Projekt nr 7

Wojciech Sobczuk

21.05.2020

1 Rozpatrywany problem

Użytkownik parasola: Mam jeden parasol i moje życie jest podzielone na n miejsc. Podróżuję tam i z powrotem na piechotę. Jeśli pada deszcz, kiedy wyruszam w podróż, zabieram parasol, jesli jest ze mna (nie zwracaj uwagi na deszcz). Pada podczas każdej podróży z prawdopodobienstwem p niezależnie dla wszystkich podróży. Jeśli jest sucho, zostawiam parasol w miejscu, z którego wyruszam. Rozważ dwa możliwe modele życia uzytkownika:

(a) Wszystkie n lokalizacji jest odwiedzanych w ustalonej kolejności.

(b) Użytkownik przemieszcza się pomiędzy miejscami losowo, niezależnie od przeszłości

Pokaż, że w obu przypadkach odsetek długoterminowych podróży, na których użytkownik parasola zmoknie, to $\frac{p(n-1)(1-p)}{1+(n-1)(1-p)}$

2 Etap I

2.1 Pojęcia teoretyczne

2.1.1 Prawdopodobieństwo warunkowe

Prawdopodobieństwem warunkowym zajścia zdarzenia A pod warunkiem zdarzenia B nazywamy liczbę $P(A|B) = \frac{P(A \cap B)}{P(B)}$, gdzie $P(B) > 0$. Pojęcie to użyję do definicji łańcucha Markowa.

2.1.2 Łańcuch Markowa

Niech S będzie zbiorem stanów. Łańcuchem Markowa nazywamy ciąg zmiennych $X_0, X_1, X_2 \ldots$ taki, że

$$P(X_n = s_n | X_{n-1} = s_{n-1} \wedge X_{n-2} = s_{n-2} \wedge \ldots \wedge X_0 = s_0) = P(X_n = s_n | X_{n-1} = s_{n-1})$$

dla każdego $n > 1$ oraz dla każdego $s_1, s_2, \ldots, s_n \in S$. Pojęcie to jest używane w wariancie (b) problemu projektowego.

2.1.3 Macierz przejścia

Rozważmy prawdopodobieństwo przejścia ze stanu $X_n = i$ do stanu
$X_{n+1} = j$.

$$P(X_n = i | X_{n+1} = j) = P_{ij}.$$

Wówczas, macierz przejścia definiujemy jako

$$\begin{bmatrix} P_{00} & P_{01} & P_{02} & \ldots \\ P_{10} & P_{11} & P_{12} & \ldots \\ P_{20} & P_{21} & P_{22} & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

gdzie $0 \leq P_{ij} \leq 1$ oraz $\sum_{j=1}^{\infty} P_{ij} = 1$. Pojęcie to jest używane w wariancie (b) problemu projektowego do wyznaczenia prawdopodobieństwa przejścia do następnego miejsca.

2.1.4 Przestrzeń zdarzeń elementarnych

Przestrzenią zdarzeń elementarnych nazywamy zbiór wszystkich możliwych wyników doświadczenia losowego. Pojedynczy element tego zbioru nazywamy zdarzeniem elementarnym. Pojęcie to użyję do definicji zmiennej losowej oraz rozkładu klasycznego prawdopodobieństwa.

2.1.5 Zmienna losowa

Zmienną losową nazywamy funkcję przyporządkowującą liczby zdarzeniom elementarnym. Pojęcie to użyję do definicji procesu stochastycznego.

2.1.6 Proces stochastyczny

Rodzina zmiennych losowych $\{X_t, t \in T\}$, gdzie T jest podzbiorem zbioru $(-\infty, \infty)$, tworzy proces stochastyczny. Poprzez ten proces realizuję zadany problem projektowy.

2.1.7 Rozkład klasyczny prawdopodobieństwa

Rozkład klasyczny prawdopodobieństwa to taki rozkład, w którym wszystkie zdarzenia elementarne są równo prawdopodobne i $P(X \in A) = \frac{|A|}{|\Omega|}$. Pojęcie to jest wykorzystywane w podpunkcie (b) problemu projektowego do wyznaczenia prawdopodobieństwa przejścia do następnego miejsca.

2.2 Plan symulacji

Problem projektowy wymaga rozpatrzenia dwóch przypadków: człowiek porusza się po n miejscach w sposób ustalony oraz człowiek porusza się po n miejscach w sposób losowy.
Każdy z tych przypadków zależy od następujących parametrów:

- p – prawdopodobieństwo że w danej podróży pada;
- n – ilość miejsc w życiu podróżnika, wartość ta będzie przyjmowała wartości $100, 1000, 10000, 100000$;
- powt – ilość powtórzeń żyć podróżnika, co w esencji jest ilością powtórzeń symulacji, wartość ta będzie stale ustalona na 10000.

Większe wartości parametrów n oraz *powt* powodują, że czas jednej symulacji zbliża się do paru godzin dlatego zdecydowałem się na ograniczenie tych wartości.

Kod odpowiedzialny za padanie deszczu w danym miejscu będzie zbliżony do kodu kroku spaceru losowego. Zakładam, że podróżą jest przejście z punktu a do punktu b, a życie to ciąg podróży. Zatem sprawdzać będziemy w ilu podróżach w trakcie życia podróżnik zmókł.

Symulacja życia będzie polegała na wyznaczeniu miejsca, do którego podróżnik przejdzie w następnej podróży oraz na sprawdzaniu warunku, czy w danej podróży człowiek zmókł, co zależy od prawdopodobieństwa p oraz od faktu, czy podróżnik ma ze sobą parasol. Fakt ten będzie sprawdzany poprzez zmienną logiczną, która przyjmie wartość True, gdy podróżnik będzie miał parasol ze sobą oraz tablicę zmiennych logicznych, wielkości n, zainicjowana poprzez podanie w całej tablicy wartości False, która będzie odpowiadała za sprawdzenie w którym miejscu został parasol.

Jeśli podróżnik pozostawi parasol w miejscu x, tablica zmiennych losowych zmieni wartość na True na pozycji z indeksem x oraz wartość zmiennej logicznej zmieni się na False. Tablica zmiennych logicznych będzie użyta tylko w przypadku drugim.

Pojedyncza podróż będzie funkcją która zwracać będzie liczbę - odsetek podróży, w których podróżnik zmókł, symulacja natomiast zwracać będzie tablicę wypełnioną odsetkami tak, aby w łatwy sposób móc używać metod z biblioteki numpy.

Człowiek porusza się w sposób ustalony

Nazywając miejsca, które człowiek odwiedzi przez kolejne cyfry mogę założyć, że otrzymamy ciąg $(1, 2, \ldots, n)$. Jest to możliwe dlatego, że dowolna permutacja tych punktów skutkować będzie takim samym wynikiem symulacji, co oznacza że nazewnictwo punktów nie ma znaczenia. Do wykonania symulacji wystarczy prosta pętla, która symulować będzie podróż po kolejnych punktach, tzn. podróżnik przejdzie z punktu 1 do punktu 2, następnie z punktu 2 do punktu 3 i tak dalej, aż do przejścia z punktu $n-1$ do punktu n.

Człowiek porusza się losowo

W treści zadania podane jest wybór miejsca, do którego podróżnik się wybierze nie zależy od przeszłości. Oznacza to że mamy do czynienia z łańcuchem Markowa. Rozpatrzmy macierz przejścia dla różnych n:

- dla n=3: $\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$
- dla n=4: $\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$
- w ogólnym przypadku:

$$\begin{bmatrix} 0 & \frac{1}{n-1} & \cdots & \frac{1}{n-1} & \frac{1}{n-1} \\ \frac{1}{n-1} & 0 & \cdots & \frac{1}{n-1} & \frac{1}{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{n-1} & \frac{1}{n-1} & \cdots & 0 & \frac{1}{n-1} \\ \frac{1}{n-1} & \frac{1}{n-1} & \cdots & \frac{1}{n-1} & 0 \end{bmatrix}$$

Można więc zauważyć, że dla prawdopodobieństwo przejścia do dowolnego miejsca jest równe $\frac{1}{n-1}$, poza przypadkiem, gdy stoimy w miejscu - wtedy prawdopodobieństwo wynosi 0. Tak więc, zmienna losowa X odpowiadająca za przydzielenie miejsca, do którego podróżnik się wybierze (poza miejscem startowym) ma rozkład klasyczny.

Programistycznie będzie się to odbywało za pomocą wylosowania liczby z przedziału $[1, n]$, do którego człowiek przejdzie. Należy jednak dodać warunek, który uniemożliwi mu stanie w miejscu.
W tym wariancie rozpatrzę trzy możliwości:

i. Życie zakończy się gdy podróżnik wróci do miejsca startu;

ii. Życie zakończy się po wykonaniu m podróży zadanych parametrem;

iii. Życie zakończy się po odwiedzeniu wszystkich n miejsc co najmniej raz.

Sprawdzę wszystkie trzy możliwości i ocenię, która z nich przynosi satysfakcjonujący wynik.

2.3 Metody statystyczne

Użyję dwóch testów statystycznych:

2.3.1 Test normalności Kołmogorowa-Smirnowa

Służy on do ustalenia, czy dany rozkład jest rozkładem normalnym. Hipotezą zerową będzie założenie, że rozkład odsetków podróży, w których człowiek zmókł jest rozkładem normalnym, hipotezą alternatywną będzie założenie, że ten rozkład nie jest rozkładem normalnym.

Normalność rozkładu jest nam potrzebna do drugiego testu.

2.3.2 Test t-Studenta dla pojedynczej próby

Służy on do weryfikacji hipotezy, że badana próba o średniej $\overline{x}$ pochodzi z populacji, dla której średnia μ to zadana wartość. Hipotezą zerową będzie założenie, że średni wynik z symulacji jest w istocie wartością, którą mamy udowodnić. Hipoteza alternatywna natomiast będzie to założenie obalać.

2.4 Analiza wyników

Dla wariantów A i B1 przeprowadzę wielokrotne symulacje zmieniając za każdym razem parametry n, p i wyliczając średni wynik symulacji. Parametr n będzie przyjmował następujące wartości: $100, 1000, 10000, 100000$, a parametr p będzie przyjmował wartości $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$.

Następnie wykonam symulacje dla każdej wartości n oraz p. Łącznie, w jednym wariancie zostanie wykonanych 36 symulacji.

W wariancie B2 parametr p będzie zmieniał się jak wyżej, parametr n przyjmie wartości $1000, 10000, 100000$ a parametr m przyjmie wartości $100, 1000, 10000, 100000$.

W wariancie B3 parametr p będzie przyjmował wartości takie jak wcześniej, a parametr n będzie przyjmował wartości $100, 1000, 2000$.

Użyję histogramu do obserwacji rozkładu wyników symulacji. Następnie naszkicuję wykres gęstości rozkładu oraz porównam go z wykresem rozkładu normalnego dla średniej i odchylenia standardowego z symulacji.

Dla każdej zmiany parametrów najpierw wykonam test normalności rozkładu, ponieważ jest on niezbędny do przeprowadzenia testu t-Studenta. Jeśli będę mógł przyjąć hipotezę zerową z tego testu dla danych parametrów, przejdę do wykonania drugiego testu.

Będę sprawdzał, czy dla danej pary (lub trójki w wariancie drugim podpunktu b) parametrów statystyka testowa p pozwala na przyjęcie hipotezy zerowej. Jeśli hipoteza zerowa będzie mogła zostać przyjęta dla parametru n (lub dodatkowo m w wariancie drugim podpunktu B), pozwalającego na wykonanie symulacji w rozsądnym czasie, oznaczać to będzie że symulacja przebiegła zgodnie z planem. Jeśli jednak zmuszony będę odrzucić hipotezę zerową i przyjąć hipotezę alternatywną dla któregokolwiek z testów, oznaczać to będzie, że założenia są niepoprawne. Jeśli natomiast czas wykonywania symulacji będzie za długi, zmniejszę parametr n lub w ostateczności *powt*.

Oczekuję, że dla dużych wartości n średnia osiągnie dokładnie wartość oczekiwaną. Aby zaobserwować jak daleko średnia znajduje się od wartości oczekiwanej, policzę wartość bezwzględną różnicy pomiędzy nimi.

3 Etap 2

3.1 Omówienie kodu

Zaimportowałem następujące biblioteki:

- **numpy** do wygodniejszego deklarowania niektórych tablic oraz funkcji losującej liczbę z przedziału $[0, 1]$,
- **random** do losowania liczby z przedziału $[0, 1]$ potrzebnej do wyznaczenia prawdopodobieństwa deszczu,
- **time**, pozwalająca na zmierzenie czasu wykonywania symulacji,
- **scipy.stats**, która jest niezbędna do przeprowadzenia testów statystycznych,
- **seaborn** do narysowania wykresu dystrybuanty rozkładu oraz histogramu.

Omówię teraz kod, który powtarza się we wszystkich 4 wariantach rozpatrywania problemu. Są to fragmenty kodu wycięte z rozpatrywania wariantu A problemu.

```
def krok(p):
    a=np.random.rand()
    if a<p:
        return 1
    else:
        return -1
```

Funkcja krok(p) służy do wyznaczania miejsc, w których pada deszcz. Parametr p odpowiada za prawdopodobieństwo, że w danej podróży pada. Liczba a to wylosowana liczba z przedziału $[0, 1]$. Jeśli będzie ona mniejsza niż p, czyli $a \in (0, p)$, to oznacza że w danej podróży będzie padał deszcz. W przeciwnym przypadku, w danej podróży będzie sucho. Funkcja zwraca 1 gdy pada oraz -1 gdy nie pada, zostanie to użyte w późniejszej funkcji.

```
def podrA(n,p):
    mok=0
    mam_par=True
    for i in range(n):
        k=krok(p)
        if k == -1:
            if mam_par:
                mam_par=False
        if k == 1:
            if mam_par:
                continue
            else:
                mok=mok+1
    return (mok/n)
```

Powyżej mamy fragment funkcji odpowiedzialnej za zliczanie ilości podróży, w których podróżnik zmókł. Parametr n to liczba miejsc, którą użytkownik ma przejść a p to prawdopodobieństwo deszczu. Funkcja ta to szereg instrukcji warunkowych, które pozwalają na wyznaczenie tych miejsc. Zmienna k przechowuje wartość 1 lub -1, na zasadzie opisanej wcześniej.

Pierwsza instrukcja warunkowa sprawdza, czy w danej podróży będzie padał deszcz. Jeśli nie, sprawdzamy czy podróżnik miał parasol przy sobie w tej konkretnej podróży. Fakt ten jest przechowywany w zmiennej logicznej *mam_par*, która jest zainicjalizowana wartością True. Jeśli podróżnik miał parasol, czyli zmienna *mam_par* była True, to zgodnie z założeniami problemowymi zostawia on go w miejscu, w którym obecnie przebywa i jednocześnie wartość zmiennej *mam_par* zmienia się na False. Jeśli nie miał parasola to nic się nie dzieje.

Jeśli jednak zmienna k przyjmie wartość jeden, to znów sprawdzamy, czy użytkownik miał parasol. Jeżeli miał, czyli wartość zmiennej *mam_par* była True, to nic konkretnego się nie dzieje, podróżnik został uchroniony od deszczu. W przeciwnym wypadku jednak on moknie i należy dodać jeden do zmiennej *mok*, która zlicza ilość podróży w których podróżnik zmókł.

Funkcja ta zwraca odsetek podróży w których człowiek zmókł.

```
def zycieA(il_pow,n,p):
    wyn=[]
    for i in range(il_pow):
        sym=podrA(n,p)
        wyn.append(sym)
    return wyn
```

Funkcja ta odpowiada za symulację. Podajemy jej parametry *il_pow*, która odpowiada ilości powtórzeń symulacji, n i p pełnią rolę opisaną wcześniej. Składa się ona z pętli wykonywanej *il_pow* razy, w której symulujemy pojedyncze życia podróżnika. Poszczególne odsetki zapisujemy do tablicy, aby łatwiej analizować wyniki.

```
def symulacjaA(il_pow,n,p):
    start = time.time()
    print("Symulacja dla parametrów n =",n,"p =",p)
    wsp=(p*(n-1)*(1-p))/(1+(n-1)*(1-p))
    sym=zycieA(il_pow,n,p)
    pvalks=round(scs.kstest(sym,'norm',args=(np.mean(sym),np.std(sym)))[1],4)
    pvalts=round(scs.ttest_1samp(sym,wsp)[1],4)
    if pvalks>1-poz_uf:
        print("Przyjmujemy hipoteze zerową w teście normalności Kołmogorowa-Smirnova. Rozkład jest normalny.")
        print("pvalue wyniosło",pvalks)
        if pvalts>1-poz_uf:
            print("Przyjmujemy hipoteze zerową w teście t-Studenta. Wartość średnia jest naszą wartością oczekiwaną.")
            print("pvalue wyniosło",pvalts)
            tekst = """
            Symulacja dla parametrów n = {n}, p = {p:5.2f}
            Przyjmujemy hipoteze zerową w teście normalności Kołmogorowa-Smirnova. Rozkład jest normalny.
            pvalue wyniosło {pvalks}
            Przyjmujemy hipoteze zerową w teście t-Studenta. Wartość średnia jest naszą wartością oczekiwaną.
            pvalue wyniosło {pvalts}
            Różnica pomiędzy średnią a wartością oczekiwaną wynosi {roz}
            """
            zmienne = {
             "n":n,
             "p":p,
             "pvalks": pvalks,
             "pvalts" : pvalts,
             "roz" : abs(np.mean(sym)-wsp)
             }
            with  open('symA.txt','a',encoding="utf-8") as myfile:
                myfile.write(tekst.format(**zmienne))
        else:
            print("Odrzucamy hipoteze zerową w teście t-Studenta. Wartość średnia nie jest naszą wartością oczekiwaną.")
            print("pvalue wyniosło",pvalts)
            tekst = """
            Symulacja dla parametrów n = {n}, p = {p:5.2f}
            Przyjmujemy hipoteze zerową w teście normalności Kołmogorowa-Smirnova. Rozkład jest normalny.
            pvalue wyniosło {pvalks}
            Odrzucamy hipoteze zerową w teście t-Studenta. Wartość średnia nie jest naszą wartością oczekiwaną.
            pvalue wyniosło {pvalts}
            Różnica pomiędzy średnią a wartością oczekiwaną wynosi {roz}
            """
            zmienne = {
             "n":n,
             "p":p,
             "pvalks": pvalks,
             "pvalts" : pvalts,
             "roz" : abs(np.mean(sym)-wsp)
             }
            with  open('symA.txt','a',encoding="utf-8") as myfile:
                myfile.write(tekst.format(**zmienne))
    else:
        print("Odrzucamy hipoteze zerową w teście normalności Kołmogorowa-Smirnova. Rozkład nie jest normalny.")
        print("pvalue wyniosło",pvalks)
        tekst = """
        Symulacja dla parametrów n = {n}, p = {p:5.2f}
        Odrzucamy hipoteze zerową w teście normalności Kołmogorowa-Smirnova. Rozkład nie jest normalny.
        pvalue wyniosło {pvalks}
        Różnica pomiędzy średnią a wartością oczekiwaną wynosi {roz}
        """
        zmienne = {
        "n":n,
        "p":p,
        "pvalks": pvalks,
         "roz" : abs(np.mean(sym)-wsp)
            }
        with  open('symA.txt','a',encoding="utf-8") as myfile:
            myfile.write(tekst.format(**zmienne))
    print("")
    end = time.time()
    print("Symulacja wykonana w",end - start,"sekund")
    return sym
```

Umieściłem zdjęcie tej funkcji ponieważ niektóre polecenia są zbyt długie, aby móc je wpisać bezpośrednio do LaTeXa. Funkcja ta służy do ujednolicenia i ułatwienia przeprowadzania wielu symulacji z różnymi parametrami. Przyjmuje ona parametry takie same jak funkcja *zycie*. Zmienne *pvalks* i *pvalts* to wartości pvalue testów statystycznych, kolejno testu normalności Kołmogorowa-Smirnowa i testu t-Studenta zaokrąglone do 4 cyfr po przecinku. Następnie rozpoczyna się blok instrukcji warunkowych, które sprawdzają, czy mogę przyjąć dane hipotezy zerowe. Zmienna *tekst* oraz słownik *zmienne* służą do zapisywania wyniku symulacji do pliku tekstowego. Funkcja time.time() służy do zmierzenia czasu wykonania symulacji. Funkcja ta zwraca tablicę wygenerowaną w poprzedniej funkcji. Funckja ta pozostaje prawie identyczna dla każdego wariantu, jedyne co się zmienia to ilość parametrów, przekazywana funkcja *zycie* oraz tytuł pliku tekstowego, do którego jest zapisywany przebieg symulacji.

3.1.1 Człowiek porusza się w sposób ustalony

Symulacja odbywa się za pomocą funkcji, które zostały pokazane na zdjęciach powyżej. Przejście po n ustalonych miejscach odbywa się za pomocą pętli iterującej po przedziale $[0, n-1]$. Numer miejsca, w którym podróżnik obecnie się znajduje to $i+1$. Reszta funkcji działa zgodnie z opisem powyżej.

3.1.2 Człowiek porusza się w sposób losowy

Wariant 1: Życie zakończy się gdy podróżnik wróci do miejsca startu

Symulacja odbywa się za pomocą funkcji krok i symulacjaB1, których działanie jest opisane w dziale z ogólnym omówieniem kodu, oraz podrB1 i zycB1.

```
def podrB1(n,p,pocz):
    mok=0
    zost_par=np.full([n+1,1],False)
    mam_par=True
    miejsce=pocz
    i=1
    while(True):
        k=krok(p)
        if k == -1:
            if mam_par:
                zost_par[miejsce]=True
                mam_par=False
        if k == 1:
            if zost_par[miejsce] or mam_par:
                zost_par[miejsce]=False
                mam_par=True
            else:
                mok=mok+1

        miejsce_pop=miejsce
        while(miejsce==miejsce_pop):
            miejsce=random.randrange(1,n+1)
        if(miejsce==pocz):
            break
        i=i+1
    return (mok/i)
```

Funkcja ta różni się od funkcji podrA dodatkowym parametrem podawanym na wejściu, jest to liczba *pocz*, która jest odpowiedzialna za ustalenie miejsca, z którego podróżnik rozpoczyna podróże. Zmienna *miejsce* odpowiada za ustalenie, w którym miejscu człowiek obecnie przebywa. Tablica *zost_par* to tablica zmiennych logicznych, która jest inicjalizowana wartością False na n pozycjach. Tablica ta jest odpowiedzialna za ustalenie, w którym miejscu został parasol. Gdy warunki na to pozwalają, podróznik zostawia parasol w miejscu *miejsce* a wartość w tablicy na pozycji *miejsce* zmienia się na True. Gdy parasol zostanie podniesiony z miejsca, w którym on poprzednio pozostał, wartość tablicy w miejscu *miejsce* zostanie zmieniona na False. Zmienna i odpowiada za zliczanie ilości podróży.

Blok instrukcji warunkowych różni się od tego omówionego w ogólnym przypadku dodaniem tablicy zmiennych logicznych do sprawdzanych warunków, co zostało opisane powyżej.

Pętla jest nieskończona ale zrywa ją warunek widoczny na samym dole, kiedy wylosowane miejsce równa się początkowemu miejscu. Oznacza to, że pętla będzie iterowała tak długo, aż nie znajdziemy się w miejscu początkowym, co jest zgodne z założeniem tego wariantu. Pętla powyżej sprawdza, czy następne miejsce jest na pewno różne od obecnego.

```
def zycieB1(il_pow,n,p):
    wyn=[]
    for i in range(il_pow):
        pocz=random.randrange(1,n+1)
        sym=podrB1(n,p,pocz)
        wyn.append(sym)
    return wyn
```

Funkcja ta różni się od przedstawionej zycieA jedynie tym, że przy każdej iteracji losujemy wartość *pocz*, która odpowiada miejscu w którym rozpoczynamy podróż.

Wariant 2: Życie zakończy się po wykonaniu m podróży zadanych parametrem

Symulacja odbywa się przy pomocy funkcji krok, podrB2, zycieB2 i symulacjaB2. Funkcje krok i symulacjaB2 nie różnią się niczym od wcześniej przedstawionych.

```
def podrB2(n,il,p,pocz):
    mok=0
    zost_par=np.full([n+1,1],False)
    mam_par=True
    miejsce=pocz
    i=0

    while(il>i):
        k=krok(p)
        if k == -1:
            if mam_par:
                zost_par[miejsce]=True
                mam_par=False
        if k == 1:
            if zost_par[miejsce] or mam_par:
                zost_par[miejsce]=False
                mam_par=True
            else:
                mok=mok+1

        miejsce_pop=miejsce
        while(miejsce==miejsce_pop):
            miejsce=random.randrange(1,n+1)
        i=i+1
    return mok/il
```

W funkcji podrB2 pętla iteruje tak długo aż powtórzy się *il* razy. Jest to parametr m podany przez użytkownika.

```
def zycieB2(il_pow,m,n,p):
    wyn=[]
    for i in range(il_pow):
        pocz=random.randrange(1,n+1)
        sym=podrB2(n,m,p,pocz)
        wyn.append(sym)
    return wyn
```

Funkcja zycieB2 różni się od funkcji zycieB1 jedynie dodatkowym parametrem m przekazywanym do funkcji podrB2.

Wariant 3: Życie zakończy się po odwiedzeniu wszystkich n miejsc co najmniej raz

Symulacja odbywa się za pomocą funkcji krok, podrB3, zycieB3 oraz symulacjaB3. Funkcje krok i symulacja B3 nie różnią się od przedstawionych wczześniej.

```
def podrB3(n,p,pocz):
    mok=0
    zost_par=np.full([n+1,1],False)
    odwiedzone=np.zeros((n+1,),dtype=int)
    odwiedzone[0]=1
    mam_par=True
    miejsce=pocz
    i=1
    while(True):
        odwiedzone[miejsce]=odwiedzone[miejsce]+1
        k=krok(p)
        if k == -1:
            if mam_par:
                zost_par[miejsce]=True
                mam_par=False
        if k == 1:
            if zost_par[miejsce] or mam_par:
                zost_par[miejsce]=False
                mam_par=True
            else:
                mok=mok+1

        miejsce_pop=miejsce
        while(miejsce==miejsce_pop):
            miejsce=random.randrange(1,n+1)
        if(all(i >= 1 for i in odwiedzone)):
            break

        i=i+1
    return mok/i
```

W tym wariancie dodatkowo dodałem tablicę *odwiedzone*, która zlicza ile razy dane miejsce zostało odwiedzone. Wartość o indeksie 0 jest zainicjalizowana wartością 1, żeby bez problemu móc sprawdzać warunek widoczny na samym końcu funkcji. Nie zmienia to przebiegu symulacji. Blok instrukcji warunkowych jest wykonywany w nieskończonej pętli, która kończona jest warunkiem sprawdzającym, czy wszystkie elementy w tablicy *odwiedzone* są większe bądź równe 1. Oznacza to, że odwiedziliśmy wszystkie miejsca conajmniej raz.

```
def zycieB3(il_pow,n,p):
    wyn=[]
    for i in range(il_pow):
        pocz=random.randrange(1,n+1)
        sym=podrB3(n,p,pocz)
        wyn.append(sym)
    return wyn
```

Funkcja zycieB3 nie różni się niczym od funkcji zycieB1 opisanej wcześniej.

3.2 Symulacja

```
il_pow=10000
n=[100,1000,10000,100000]
p=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
poz_uf=0.95

fig,axes = plt.subplots(len(n),len(p),figsize=(30,8))
for i in range(len(n)):
    for j in range(len(p)):
        sym = symulacjaA(il_pow,n[i],p[j])
        sns.distplot(sym,ax=axes[i,j]).set_title("n="+str(n[i])+",p="+str(p[j]))
plt.savefig("sym.png")
plt.show()
```

Powyżej zostal umieszczony kod, który przeprowadza symulacje. Funkcja *sns.distplot* pozwala na wygenerowanie wykresu gęstości z histogramem rozkładu.

Wyniki symulacji będą przedstawione w formie tabelki. Wierszami są wartości parametru n, a kolumnami wartości parametru p. W pierwszej tabelce umieszczone będą wartości pvalue dla testu normalności Kołmogorowa-Smirnowa, a w drugiej wartości pvalue dla testu t-Studenta. Jeśli odrzucona została hipoteza zerowa dla testu normalności dla danych parametrów, to w dolnej tabelce zostanie wpisana wartość n/w. Na zielono zostaną zaznaczone wartości pvalue, dla których przyjęta została hipoteza zerowa, a na czerwono wartości, dla których została ona odrzucona.

Zamieszczone zdjęcia mają na celu zobaczenie na pierwszy rzut oka, jak wygląda rozkład dla danych parametrów.

Poziom ufności ustaliłem na 0.95.

3.2.1 Człowiek porusza się w sposób ustalony

Tablica 1: Wartości pvalue dla testu Kołmogorowa-Smirnowa

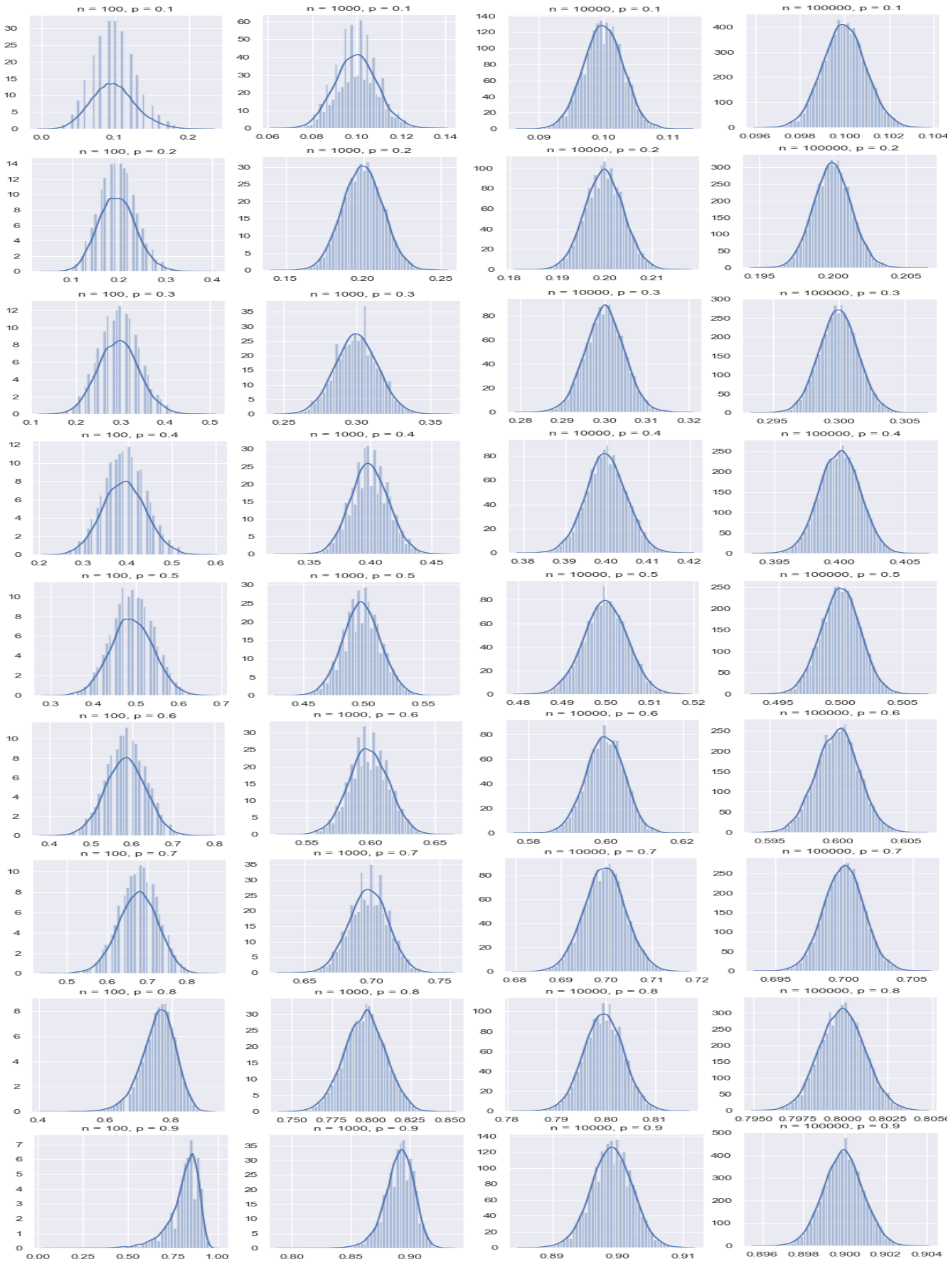
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1000	0.000	0.002	0.005	0.000	0.002	0.001	0.000	0.000	0.000
10000	0.239	0.366	0.325	0.151	0.692	0.077	0.284	0.585	0.202
100000	0.832	0.689	0.976	0.491	0.278	0.241	0.849	0.329	0.680

Z tabelki powyżej wynika, że dla dużych wartości n mogę przyjąć, że rozkład jest normalny.

Tablica 2: Wartości pvalue dla testu t-Studenta

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
1000	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
10000	0.672	0.037	0.151	0.446	0.705	0.703	0.263	0.488	0.590
100000	0.070	0.298	0.647	0.516	0.834	0.520	0.191	0.412	0.058

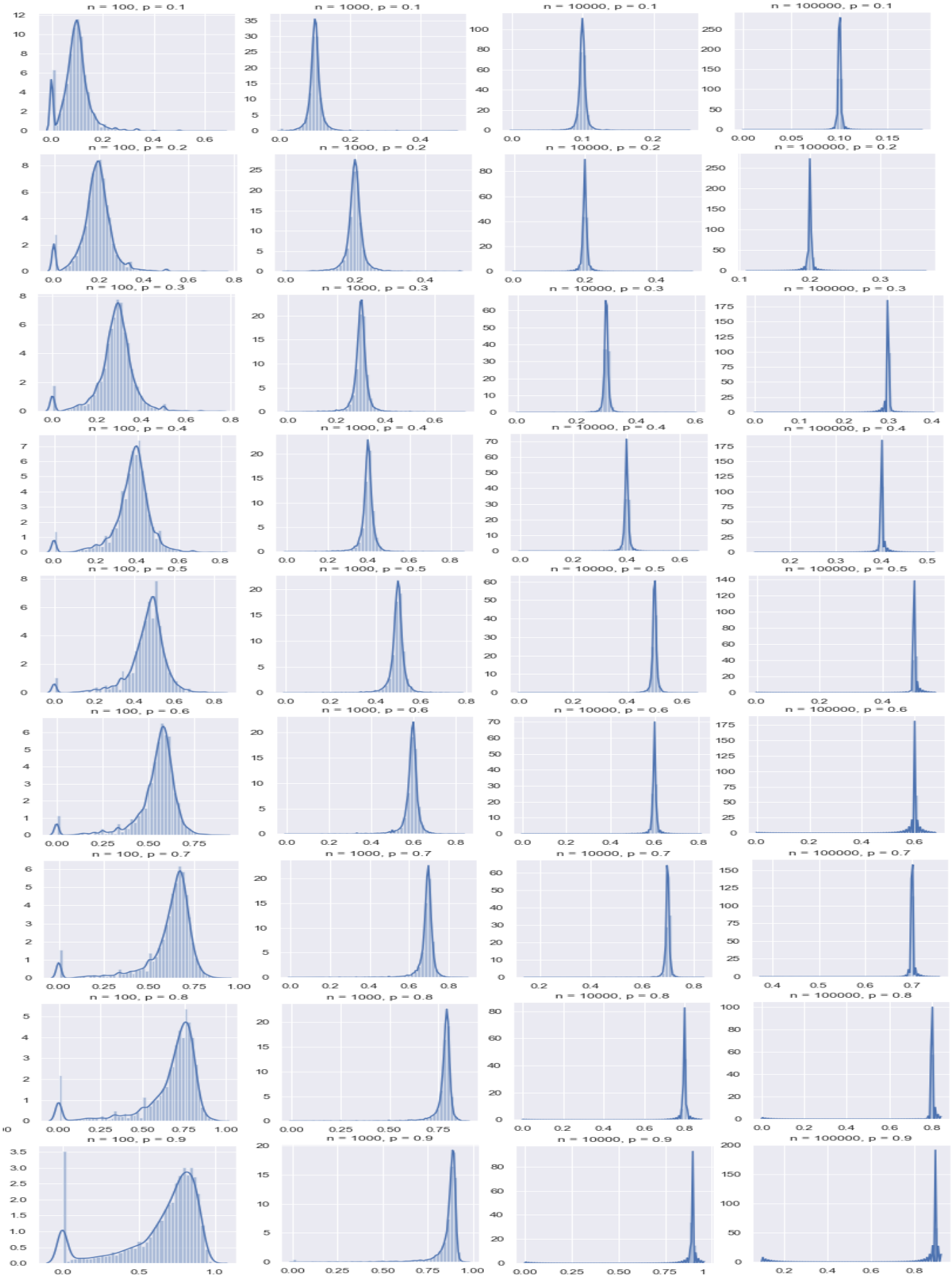
Z tabelki powyżej wynika, iż dla dużych wartości n mogę przyjąć, że wartość średnia jest naszą wartością oczekiwaną. Udowodniłem, więc problem projektowy dla wariantu A. Bardziej szczegółowe wyniki otworzą się po kliknięciu w załącznik.



Rysunek 1: Wyniki symulacji dla wariantu A, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja

3.2.2 Człowiek porusza się w sposób losowy

Wariant 1: Życie zakończy się, gdy podróżnik wróci do miejsca startu
W tym przypadku nie będzie potrzebna tabelka. Dla każdej pary parametrów wartość pvalue dla testu normalności wyniosła 0. Oznacza to, że mamy podstawy do stwierdzenia, że rozkład w tym wariancie nie jest normalny. Moim zdaniem wynika to z faktu, że długości podróży mogą przyjmować wartości o bardzo dużym rozstrzale, konkretnie od 2 do nieskończoności. Przez to odsetek również przyjmuje szeroko rozproszone wartości i nie skupia się wokół średniej w sposób normalny. Fakt ten można zaobserwować na zdjęciu poniżej, szczególnie w pierwszej kolumnie. Szczegółowe wyniki symulacji pokażą się po kliknięciu w załącznik.



Rysunek 2: Wyniki symulacji dla wariantu B1, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja

Wariant 2: Życie zakończy się po wykonaniu m podróży zadanych parametrem
W tym wariancie zmieniamy aż trzy parametry i w związku z tym zdecydowałem się na umieszczenie tabel dla różnych wartości parametru n. W tabelach poniżej kolumnami są wartości parametru p, a wierszami wartości parametru m.
Poniżej znajdują się symulacje dla parametru $n = 1000$.

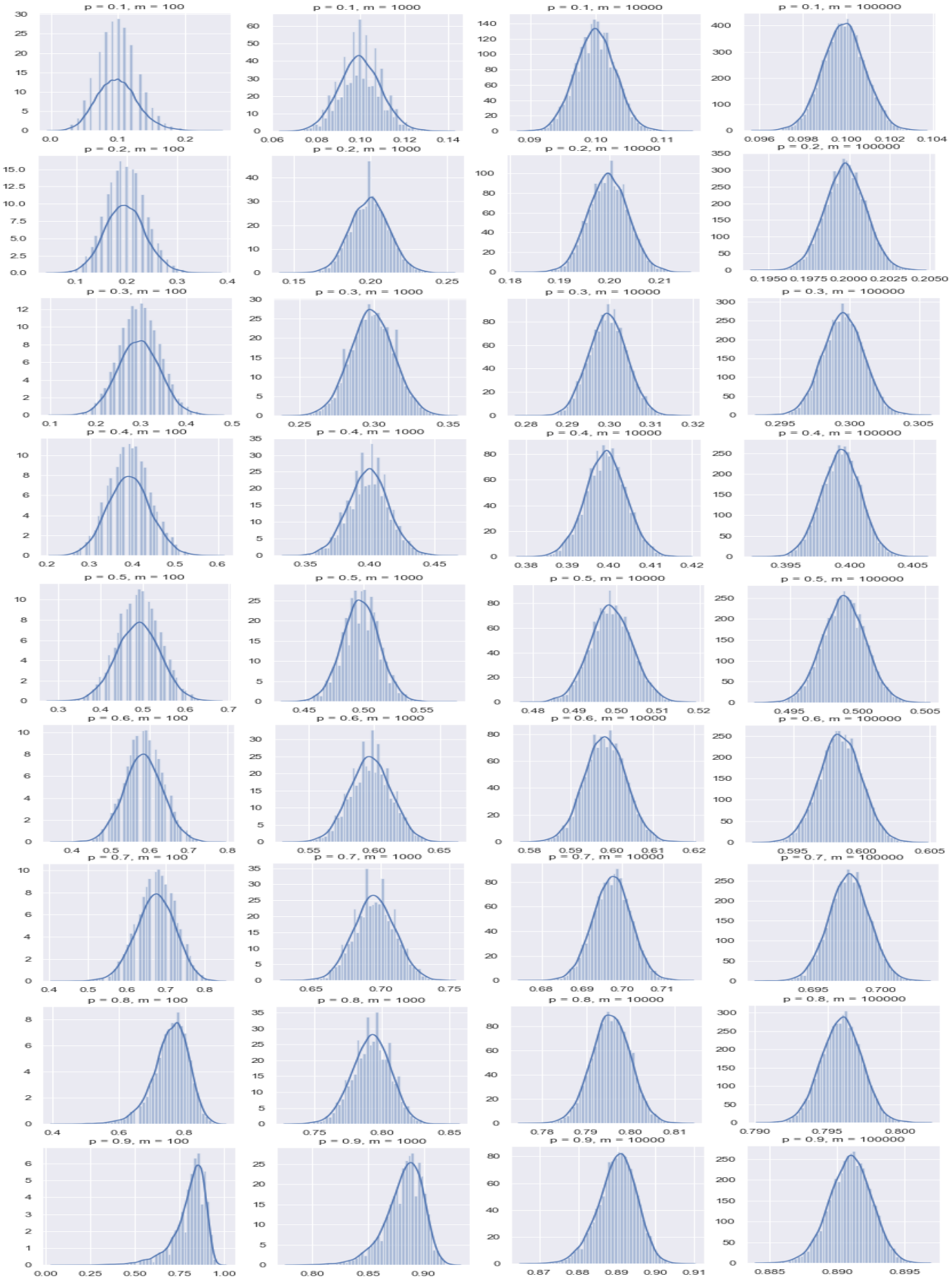
Tablica 3: Wartości pvalue dla testu Kołmogorowa-Smirnowa dla n=1000

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1000	0.000	0.000	0.020	0.001	0.016	0.016	0.007	0.000	0.000
10000	0.235	0.318	0.698	0.489	0.148	0.219	0.160	0.283	0.000
100000	0.649	0.679	0.917	0.390	0.734	0.867	0.784	0.641	0.040

Z tabeli powyżej wynika, że mogę założyć normalność rozkładu dla wszystkich p poza 0.9, więc dla tej wartości n teza problemowa nie może zostać potwierdzona. Dla formalności, poniżej wartości pvalue dla testu t-Studenta.

Tablica 4: Wartości pvalue dla testu t-Studenta dla n=1000

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
1000	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
10000	0.721	0.995	0.435	0.067	0.009	0.023	0.000	0.000	n/w
100000	0.377	0.240	0.179	0.458	0.743	0.050	0.161	0.000	n/w



Rysunek 3: Wyniki symulacji dla wariantu B2 oraz $n = 1000$, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja

Poniżej znajdują się symulacje dla parametru $n = 10000$.

Tablica 5: Wartości pvalue dla testu Kołmogorowa-Smirnowa dla n=10000

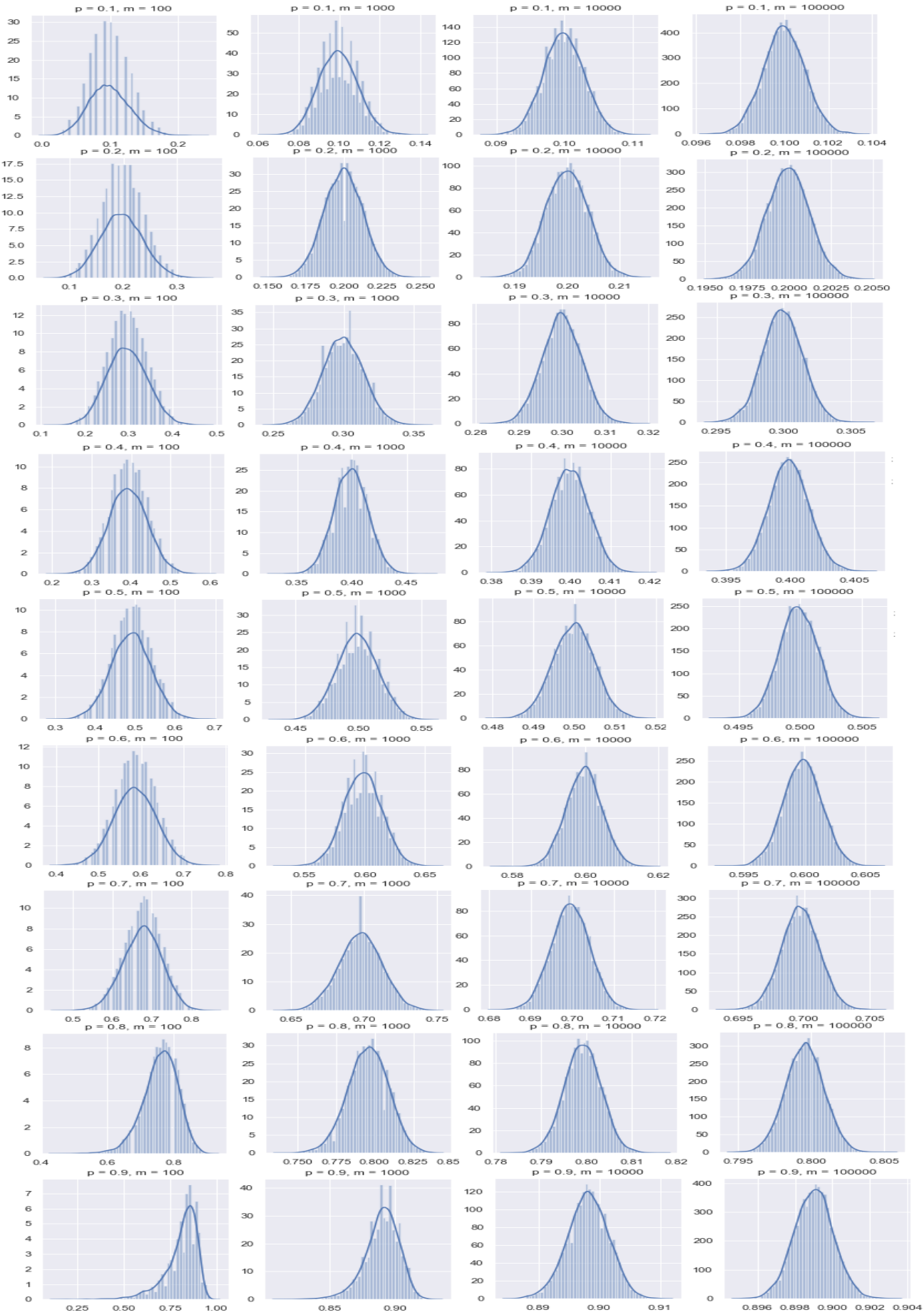
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1000	0.000	0.001	0.000	0.003	0.005	0.009	0.003	0.000	0.000
10000	0.054	0.179	0.301	0.340	0.083	0.228	0.490	0.500	0.022
100000	0.654	0.889	0.445	0.698	0.448	0.941	0.640	0.830	0.343

Z tabelki powyżej wynika, że dla większej wartości parametru n można założyć, że rozkład jest normalny.

Tablica 6: Wartości pvalue dla testu t-Studenta dla n=10000

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
1000	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
10000	0.025	0.229	0.689	0.023	0.075	0.014	0.001	0.000	n/w
100000	0.811	0.064	0.077	0.406	0.500	0.704	0.018	0.042	0.000

Hipoteza zerowa w teście t-Studenta może zostać przyjęta dla dużych parametrów m, ale jedynie dla p mniejszych bądź równych 0.6. Powoduje to, że dla $n = 10000$ również teza problemowa jest obalona.



Rysunek 4: Wyniki symulacji dla wariantu B2 oraz $n = 10000$, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja

Poniżej znajdują się symulacje dla parametru $n = 100000$.

Tablica 7: Wartości pvalue dla testu Kołmogorowa-Smirnowa dla n=100000

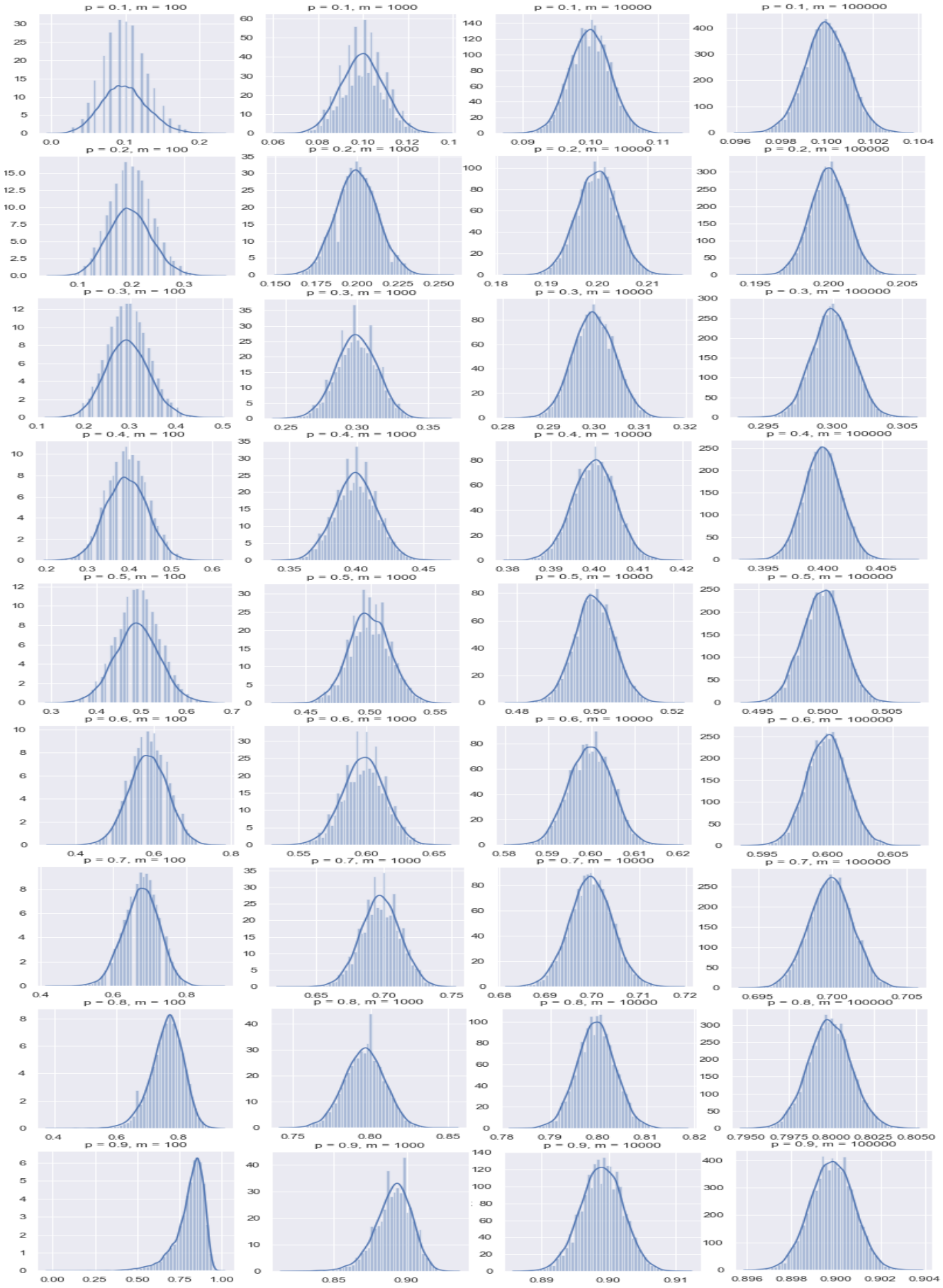
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1000	0.000	0.000	0.011	0.021	0.000	0.002	0.008	0.000	0.000
10000	0.211	0.136	0.082	0.344	0.222	0.128	0.392	0.291	0.052
100000	0.572	0.791	0.808	0.948	0.359	0.875	0.745	0.601	0.110

Podobnie jak w poprzednim przypadku, mogę założyć, że rozkład jest normalny.

Tablica 8: Wartości pvalue dla testu t-Studenta dla n=100000

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
1000	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w	n/w
10000	0.042	0.150	0.688	0.030	0.184	0.003	0.000	0.000	0.000
100000	0.390	0.777	0.800	0.422	0.426	0.069	0.000	0.000	0.000

Z tabelki powyżej wynika, że mogę przyjąć hipotezę zerową tylko dla p mniejszych bądź równych 0.6.



Rysunek 5: Wyniki symulacji dla wariantu B2 oraz $n = 100000$, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja

Dla wariantu B2 nie mogę jednoznacznie stwierdzić, że dla wszystkich wartości parametru p wartość średnia jest wartością oczekiwaną. Mogę jednak stwierdzić, że dla małych wartości p założenie problemowe jest prawdziwe. Bardziej szczegółowe wyniki symulacji otworzą się po kliknięciu w załącznik.

Wariant 3: Życie zakończy się po odwiedzeniu wszystkich n miejsc co najmniej raz

W tym wariancie zredukowałem parametr *powt* do 1000, a parametr n został zredukowany tak, jak opisałem to w etapie 1. Jest to spowodowane czasem wykonywania symulacji. Przewidywałem, że czas wykonywania symulacji będzie długi i moje przypuszczenia sprawdziły się, dlatego też zdecydowałem się na taki ruch.

Tablica 9: Wartości pvalue dla testu Kołmogorowa-Smirnowa

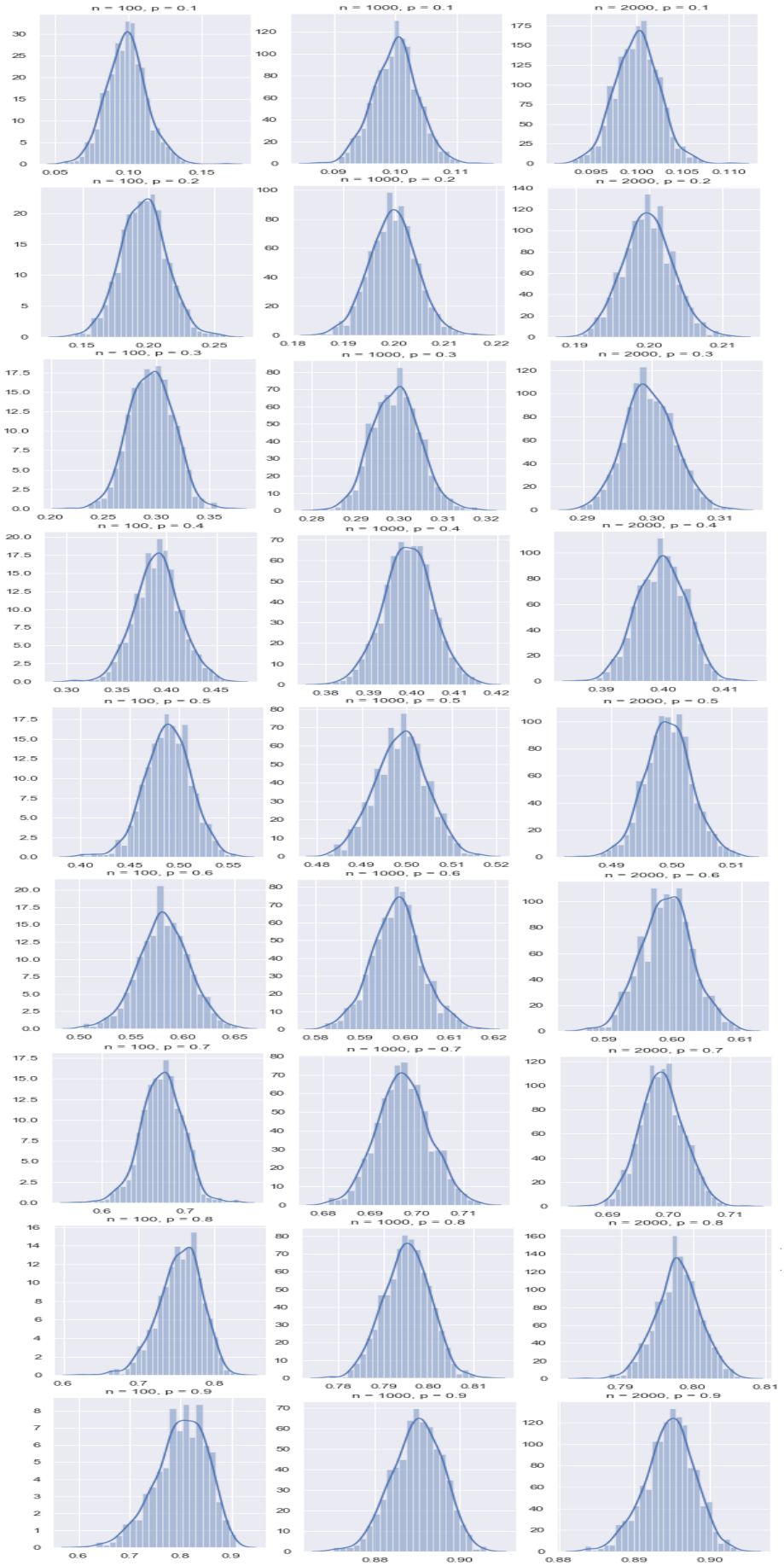
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.425	0.752	0.784	0.766	0.503	0.603	0.800	0.063	0.030
1000	0.388	0.955	0.913	0.929	0.659	0.552	0.948	0.946	0.432
2000	0.766	0.857	0.276	0.779	0.799	0.827	0.522	0.147	0.280

Z tabelki powyżej wynika, że mogę przyjąć, że dla wszystkich rozpartywanych wartości n rozkład jest normalny.

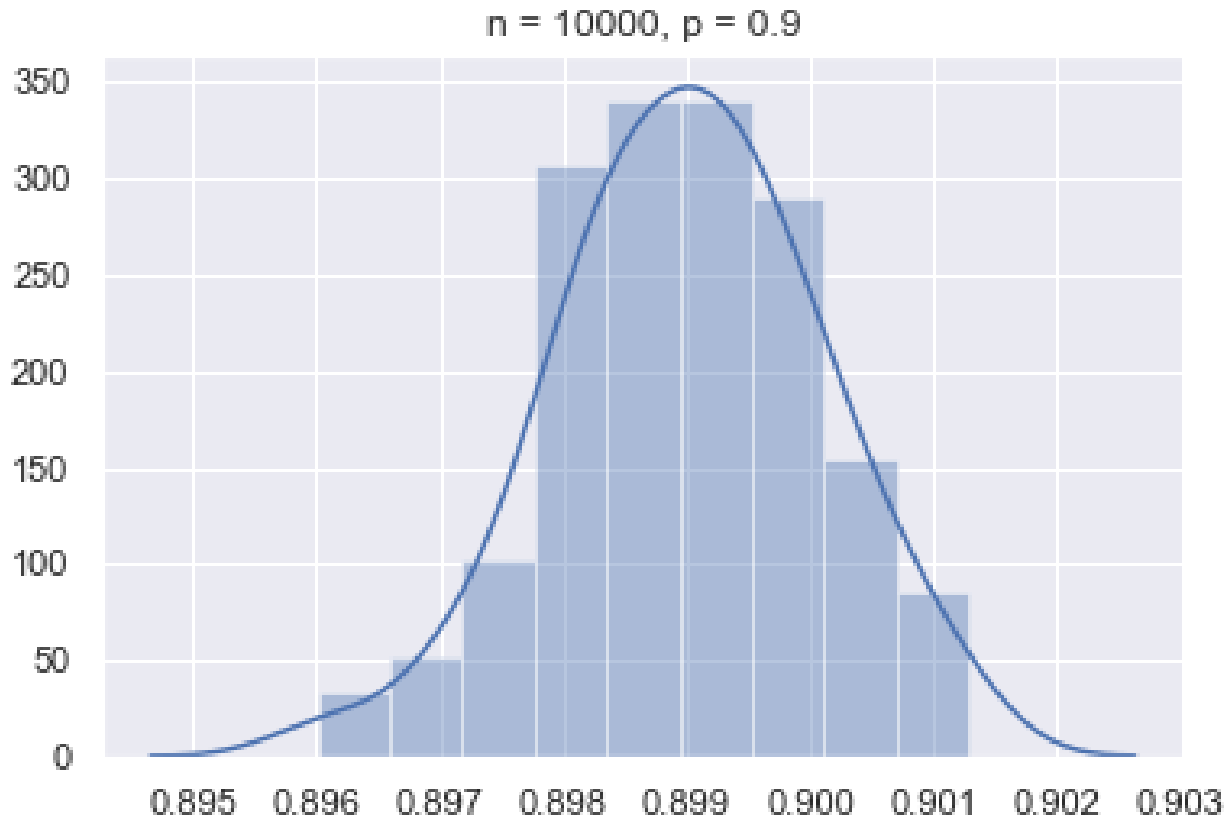
Tablica 10: Wartości pvalue dla testu t-Studenta

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
100	0.898	0.112	0.448	0.000	0.098	0.000	0.000	0.000	n/w
1000	0.128	0.892	0.127	0.628	0.088	0.829	0.002	0.000	0.000
2000	0.279	0.820	0.422	0.916	0.683	0.637	0.839	0.266	0.000

Z tabelki powyżej wynika, że dla wartości n większych od 1000 wartość średnia jest wartością oczekiwaną. Zgadza się to dla każdej wartości p oprócz 0.9. Dlatego też zdecydowałem się na wykonanie symulacji dla p=0.9 i n=10000. Próbowałem wykonać tę symulację dla parametru *powt*=1000, ale czas oczekiwania na wynik był zdecydowanie za długi (po paru godzinach zrezygnowałem z oczekiwania na wynik), dlatego obniżyłem wartość *powt* do 100. Wynik symulacji był następujący: pvalue dla testu normalności Kołmogorowa-Smirnowa wyniósł 0.997, więc ten rozkład jest normalny, a pvalue dla testu t-Studenta wyniosła 0.355, więc wartość oczekiwana to wartość średnia. Tym samym udowodniłem dla tego wariantu problem projektowy. Bardziej szczegółowe wyniki otworzą się po kliknięciu w załącznik.



Rysunek 6: Wyniki symulacji dla wariantu B3, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji, nad poszczególnymi wykresami są wypisane parametry dla których była wykonywana symulacja



Rysunek 7: Wynik dodatkowej symulacji dla wariantu B3, oś pozioma oznacza odsetek podróży, w których podróżnik zmókł a oś pionowa oznacza ilość powtórzeń tego odsetka w danej symulacji

3.2.3 Wnioski

Na początku tej sekcji chciałbym odnieść się do pewnych mankamentów rozpatrywania przeze mnie tego problemu. Nie został on potraktowany w sposób idealny, jednakże zadanie zostało wykonane. W idealnych warunkach, parametry n, m i *powt* powinny być znacznie większe, niż wartości przeze mnie sprawdzane, jednak nawet dla stosunkowo małych wartości teza problemowa została udowodniona. Utrudnienia wynikają z ograniczonej zdolności obliczeniowej mojego sprzętu, a co za tym idzie, brakiem czasu. Komputer podczas przeprowadzania symulacji był bardzo obciążony, a obecna sytuacja wymaga użycia komputera, chociażby do uczestnicwa w zajęciach.
Tak jak wcześniej wspomniałem, teza problemowa została udowodniona dla wariantu A oraz dla wariantu B3, co potwierdzają testy statystyczne oraz bardzo mała różnica pomiędzy wartością średnią a wartością oczekiwaną dla dużych wartości parametru n (ta różnica przyjmuje wartości rzędu 10^{-5} dla obu wariantów). Wariant B1 nie miał szans na powodzenie, nawet przy ogromnych wartościach parametrów. Moim zdaniem, wariant B2 ma szansę być prawidłowym przy użyciu większych wartości parametrów n oraz m.

4 Wykorzystane źródła

Inspirację czerpałem z następujących stron:

- stackoverflow, w szczególności sposób na zgrywanie większej ilości linijek tekstu do pliku widoczny tutaj,
- Quora,
- Wikipedia,
- Dokumentacje bibliotek scipy, matplotlib.pyplot, numpy oraz seaborn.