Unutar četvrte cjeline nema programskih dijelova sve do naslova 4.6 -> Primjena R-a za računanje uvjetnih vjerojatnosti. Naravno, ovdje će se nalaziti Python ekvivalenti za računanje uvjetnih vjerojatnosti.

Za početak, simulirat ćemo bacanje kockica. Konkretno, 4 kockice ćemo bacati 12 puta.

```
import numpy as np

# Postavljanje početnog stanja generatora slučajnih brojeva
np.random.seed(42)

# Simulacija bacanja četiri kockice 12 puta
broj_bacanja = 12
broj_kockica = 4
DePom = np.random.choice(6, size=(broj_kockica, broj_bacanja), replace=True)

# Ispis rezultata simulacije
print(DePom)

# Dijeljenje svih elemenata matrice DePom s 6
DePom6 = DePom // 6

# Ispis rezultata dijeljenja
print(DePom6)

# Izračun maksimalnih vrijednosti u svakom stupcu
maksimalne_vrijednosti = np.max(DePom6, axis=0)

# Ispis maksimalnih vrijednosti
print(maksimalne_vrijednosti)

# Izračun prosječne vjerojatnosti kod višestrukog bacanja kockica
prosjecna_vjerojatnost = np.mean(maksimalne_vrijednosti)

# Ispis prosječne vjerojatnosti
print(prosjecna_vjerojatnost)
```

Zatim možemo pokušati napraviti isto, ali sa 100000 ponavljanja.

```
broj_simulacija = 100000
DeMere1 = np.random.choice(6, size=(broj_kockica, broj_simulacija), replace=True)

# Izračun maksimalnih vrijednosti u svakom nizu i prosječna vjerojatnost
maksimalne_vrijednosti_100k = np.max(DeMere1 // 6, axis=0)
prosjecna_vjerojatnost_100k = np.mean(maksimalne_vrijednosti_100k)

# Ispis prosječne vjerojatnosti za 100.000 ponavljanja
print(prosjecna_vjerojatnost_100k)
```

Ovaj kod simulira bacanje četiri kockice po 12 puta, izračunava maksimalne vrijednosti za svaku seriju bacanja i prosječnu vjerojatnost da će pasti određeni broj na kockici. Nakon toga, ista simulacija ponavlja se 100.000 puta i izračunava se prosječna

vjerojatnost za taj veći broj simulacija. Ovo može poslužiti za procjenu vjerojatnosti u igrama s kockicama i druge statističke analize.

Uz malu modifikaciju prethodnog koda možemo izračunati i prosječnu vjerojatnost da će pasti dvije šestice nakon 24 bacanja:

```
import numpy as np

# Postavljanje početnog stanja generatora slučajnih brojeva
np.random.seed(42)

# Broj simulacija
n_simulations = 100000

# Broj bacanja i broj kockica
n_bacanja = 24
n_kockica = 2

# Simulacija bacanja kockica
kockice = np.random.randint(1, 7, size=(n_simulations, n_bacanja, n_kockica))

# Provjerava jesu li oba rezultata u svakom bacanju jednaka 6 (šestice)
is_šestica = (kockice == 6).all(axis=2)

# Računanje maksimalnog broja šestica u 24 bacanja
maksimalne_šestice = is_šestica.sum(axis=1)

# Računanje prosječne vjerojatnosti
prosjecna_vjerojatnost = (maksimalne_šestice >= 2).mean()

print(f"Prosječna vjerojatnost da će pasti dvije šestice nakon 24 bacanja:
{prosjecna_vjerojatnost:.5f}") #.5f nam formatira output na 5 decimala
```

Možemo probati i s bacanjem četiri kockice 12 puta na sličan način.

```
import random

# Postavljanje početnog stanja generatora slučajnih brojeva (za reproduktivnost)
random.seed(42)

# Simulacija bacanja četiri kockice 12 puta
DePom = [[random.randint(1, 6) for _ in range(4)] for _ in range(12)]

# Ispis rezultata simulacije
for row in DePom:
    print(row)
```

Možemo simulirati i bacanje kockice u 12 serija po 12 puta na sljedeći način:

```
import numpy as np

# Generiranje uzorka podataka
np.random.seed(42)
```

```
DePom = np.random.randint(1, 7, size=(12, 12))

# Izračun maksimalnih vrijednosti u svakom stupcu
DePom2 = np.max(DePom, axis=0)

# Ispis maksimalnih vrijednosti
print(DePom2)

# Izračun postotka puta kada je maksimalna vrijednost 6
count_6 = np.sum(DePom2 == 6)
percentage_6 = count_6 / 12
print(percentage_6)
```

Zatim ispisujemo postotak slučajeva kada smo dobili 6.

Ukoliko želimo testirati velik broj slučajeva, možemo isprobati 100000 bacanja

```
import numpy as np

# Postavljamo širinu za bolju čitljivost
np.set_printoptions(linewidth=75)

# Generiramo simulaciju bacanja kockica
np.random.seed(42)
DePom = np.random.randint(1, 7, size=(12, 4))

# Ispisujemo logički vektor koji označava jesu li maksimalne vrijednosti u svakom
stupcu jednake 6
print(DePom == 6)

# Računamo broj puta kada je maksimalna vrijednost 6
print(np.sum(DePom == 6))

# Generiramo simulacije bacanja kockica
DeMere1 = np.random.randint(1, 7, size=(100000, 4))

# Računamo postotak puta kada je maksimalna vrijednost 6
print(np.sum(np.max(DeMere1, axis=1) == 6) / 100000)
```

Moguće je i simulirati 12 serija bacanja 4 kockice i izračunati vjerojatnost da je u barem jednom pala šestica.

```
import numpy as np

# Postavljamo početno stanje generatora slučajnih brojeva
np.random.seed(42)

# Postavljamo broj ponavljanja i broj bacanja
brpon = 12
brbac = 4
brojac = 0

# Generiramo simulaciju bacanja kockica
```

```
DePom = np.random.randint(1, 7, size=(brbac, brpon))

# Računamo procijenjenu vjerojatnost korištenjem petlji i if-ova
for i in range(brpon):
    j = 0
    while j < brbac:
        if DePom[j, i] == 6:
            brojac += 1
            j = brbac
        else:
            j += 1

# Ispisujemo procijenjenu vjerojatnost
print("Procijenjena vjerojatnost je", brojac / brpon)
```

Moguće je u Pythonu simulirati i rođenje djece. Ovaj kod simulira rođenje dvoje djece, gdje svako dijete može biti ili muško (označeno s 0) ili žensko (označeno s 1). Zatim računa uvjetnu vjerojatnost da će barem jedno od djece biti žensko, koristeći dva vektora: dijoba, koji označava jesu li oba djeteta ženskog spola, i dijjedno, koji označava je li barem jedno dijete žensko

```
# Python
import random

# Postavljanje početnog stanja generatora slučajnih brojeva
random.seed(42)

# Generiranje niza dij1 koji sadrži 20 nezavisnih bacanja prvog djeteta
dij1 = [random.choice([0, 1]) for _ in range(20)]

# Generiranje niza dij2 koji sadrži 20 nezavisnih bacanja drugog djeteta
dij2 = [random.choice([0, 1]) for _ in range(20)]

# Formiranje vektora dijoba koji označava je li barem jedno dijete žensko
dijoba = [d1 * d2 for d1, d2 in zip(dij1, dij2)]

# Formiranje podataka o djeci
dijpom = list(zip(dij1, dij2))

# Računanje vektora dijjedno koji označava imaju li oba djeteta barem jedno žensko
dijete
dijjedno = [max(d1, d2) for d1, d2 in dijpom]

# Izračun uvjetne vjerojatnosti
print("Tražena uvjetna vjerojatnost je", sum(dijoba) / sum(dijjedno))
```

Istu funkcionalnost možemo postići korištenjem NumPy-a, samo što će kod s NumPy-jem raditi brže i efikasnije.

```
# Python
import numpy as np

# Postavljanje početnog stanja generatora slučajnih brojeva
```

```
np.random.seed(42)

# Generiranje niza dij1 koji sadrži 20 nezavisnih bacanja prvog djeteta
dij1 = np.random.choice([0, 1], 20, replace=True)

# Generiranje niza dij2 koji sadrži 20 nezavisnih bacanja drugog djeteta
dij2 = np.random.choice([0, 1], 20, replace=True)

# Izračun broja slučajeva u kojima barem jedno dijete je žensko
nC = np.sum((dij1 == 1) | (dij2 == 1))

# Izračun broja slučajeva u kojima oba djeteta su ženska
nAC = np.sum((dij1 == 1) & (dij2 == 1))

# Izračun uvjetne vjerojatnosti
print("Tražena uvjetna vjerojatnost je", nAC / nC)
```

Na istu tematiku može se napraviti još sličnih simulacija, pomoću random.choice i zip.a(zip služi za povezivanje dvaju Python objekata -> u ovom slučaju listi):

```
import random

random.seed(42)

dij1 = [random.choice([0, 1]) for _ in range(100000)]
dij2 = [random.choice([0, 1]) for _ in range(100000)]

dijoba = [d1 * d2 for d1, d2 in zip(dij1, dij2)]
dijpom = list(zip(dij1, dij2))
dijjedno = [max(d1, d2) for d1, d2 in dijpom]

uvjetna_vjerojatnost = sum(dijoba) / sum(dijjedno)

print(f"Tražena uvjetna vjerojatnost je {uvjetna_vjerojatnost}")

import random

random.seed(42)

dij1 = [random.choice([0, 1]) for _ in range(100000)]
dij2 = [random.choice([0, 1]) for _ in range(100000)]

nC = sum(d1 == 1 or d2 == 1 for d1, d2 in zip(dij1, dij2))
nAC = sum(d1 == 1 and d2 == 1 for d1, d2 in zip(dij1, dij2))

uvjetna_vjerojatnost = nAC / nC
```

Za sam kraj, pozabavit ćemo se izvlačenjem kuglica crvene i plave boje. Simulirat ćemo situaciju s plavim (označeno s "p") i crvenim (označeno s "c") kuglicama te računati uvjetnu vjerojatnost da će, pod uvjetom da je donesena odluka o odabiru kuglice, kuglica zaista biti crvena.

```python
import random

random.seed(42)
kug = [0] * 100000

nov = [random.choices([0, 1], [0.45, 0.55])[0] for _ in range(100000)]

for i in range(100000):
    if nov[i] == 0:
        kug[i] = random.choices(["p", "c"], [1/3, 2/3])[0]
    else:
        kug[i] = random.choices(["p", "c"], [2/3, 1/3])[0]

nA = kug.count("p")
nAH = sum(1 for i in range(100000) if nov[i] == 0 and kug[i] == "p")

print("Tražena uvjetna vjerojatnost je", nAH / nA)
```