UNIWERSYTET
EKONOMICZNY
W POZNANIU

Wydział Informatyki
i Gospodarki Elektronicznej

Konrad Jadwiżyc

Porównanie wybranych języków programowania na przykładzie próbkowania z wielowymiarowego rozkładu normalnego

Comparison of selected programming languages on the example of sampling from multivariate normal distribution

Praca licencjacka

Promotor: dr Maciej Beręsewicz

Pracę przyjęto dnia:

Podpis promotora

Kierunek: Informatyka i ekonometria

Specjalność: Analityka Gospodarcza

Poznań 2019

Spis treści

Wstęp

W badaniach statystycznych oraz w przedsiębiorstwach wykorzystuje się na szeroką skalę różne modele regresji. Istotną rolę w tego rodzaju badaniach odgrywa język programowania, który zostanie wykorzystania do przeprowadzenia analizy. Przykładem skutków wynikających ze złego dobrania języka programowania są opóźnienie projektu, a także problemy integracji z infrastrukturą przedsiębiorstwa. W konsekwencji istnieje ryzyko poniesienia przez firmę strat finansowych. Dlatego dobranie odpowiedniego języka do analizy regresji jest równie ważne co czas wykonania analizy. Trafne prognozy analizy regresji pozwalają przyspieszyć pracę oraz podnieść konkurencyjność firmy na rynku.

Niniejsza praca ma na celu porównanie szybkości wybranych języków *open source* na przykładzie generowania zmiennych z wielowymiarowego rozkładu normalnego. Ocenie poddane zostaną trzy języki – R, Python oraz Julia, które są narzędziami wykorzystywanymi do obliczeń numerycznych czy szerzej do analizy danych.

Celem utylitarnym pracy jest wybranie optymalnego, pod względem szybkości, języka programowania przy wykorzystaniu generowania rozkładu wielowymiarowego normalnego oraz dekompozycji macierzy. Optymalny czas pracy języka można określić na podstawie szybkości przeprowadzonych dekompozycji macierzy kowariancji.

Praca składa się z 3 rozdziałów, z których pierwsze dwa są to rozdziały teoretyczne, natomiast trzeci ma charakter badawczy. Dopełnieniem struktury pracy jest wstęp oraz posumowanie.

W pierwszym rozdziale została omówiona istota symulacje w ekonomii, a także przedstawione zostały jej metody. Następnie opisano komputerowe pakiety statystyczne, począwszy od generowania liczb pseudolosowych poprzez generowanie z rozkładów prawdopodobieństw. Następnie opisano złożoność obliczeniową algorytmów oraz opisano wybrane języki programowania.

W drugim rozdziale szczegółowo omówione zostały własności rozkładu normalnego oraz

wielowymiarowego rozkładu normalnego. Ponadto przedstawiono metody generowania wielowymiarowego rozkładu normalnego.

W ramach trzeciego rozdziału przeprowadzono badanie symulacyjne, w celu wyboru najlepszego języka programowania wykorzystywanego do regresji. Badanie polegało na stworzeniu macierzy Σ kowariancji przy pomocy rozkładu normalnego oraz macierzy X z rozkładu normalnego. Następnie napisano funkcję dekompozycji składającą się z trzech metod dekompozycji Choleskiego, Według wektorów własnych (ang. Singular Value Decoposition; SVD) oraz Spektralnej dzięki którym można generować wielowymiarowy rozkład normalny. Na podstawie napisanej funkcji dokonano 100 losowań dla wybranych wymiarów macierzy Σ oraz X wymiary zwiększano co 100 zaczynając od 200 kończąc na 1000. Pozwoliło to uzyskać zbiór 900 pomiarów czasowych dla jednej metody dekompozycji. Pozwoliło to uzyskać łączny zbiór 8100 pomiarów czasowych. Na końcu przedstawione zostały wyniki przeprowadzonej analizy.

Przy doborze literatury ograniczono się do najważniejszych pozycji, w których szczegółowo opisano symulacje oraz generowanie wielowymiarowego rozkładu normalnego. Wszelkie obliczenia zostały przeprowadzone w języku R, Python, Julia natomiast analiza wyników została przeprowadzona w języku R w oparciu o autorski kod.

Rozdział 1

Symulacje w ekonomii

1.1 Pojęcie symulacji w ekonomii

Według definicji symulacji w Encyklopedii powszechnej PWN – Symulacja „jest to sztuczne odtwarzanie właściwości danego zjawiska lub obiektów występującego w naturze, lecz trudnych do obserwowania, zbadania, powtórzenia. Symulacja jest techniką służącą do imitowania działania całego systemu bądź naśladowania jego konkretnych informacji" (PWN, 1999). W odniesieniu do definicji „System jest to zestaw wzajemnie powiązanych ze sobą elementów, funkcjonujących jako jedna całość np system operacyjny".

W matematyce symulacja może być przedstawiona przez modele matematyczne. Modele te opisuje się za pomocą równań lub w postaci układów równań które potrafią scharakteryzować dane zjawisko lub proces i na drodze obliczeń da się je przewidzieć. Takie podejście wykorzystuje się biznesie oraz projektowaniu systemów informatycznych (Maciąg, Pietroń & Kukla, 2013).

Natomiast w literaturze odnoszącej się do symulacji i zarządzania możemy znaleźć inne definicje w których mówimy, że „Symulacje są to metody odtwarzania istoty systemu bez konieczności rzeczywistego uruchamiania modelu". Dlatego wykorzystujemy je do modelowania i przedstawiania zachowania w czasie badanych charakterystyk systemu lub procesu. Taki model możemy dowolnie modyfikować w sposób przy jakim realny system byłby niepraktyczny lub byłoby to nieosiągalne dla systemu. Zakładamy, że dana symulacja pozwoli nam na zbadanie i wyciągnięcie wniosków na temat dynamiki, zachowań badanego modelu oraz do weryfikacji owego modelu (Gajda, 2017).

Modele symulacyjne pozwalają na ocenę oraz podjęcie odpowiedniej decyzji w okoliczno-

ściach, gdy możliwa jest kategoryzacja relacji w modelu ze względu na losowy charakter zmiennych czy np. zmianie w czasie zależności między zmiennymi. Mechanizm podejmowania decyzji można łatwo zaimplementować do ciągle zmieniających się warunków i otoczenia (Witkowski, 2000).

W modelowaniu powinniśmy zwrócić uwagę na zmienne decyzyjne, które bierzemy jako przyczyny (objaśniające dane zjawisko) i skutek działania owych przyczyn. Zazwyczaj wystarczy wyznaczyć postać funkcyjną określoną wzorem 1.1

$$y_t = f(x_{1t}, x_{2t}, ..., x_{kt}). \tag{1.1}$$

Równanie to pomocne jest przy rozpoznawaniu kreowania się wyniku y_t pod wpływem czynników x_{it} dla $i = 1, ..., k$, $t = 1, ..., T$. Żeby dopatrzyć się reakcji musimy wybrać próbę losową na podstawie której będziemy mogli wyznaczyć odpowiedni model. Jeśli w modelu występują zmienne losowe lub nielosowe możemy mieć do czynienia z modelem stochastycznym lub deterministycznym (Gajda, 2001).

Symulacje zwykle wykorzystuje się wtedy gdy rozwiązanie analityczne nie istnieje z powodu skomplikowanych równań różniczkowo-różnicowych lub nie istnieje owe rozwiązanie oraz jeśli rozwiązanie pociąga za sobą duże koszty i nakłady pracy (Welfe, 2009).

W rzeczywistości modelowanie systemów niesie ze sobą szereg komplikacji takich jak występowanie funkcji nieliniowych składających się z skomplikowanej sieci wielu powiązań, powstają też losowe zakłócenia. Losowo zmienne bodźce oraz losowe elementy modelu mogą realizować się z pewnym prawdopodobieństwem jak i uwzględnienia źródeł niepewności w zachowaniu systemu (Gajda, 2017).

W symulacyjnym podejściu do problemu wyróżnia się 3 typy symulacji:

- Symulacje Dyskretne
- Symulacje Ciągłe
- Symulacje Monte Carlo

1.1.1 Symulacje dyskretne

Symulacje dyskretnym nazywać można proces oparty na zdarzeniach dyskretnych, czyli zdarzeniach które powstają w pewnych dokładnie określonych momentach czasowych. Zdarzenia

Tabela 1.1. Zalety i wady symulacji

Zalety	Wady
Uniwersalność zastosowania dla wielu dziedzin oraz łatwe wprowadzanie odpowiednich parametrów	Brak reguł tworzenia modeli dziedzinowych
Możliwość eliminacji w badaniu wpływu czasu (zjawiska mogą być spowalniane lub przyspieszane)	Brak uniwersalnych metod rozstrzygania o poprawności budowanych modeli
Powtarzalność symulacji przy tych samych warunkach	Brak możliwości automatyzacji procedury budowy modeli
Możliwość badania systemu dla warunków nie możliwych do osiągnięcia w rzeczywistości	Czasochłonność i duże koszty budowy modeli
Przeprowadzenie badania nie niszcząc systemu	Duża wrażliwość wyników na błąd ludzki (złe określenie celu modelu), brak wystarczającego dowodu na potwierdzenie wiarygodności
Możliwość realizacji badania systemu bez potrzeby budowania jego prototypu	Błędne wnioskowanie

Źródło: Opracowanie własne.

wywołują zmianę stanów procesu i systemu w momencie ich wystąpienia, zachodzące zmiany odbywają się więc w sposób nieciągły - dyskretny.

Procesy te często wykorzystywane są w badaniach operacyjnych lecz duże zróżnicowanie modeli powoduje że przedstawienie problemów wymaga dużego poziomu abstrakcji. Do przedstawienia upływu czasu wykorzystuje się najczęściej koncepcje następstw zdarzeń (Maciąg i in., 2013). Wyróżniamy cztery podejścia (Gajda, 2017).

Pierwsze z nich to metoda planowania zdarzeń. W metodzie tej każdemu etapowi dokonywanej działalności przyporządkowujemy następstwo, które jest skutkiem realizacji zjawiska w dotychczasowych etapach (Janiga-Ćmiel, 2016). Możemy również poddawać szczegółowej analizie kroki jakie trzeba podjąć, żeby kontynuować proces zjawiska.

Drugim podejściem jest metoda przeglądu i wybieranie zdarzeń które po zakończeniu jednego procesu powinny być w dalszej części realizowane. Najczęstszą metodą podziału dalszej realizacji jest podział na realizacje optymistyczne, przeciętne i pesymistyczne w stanach optymistycznych bierzemy pod uwagę stany korzystniejsze od oczekiwanych i na odwrót w przypadku stanów pesymistycznych.

Trzecie podejście to symulacje interakcji procesów i jest połączeniem dwóch wyżej wymienionych metod. Rozpatrywane działania Polegają grupowaniu w zależności kiedy się pojawiają

i kiedy znikają.

Czwartym podejściem jest to metoda analizy zdarzeń dotyczy ona wymienionych procesach ponieważ łączy w sobie planowanie i interakcji procesów. Celem tej metody jest pozyskanie informacji, które w każdej sytuacji, w dowolnym momencie pozwolą ocenić słuszność podjętej decyzji, jednocześnie dokonując oceny skutków ex post i ex ante (Janiga-Ćmiel, 2016).

Model symulacyjny można przedstawić za pomocą układów równań opisany wzorem (1.2)

$$\begin{cases} z_1 = f(x_l, ..., x_k, z_2, ..., z_m), \\ z_2 = f(x_l, ..., x_k, z_l, z_3, ..., z_m), \\ \vdots \\ z_m = f(x_l, ..., x_k, z_l, ..., z_{m-1}). \end{cases} \tag{1.2}$$

gdzie z jest to badane zjawisko x jest to czynnik kształtujący badane zjawisko natomiast $z_1, ..., z_m$ ją to czynniki na które wpływ ma czynnik x. zjawisko badane z zapisuje się za pomocą funkcji $z = f(x_1, ..., x_n, z_1, ..., z_m)$

Symulacje dyskretne mają szerokie zastosowanie w ekonomii i zarządzaniu na różnych poziomach produkcyjnym, logistycznym, transportowym oraz obsłudze klientów. Metody te można wykorzystać w rozwiązywaniu problemów planowania i harmonogramowania produkcji, ocenie wykorzystanych zasobów oraz identyfikacji ograniczeń produkcji.

W badaniu i analizie przedsiębiorstwa metody symulacyjne znajdują zastosowanie w:

- analizie łańcucha dostaw,
- ocenie oraz przewidywaniu skutków działania systemów, które są oparte na kluczowych miernikach (np. koszt, przepustowość, wykorzystanie zasobów),
- planowaniu kadr,
- planowaniu wyposażenia technicznego oraz potrzebnych materiałów.

Maciąg i in. (2013)

1.1.2 Symulacje ciągłe

W symulacji ciągłej czas ma postać ciągłą czyli może wykonać się w dowolnym momencie czasu a nie w ściśle określonym czasie jak zmienna dyskretna. W symulacji ciągłej model systemu

dynamicznego ma postać równania różniczkowego może mieć postać układu równań różniczkowych lub różnicowych są to równania stanu. Stanu w którym znajduje się system.

Model układu równań stanu kreowany jest na podstawie wyszczególnionych cech systemu. Jeżeli n cechom danego systemu, traktowanym jako istotne w kontekście modelu, przyporządkowano ich wartości, określa to chwilowy stan systemu który jest opisany wektorem(1.3):

$$z = [x_1, x_2, ..., x_n], \tag{1.3}$$

gdzie z jest cechą opisaną przez zmienne $x_1, x_2 ...., x_n$. Wektor ten oznacza zmianę cech w czasie.

Jeśli cechy systemu są zmienne w czasie, to mówi się zmianie stanu systemu. Stan systemu opisywany jest wektorem zmiennych funkcji czasu - zmiennymi stanu(1.4):

$$z(t) = [x_1(t), x_2(t), ..., x_n(t)] \tag{1.4}$$

Gdzie $z(t)$ jest to funkcja czasu a zmienne $x_1(t), ..., x_(t)$ opisujące zmiany stanu.

Zmienne wyznaczają przestrzeń, w której każdy z punktów określa chwilowy stan systemu. Miejsce to jest nazywane przestrzenią stanów systemu. Zmiany wartości zmiennych opisujących stan można zapisać za pomocą funkcji pochodnych tych zmiennych tzn(1.5):

$$\dot{x}_1(t), \dot{x}_2(t), ..., \dot{x}_n(t). \tag{1.5}$$

Pochodne te wyznaczają kierunek z jaką zmienia się ruchu punktów systemu w przestrzeni stanu. Znając stan początkowy systemu w chwili t_0 (najczęściej t_0 = 0), tj. wartości: $x_1(t_0), x_2(t_0), ..., x_n(t_0)$ oraz relacje pomiędzy zmiennymi i ich funkcjami pochodnymi, można wyznaczyć zmiany systemu w przestrzeni stanu (Maciąg i in., 2013).

1.1.3 Symulacje Monte Carlo

Monte Carlo jest to technika rozwiązywania modelu stochastycznego wielkości losowych z wybranego rozkładu prawdopodobieństwa. Podstawy tej metody zastosował 1733r Buffon francuski filozof oraz matematyk który próbował oszacować liczbę π po przesz rzucanie igły o długości l na drewnianą podłogę o szerokości deski $2l$ i obliczeniu ilości rzutów do ilości rzutów w których igła spadła w poprzek szczeliny między deskami.

Natomiast metodę tę opracował 1949 matematyk Stanisław Ulam i pierwszy raz zastosował podczas tworzenia bomby termojądrowej. Metoda ta jest nazywana symulacją stochastyczną.

Mówimy o takim podejściu w odniesieniu do metody próbkowania z modelu opierającej się na badaniu własności modelu metodą Monte Carlo polega to na tym, że w wybrane fragmenty modelu które stanowią źródła niepewności są wstawiane zaburzenia wylosowane z odpowiednich rozkładów prawdopodobieństwa następnie są wyznaczane rozwiązania modelu. Rozwiązania te nazywamy replikacją (Gajda, 2017).

Metoda Monte Carlo posiada dwie właściwości. Pierwszą właściwością jest prosta struktura algorytmu obliczeń. Zazwyczaj tworzy się program jednego zdarzenia losowego. Później należy wybrać losowy punkt w kwadracie i zobaczyć czy zdarzenie należy do obrębu kwadratu czy nie. Następnie zdarzenie powtarza się N-krotnie a z wyniki wszystkich doświadczeń uśrednia się.

Drugą właściwością Monte Carlo jest zbieżność obliczeń do reguły $\sqrt{D/N}$ gdzie D jest pewną stałą, a N jest liczbą prób. W celu 10-krotnego zmniejszenia błędu, trzeba zwiększyć N 100 razy (Sobol, 2017).

Metoda ta stosowana jest do modelowania zbyt złożonych matematycznych procesów, aby można było oszacować ich wynik za pomocą podejścia analitycznego. Na przykład komputerowa symulacja przybliżania liczby π oraz wiele innych zastosowań w biznesie. Jednym z wielu zastosowań Monte Carlo może być wycena opcji, całkowanie, rozwiązywanie układów równań liniowych, nie liniowych, różniczkowych oraz wszędzie tam gdzie nie można wykorzystać podejścia analitycznego (Ziętek-Kwaśniewska, 2006).

1.1.4 Zastosowanie symulacji w ekonomii

Kreowanie odpowiednich postaw przedsiębiorczych jest procesem skomplikowanym. Zawiera on wykorzystanie niezbędnej wiedzy oraz nabycia odpowiednich postaw w prowadzeniu biznesu. Droga przedsiębiorcy wymaga ciągłego radzenia sobie z problemami (Bizon & Poszewicki, 2013).

Dlatego w ekonomii metody przewidywania znajdują zastosowanie zarówno na poziomie makro w skali gospodarki oraz mikro skali przedsiębiorstw. Takie podejście które oparte jest na prognozach wykorzystywane jest w wielu obszarach ekonomii i działalności człowieka (Maciąg i in., 2013). W przedsiębiorstwo kierownictwo musi zmagać się z wieloma trudnymi decyzjami które obarczone jest dużym ryzykiem. Podejmowane działania muszą być poparte solidnymi informacjami. Wyróżniamy prognozowanie wielkości sprzedaży, prognozy gospodarcze, prognozy społeczne oraz w badaniach marketingowych i strategii przedsiębiorstw (Maciąg i in., 2013). Przykładami takich symulacji może być ewaluacja modeli prognostycznych, Analiza wraż-

liwości modeli na zmianę założeń. W Przypadku prognoz gospodarczych na poziomie państwa może być szacowanie zapotrzebowania na energię elektryczną lub w górnictwie szacowanie zapotrzebowania na surowce.

1.2 Komputerowe pakiety statystyczne i zastosowanie w symulacji

1.2.1 Generatory liczb pseudolosowych

Otaczający nas świat pełny jest zdarzeń losowych możemy je zaobserwować w każdym aspekcie życia oraz w ekonomii ale również w matematyce przykładem losowości może być wstępowanie liczb pierwszych choć matematycy na całym świecie próbują wyjaśnić i przewidzieć powstawanie owych liczb. W swojej pracy dyplomowej chciałbym przybliżyć istotę generowania liczb losowych oraz cele ich wykorzystania w ekonomii.

Liczby losowe można wykorzystać w reprezentatywnych badaniach statystycznych (przy losowaniu próby z populacji generalnej lub, szerzej, planowaniu schematu losowania), a zatem w zagadnieniach statystycznej kontroli jakości produktów, wszelkich badaniach ekonomicznych, społecznych, marketingowych (Kotulski, 2001).

Innym zastosowaniem generowania liczb losowych są gry komputerowe w których wykorzystuje się liczby losowe do uzyskania realizmu procesów graficznych, gry zręcznościowe oraz grach strategicznych ekonomicznych lub społecznych.

Programy komputerowe służące do symulacji stochastycznych posiadają wiele generatorów wyróżniamy:

Generator fizyczny – proces ten polega na wytwarzaniu liczb pseudolosowych z mierzalnych parametrów procesu fizycznego przebiegającego w sposób losowy (Kotulski, 2001). Do takich generatorów należą:

- rzut monety,
- urny z których losujemy kule,
- kostki do gry,
- licznik impulsów elektrycznych z procesorów kart graficznych oraz innych podzespołów komputera,

- systemy pobierania losowych bitów z algorytmów komputera,
- specjalne urządzenia do generacji liczb pseudolosowych.

Generatory komputerowe – metoda ta została zapoczątkowane przez Johna von Neumanna. Naukowiec zaproponował algorytm w którym wybraną liczbę n-cyfrową (przy parzystym n) tak zwane ziarno. Jest to początek algorytmu, podnosimy liczbę tą do kwadratu po czym z otrzymanej liczby $2n$-cyfrowej wybieramy środek n cyfr z których tworzona jest pierwsza liczba pseudolosowa po czym zostaje on podniesiona do kwadratu i z nowo pozostałej liczby wybieramy kolejną n cyfrę i powstaje druga liczba pseudolosowa itd. Lecz algorytm ten nie stał się zbyt użyteczny lecz stanowiło podstawy do dalszego rozwoju generatorów. Najczęściej stosowane dziś generatory to generatory liniowe(1.6)

$$X_i = (aX_{i-1} + b) mod c, \tag{1.6}$$

gdzie a,b,c – są to parametry, X_{i-1} poprzednia wystąpienie zmiennej pseudo losowej, $a\ mod\ c$ oznacza całkowitoliczbową resztę z dzielenia przez c. Generatory liniowe tworzą okresowe ciągi liczb.

W praktyce z takich generatorów korzystamy w programach takich jak Excel oraz językach programowania Pascal, Fortran, C++.

Generowane liczby za pomocą komputera są liczbami pseudolosowymi. Wynika to z faktu iż, liczby te mają emulować losowość przez co nadają one wynikom wiarygodności. Możemy się spotkać z wykorzystaniem liczb pseudolosowych do weryfikacji własności narzędzi statystycznych, metod Monte Carlo, kryptografii, jak i grach komputerowych.Służą też jako istotny element w podejmowaniu decyzji strategicznych które muszą być poparte symulacjami (Gągolewski, 2014).

Jednym z problemów jakie można napotkać podczas generowania liczb pseudolosowych jest to, że komputer jest narzędziem o zaprogramowanym przewidywalnym zachowaniu (Gągolewski, 2014). Wszystkie procesy są określone. Jeśli wprowadzimy takie same dane na wejściu możemy uzyskać ten sam wynik. Stąd nazwa liczb pseudolosowych ponieważ generowanie ich za pomocą komputera imituje losowość (Gągolewski, 2014).

1.2.2 Rozkłady ciągłe i ich generatory

Podstawowe typy generatorów liczb losowych wytwarzają liczby losowe podlegające rozkładowi jednostajnemu.Przedstawiono jakie Rozkłady ciągłe wchodzą w skład ty generatorów.

1.2.2.1 Rozkład normalny

Rozkład ten jest najczęściej występującym rozkładem. Rozkład ten szczegółowo został opisany w rozdziale drugim.

1.2.2.2 Rozkład logarytmiczno-normalny

Zmienna ma rozkład logarytmiczno-normalny, jeśli jej logarytm ma rozkład normalny. Jest to sytuacja szczególnie często spotykana w badaniach ekonomicznych, w których występują zmienne o wartościach dodatnich rozłożonych asymetrycznie (Gajda, 2017). Rozkład ten przydatny jest szczególnie przy tym żeby powiązać asymetryczne rozkłady zmiennych w rozkładach normalnych oraz rozkładzie Pareto ponieważ rozkład ten odpowiednia obserwacją empirycznych w tzw ”prawym ogonie”podczas gdy rozkład logarytmiczno-normalny skupia się w okolicach zera. Gęstość prawdopodobieństwa dana jest wzorem(1.7)

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(\ln x - \mu)^2}{2\sigma^2}\right) 1_{(0,\infty)} \tag{1.7}$$

gdzie π - liczba Pi, $\ln x$ - logarytm naturalny z x, μ to wartość średnia, σ to odchylenie standardowe, σ^2 oznacza wariancję.

1.2.2.3 Rozkład χ^2

Sumując k zmiennych normalnych podniesionych do kwadratu, otrzymujemy zmienną o rozkładzie χ^2 z k stopniami swobody (oznaczane $\chi^2(k)$ lub $\chi(k)$) (Gajda, 2017). O zastosowaniu tego generatora w symulacjach decyduje zmienność jego kształtu czyli dla k bliskiego 1 rozkład jest bliski symetrycznemu dla $k = 20 - 30$ rozkład jest bliski normalnemu. Funkcja gęstości prawdopodobieństwa dana jest wzorem(1.8)

$$f(x) = \frac{(1/2)^{k/2}}{\Gamma(k/2)} x^{k/2-1} e^{-x/2} \tag{1.8}$$

k to stała określająca liczbę stopni swobody, e oznacza stałą Eulera, Γ to funkcja Gamma.

1.2.2.4 Rozkład t-Studenta

Jest to rozkład często stosowany w procedurach testowania hipotez statystycznych oraz do oceny niepewności pomiaru. Zmienną o rozkładzie t-studenta uzyskujemy przez podział standardowej zmiennej normalnej przez zmienną o rozkładzie χ^2. funkcja gęstości prawdopodobieństwa dana jest(1.9)

$$f(x) = \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})\sqrt{n\pi}}(1+\frac{x^2}{n})^{-\frac{n+1}{2}}, \tag{1.9}$$

gdzie Γ - funkcja Gamma, $n = 1,2,..,n$, π - liczba Pi

1.2.2.5 Rozkład Pareto

Zastosowanie tego rozkładu służy do opisu lokacji dóbr w społeczeństwie do rozkładu miast, używany jest również w geofizyce.

Generator Par(a, c) generuje zmienną o rozkładzie Pareto, na odcinku (a, +∞), c > 0 – dominanta, funkcja gęstości $f(x) = c(a^c/x^{c+1})$ różna od zera i malejąca w przedziale (a, ∞), dystrybuanta $F(x) = 1(a/x)^c$, odwrotność dystrybuanty $F^1(y) = a(1y)^{1/c}$, średnia rozkładu $ca/(c1)$ istnieje dla $c > 1$, wariancja $ca^2/[(c1)^2(c2)]$ istnieje dla c. Funkcja gęstości prawdopodobieństwa dana jest wzorem(1.10)

$$f(x) = \frac{kx_m}{x^{k+1}}, \tag{1.10}$$

gdzie k to liczba stopni swobody

1.2.2.6 Rozkład wykładniczy

Rozkład ten podobny jest do rozkładu Pareto oraz jest obcięty z lewej strony posiada jednak skończoną średnią oraz wariancje. jeżeli w jednostce czasu ma zajść λ niezależnych zdarzeń, to rozkład wykładniczy opisuje odstępy czasu pomiędzy kolejnymi zdarzeniami.d wykładniczy może skutecznie przybliżać rozkłady czasu przetrwania np. samochodu lub przeżycia jednostki biologicznej. Funkcja gęstości prawdopodobieństwa(1.11)

$$f(x) = \lambda e^{-\lambda x} \tag{1.11}$$

gdzie λ - parametr funkcji , e - stała Eulera.

1.2.3 Złożoność obliczeniowa

Notacja dużego O(omikron) została pierwszy raz zaproponowana przez niemieckiego matematyka Paula Bechmanna w roku 1894. Notacja ta została spopularyzowana później przez Edmunda Landau niemieckiego matematyka przez co notacja ta nazywana jest też notacją Landau.

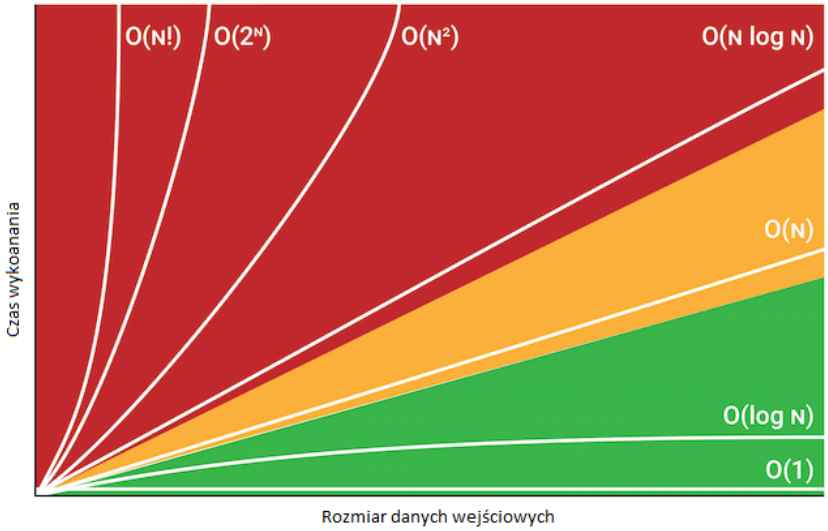
Badanie wydajności służy do określenia czasu pracy i stopnia skomplikowania danego algorytmu. Przykładem może być Asymptotyczne tempo wzrostu jest to pokazanie jak zachowują się wartości funkcji wraz ze wzrostem argumentów. Metoda ta jest stosowana do badania złożoności obliczeniowej. W zależności od danych wejściowych oraz dostępnych zasobów takich jak czas lub pamięć procesora algorytm asymptotycznego wzrostu pokaże nam jak szybko nada funkcja rośnie lub maleje (Cormen, Leiserson, Rivest & Stein, 2012). Przyjęte zostało, że największą złożonością jaka jest akceptowana to złożoność wielowymiarowa (Cormen i in., 2012).

Przykładowe rzędy złożoności:

- O(1) – złożoność stała
- O($\log n$) – złożoność logarytmiczna
- O(n) – złożoność liniowa
- O($nlogn$) – złożoność liniowo-logarytmiczna
- O(n^2)– złożoność kwadratowa
- O(n^k) – złożoność wielomianowa, gdzie k jest stałą
- O(k^n) – złożoność wykładnicza, gdzie k jest stałą
- O($n!$) – złożoność rzędu silnia

Kolor zielony oznacza że algorytm wraz ze wzrostem rozmiaru danych będzie niski czas wykonania. Kolor żółty oznacza, że wraz ze wzrostem danych czas wzrasta liniowo. Kolor czerwony oznacza, że algorytm jest nieefektywny ponieważ wraz ze wzrostem danych czas wzrasta bardzo szybko(wykładniczo). Dla przykładu:

- algorytm $O(NlogN)$
  - 1 element, 1 sek,



Rysunek 1.1. Notacja duże O

  - 10 elementów, 2 sek
  - 100 elementów, 3 sek

- Algorytm $O(n)$
  - 1 element, 1 sek
  - 10 elementów, 10 sek
  - 100 elementów, 100 sek

- algorytm $O(n^2)$
  - 1 element, 1 sek
  - 10 elementów, 100 sek
  - 100 elementów, 10,000 sek

1.3 Wybrane języki programowania

1.3.1 Język R

Język R powstał w 1997 na Uniwersytecie w Auckland w Nowej Zelandii. Pierwsza wersja 1.0 została wydana w 2000r a w roku 2013 została wydana wersja 3.0. Jądro R składa się z implementacji napisanej w C i Fortran i jest to implementacja wciąż rozwijanego języka S. R jest

to otwarte i darmowe środowiskiem oraz językiem służącym do przeprowadzania odtwarzalnych obliczeń statystycznych i numerycznych, analizy danych oraz tworzenia wysokiej jakości grafiki. licencja GNU GPL (ang. General Public License) pozwala na komercyjne wykorzystanie R. Na świecie jest wiele języków programowania i żaden nie jest idealny ponieważ każdy język jest dobry w określonych zastosowaniach. Patrząc na wykorzystanie i ukierunkowanie na obliczenia naukowe podobnym do R jest język użyty w pakiecie Matlab oraz Python z pakietem SciPy/NumPy. Natomiast składnia R przypomina C i C++ ale semantyka zbliżona jest do funkcyjnego Scheme(Gągolewski, 2014).

Cechy języka R (Gągolewski, 2014):

- R jest językiem ogólnego zastosowania dlatego można w nim zaimplementować praktyczne każdy algorytm co odróżnia go od języków szczegółowego zastosowania jak np język SQL
- Skupia się na pisaniu raczej małych programów oraz samych obliczeniach ze względu na określony rodzaj użytkowników którymi są ściśle określeni specjaliści.
- Jest językiem interpretowalnym co pozwala pracować w sposób interaktywny czyli prawie natychmiast otrzymujemy wynik obliczeń. Można się odwoływać do gotowych składni z zewnętrznych bibliotek oraz łatwo tworzyć swoje biblioteki.
- R posiada zwięzłą składnie czyli mała ilości kodu daje dość duży i złożony efekt.
- Posiada dość mocno rozbudowane możliwości tworzenia wysokiej jakości grafiki lub przedstawić wyniki za pomocą aplikacji webowej korzystając z pakietu Shiny.
- Ma dużą i łatwo dostępną dokumentację.

Podczas badania do budowy funkcji generującej wielowymiarowy rozkład normalny wykorzystano następujące pakiety:

- `dplyr` jest to kompletne narzędzie do szybkiej pracy na ramkach danych oraz obiektów (Wickham, François, Henry & Müller, 2018),
- `ggplot2` pakiet ten służy do graficznego przedstawienia danych (Wickham, 2016).

1.3.2 Język Python

Python jest to język programowania wysokiego poziomu ogólnego przeznaczenia. Ideą przewodnią jest czytelność i klarowność kodu źródłowego. Pierwsza wersja 1.0 została wydana w 1991 roku przez Guido van Rossum. Natomiast najnowsza wersja 3.7.3 została wydana 25.03.2019 r. Język ten jest rozwijany jako projekt Open Source. Python wspiera programowanie obiektowe, imperatywny oraz funkcyjny. Język ten można wykorzystać do obliczeń naukowych za pomocą pakiety SciPy oraz NumPy oraz jest wykorzystywany do analizy danych i szeroko pojętego Data Science.

Cechy języka Python:

- Jest to język skryptowy
- Struktury danych są wysokiego poziomu oraz dynamiczny system typów zwiększa wydajność oraz poprawia produktywność programistów.
- interpreter dostępny jest na wszystkich głównych platformach DOS, Windows, LINUX/Unix, Mac OS
- Posiada bogatą bibliotekę pakietów startowych
- Łatwość integracji programów napisanych w Pythonie z innymi częściami aplikacji napisanych w innych językach.

Podczas badania do budowy funkcji generującej wielowymiarowy rozkład normalny wykorzystano następujące pakiety:

- `SciPy` – pakiet ten jest zbiorem algorytmów matematycznych oraz funkcji zbudowanych na pakiecie Numpy. Dodaje znaczących przyspieszeń interaktywnej sesji pythona, udostępnia polecenia wysokiego poziomu do manipulowania oraz wizualizacji danych (Jones, Oliphant & Peterson, 2001)
- `Numpy` – pakiet umożliwiający zaawansowane obliczenia matematyczne oraz obliczenia numeryczne (mnożenie, dodawanie, diagonalizacje oraz odwracanie o całkowanie, itd). Daje to możliwości tworzenia specjalnych typów danych, operacji oraz funkcji których nie zawiera Python (Oliphant, 2006).

- `Pandas` – Jest to pakiet zapewniający szybkie, elastyczne oraz ekspresyjne struktury danych zaprojektowane tak, aby praca z danymi była łatwa i intuicyjna. Służy do wykonywania analiz (McKinney, 2010). DataFrame funkcja zawarta w pakiecie Pandas może tworzyć dwu wymiarowa, zmienna oraz potencjalnie niejednorodną strukturę danych. Operacje arytmetyczne są wykonywane na kolumnach i wierszach tabeli(McKinney, 2010)
- `time` – pakiet służący do pomiaru czasu

1.3.3 Język Julia

Język programowania Julia jest młodym językiem prace rozpoczęto w 2009 roku pierwsza wersja została wydana w 2012 roku a w 2018 wydano wersje 1.0. Jest to język wysokiego poziomu ogólnego przeznaczenia (ang. high-level general-purpose)(Bezanson, Edelman, Karpinski & Shah, 2018). Charakterystycznym aspektem w Juli jest to, że zawiera polimorfizm parametryczny oraz jest w pełni dynamicznym językiem programowania. Julia umożliwia współbieżne, równoległe i rozproszone przetwarzanie oraz bezpośrednie wywoływanie bibliotek C i Fortran bez kodu kleju (ang. glue code)(Bezanson i in., 2018). Projekt jest przeznaczony do prowadzenia wydajnych obliczeń numerycznych oraz naukowych oraz systemów niskiego poziomu. Julia wykorzystywana jest do analizy danych, przetwarzania obrazów, analizy statystycznej oraz obliczeń w ekonometrii(Bezanson i in., 2018). Julia zapewnia łatwość i wyrazistość dla kompilacji numerycznej wysokiego poziomu, w w taki sam sposób jak języki R, MATLAB i Python(Bezanson i in., 2018). Cechy języka Julia :

- kompilacja kodu do kodu maszynowego w czasie działania programu dzięki LLVM przez co np. pętle działają szybciej.
- Możliwość tworzenia grafiki na wysokim poziomie
- możliwość pisania własnych funkcji, tym samym na rozszerzanie Octave
- Automatyczne generowanie wydajnego, wyspecjalizowanego kodu dla różnych typów argumentów
- dobra wydajność znacznie szybciej

LLVM jest to nisko poziomowa maszyna wirtualna(ang. Low Level Virtual Machine) kompilator napisany w języku C++. Stanowi warstwę pośrednią, która pobiera kod z kompilatora danego języka i go optymalizuje.

Podczas badania do budowy funkcji generującej wielowymiarowy rozkład normalny wykorzystano następujące pakiety:

- Pkg – podstawowy pakiet do zarządzania oraz instalacji innych pakietów.
- StatsBase – Pakiet zapewnia wsparcie dla statystyki. Implementuje różne funkcje takie jak kowariancje, regresję itd (Arslan, 2019).
- DataFrame – pakiet ten pozwala zapisać dane w formie tabeli. Umożliwia prowadzenie operacji arytmetycznych na kolumnach lub wierszach (Julia Data, 2019).
- LinearAlgebra – pakiet ten pozwala na implementacje wielu operacji algebry liniowej (Julia Team, 2019a).
- Statistics – Pakiet zawierający zawiera funkcje statystyczne oraz testy wykorzystywane w statystyce (Julia Team, 2019b).

Julia z racji tego, że jest językiem nowo powstałym i rozwijanym stanowy dobrą alternatywne dla takich języków jak R czy Python.

1.4 Podsumowanie

Rozdział ten wprowadził od ogólnego pojęcia symulacji w ekonomii po przez coraz to głębsze zagłębianie się i wyjaśnianie pojęci związanych z symulacją. Pozwoliło to dać podstawy i pogląd na temat kolejnego rozdziału. Jak wynika z powyższej treści symulacje się nie odłączną częścią ekonomii oraz przedsiębiorstwa. Proces Symulacji jest związany z przewidywanie pewnych zachowań. Do badania owych zachowań np zachowań konsumenta lub przedsiębiorstwa przydatnym elementem mogą być różne modele regresji liniowej lub nie liniowej. Natomiast podstawię modeli stanowią różne rozkłady jak np rozkład wielowymiarowy normalny, rozkład normalny lub rozkład t-studenta. W rozdziale tym opisane zostały również narzędzia niezbędne to modelowania regresji oraz badania efektów symulacji. Rozdział 2 jest poświęcony rozkładowi wielowymiarowemu oraz jak można generować ten rozkład który stanowi podstawę dalszych badań.

Rozdział 2

Wielowymiarowy rozkład normalny

W rozdziale została przybliżona metody generowania oraz w jaki sposób generować liczby z rozkładu wielowymiarowego normalnego. Rozkład wielowymiarowy normalny jest najczęściej wykorzystywany w regresji liniowej. Modele liniowe są jednymi z najpopularniejszych technik modelowania zależności między zmiennymi (Biecek, 2011).

Analiza ta zajmuje się badaniem zależności między badanymi zmiennymi, mając na uwadze skonstruowanie modelu który dobrze wyjaśni tą zależność(Larose, 2012). Zaletą modeli liniowych jest przejrzystość oraz łatwość wyznaczania statystyk. Wyróżniamy również modele nieliniowe, i modele mieszane. Wszystkie te modele łączy jedna cecha korzystają one z wielowymiarowego rozkładu normalnego. Natomiast rozkład wielowymiarowy normalny tworzony jest z wykorzystaniem wektora zmiennych losowych o rozkładzie normalnym.

2.1 Rozkład normalny i jego właściwości

Rozkład normalny (rozkład Gaussa) jest najczęściej występującym rozkładem oraz najważniejszych rozkładów prawdopodobieństw. Model ten jest wszechobecny w aplikacjach z zakresu biologi, chemii, fizyki, informatyki oraz nauk społecznych. Odkrycie datowane jest na rok 1738 przez Abrahama de Moivre (Francis & Golmant, 2017).

Lecz krzywą normalną jako model rozkładu błędów w teorii naukowej jest najczęściej przypisywany do niemieckiego matematyka Carla Friedricha Gaussa który w 1809 znalazł nową pochodną wzoru na krzywą dlatego nazywana jest czasem krzywą Gaussa lub rozkładem Gaussa (Gordon, 2006).

Co więcej wiele rozkładów zbliża się do rozkładu normalnego. Okazuje się, że sumy niezależ-

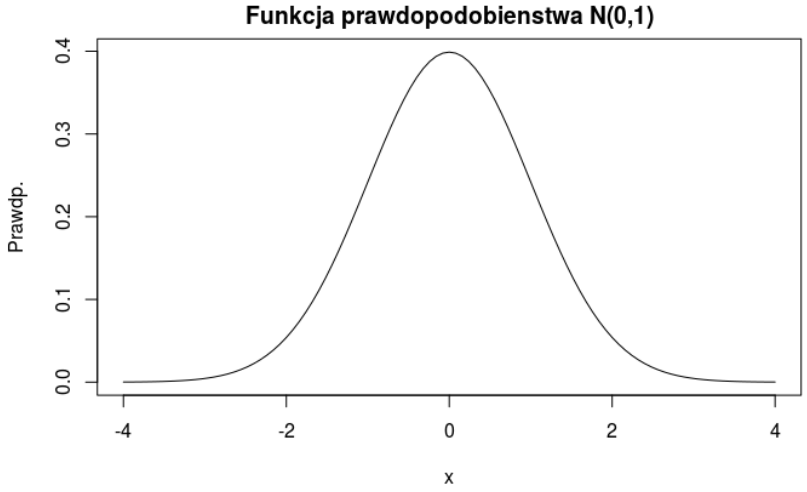
nych od siebie zmiennych losowych o podobnych rozkładach, zbiegają do rozkładu normalnego wraz ze wzrostem liczb dodawanych zmiennych. To zjawisko umożliwia wykorzystanie rozkładu normalnego wszędzie tam, gdzie występuje sumowanie niezależnych czynników o jednakowych rozkładach (Gajda, 2017). „Wszystko to sprawia, że olbrzymia część metod wnioskowania statystycznego jest związana z rozkładem normalnym” (Gajda, 2017).

Funkcja gęstości rozkładu normalnego dana jest wzorem (**??**)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \tag{2.1}$$

gdzie σ jest to odchylenie standardowe, μ jest to wartość średnia, σ^2 jest to wariancja

Za każdym razem kiedy jest możliwe zamieszanie pomiędzy zmienną losową X a wartością realną x funkcja gęstości prawdopodobieństwa reprezentowana jest za pomocą $f(x)$. Rozkład normalny zmiennej X jest zazwyczaj przedstawiany za pomocą (Ribeiro, 2004).



Rysunek 2.1. Wykres gęstości prawdopodobieństwa dla rozkładu normalnego standaryzowanego

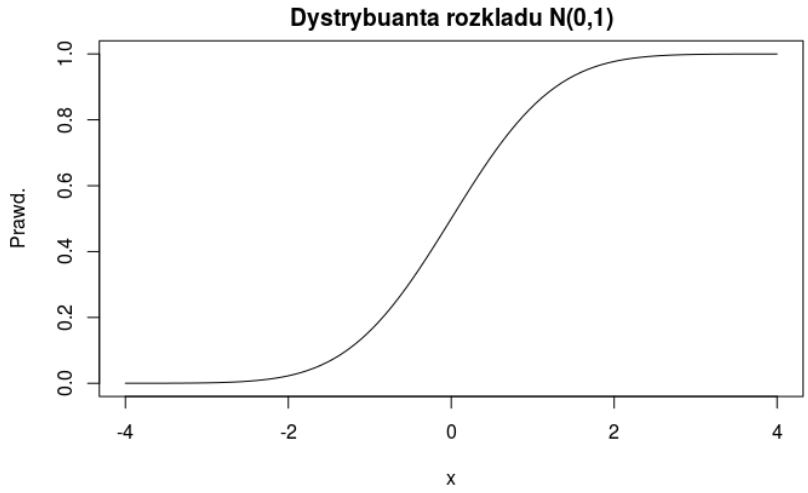
Dystrybuanta rozkładu normalnego dana jest wzorem (2.2)

$$f(X \leq x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx, \tag{2.2}$$

gdzie dla $x \in \mathbb{R}$.

Dystrybuantę definiujemy jako prawdopodobieństwo więc zbiorem wartości tej funkcji jest przedział od 0 do 1. Asymptotami poziomymi są proste $y = 0$ oraz $y = 1$.

Zmienna losowa X z rozkładu normalnego z wartością oczekiwaną μ oraz wariancją σ^2



Rysunek 2.2. Wykres dystrybuanty prawdopodobieństwa dla rozkładu normalnego standaryzowanego

można przedstawić w postaci (2.3):

$$X \sim N(\mu, \sigma^2) \tag{2.3}$$

Rozkład Gaussa jest rozkładem symetrycznym wokół średnie μ oraz wykresem jest krzywa w kształcie dzwonu (Biecek, 2011).

Rozkład ten charakteryzują dwa parametry jest to wartość średnia oraz wariancja (Ribeiro, 2004)

$$\mu = E[X] \tag{2.4}$$

$$\sigma^2 = E[(X - \mu)^2] \tag{2.5}$$

W tym wypadku średnia jest również wartością oczekiwaną oraz punktem w którym funkcja gęstości prawdopodobieństwa osiąga maksimum a wariancja jest miarą rozproszenia zmiennej losowej (Biecek, 2011).

2.1.0.1 Rozkład normalny standaryzowany

Jeśli $X \sim N(\mu, \sigma)$ a zmienna losowa X ma rozkład normalny z parametrami μ,σ, to zmienna losowa Z

$$Z = \frac{X - \mu}{\sigma}, \tag{2.6}$$

gdzie Zmienna Z ma standardowy rozkład normalny $Z \sim N(0,1)$.

Standaryzacja zmiennej losowej o rozkładnie normalnym umożliwia korzystanie z tablic statystycznych dla rozkładu normalnego o dowolnych parametrach przy zachowaniu następującej zależności (Hellwig, 1998):

$$P(X \leq x) = \phi \frac{x-\mu}{\sigma}. \tag{2.7}$$

2.2 Wielowymiarowy rozkład normalny i jego właściwości

Rozkład wielowymiarowy normalny (MV-N) jest wielowymiarowym uogólnieniem jednowymiarowego rozkładu normalnego. W swojej najprostszej postaci, zwanej „standardowym" rozkładem MV-N (ang. *Multivariate Normal*), opisuje on łączny rozkład losowego wektora, którego wpisy są wzajemnie niezależnymi zmiennymi losowymi jednowymiarowymi mającymi zerową średnią i wariancję jednostki. W swojej ogólnej formie opisuje łączny rozkład losowego wektora, który można przedstawić jako liniową transformację standardowego wektora MV-N.

Wielowymiarowy rozkład normalny można zdefiniować następująco n-wymiarowa zmienna losowa $X = [x_1, ..., x_n]^T$ podlega n-wymiarowemu rozkładowi normalnemu jeśli dowolna kombinacja liniowa $Y = a_1x_1 + ... + a_nx_n$ jej składowych ma rozkład normalny. Prognozowanie na podstawie wielowymiarowego rozkładu jest podobne do prognozowania za pomocą modelu jednorównaniowego. W modelu wielowymiarowym występuje wektor składników losowych co lepiej może opisywać zmienną objaśnianą (Błaszczuk, 2006)

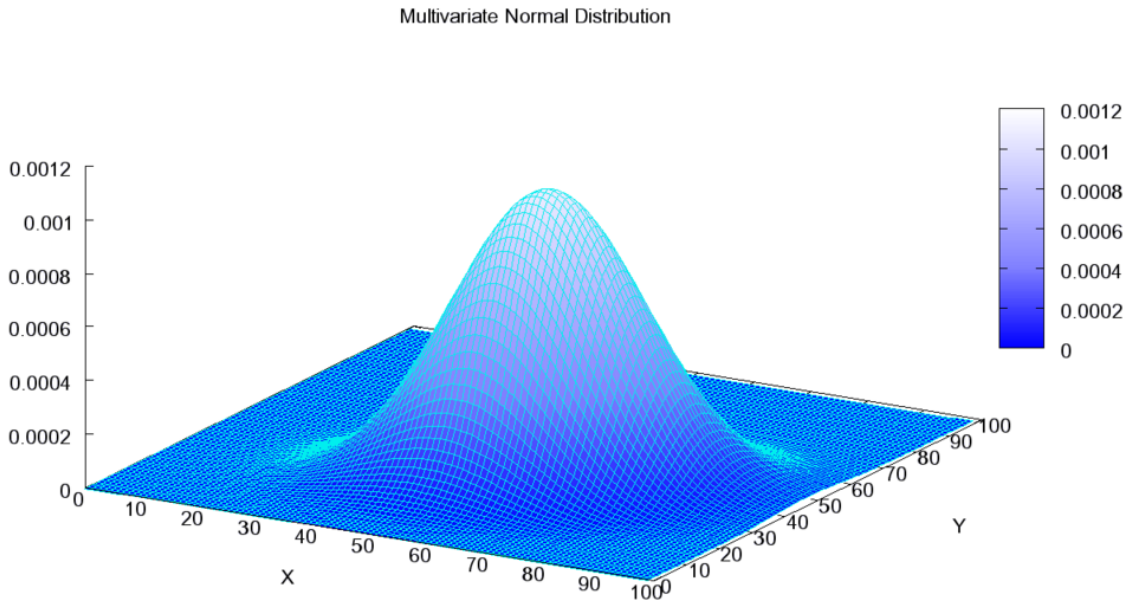
Funkcja gęstości przedstawia się następująco

$$f(x) = \frac{1}{(2\pi)^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)), \tag{2.8}$$

którą można oznaczyć skrótowo

$$X \sim N(\mu, \Sigma) \tag{2.9}$$

Gdzie Σ to macierz kowariancji, symetryczna dodatnio określona o wymiarach n a μ to wektor średnich o długości n. Wizualizacja gęstości dwuwymiarowego rozkładu wielowymiarowego przedstawia wykres 2.3.



Rysunek 2.3. Rozkład wielowymiarowy normalny

2.2.1 Estymacja parametrów

Jeśli posiadając dane N wektorów które zostały obrane z wielowymiarowego rozkładu normalnego o wektorze wartości oczekiwanych μ oraz macierzy kowariancji Σ parametry możemy oszacować następująco. Estymator wartości oczekiwanej:

$$\widehat{\mu} = \frac{1}{N}\sum_{i-1}^{n}(X_i), \tag{2.10}$$

Natomiast, estymator macierzy kowariancji największej wiarygodności

$$\widehat{\Sigma} = \frac{1}{N}\sum_{i=N}^{N}(X_i - \widehat{\mu})(X_i - \widehat{\mu})^T. \tag{2.11}$$

2.3 Metody generowania rozkładów wielowymiarowych

Zmienne z rozkładu wielowymiarowego normalnego uzyskujemy przez wektor średnich μ oraz macierzy kowariancji Σ.

W pierwszym kroku generuje się n niezależnych zmiennych o rozkładzie normalnym. W drugim kroku przekształca się te zmienne by otrzymać średnią μ i macierz kowariancji Σ. Korzysta

się z faktu, że jeżeli $Z \sim N(0,I)$, to

$$CZ = \mu = N_d(\mu, CC^T) \tag{2.12}$$

Żeby wylosować zmienne których potrzebujemy wystarczy wylosować wektor niezależnych zmiennych Z, a następnie przekształcić je liniowo $X = CZ + \mu$. W tym celu macierz Σ musimy rozłożyć na iloczyn CC^T. Można to zrobić używając różnych dekompozycji macierzy Σ (Robert, 2011).

2.3.1 Dekompozycja Spektralna

Wektory własne V_i (macierz wektorów własnych ułożonych kolumnami oznaczamy V) i wartości własne λ_i (macierz diagonalna, na której przekątnej są wartości własne oznaczamy D) mają następujące właściwości:

$$Av = \lambda \qquad AV = VD \tag{2.13}$$

Jeżeli macierz A jest symetryczna to

$$CC^T = I \qquad \lambda_i \in R \qquad tr(A^p) = \Sigma_i + \lambda_i^p \tag{2.14}$$

Dla dowolnego p, gdzie $tr(X)$ oznacza ślad macierzy X, czyli sumę wartości na przekątnej (Biecek, 2011). W programie R dekompozycję spektralną można wyznaczyć funkcją `eigen()` jest to funkcja podstawowa R nie potrzebna jest dodatkowa instalacja bibliotek. W języku Python za pomocą `eig()` importowaną z pakietu Numpy a w języku Julia funkcja `eigen()` z Pakietu LinearAlgebra (Biecek, 2011).

$$C = \Sigma^{\frac{1}{2}} = P\Gamma^{\frac{1}{2}}P^{-1} \tag{2.15}$$

Gdzie Γ jest macierzą diagonalną wartości własnych a P jest macierzą wektorów własnych $P^{-1} = P^T$.

Dekompozycja Spektralna dla macierzy 3x3 wylosowanej z rozkładu normalnego przedstawia się następująco:

$$\Gamma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad P = \begin{bmatrix} \lambda_{11} & a_{12} & a_{13} \\ a_{21} & \lambda_{22} & a_{23} \\ a_{31} & a_{32} & \lambda_{33} \end{bmatrix}$$

2.3.2 Dekompozycja SVD

Dekompozycja ta jest uogólnieniem dekompozycji spektralnej na macierze prostokątne. Funkcje dla biblioteki LAPACK jest to biblioteka procedur numerycznych dla algebry liniowej napisana w fortranie. Pozwala na uwzględnienie tego, że rozkład macierzy jest symetryczny i dodatnio określony jednakże wrapper R już tego nie uwzględniają, więc ta transformacja jest mniej efektywna. macierze U , V, D dla macierzy Σ można znaleźć z użyciem funkcji svd() w języku R natomiast w Pythonie używa się `svd()` z pakietu SciPy.linalg oraz w Juli funkcja `svd()` z pakietu LinearAlgebra (Biecek, 2011).

$$C = \Sigma^{\frac{1}{2}} = UD^{\frac{1}{2}}V^T \tag{2.16}$$

gdzie U, D oraz V są określone następująco

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad V = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 2 & 9 \end{bmatrix}^T .$$

2.3.3 Dekompozycja LU

Jeśli przyjmiemy Σ jako kwadratową macierz bez zerowych minorów głównych [1], to

$$\Sigma = LU \tag{2.17}$$

L jest dolną macierzą trójkątną, a U jest górną macierzą trójkątną (Petersen & Petersena, 2008).

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

[1] Minor to inaczej wyznacznik macierzy który powstaje przez skreślenie pewnej ilości wierszy i kolumn natomiast minor główny macierzy powstaje przy wykreślaniu tak aby pozostały wiersze i kolumny o równych indeksach

2.3.4 Dekompozycja Choleskiego

Jeśli przyjmiemy Σ jako dodatnią symetryczną macierz kwadratową, gdzie

$$\Sigma = U^T U = LL^T, \tag{2.18}$$

to wtedy

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}^T$$

gdzie U jest dolną macierzą trójkątną a L jest dolną macierzą trójkątną (Petersen & Petersena, 2008). Macierz U dla macierzy Σ można znaleźć przy użyciu funkcji `chol()` funkcja używana w języku R natomiast w języku Python wykorzystuje się metodę `cholesky()` z pakietu Numpy a w Juli z metody `cholesky()` z pakietu LinearAlgebra.

W programie R nie musimy do tworzenia zmiennych z rozkładu wielowymiarowego normalnego ręcznie wykonywać dekompozycji macierzy Σ. Należy użyć funkcji `rmvnorm(mvtnorm)`. Pozwala ona wskazać metodę dekompozycji argumentem method = (”eigen”, ”svd”, ”chol”). Domyślnie wybierana jest dekompozycja na wartości spektralne na wypadek gdyby macierz Σ była niepełnego rzędu, ale jeżeli macierz Σ jest dodatnio określona to lepiej zastosować dekompozycje pierwiastka Choleskiego (Biecek, 2011).

2.4 Podsumowanie

Rozdział ten przybliżył istotę rozkładu normalnego oraz pokazał właściwości owego rozkładu. Zagłębiając się dalej przedstawiono rozkład wielowymiarowy normalny oraz własności owego rozkładu. Pokazano również metody dekompozycji macierzy dzięki którym można generować wielowymiarowy rozkład normalny. Pozwoliło to wyjaśnić działanie funkcji zawartych w kolejnym rozdziale oraz wiedza ta umożliwiła napisanie funkcji które stanowią szkielet badania. Przedstawione badanie ma na celu ukazanie optymalnego języka programowania na podstawie generowania wielowymiarowego rozkładu normalnego dzięki czemu można będzie potwierdzić hipotezę zawartą we wstępie lub odrzucić ową hipotezę.

Rozdział 3

Empiryczna ocena wydajności na podstawie próbkowania z wielowymiarowego rozkładu normalnego

3.1 Opis procedury badawczej

3.1.1 Ekonomiczny aspekt badania

W celu sprawdzenia wydajności języków programowania R, Python, Julia zbudowano w każdym z wymienionych programów funkcje generującą wielowymiarowy rozkład normalny. Funkcje zbudowano tak aby można było wybrać metodę dekompozycji oraz wielkość wymiaru macierzy. Powstały w ten sposób zbiór danych został poddany analizie której celem jest pokazanie który program działa najszybciej oraz jak to wpływa na koszt korzystania z chmur obliczeniowych np Microsoft Azure lub Amazon Web Services (AWS).

Amazon Web Services (AWS) Infrastruktura chmurowa stworzona przez Amazon. Umożliwia korzystanie z zasobów chmurowych oraz całej infrastruktury. Znajdują się tam wirtualne maszyny do Machnie Lerningu, analizy danych oraz do tworzenia hurtowni danych. AWS umożliwia też obliczenia na kartach graficznych oraz a procesorach. Wykorzystanie kart graficznych znacznie przyśpiesza obliczenia. W AWS płaci się za faktyczne wykorzystanie zasobów oraz czas korzystania z infrastruktury. Pozwala to zaoszczędzić czas i pieniądze oraz podczas intensywnego obłożenia serwera przez klientów automatycznie zwiększą się potrzebne zasoby co uniemożliwi przeciążenia serwera.

Microsoft Azure Jest to rozwiązanie podobne to AWS tylko stworzone przez Microsoft. Tak samo jak AWS Azure udostępnia przestrzeń chmurową do obliczeń i przechowywania danych. Azure posiada też gotowe narzędzia do analiz takie jak Python oraz R. Posiada również możliwość przechowywania danych w przystosowanych do tego bazach danych lub hurtowniach danych. Podobnie jak w AWS w Azure płacimy za faktyczne wykorzystanie zasobów oraz wykorzystanej mocy obliczeniowe.

Tabela 3.1. Koszt wykorzystania AWS

Czas wykorzystania AWS	Cena w dolarach
1 min wykorzystania AWS	0,003 $
2 min wykorzystania AWS	0,006 $
3 min wykorzystania AWS	0,013 $
6 min wykorzystania AWS	0,020 $
7 min wykorzystania AWS	0,021 $
15 min wykorzystania AWS	0,04 $
20 min wykorzystania AWS	0,05 $

Źródło: Opracowanie na podstawie aws.amazon.com

Rozwiązanie Chmurowe staje się teraz praktycznie codziennością ponieważ już nie opłaca się budowania własnej struktury serwerowej oraz utrzymanie obsługi serwerów. Firmy coraz częściej przenoszą przetwarzanie danych na chmury ponieważ przyspiesza to obliczenia oraz pozwala lepiej zabezpieczyć dane klientów. Mając to na uwadze wykonane badanie ma pokazać który język najlepiej nadaje się do obliczeń oraz prazy z dużymi zasobami danych.

3.1.2 Układ badania

Do badania zbiór danych został podzielony ze względu na język programowania oraz metodę dekompozycji. W zbiorze znajdują się 3 rodzaje dekompozycji choleskiego, SVD oraz Spektralna.

Dla każdej metody zostały wylosowane 100 razy macierze za pomocą rozkładu normalnego. Każda próba została przeprowadzona na macierzach dla których wymiar zwiększał się o 100. Oznacza to że dla dekompozycji choleskiego przeprowadzono 100 losowań oraz zwiększano wymiar macierzy z 200 na 100 do 300 na 200 itd. Pozwoliło to na wygenerowanie zbioru z 8 tyś danych czasowych dzięki czemu dokładniej oszacowano czas dekompozycji w poszczególnych językach.

Tabela 3.2. Metoda Badania

Metoda dekompozycji	Ilość powt
Dekompozycja Choleskiego	100 losowań
Dekompozycja SVD	100 losowań
Dekompozycja Spektralna	100 losowań
Wymiar macierzy	200 300 400 ... do 1000
Ilość losowań dla każdego wymiaru macierzy	100 losowań

Źródło: Opracowanie własne.

3.1.3 Autorskie skrypty wykorzystane w badaniu

```
library(tidyverse)
library(rbenchmark)

funkcja_losowanie = function(n, d, method) {
  # tworze macierz sigma
  Sigma = cov(matrix(rnorm(n*d),n,d))
  # tworze macierz X
  X = cov(matrix(rnorm(n*d),n,d))
  # jesli method == "CHOL" to odpal ta czesc kodu
  if(method == "CHOL") {
    # dekompozycja Choleskiego
            Q = chol(Sigma)
            wynik = X \% * \% Q
            # mnozenie macierzy
  # jesli method == "SVD" to odpal ąt ęśćcz kodu
  } else if(method == "SVD") {
    # dekompozycja SVD
          tmp = svd(Sigma)
          wynik = X \% * \%
          (tmp$u \% * \% diag(sqrt(tmp$d)) \% * \% t(tmp$v))

    # jesli method == "SPEKTR" to odpal ta czesc kodu

  } else if(method == "SPEKTR") {
    # dekompozycja spektralna
      tmp = eigen(Sigma, symmetric=T)
       wynik = X \% * \%
       (tmp$vectors \% * \% diag(sqrt(tmp$values)) \% * \% t(tmp$vectors))

  } else {
    # w~przypadku podania zlego argumentu method pojawi ęsi ten komunikat
    return("Podano zly argument funkcji")
  }
  return(wynik)
}
```

Program 3.1. Program generowania rozkładu wielowymiarowego w R

Przedstawiony kod w języku R tworzy funkcję generowania wielowymiarowego rozkładu normalnego po przez wybranie odpowiedniej funkcji. Na początku określa się wymiar macierzy oraz metodę dekompozycji jaką ma zostać wykonane generowanie. W zależności jaką metodę podano to funkcja wykonuję odpowiednią część kodu.

```
import numpy as np
import random as random
import math
from scipy.linalg import svd
from numpy import linalg as LA
import timeit
import datetime
import time
import pandas as pd

def fun_test(n, d, method):

    X = np.random.normal(size = (n,d))
    Sigma = np.cov(X.transpose())
    if(method == "CHOL"):
        Q = np.linalg.cholesky(Sigma)
        wynik = np.dot(X,Q)

    elif(method == "SVD"):
        U, s, VT = svd(Sigma)
        Q = np.dot(U,np.diag(np.sqrt(s)))
        W~= np.dot(Q,VT.transpose())
        wynik = np.dot(X,W)

    elif(method == "SPEKTR"):
        w, v = LA.eig(Sigma)
        Q = np.dot(v,np.diag(np.sqrt(w)))
        W~= np.dot(Q,v.transpose())
        wynik = np.dot(X,W)

    else:
        print("podano zla wartosc")
    return(wynik)
```

Program 3.2. Program generowania rozkładu wielowymiarowego w Python

Przedstawiony kod w języku Python tworzy funkcję generowania wielowymiarowego rozkładu normalnego. Na początku wczytano potrzebne pakiety z których korzysta funkcja. Podczas włączania funkcji określa się wymiar macierzy oraz metodę dekompozycji jaką ma zostać wykonane generowanie. W zależności jaką metodę podano to funkcja wykonuję odpowiednią część kodu.

```
using Pkg
using StatsBase
using Statistics
using DataFrames
using LinearAlgebra

#funkcja dekompozycji
    function dekompozycja(n,d,metoda = "CHOL")
        Sigma = cov(randn(n,d))
        X = transpose(Sigma)
            if metoda == "CHOL"
                Q = cholesky(Sigma)
                wynik = X * Q.factors
            elseif metoda == "SVD"
                Q = svd(Sigma)
                wynik = (Q.U * Diagonal(sqrt.(Q.S))) * transpose(Q.Vt)
            else metoda == "SPEK"
                Q = eigen(Sigma)
                wynik = (Q.vectors*(Diagonal(sqrt.(Q.values))*transpose(Q.
                    vectors)))
```

```
        end
        return wynik
    end
```

Program 3.3. Program generowania rozkładu wielowymiarowego w Julia

Przedstawiony kod w języku Julia tworzy funkcję generowania wielowymiarowego rozkładu normalnego. Na początku wczytano potrzebne pakiety z których korzysta funkcja. Podczas włączania funkcji określa się wymiar macierzy oraz metodę dekompozycji jaką ma zostać wykonane generowanie. W zależności jaką metodę podano to funkcja wykonuję odpowiednią część kodu.

3.2 Wyniki badania

3.2.1 Dekompozycja SVD

Analiza zebranych pomiarów czasowych została przeprowadzona w języku R. Przedstawiono wyniki pomiaru czasu dla funkcji wybierając metodę dekompozycji SVD

```
analiza_all %>% filter(metoda == "svd_jl" | metoda == "svd_py" |
metoda == "SVD") %>%
ggplot2::ggplot(aes(wymiar, srednia, color = jezyk)) +
  geom_line(aes(linetype = metoda))+
  ylab("Średni czas w s")+
  xlab("Wymiar")+
  ggtitle("Średni czas dekompozycji SVD wraz ze wzrostem wymiaru macierzy")
```

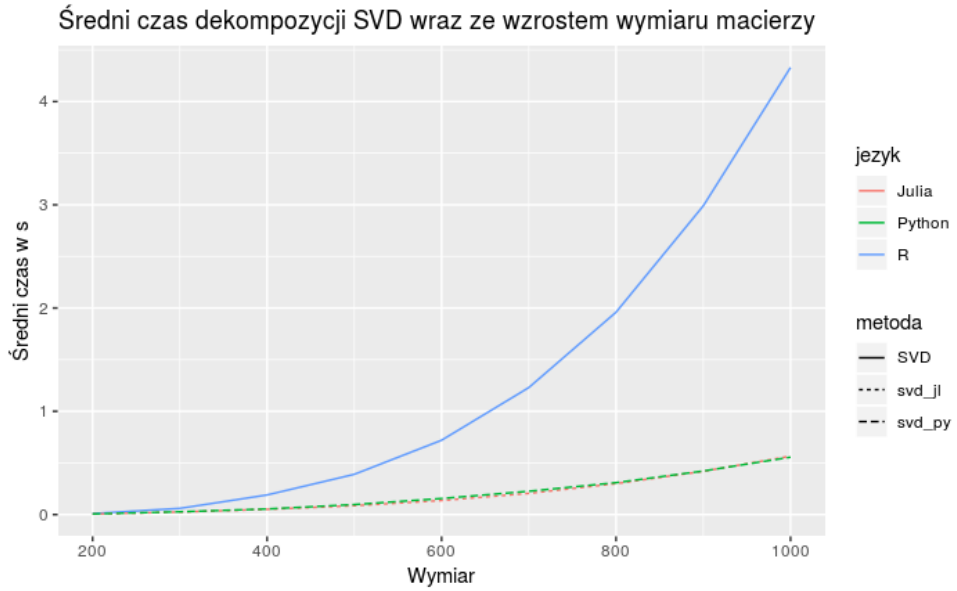
Program 3.4. Kod tworzący wykres 3.1

Wykres 3.1 przedstawia wykładniczy wzrost czasu wraz ze wzrostem wymiaru macierzy. SVD oznacza dekompozycje w R, svd_py oznacza dekompozycje w Pythonie, a svd_jl oznacza dekompozycje w Julii. Na wykresie przedstawiono średni czas dla wybranej metody. Zauważono mocny wzrost czasu dla języka R natomiast średni czas dla języka Python oraz Julia jest bardzo zbliżony

3.2.2 Dekompozycja Choleskiego

Wyniki pomiaru czasu dla funkcji wybierając metodę dekompozycji Choleskiego.

```
analiza_all %>% filter(metoda == "chol_jl" |
                         metoda == "chol_py" |
                         metoda == "CHOL") %>%
ggplot2::ggplot(aes(wymiar, srednia, color = jezyk)) +
  geom_line(aes(linetype = metoda))+
  ylab("Średni czas w s")+
  xlab("Wymiar")+
```

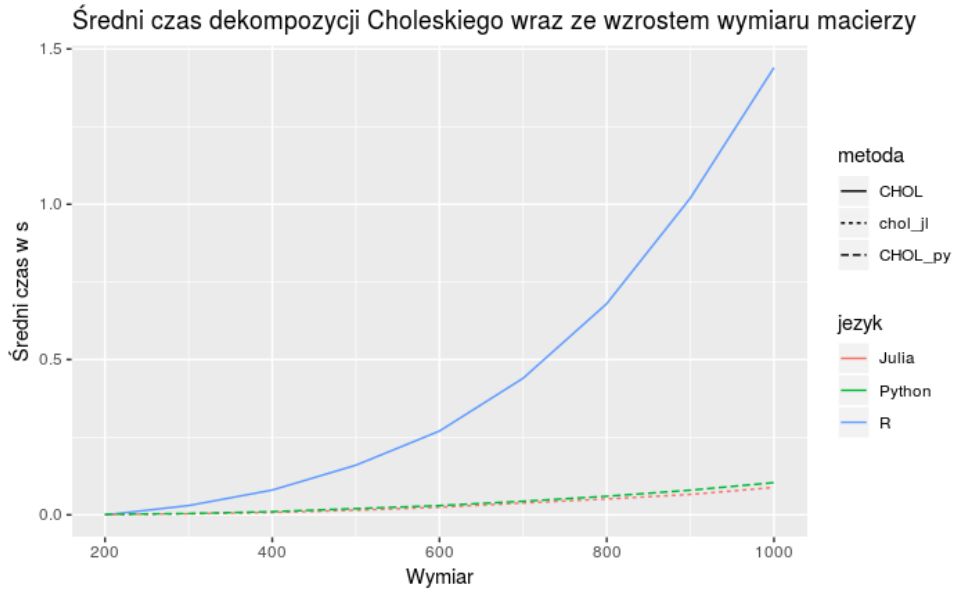


Rysunek 3.1. Czas dekompozycji SVD w R, Python i Jula

Źródło: Opracowanie własne.

```
ggtitle("Średni czas dekompozycji Choleskiego wraz ze wzrostem wymiaru macierzy")
```

Program 3.5. Kod tworzący wykres 3.2



Rysunek 3.2. Czas dekompozycji Choleskiego w R, Python i Jula

Źródło: Opracowanie własne.

Wykres 3.2 Przedstawia średni czas wraz ze wzrostem wymiaru macierzy. CHOL oznacza dekompozycje w R, CHOL_py, a SVD_jl oznacza dekompozycje w Julii. Język R znacząco odbiega od

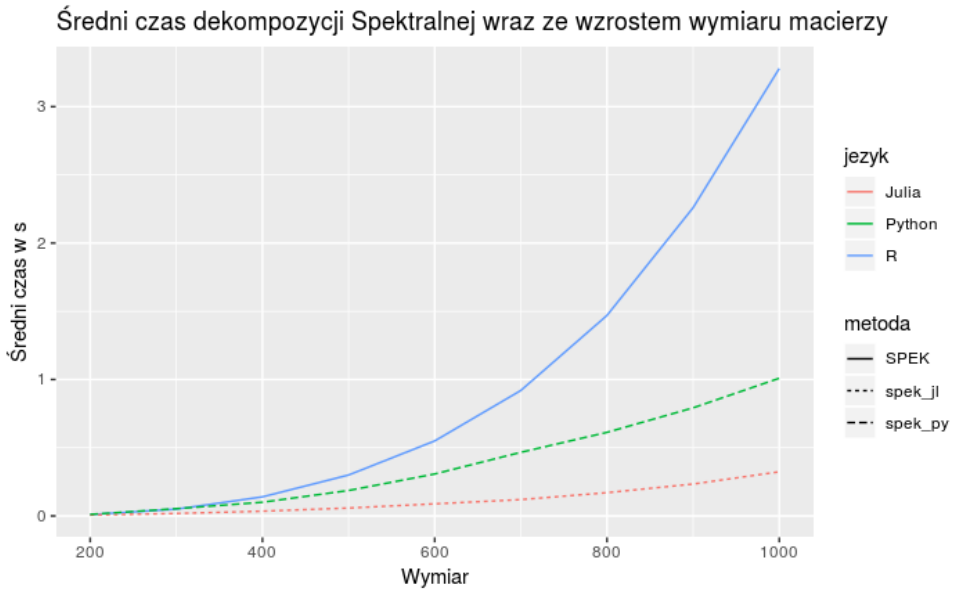
Pythona oraz Juli, natomiast w przypadku Pythona oraz Juli średni czas wykonania jest bardzo zbliżony.

3.2.3 Dekompozycja Spektralna

Wyniki pomiaru czasu dla funkcji wybierając metodę dekompozycji Spektralnej

```
analiza_all %>% filter(metoda == "spek_jl" |
metoda == "spek_py" |
metoda == "SPEK") %>%
ggplot2::ggplot(aes(wymiar, srednia, color = jezyk)) +
  geom_line(aes(linetype = metoda))+
  ylab("Średni czas w s")+
  xlab("Wymiar")+
  ggtitle("Średni czas dekompozycji Spektralnej wraz ze wzrostem wymiaru
     macierzy")
```

Program 3.6. Kod tworzący wykres w R



Rysunek 3.3. Czas dekompozycji SVD w R, Python i Jula

Źródło: Opracowanie własne.

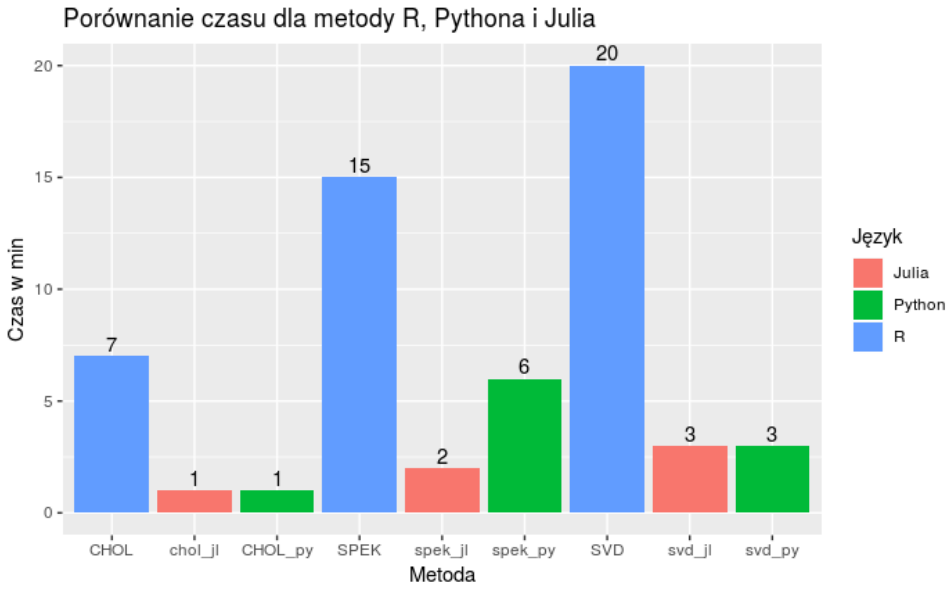
Wykres 3.3 Przedstawia średni czas dekompozycji spektralnej wraz ze wzrostem wymiaru macierzy. SPEK oznacza dekompozycje w R, spek_py oznacza dekompozycje w Pythonie, a spek_jl oznacza dekompozycje w Julii. Przedstawione funkcje wyraźnie różnią się oraz czasy wzrasta wykładniczo. Najszybciej dekompozycję przeprowadza język Julia natomiast najwolniej język R.

3.2.4 Symulacje a koszt badania

```
analiza_all_jezyk = data_R_and_py %>%
  group_by(metoda,jezyk) %>%
    summarise(srednia = round(mean(time),2),
              suma = sum(time)) %>% mutate(czas_w_min = round(suma/60,0))
```

Program 3.7. Kod tworzący zbiór danych w R

Zbiór danych dla którego policzono średni czas, sumę czasu oraz czas w minutach. Wyniki pogrupowano ze względu na język oraz metodę dekompozycji.



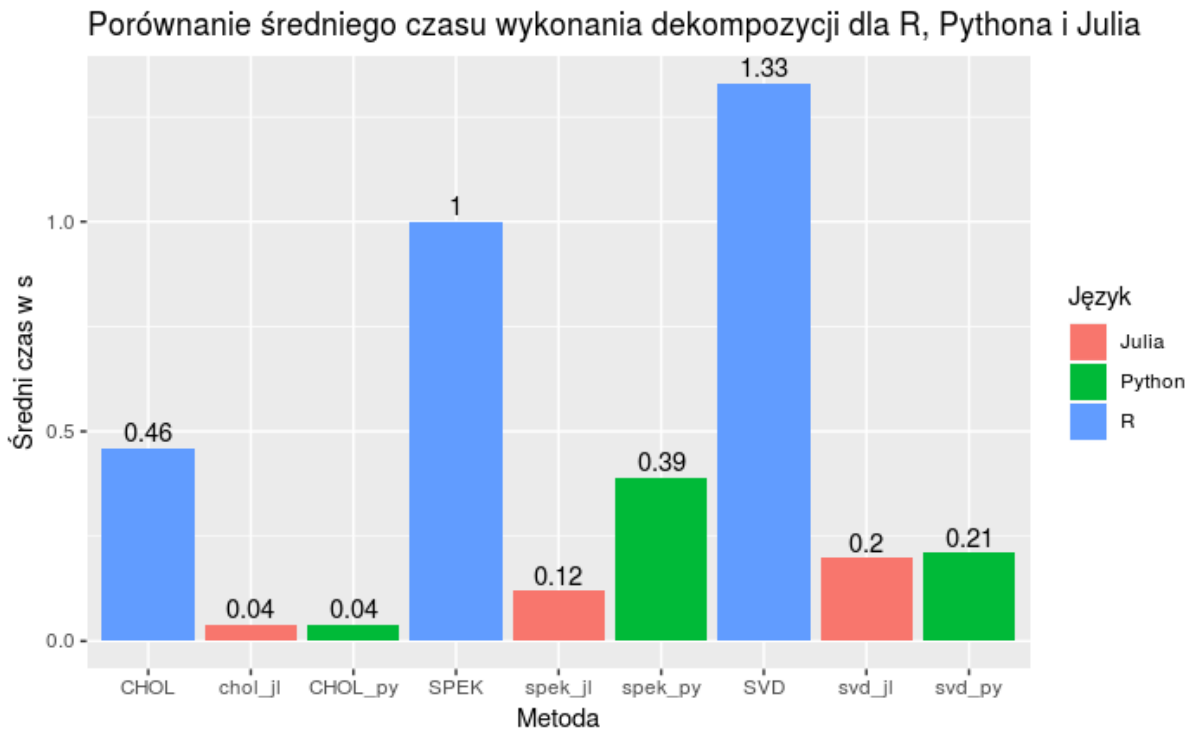
Rysunek 3.4. Czas dekompozycji w R, Python i Jula

Źródło: Opracowanie własne.

Wykres 3.4 Przedstawia czas wykonania dekompozycji w języku R, Python oraz Julii. Czas pracy w języku Python jest znacznie szybszy niż w języku R oraz mocno zbliżony do Julii natomiast w Julii funkcja wykonała się najszybciej. Jak można zauważyć dla metody dekompozycji choleskiego czas w Pythonie wynosi tyle samo co w Julii czyli 1 min, w R jest to 7 min.

Dla dekompozycji SVD czas w Pythonie wynosi 3 min tak samo jak w Juli natomiast w R jest to 20 min. Jest to największa różnica. Dla dekompozycji Spektralnej czas w Pythonie wynosi około 6 min w R jest to 15 min w Juli czas wynosi 2 min.

Wykres 3.5 przedstawia średni czas wykonania dekompozycji. Można zauważyć że średni czas wykonania zawiera się w przedziale 0 do nieco ponad 1 sekundy. R w tym przypadku odstaje mocno od Pythona oraz Julii ma to miejsce również podczas wykonania całej funkcji dekompozycji.

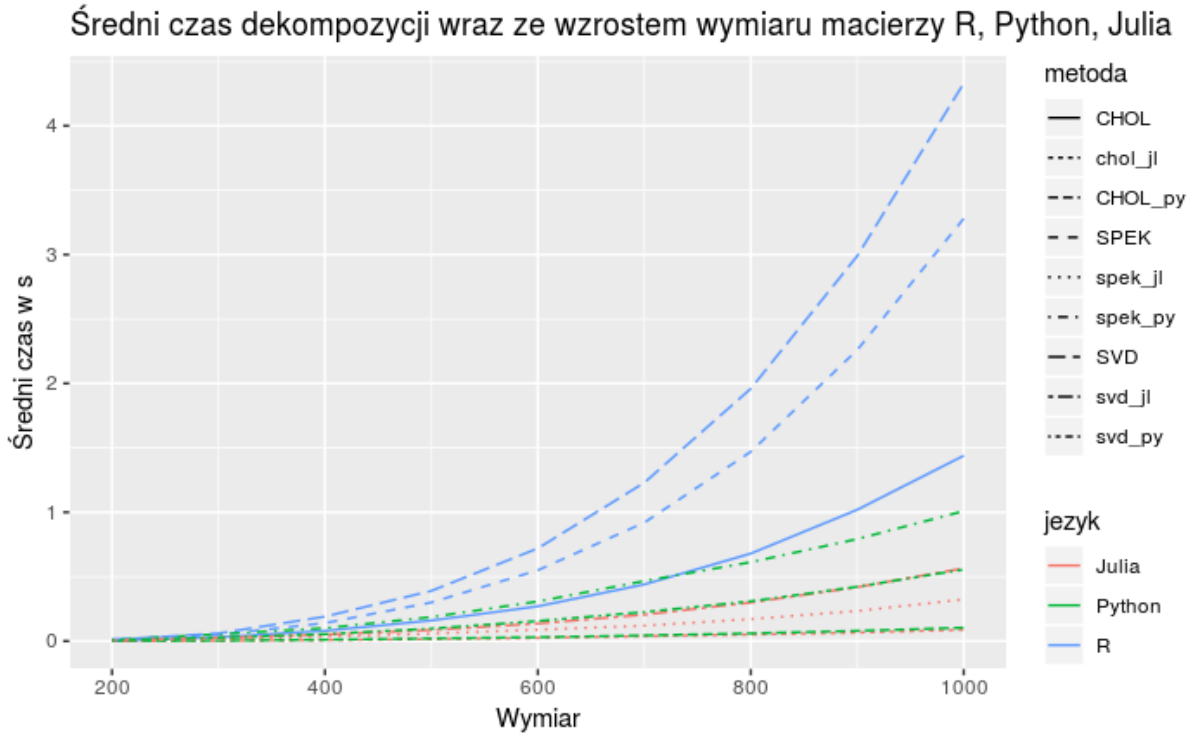


Rysunek 3.5. Średni czas dekompozycji R, Python i Jula

Źródło: Opracowanie własne.

Wykres 3.6 przedstawia średni czas wykonania poszczególnych dekompozycji w raz ze wzrostem wymiaru macierzy, gdzie oznaczenia są takie same jak na poprzednich rysunkach. Zależność ma charakter wykładniczy wraz ze wzrostem wymiaru macierzy czas wzrasta wykładniczo. Jest to niekorzystne zjawisko przy dużych zbiorach danych. Wykres dla języka Python wzrasta powoli podobnie jak wykres dla języka Julia natomiast dla języka R wykres wzrasta dość gwałtownie. Można też zauważyć dość dużą rozbieżność pomiędzy wykresami (linie niebieska R oraz zielona Python).

Z przeprowadzonego badania wywnioskować można że wykorzystanie Juli najlepszą decyzją ponieważ wszystkie funkcje są wykładnicze ale przyrastają najwolniej oraz najważniejsze czas wykonania znacznie różni się od czasu w R. Pełny czas wykonania dekompozycji choleskiego dla języka R zajmuje około 7 min natomiast dla języka Python jest są to około 1 min można stwierdzić, że w tym przypadku Python jest 7 razy szybszy taki sam czas jak Python uzyskał język programowania Julia. Założono przykładowy koszt 0.000004897 dolara za wykorzystania chmury obliczeniowej AWS dla 100ms. Oznacza to, że za 1 min wykorzystania AWS zapłacimy 0,003 dolara. Przekładając to na wyniki badania za język R zapłacimy:



Rysunek 3.6. średni czas dla R, Python i Jula

Źródło: Opracowanie własne.

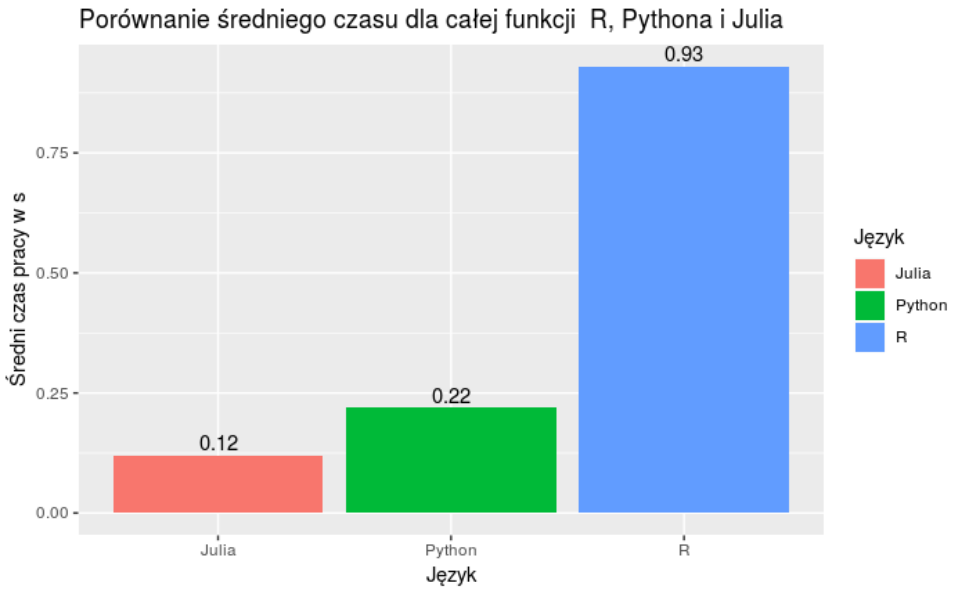
- dekompozycja Choleskiego 0,021 dolara
- dekompozycja SVD 0,04 dolara
- dekompozycja Spektralna 0,05 dolara

za język Python zapłacimy:

- dekompozycja Choleskiego 0,003 dolara
- dekompozycja SVD 0,013 dolara
- dekompozycja Spektralna 0,020 dolara

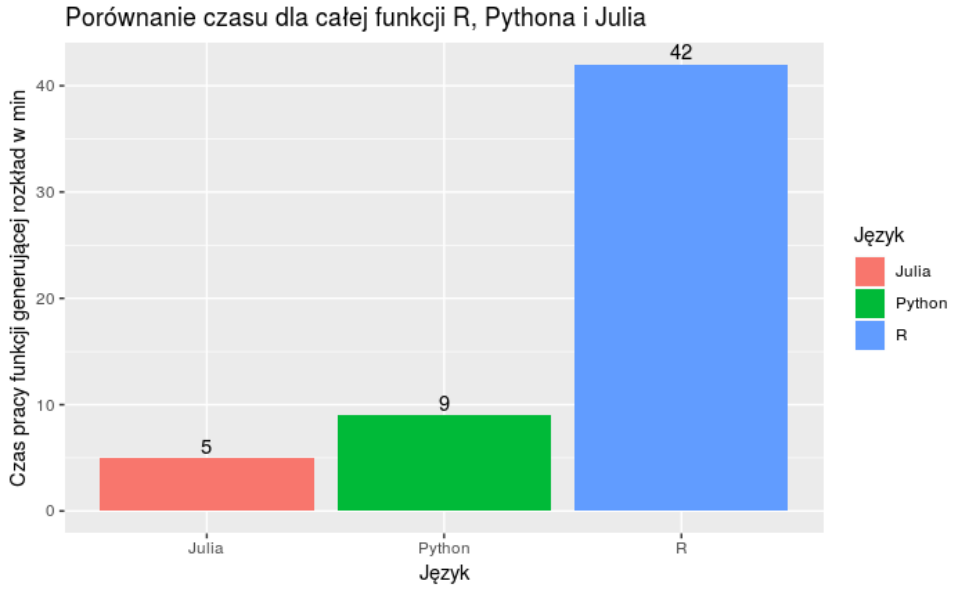
za język Julia zapłacimy

- dekompozycja Choleskiego 0,003 dolara
- dekompozycja SVD 0,013 dolara
- dekompozycja Spektralna 0,006 dolara



Rysunek 3.7. Średni czas ogółem dla R, Python, Julia

Źródło: Opracowanie własne.



Rysunek 3.8. Całkowity czas dla R, Python, Julia

Źródło: Opracowanie własne.

Jak widać wykres 3.7 przedstawia średni czas wykonania całej funkcji bez rozbicia na poszczególne metody dekompozycji. Najszybszy jest język Julia.

Wykres 3.8 Pokazuje jak wygląda rzeczywisty czas wykonania całej funkcji bez rozbicia na poszczególne metody w tym przypadku R bardzo ostaje od pozostałych języków różnica po-

między R a Julią wynosi 37 min natomiast różnica pomiędzy Pythonem a R wynosi 33 min dla porównania różnica pomiędzy Julią i Pythonem wynosi 4 min.

Badanie pokazało, że w zastosowaniach komercyjnych najlepiej byłoby wybrać język Julia jest to najszybszy język i stosunkowo młody. Lecz niesie to za sobą inne konsekwencje język ten jeszcze w pełni nie dojrzał a w zastosowaniach komercyjnych potrzebne jest pewne i stabilne narzędzie. Takim narzędziem jest właśnie język Python który jest stabilnym językiem z dużym zapleczem wsparcia dla przedsiębiorstw. Pod względem szybkości dorównuje Julii. Natomiast Język R jest językiem w dużej mierze wykorzystywany przez środowisko akademickie gdzie nie jest kładziony duży nacisk na szybkość wykonywania obliczeń. Optymalnym wyborem na podstawie przeprowadzonych badań jest język Python co ma odwzorowanie w rzeczywistości ponieważ obecnie do analiz najczęściej wykorzystuje się Pythona. Każdy z badanych języków jest w większym lub mniejszym stopniu używany w przedsiębiorstwach i powinno się rozwijać w wykorzystanie hybrydowe języków.

Podsumowanie

Przedmiotem pracy było wykorzystanie wielowymiarowego rozkładu normalnego oraz zweryfikowanie wydajności obliczeniowej języków R, Python i Julia. Znajomość wydajności wybranych programów jest kluczowa przy wyborze języka do potencjalnego projektu lub planowania analizy. Wiedza ta pozwoli zaoszczędzić środki wydane na badania oraz czas.

W badaniu wykazano, że istotny jest dobór odpowiedniego języka. Następnie opisano platformy udostępniające serwery oraz moce obliczeniowe w chmurze. Platformy to Amazon Web Services oraz Microsoft Azure. Analiza platform pozwoliła wykazać istotne czynniki takie jak Cena wykorzystania zasobów chmurowych oraz cenę za godzinę wykorzystania mocy obliczeniowej. W związku z tym zbudowano odpowiednie funkcje generujące rozkład wielowymiarowy normalny wykorzystujące dekompozycje Choleskiego, Spektralna oraz SVD. Na podstawie których zbudowano zbiór pomiarów czasowych. Jak wykazały badania najszybciej symulacje wykonał język Julia, najszybciej została wykonana dekompozycja Choleskiego następnie dekompozycja Spektralna później dekompozycja SVD. Zauważyć można prawidłowość, że wraz ze wzrostem wymiaru macierzy czas wzrasta wykładniczo. Zauważyć można też inną prawidłowość w każdym języku dekompozycja Choleskiego wykonuje się najszybciej.

Analiza pokazała, że dzięki metodzie generowania rozkładu wielowymiarowego normalnego można wykazać który język jest najszybszy. Przeprowadzone badanie pozwoliło potwierdzić hipotezę, że optymalny czas pracy języka można określić za pomocą generowania rozkładu wielowymiarowego normalnego oraz na podstawie szybkości przeprowadzonych dekompozycji na macierzach.

Na podstawie przeprowadzonych analiz można wybrać optymalny język który pozwoli zaoszczędzić czas oraz środki przeznaczone na badania. Korzystając z AWS oraz wybierając język Julia pozwala to zaoszczędzić dwa razy więcej niż wykorzystując język Python oraz osiem razy szybciej pracuje niż R.

Julia jest stosunkowo świeżym językiem na rynku natomiast języki Python oraz R posiadają

ugruntowaną pozycję na rynku. Powoduje to większe zaufanie do stabilnych języków. Dlatego język Python jest obecnie najpopularniejszy w środowisku analityków oraz porównywalnie wydajny do języka Julia który powoli zaczyna konkurować na rynku oraz długo na nim pozostanie.

Bibliografia

Arslan, A. (2019). Stat Base. Dostęp z https://docs.scipy.org/doc/scipy/reference/tutorial/general.html

Błaszczuk, D. (2006). *Wstęp do prognozowania i symulacji*. Wydawnictwo Naukowe PWN.

Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. (2018). Julia: A fresh approach to numerical computing.

Biecek, P. (2011). *Analiza danych z programem R. Modele liniowe z efektami stałymi, losowymi i mieszanymi*. Wydawnictwo Naukowe PWN.

Bizon, W. & Poszewicki, A. (2013). *Efektywność innowacyjnych narzędzi dydaktycznych w procesach kształtowania postaw przedsiębiorczych*. Wydawnictwo Uniwersytetu Gdańskiego.

Cormen, T. H., Leiserson, C. E., Rivest, R. & Stein, C. (2012). *Wprowadzenie do algorytmów*. Wydawnictwo Naukowe PWN.

Francis, A. & Golmant, N. (2017). Derivations of the Univariate and Multivariate Normal Density.

Gągolewski, M. (2014). *Programowanie w języku R*. Wydawnictwo Naukowe PWS SA.

Gajda, J. B. (2001). *Prognozowanie i symulacja a decyzje gospodarcze*. Wydawnictwo C.H. Beck.

Gajda, J. B. (2017). *Prognozowanie i symulacjie w ekonomii i zarządzaniu*. Wydawnictwo C.H. Beck.

Gordon, S. (2006). The Normal Distribution.

Hellwig, Z. (1998). *Elementy rachunku prawdopodobieństwa i statystyki matematycznej*. Wydawnictwo Naukowe PWN.

Janiga-Ćmiel, A. (2016). Zastosowanie metody symulacji dyskretnej do optymalizacji podejmowanych decyzji ekonomicznych.

Jones, E., Oliphant, T. & Peterson, P. (2001). SciPy: Open source scientific tools for Python. Dostęp z http://www.scipy.org/scipylib/index.html

Julia Data. (2019). Data Frame. Dostęp z https://juliadata.github.io/DataFrames.jl/stable/man/getting_started.html

Julia Team. (2019a). Linear Algebra. Dostęp z https : / / docs . julialang . org / en / v1 / stdlib / LinearAlgebra/index.html

Julia Team. (2019b). Statistics.

Kotulski, Z. (2001). Generatory liczb losowych: algorytmy, testowanie, zastosowania.

Larose, D. T. (2012). *Metody i modele eksploracji danych*. Wydawnictwo Naukowe PWN.

Maciąg, A., Pietroń, R. & Kukla, S. (2013). *Prognozowanie i symulacje w przedsiębiorstwie*. Polskie Wydawnictwo Ekonomiczne.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. W S. van der Walt & J. Millman (Red.), *Proceedings of the 9th Python in Science Conference* (s. 51–56).

Oliphant, T. (2006). NumPy: A guide to NumPy. USA: Trelgol Publishing. [Online; accessed <24.06.2019>].

Petersen, K. & Petersena, M. (2008). *The Matrix cookBook*. Technical University of Denmark.

PWN. (1999). *Encyklopedia powszechna PWN*.

Ribeiro, M. I. (2004). Gaussian Probability Density Functions: Properties and Error Characterization.

Robert, C. (2011). *James E. Gentle: Computational statistics (Statistics and Computing Series)*.

Sobol, I. (2017). *Metoda Monte Carlo*. Moskwa, Nauka Główna Redakcja Literatury Matematyczno-Fizycznej.

Welfe, A. (2009). *Ekonometria metody i ich zastosowanie*. Polskie Wydawnictwo Ekonomiczne.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H., François, R., Henry, L. & Müller, K. (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.6.

Witkowski, T. (2000). *Decyzje w zarządzaniu przedsiębiorstwem*. Wydawnictwo Naukowo Techniczne.

Ziętek-Kwaśniewska, K. (2006). Symulacje Monte Carlo jako metoda wyceny opcji.

Spis tabel

Spis rysunków

Spis Programów