„Przeszłość Demograficzna Polski" 37, 2015, nr 4
DOI: 10.18276/pdp.2015.4.37-07

Dariusz K. Chojecki
Szczecin

O programowaniu w środowisku R na przykładzie algorytmu obliczeniowego dla formuły do wykrywania kryzysów demograficznych*

Stary porządek demograficzny odznaczał się m.in. występowaniem (cyklicznym) kryzysów demograficznych, które były wywoływane – by użyć przenośni – przez czterech jeźdźców Apokalipsy, przede wszystkim głód i zarazy[1]. Samo pojęcie kryzysu demograficznego, w uproszczeniu oznaczające załamanie się dotychczasowych relacji między zmiennymi demograficznymi, ma jednak dość nieostry charakter, można nawet rzec: uznaniowy. Nie ulega jednak wątpliwości, że przed nastaniem transformacji demograficznej to przede wszystkim ruch zgonów był czułym miernikiem normalności i anormalności zmian, przy czym należy pamiętać, że jego rejestracja w dobie wysokiej umieralności ze względu na zaburzenia w ładzie społeczno-administracyjnym mogła być daleka od kompletności. Mimo tego w środowisku naukowym metoda „ostrego" typowania kryzysów demograficznych na podstawie rozwoju liczby zgonów, oparta na standaryzacyjnej formule zaproponowanej przez Jacques'a Dupâquiera, zdobyła sobie zwolenników[2]. Jej wada – jednowymiarowość – zważywszy na niedobory różnych danych demograficznych dla epoki przedstatystycznej, stała się zarazem jej zaletą. Mało tego – moim zdaniem – formuła ta, z uwagi na swą standaryzacyjną postać, nie musi bazować wyłącznie na liczbach zgonów, by wykrywać kryzysy

* Wyniki obliczeń zamieszczone są też na platformie wydawniczej (do pobrania jako pliki csv lub txt).

[1] Zob. Massimo Livi Bacci, *Europa und seine Menschen. Eine Bevölkerungsgeschichte*, aus dem Italienischen von Rita Seuß, München 1999, s. 56–120.

[2] Szerzej: Cezary Kuklo, *Demografia Rzeczypospolitej przedrozbiorowej*, Warszawa 2009, s. 249–257.

demograficzne. Może również „działać" na danych o urodzeniach (chrztach)[3] czy ślubach, co nawet jest bardziej zasadne, gdyż w tym wypadku czynnik wadliwej rejestracji w trudnych latach odgrywa o wiele mniejszą rolę[4].

Oczywiście zadaniem tego artykułu nie jest roztrząsanie istoty kryzysów demograficznych i sposobów ich pomiaru, a zapoznanie – w nawiązaniu do treści przedstawionych już w innym artykule, niezbędnych także do zrozumienia części zagadnień tu omawianych[5] – czytelnika z podstawami programowania w środowisku **R** z wykorzystaniem docelowo formuły Dupâquiera do ukazania działania pętli typu for[6]. I tym razem droga do celu odmierzana jest małymi krokami, a nawet na przykładach niezwiązanych bezpośrednio z rzeczoną formułą. Są one jednakże ważną wprawką do samodzielnego programowania i, co ważne, pozwalają zrozumieć zaprezentowane na końcu tego artykułu kody funkcji przygotowanych na bazie formuły Dupâquiera: *kryzysyDem1*, *kryzysyDem2*, *kryzysyDem3*.

Niezmodyfikowaną formułę Jacques'a Dupâquiera można zapisać następująco:

$$I_{\mathrm{k1}} = \frac{Z_{x+10} - \overline{Z}_{x-x+9}}{S(z)_{x-x+9}},$$

gdzie:

I_{k1} – wskaźnik intensywności kryzysu demograficznego dla jedenastego okresu (i następnych w analizowanym szeregu czasowym);

x – rok bądź indeks danych w szeregu czasowym;

Z_{x+10} – liczba zgonów w okresie jedenastym (i następnych w szeregu czasowym);

$\overline{Z}_{x-x+9}$ – średnia arytmetyczna liczby zgonów z dziesięciu okresów poprzedzających okres $x+10$;

---

[3] W tym wypadku należy jednak uwzględnić przesunięcie w czasie; wykrywanie kryzysu demograficznego następuje na podstawie poziomu zdarzeń zanotowanego po upływie pewnego czasu. Zob. też Piotr Miodunka, *Kryzysy demograficzne w Małopolsce w końcu XVII i pierwszej połowie XVIII wieku. Zarys problematyki*, „Przeszłość Demograficzna Polski" (dalej: PDP) 37, 2015, nr 4, s. 7–37.

[4] W dobie kryzysu demograficznego dla urodzeń (chrztów) i ślubów wartości wskaźnika będą przyjmować wartości ujemne, np. grubo poniżej –1, a w dobie koniunktury demograficznej – grubo powyżej 1.

[5] Dariusz K. Chojecki, *O programowaniu w środowisku R na przykładzie algorytmu obliczeniowego dla sezonowości zjawisk (metoda średnich jednoimiennych okresów)*, PDP 2014 (nr 35), s. 75–92. Zob. też: Radosław Poniat, *O wykorzystaniu wykresów pudełkowych do prezentacji danych demograficznych i o pożytku z użycia środowiska R z pakietem ggplot2*, PDP 2014 (nr 34), s. 103–120.

[6] Wprowadzenie: Przemysław Biecek, *Przewodnik po pakiecie R*, wydanie II rozszerzone, Wrocław 2011, s. 50–51; Marek Gągolewski, *Programowanie w języku R. Analiza danych, obliczenia, symulacje*, Warszawa 2014, s. 107–110.

$S(z)_{x-x+9}$ – nieobciążone odchylenie standardowe liczby zgonów dla dziesięciu okresów poprzedzających okres $x+10$.

Wskaźnik kryzysów demograficznych może teoretycznie przyjmować wartości od minus nieskończoności do plus nieskończoności, przy czym z reguły oscyluje pomiędzy –1 a 1 (lata normalne). Jeśli zawiera się w przedziale od 1 do mniej niż 2, to mówimy o lekkim kryzysie, jeśli od 2 do mniej niż 4 – o średnim, od 4 do mniej niż 8 – o silnym, od 8 do mniej niż 16 – o wielkim, od 16 do mniej niż 32 – o superkryzysie, a gdy wynosi 32 i więcej – o katastrofie demograficznej[7].

Wskaźnik ten ma jednakże pewien mankament: tylko wówczas dobrze pozwala ocenić natężenie zjawiska kryzysowego, gdy to ostatnie pierwszy raz występuje po dziesięciu „normalnych" latach. Problem pojawia się wraz z powtórnymi wystąpieniami lat kryzysowych w danym dziesięcioleciu lub z trwałym wzrostem (spadkiem) liczby zgonów wskutek procesów migracyjnych. Aby osłabić wpływ tych czynników, zaproponowałem badanie zjawisk kryzysowych z trzech perspektyw czasu, mianowicie przeszłej – co odpowiada niezmodyfikowanej formule Dupâquiera – w której płaszczyzną odniesienia jest rozwój zjawiska we wcześniejszym dziesięcioleciu; „teraźniejszej", w której płaszczyzną odniesienia jest rozwój zjawiska w pięcioleciu poprzedzającym obserwowany okres i w pięcioleciu następnym po nim (zob. wzór na I_{k2}); przyszłej, w której płaszczyzną odniesienia jest rozwój zjawiska w następnym dziesięcioleciu (zob. wzór na I_{k3}).

$$I_{k2} = \frac{Z_{x+10} - \overline{Z}_{(x+5-x+9),\ (x+11-x+15)}}{S(z)_{(x+5-x+9),\ (x+11-x+15)}}$$

$$I_{k3} = \frac{Z_{x+10} - \overline{Z}_{x+11-x+20}}{S(z)_{x+11-x+20}}$$

W przygotowanych funkcjach *kryzysyDem2*, *kryzysyDem3* poszczególne wartości wskaźników kryzysów demograficznych są średnią **arytmetyczną** z wyników uzyskanych dla trzech perspektyw czasowych, przy czym w wypadku funkcji *kryzysyDem3*, po określeniu granic poziomu kryzysu – wskaźników (np. wyniki, których wartość jest większa lub równa 2), wyświetlane są tylko te wartości, które we **wszystkich** perspektywach czasu spełniają wymagany warunek. Ten ostatni sposób przedstawiania wyników nazywam metodą „jednorękiego

[7] C. Kuklo, *Demografia* [2], s. 250.

bandyty”[8], gdyż zachodzi tu oczywiste skojarzenie ze znaną grą hazardową (zob. też opisy działania funkcji w końcowych partiach artykułu).

Szczegółowe wyniki działania funkcji zamieszczone są na końcu artykułu i odnoszą się do rozwoju liczby zgonów w Gdańsku w latach 1601–1846 oraz udzielonych ślubów na Pomorzu Zachodnim (Brandenburskim) w latach 1726–1805 (tabele 2 i 3). Analizując je, należy mieć na uwadze, że uzyskane oceny poziomu kryzysu na podstawie funkcji *kryzysyDem2* i *kryzysyDem3* pod względem wartości pokrywają się, z tym że dla drugiej z wymienionych funkcji, po wyborze określonego przedziału poziomu kryzysu (wskaźnika) – powtórzmy – są tylko i tylko wtedy wyświetlane wartości, gdy w każdej perspektywie czasowej osiągane są wyniki zawierające się w wybranym przedziale (zob. wykresy 2–4). Przyjęcie takiego rozwiązania sprawia, że funkcja *kryzysyDem3* zwraca mniej wyników niż funkcja *kryzysyDem2* (w szczególnym wypadku liczba uzyskanych wyników może być taka sama). Choć jest ich z reguły mniej, to jednak zyskują one swoją „legitymizację” właśnie dzięki temu rygorystycznemu rozwiązaniu, które – jak każde z wymienionych – nie jest wolne od wad. Dajmy na to, jeśli jest wybrany przedział 2–4 (średni kryzys dla ruchu zgonów) – *kryzysyDem3*(*ramka*, 2, 4) – to funkcja *kryzysyDem3* nie wyświetli wyników dla danego roku w sytuacji, gdy w jednej z perspektyw czasu – bez znaczenia, w której – rezultat obliczeń wyniesie mniej niż 2 lub będzie równy lub większy od 4. Oczywiście problem ten można w pewnej mierze obejść, ustawiając w argumentach funkcji otwarty przedział poziomu kryzysu, przykładowo na 2 i więcej: *kryzysyDem3*(*ramka*, 2,) – ale wyświetlone wyniki mogą „pochodzić” z bardzo niejednorodnego uśrednienia, jeśli uwzględnimy możliwość wystąpienia znacznych różnic między wartościami uzyskanymi dla trzech perspektyw czasu. Rezultaty obliczeń funkcji *kryzysyDem1* z reguły będą odznaczać się wyższymi wartościami, natomiast *kryzysyDem2* – niższymi. Po ustaleniu przedziału poziomu kryzysu (wskaźnika) może dojść do takiej sytuacji, że jedna funkcja wykaże oczekiwaną przez nas wartość dla danego roku, a druga nie, co oczywiście związane jest z uwzględnieniem różnych perspektyw czasu. I tak, słabością klasycznej formuły (tylko „przeszła” perspektywa), przypomnijmy, jest jej czułość na zjawisko kryzysowe występujące pierwszy raz po upływie „normalnych” dziesięciu lat. Jeżeli w następnych, bliskich sobie, acz niekoniecznie sąsiadujących ze sobą latach wartości badanego zjawiska są bardzo wysokie (czy bardzo niskie), to funkcja *kryzysyDem1* traci swoją wrażliwość. Inaczej rzecz wygląda w wypadku funkcji *kryzysyDem2*, ale ta z kolei nie jest aż tak czuła na wystąpienie zjawiska

[8] Por. Dariusz K. Chojecki, *Od społeczeństwa tradycyjnego do nowoczesnego. Demografia i zdrowotność głównych ośrodków miejskich Pomorza Zachodniego w dobie przyśpieszonej urbanizacji i industrializacji w Niemczech (1871–1913)*, Szczecin 2014, s. 190–192.

kryzysowego po raz pierwszy, a ponadto, z powodu uwzględnienia perspektywy „przyszłej", jej wyników nie można wyznaczyć nie tylko dla pierwszych dziesięciu lat w szeregu czasowym, lecz również dla ostatnich dziesięciu (zob. szare pola na wykresach 2–4). Tak czy inaczej rezultaty tych funkcji można rozpatrywać komplementarnie.

Krótkie wprowadzenie do programowania

Pętla typu for jest najczęściej wykorzystywana w obliczeniach w sytuacji, gdy można w miarę określić ich liczbę powtórzeń[9]. Jej składnia zawiera trzy elementy: na początku polecenie „for" inicjujące działanie według określonego schematu; następnie w nawiasie zwykłym słowo kluczowe „in" poprzedzone zmienną, nazywaną licznikiem czy iteratorem „i", która pobiera wartości indeksów z podanego po słowie kluczowym wektora[10] (indeksy z wektora określają liczbę powtórzeń – iteracji wykonania określonego działania); na końcu zaś w opcjonalnym nawiasie klamrowym instrukcję wykonywania, z wykorzystaniem zmiennej „i", określonego (powtarzalnego) działania na wskazanych danych.

for (zmienna in wektor) {instrukcja}

Rozpatrzmy kilka prostych przykładów, by nabrać wprawy w posługiwaniu się pętlą typu for. Jednym ze wskaźników dynamiki, często stosowanym w badaniach historycznych, jest prosty indeks łańcuchowy. Wyraża on stosunek wartości określonego zjawiska w okresie (momencie) badanym (y_t) do wartości w okresie (momencie) poprzedzającym okres badany (y_{t-1}). Wynik tej relacji mnożony jest z reguły przez wartość stałą (c), najczęściej 100. Wzór na prosty indeks łańcuchowy przyjmuje wtedy postać:

$$\frac{y_t}{y_{t-1}} \cdot c$$

Na podstawie poniższych informacji o liczbie zgonów w Gdańsku w latach 1601–1610 (tabela 1) dla dziesięcioelementowego szeregu danych można obliczyć dziewięć indeksów łańcuchowych dla lat 1602–1610. Pierwszy rok nie wchodzi w rachubę, gdyż nie znamy liczby zgonów z okresu wcześniejszego. A zatem

---

[9] W środowisku R istnieje wiele funkcji działających na pętli typu for. Na przykład runmean() z pakietu caTools (średnia ruchoma) czy cały szereg funkcji z rodziny apply. Por. Mikołaj Rybiński, *Krótkie wprowadzenie do R dla programistów, z elementami statystyki opisowej*, 22 lutego 2011, www.mimuw.edu.pl/~trybik/edu/rps/r-skrypt.pdf (14.11.2015). Składam podziękowania dla Pawła Chudziana za udzielone wskazówki.

[10] Może to być również lista lub inny obiekt zawierający zbiór informacji. W obliczeniach statystycznych najczęściej jest to jednak wektor.

nasze działanie sprowadzi się do wykonania dziewięciu działań, w których będą się tylko zmieniać dane:

$$\frac{y_{1602}}{y_{1601}} \cdot c \; \frac{y_{1603}}{y_{1602}} \cdot c \; \frac{y_{1604}}{y_{1603}} \cdot c \cdots \frac{y_{1610}}{y_{1609}} \cdot c \quad \text{czyli} \quad \frac{16919}{1361} \cdot 100 \; \frac{1531}{16919} \cdot 100 \cdots \frac{2495}{2782} \cdot 100 \; .$$

Tabela 1. Liczba zgonów w Gdańsku w latach 1601–1610

Rok (t)	Zgony (y)
1601	1361
1602	16919
1603	1531
1604	2030
1605	2255
1606	2903
1607	2170
1608	1970
1609	2782
1610	2495

Źródło: Jan Baszanowski, *Przemiany demograficzne w Gdańsku w latach 1601–1846 w świetle tabel ruchu naturalnego*, Gdańsk 1995, s. 348 (zob. też tabela 2 i wykres 1).

Nasza tabela (ramka danych) składa się *de facto* z dwóch wektorów o nazwie *Rok* i *Zgony*. Podczas obliczeń będą interesowały nas nie tylko wyniki, lecz również ich przyporządkowanie do konkretnych jednostek czasu. I od tego ostatniego zacznijmy. Jak już powiedziano, pierwszy rok nie wchodzi w rachubę. Stwórzmy najpierw wektor jednostek czasu. Żeby to uczynić, powinniśmy dysponować informacją o liczbie elementów wchodzących w skład wektora (wiemy, że jest ich 10). Do tego celu można wykorzystać funkcję length(). Ponadto przydatna będzie wiedza, że z użyciem znaku dwukropka jesteśmy w stanie utworzyć nowy wektor danych o określonej przez nas długości (liczbie elementów), początku i końcu:

```
> 1:10
[1]  1  2  3  4  5  6  7  8  9 10
> 4:8
[1] 4 5 6 7 8
```

Niech nasza ramka danych nazywa się *zgonyGdansk*, a jej pierwsza kolumna zawiera informacje o jednostkach czasu, tu: latach, podobnie jak w tabeli 1. Wyświetlmy je:

```
> zgonyGdansk[,1]
[1] 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610
```

Inaczej, bo z podaniem pozycji:

```
> zgonyGdansk[,1][1:10]
[1] 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610
```

Wynikiem jest wektor utworzony na podstawie danych z pierwszej kolumny ramki, które mają pozycję (indeks) od 1 do 10. Jeśli chcielibyśmy wybrać lata 1605–1608, mające pozycję 5–8, to wówczas zapis wyglądałby następująco:

```
> zgonyGdansk[,1][5:8]
[1] 1605 1606 1607 1608
```

Ażeby nasz zapis uwzględnił zmiany długości wektora (badanie może np. objąć 100 lat), należy wykorzystać w nim funkcję length(), podstawiwszy ją za ostatnią pozycję rozpatrywanego szeregu. Najpierw zobaczmy wynik działania samej funkcji:

```
> length(zgonyGdansk[,1])
[1] 10
```

A następnie ją podstawmy:

```
> zgonyGdansk[,1][1:length(zgonyGdansk[,1])]
[1] 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610
```

Nas jednak nie interesuje pierwszy rok. Zmodyfikujmy zatem zapis:

```
> zgonyGdansk[,1][2:length(zgonyGdansk[,1])]
[1] 1602 1603 1604 1605 1606 1607 1608 1609 1610
```

Zmienną *zgonyGdansk*, zawierającą kolumny z danymi o jednostce czasu i poziomie zjawiska, nazwijmy ogólnie *ramka*:

```
> ramka[,1][2:length(ramka[,1])]
```

Ten ogólny zapis wykorzystamy nieco dalej przy tworzeniu zwartej funkcji.

Pora przejść do spraw związanych z tworzeniem wektora wyników indeksu łańcuchowego z wykorzystaniem pętli typu for i funkcji length, z zaznaczeniem, że tym razem będą pobierane dane z drugiej kolumny ramki (Zgony):

```
> for (i in 1 : 9) {
+ zgonyGdansk[,2][i+1] / zgonyGdansk[,2][i]*100
+ }
>
```

Najpierw określamy liczbę obrotów pętli. Wiemy, że na przykładzie naszych danych mamy wykonać dziewięć obliczeń indeksów łańcuchowych i tyle też musi być powtórzeń – obrotów pętli. Licznik pętli – „i" – powinien zatem przyjąć kolejno wartości: 1, 2, 3, 4, 5, 6, 7, 8, 9. W pierwszym obrocie pętli w obliczeniach (instrukcja) jest wykorzystywana wartość „i" równa 1, w drugim – 2, w trzecim – 3 itd. aż do 9. Jak zatem będzie wyglądać działanie instrukcji, która jest transpozycją wzoru na indeks łańcuchowy? Weźmy przykładowo trzeci obrót pętli (i = 3) i dokonajmy odpowiedniego podstawienia:

```
zgonyGdansk[,2][3+1] / zgonyGdansk[,2][3]*100
Czyli:
zgonyGdansk[,2][4] / zgonyGdansk[,2][3]*100
```

W tym wypadku z drugiej kolumny ramki *zgonyGdansk* zostanie najpierw pobrana wartość o indeksie 4 (2030), następnie będzie ona podzielona przez wartość o indeksie 3 (1531), zaś wynik tego działania zostanie przemnożony razy 100 itd. W tym miejscu należy zaznaczyć, że wyświetlenie wyników wymaga jeszcze pewnej czynności, o czym będzie nieco dalej mowa.

Do utworzenia wektora, z którego pobierana jest liczba obrotów pętli, wykorzystaliśmy „sztywny" zapis (i in 1 : 9). A przecież nasza liczba kolejnych obliczeń, w zależności od długości szeregu czasowego, może przyjmować różne wartości. Aby to uwzględnić, liczbę 9 należy zastąpić funkcją, która zwróci długość „iteracyjnego" wektora, co pozwoli zautomatyzować proces obliczeń. Sięgnijmy znów po length():

```
> for (i in 1 : (length(zgonyGdansk[,1])-1)) {
+ zgonyGdansk[,2][i+1] / zgonyGdansk[,2][i]*100
+ }
>
```

Funkcja length(*zgonyGdansk*[,1]) zwróci wartość dziesięć. Ale w grę wchodzi dziewięć obliczeń, dlatego też jej wynik pomniejszamy o jeden: (length(*zgonyGdansk*[,1])-1), otrzymując dla analizowanego przykładu liczbę 9.

Podobnie jak w wypadku wektora jednostek czasu, tak i tu zastosujmy bardziej ogólny zapis tworzenia wektora wartości indeksów łańcuchowych:

```
> for (i in 1 : (length(ramka[,1])-1)) {
+ ramka[,2][i+1] / ramka[,2][i] * 100
+ }
>
```

Pozostaje już tylko wyświetlić wyniki obliczeń indeksów łańcuchowych. Innymi słowy, zwrócić je do nowo utworzonego wektora c(). Bez wchodzenia w szczegóły informatyczne ukażmy rzecz całą schematycznie:

```
> mojNowyWektor <- c()
> for (zmienna in wektor) {
+ mojNowyWektor[i] <- instrukcja
+ }
> mojNowyWektor
>
```

Powyższy schemat wykorzystamy w kolejnych przykładach – najpierw przy tworzeniu wektora wyników indeksu łańcuchowego, któremu nadajmy nazwę: *wynikiIndeksLancuchowy*:

```
> wynikiIndeksLancuchowy <- c()
> for (i in 1 : (length(zgonyGdansk[,1])-1)) {
+ wynikiIndeksLancuchowy[i] <- zgonyGdansk[,2][i+1] / zgonyGdansk[,2][i]*100
+ }
> wynikiIndeksLancuchowy
[1] 1243.130051 9.048998 132.593076 111.083744 128.736142 74.750258 90.783410
↑141.218274 89.683681
>
```

***Uwaga**: symbol „↑” w wierszu oznacza, że wiersz uległ zawinięciu w edytorze tekstu.*

Posiadając stosowną wiedzę, utwórzmy nową funkcję o nazwie *indeksLancuchowy*. W funkcji tej zastosujemy ogólny zapis danych: *ramka*. Ramkę – argument funkcji w wypadku naszych obliczeń, niech stanowi dwukolumnowa tabela z nagłówkami zawierająca jednostki czasu i przyporządkowane im poziomy zjawiska wyrażone w liczbach. Wewnątrz funkcji pierwszą kolumnę ramki nazwijmy *czas*, drugą – *zdarzenia*, by uprościć dalszy zapis.

```
> indeksLancuchowy <- function(ramka) {
+ czas <- ramka[,1]
+ zdarzenia <- ramka[,2]
+ jednostkiCzasuBezPierwszej <- czas[2:length(czas)]
+ wynikiIndeksLancuchowy <- c()
+ for (i in 1 : (length(czas)-1)) {
+ wynikiIndeksLancuchowy[i] <- zdarzenia[i+1] / zdarzenia[i]*100
+ }
+ wynikiIndeksLancuchowy
+ data.frame(jednostkiCzasuBezPierwszej, wynikiIndeksLancuchowy)
+ }
>
```

Jak widać, ostatnim działaniem wykonywanym przez naszą funkcję *indeksLancuchowy* jest wstawienie dwóch wektorów, tj. *jednostkiCzasuBezPierwszej* i *wynikiIndeksLancuchowy*, do jednej ramki za pomocą funkcji data.frame(**wektor1**, **wektor2**, ...).

Podstawmy w naszej nowej funkcji za argument *ramka* zmienną *zgonyGdansk*:

```
> indeksLancuchowy(zgonyGdansk)
  jednostkiCzasuBezPierwszej wynikiIndeksLancuchowy
1                       1602                1243.130051
2                       1603                   9.048998
3                       1604                 132.593076
4                       1605                 111.083744
5                       1606                 128.736142
6                       1607                  74.750258
7                       1608                  90.783410
8                       1609                 141.218274
9                       1610                  89.683681

>
```

Inną prostą miarą używaną w charakterystyce rozwoju zjawisk w czasie jest średnia ruchoma. Skupmy tu uwagę na jej najprostszej postaci, a mianowicie na średniej ruchomej trzyletniej, by następnie utworzyć uniwersalną funkcję na średnią ruchomą liczoną z okresów o nieparzystej liczbie jednostek czasu (trzyletnia, pięcioletnia, siedmioletnia itd.).

$$\frac{y_1+y_2+y_3}{3}\ \frac{y_2+y_3+y_4}{3}\ \frac{y_3+y_4+y_5}{3}\ldots\ \frac{y_{n-2}+y_{n-1}+y_n}{3}$$

Na podstawie danych o zgonach w Gdańsku w latach 1601–1610 zapis średniej trzyletniej przyjmie następującą postać:

$$\frac{y_{1601}+y_{1602}+y_{1603}}{3}\ \frac{y_{1602}+y_{1603}+y_{1604}}{3}\ \frac{y_{1603}+y_{1604}+y_{1605}}{3}\ldots\ \frac{y_{1608}+y_{1609}+y_{1610}}{3}$$

Czyli:

$$\frac{1361+16919+1531}{3}\ \frac{16919+1531+2030}{3}\ \frac{1531+2030+2255}{3}\ldots\ \frac{1970+2782+2495}{3}$$

Z powyższego wynika, że dla dziesięcioelementowego szeregu można obliczyć osiem średnich ruchomych, pierwszą na poziomie 1602 roku, a ostatnią – 1609 roku. Tyle też powinno być wziętych pod uwagę jednostek czasu. Zacznijmy od „sztywnego" rozwiązania generującego szereg ośmiu jednostek czasu, dla których zostaną przyporządkowane średnie ruchome:

```
> jednostkiCzasuBezSkrajnych3 <- c()
> for (i in 1 : 8) {
+ jednostkiCzasuBezSkrajnych3[i] <- zgonyGdansk[,1][i+1]
+ }
> jednostkiCzasuBezSkrajnych3
[1] 1602 1603 1604 1605 1606 1607 1608 1609
```

Aby zautomatyzować powyższe wyznaczanie szeregu jednostek czasu dla trzyletniej średniej ruchomej, wystarczy liczbę 8 zastąpić funkcją length, której argumentem będzie pierwsza kolumna ramki *zgonyGdansk*:

```
> jednostkiCzasuBezSkrajnych3 <- c()
> for (i in 1 : (length(zgonyGdansk[,1])-2)) {
+ jednostkiCzasuBezSkrajnych3[i] <- zgonyGdansk[,1][i+1]
+ }
> jednostkiCzasuBezSkrajnych3
[1] 1602 1603 1604 1605 1606 1607 1608 1609
```

Po utworzeniu wektora jednostek czasu utwórzmy wektor średnich ruchomych, odwołując się do drugiej kolumny ramki *zgonyGdansk*, wykorzystawszy także funkcję sumy, tj. sum():

```
> wynikiSredniaRuchoma3 <- c()
> for (i in 1: (length(zgonyGdansk[,1])-2)) {
+ wynikiSredniaRuchoma3[i] <- sum(zgonyGdansk[,2][i:(i+2)])/3
+ }
> wynikiSredniaRuchoma3
[1] 6603.667 6826.667 1938.667 2396.000 2442.667 2347.667 2307.333 2415.667
```

Po wyznaczeniu obu wektorów przejdźmy do utworzenia nowej funkcji. Nazwę konkretnej ramki (tu: *zgonyGdansk*) zastąpmy, tak jak uprzednio, ogólną nazwą: *ramka*, pamiętając, że zmienna ta będzie odnosić się do dwukolumnowej źródłowej tabeli z jednostkami czasu i przyporządkowanymi im poziomami zjawiska. Wewnątrz funkcji pierwszą kolumnę ramki nazwijmy *czas*, drugą – *zdarzenia*. Niech nowa funkcja nosi nazwę *sredniaRuchoma3*:

```
> sredniaRuchoma3 <- function(ramka) {
+ czas <- ramka[,1]
+ zdarzenia <- ramka[,2]
+ jednostkiCzasuBezSkrajnych3 <- c()
+ for (i in 1 : (length(czas)-2)) {
+ jednostkiCzasuBezSkrajnych3[i] <- czas[i+1]
+ }
+ jednostkiCzasuBezSkrajnych3
+ wynikiSredniaRuchoma3 <- c()
+ for (i in 1: (length(czas)-2)) {
+ wynikiSredniaRuchoma3[i] <- sum(zdarzenia[i:(i+2)])/3
+ }
+ wynikiSredniaRuchoma3
+ data.frame(jednostkiCzasuBezSkrajnych3, wynikiSredniaRuchoma3)
+ }
>
```

Podstawmy w naszej nowej funkcji za argument *ramka* zmienną *zgonyGdansk*:

```
> sredniaRuchoma3(zgonyGdansk)
  jednostkiCzasuBezSkrajnych3 wynikiSredniaRuchoma3
1                        1602                6603.667
2                        1603                6826.667
3                        1604                1938.667
4                        1605                2396.000
5                        1606                2442.667
6                        1607                2347.667
7                        1608                2307.333
8                        1609                2415.667
```

Przypomnijmy, że nasze działanie jest nakierowane na stworzenie uniwersalnej funkcji na średnią ruchomą liczoną z okresów o nieparzystej liczbie jednostek czasu (trzyletnia, pięcioletnia, siedmioletnia itd.). Aby uwzględnić ową uniwersalność, należy dodać drugi argument do nowo tworzonej funkcji celem określenia, z ilu wartości ma być liczona średnia ruchoma. Niech argument ten nosi nazwę *okres* i będzie liczbą nieparzystą (3, 5, 7 itd.). I tym razem musimy najpierw wyznaczyć wektor jednostek czasu, który w wypadku średniej trzyletniej nie będzie obejmował pierwszej i ostatniej wartości, w wypadku średniej pięcioletniej – dwóch pierwszych i dwóch ostatnich, w wypadku średniej siedmioletniej – trzech pierwszych i trzech ostatnich itd. Jak już powiedziano, nasza nowa funkcja będzie zawierać drugi argument *okres*. Uczyńmy tak, by na podstawie jego wartości można było wyznaczyć pozycję pierwszej wartości w wektorze branych pod uwagę jednostek czasu (zmienna: *czasStart*). Wiadomo, że w wypadku średniej trzyletniej będzie chodzić o indeks o wartości 2, w wypadku średniej pięcioletniej – 3, w wypadku średniej siedmioletniej – 4 itd. Wartości te uzyskamy, np. wykorzystując funkcję mediany (wartości środkowej), średniej arytmetycznej czy w jeszcze inny sposób:

```
    czasStart <- median(1:okres)
lub
    czasStart <- mean(1:okres)
lub
    czasStart <- 0.5*okres + 0.5
```

Wybierzmy pierwszy z przedstawionych wyżej wariantów. Następnie wyznaczmy, na której pozycji (indeksie) powinno się zakończyć tworzenie wektora jednostek czasu. I tu z pomocą przychodzi nam funkcja length(), którą wykorzystamy w połączeniu ze zmienną *czasStart*. Wiadomo, że w naszym konkretnym

przykładzie (*zgonyGdansk*) długość wektora jednostek czasu wynosi 10. Jeżeli od tej wartości – w wypadku obliczeń średniej trzyletniej – odejmiemy wartość zmiennej *czasStart* (2), to otrzymamy wynik równy 8. Ale dla tej średniej wektor jednostek czasu powinien kończyć się na indeksie 9. Dlatego też od wyniku funkcji length() należy odjąć wartość zmiennej *czasStart* pomniejszoną o 1, co przedstawia poniższy fragment kodu, tu: samodzielnej funkcji:

```
> jednostkiCzasuBezSkrajnych <- function(ramka, okres) {
+ czas <- ramka[,1]
+ czasStart <- median(1 : okres)
+ czas[czasStart : (length(czas)-(czasStart-1))]
+ }
>
```

Na podstawie analizowanego przykładu otrzymamy następujący wektor jednostek czasu dla średniej ruchomej trzyletniej:

```
> jednostkiCzasuBezSkrajnych(zgonyGdansk, 3)
[1] 1602 1603 1604 1605 1606 1607 1608 1609
```

dla średniej ruchomej pięcioletniej:

```
> jednostkiCzasuBezSkrajnych(zgonyGdansk, 5)
[1] 1603 1604 1605 1606 1607 1608
```

itd.

Zapoznanie się z powyższym powinno ułatwić zrozumienie działania funkcji obliczającej wartości średnich ruchomych w zależności od podanej zmiennej *okres*. Przejdźmy zatem do konkretnej funkcji (z pętlą for), która ma temu służyć. Niech przyjmie ona następujący zapis:

```
> wynikiSredniaRuchoma <- function(ramka, okres) {
+ zdarzenia <- ramka[,2]
+ wyniki <- c()
+ for (i in 1 : (length(zdarzenia) - (okres-1))) {
+ wyniki[i] <- sum(zdarzenia[i : ((i+okres)-1)]) / okres
+ }
+ wyniki
+ }
```

Dlaczego akurat taki? Najlepiej będzie rzecz całą wytłumaczyć na przykładzie obliczeń trzyletniej ruchomej średniej dla ramki – zmiennej *zgonyGdansk* (dla przypomnienia: dziesięć wartości zdarzeń) w pierwszym (i = 1) i ostatnim (i = 8) obrocie pętli. W funkcji *wynikiSredniaRuchoma* „iteracyjny" wektor został tak wyznaczony, by przyjmować długość równą liczbie powtarzających się obliczeń, zaczynając od 1. Jeżeli zatem argument *okres* – bo z uwzględnieniem niego wszystko jest tu przygotowywane – wyniesie 3, to „iteracyjny" wektor przyjmie postać 1 : 8, jeżeli wyniesie 5, to 1 : 6, jeżeli 7, to 1 : 4, jeżeli 9, to 1 : 2 (i to już koniec możliwości w dziesięcioelementowym zbiorze wyjściowych wartości). Wróćmy do średniej ruchomej trzyletniej. W pierwszym obrocie pętli zostanie podstawiona za „i" wartość 1. W części instrukcji związanej z poleceniem for poskutkuje to wykonaniem następującego działania: najpierw zostaną zsumowane wartości z wektora *zdarzenia* (druga kolumna ramki *zgonyGdańsk*) o pozycji od 1 do 3[11], czyli 1361 [1], 16919 [2] i 1531 [3], a następnie podzielone przez zmienną *okres*, tu: 3. Natomiast w ostatnim obrocie pętli zostaną zsumowane wartości z rzeczonego wektora o pozycji od 8 do 10[12]. W tej sytuacji będą dodane do siebie wartości 1970 [8], 2782 [9] i 2495 [10], które, jak uprzednio, zostaną podzielone przez 3. Odwołanie się w algorytmie obliczeń do funkcji length() i argumentu *okres* (a także pomniejszenie jego wartości o jeden) ma kluczowe znaczenie dla automatyzacji obliczeń, tzn. wyznaczania określonej liczby wartości średniej ruchomej w zależności od liczby elementów wchodzących w skład wektora danych wyjściowych i rodzaju średniej. Poznawszy poszczególne elementy nowo utworzonych dwuargumentowych funkcji: *jednostkiCzasuBezSkrajnych*() i *wynikiSredniaRuchoma*(), użyjmy je w jednej funkcji, tj. *sredniaRuchoma*, która również będzie zawierać te same argumenty, co używane wyżej.

```
> sredniaRuchoma <- function(ramka, okres) {
+ czas <- ramka[,1]
+ zdarzenia <- ramka[,2]
+ czasStart <- median(1 : okres)
+ jednostkiCzasuBezSkrajnych <- czas[czasStart : (length(czas)-(czasStart-1))]
+ wynikiSredniaRuchoma <- c()
+ for (i in 1 : (length(zdarzenia) - (okres-1))) {
+ wynikiSredniaRuchoma[i] <- sum(zdarzenia[i : ((i+okres)-1)]) / okres
+ }
+ wynikiSredniaRuchoma
+ data.frame(jednostkiCzasuBezSkrajnych, wynikiSredniaRuchoma)
+ }
>
```

[11] Liczba 3 jest tu wynikiem działania: (1 + 3) – 1.

[12] Liczba 10 jest tu wynikiem działania: (8 + 3) – 1.

Przetestujmy działanie nowej funkcji dla średniej dziewięcioletniej ruchomej:

```
> sredniaRuchoma(zgonyGdansk, 9)
  jednostkiCzasuBezSkrajnych wynikiSredniaRuchoma
1                       1605                3769
2                       1606                3895
```

Pozostaje już tylko „uszlachetnić" funkcję, dodając warunek, który ograniczy jej działanie. W nowo utworzonej funkcji *sredniaRuchoma* argument *okres* powinien być dodatnią liczbą nieparzystą. Mimo to użytkownik może podstawić za ten argument jakąś inną liczbę, np. ujemną lub parzystą czy zmiennoprzecinkową, co spowoduje wypaczenie wyników. W pierwszym wypadku wystarczy zastosować gotową funkcję abs(), zwracającą wartość bezwzględną danej liczby, w drugim można posłużyć się instrukcją warunkową typu if … else …, wykorzystując w jej teście jako „detektor" nieparzystości tzw. dzielenie modulo. To ostatnie jest resztą z dzielenia przez siebie dwóch wartości (operator matematyczny %%):

```
> 5 %% 2
[1] 1
> 4 %% 2
[1] 0
```

Wykorzystajmy zatem fakt, że podzielenie jakiejkolwiek liczby nieparzystej przez dwa daje zawsze resztę o wartości 1, ponadto ustawmy dla argumentu *okres* wartość domyślną równą 3, zaś komunikat tekstowy o błędzie wyświetlmy bez cudzysłowu, używając do tego celu funkcji cat():

```
> sredniaRuchoma <- function(ramka, okres = 3) {
+ if (okres %% 2 == 1) {
+ czas <- ramka[,1]
+ zdarzenia <- ramka[,2]
+ okresBezwzgl <- abs(okres)
+ czasStart <- median(1 : okresBezwzgl)
+ jednostkiCzasuBezSkrajnych <- czas[czasStart : (length(czas)-(czasStart-1))]
+ wynikiSredniaRuchoma <- c()
+ for (i in 1 : (length(zdarzenia) - (okresBezwzgl - 1))) {
+ wynikiSredniaRuchoma[i] <- sum(zdarzenia[i:((i+okresBezwzgl)-1)]) / okresBezwzgl
+ }
+ wynikiSredniaRuchoma
+ data.frame(jednostkiCzasuBezSkrajnych, wynikiSredniaRuchoma)
+ } else {
+ cat("Drugi argument funkcji – tylko dodatnia liczba nieparzysta.")
+ }
+ }
>
```

Na szczególną uwagę zasługuje jeszcze jedna rzecz, mianowicie działanie pętli na różnych fragmentach tego samego wektora. Załóżmy, że chcemy obliczyć ruchomą średnią arytmetyczną z czterech wartości, które w pierwszym obrocie pętli mają następujące pozycje: 1, 2 oraz 4, 5; w drugim – 2, 3 oraz 5, 6; w trzecim – 3, 4 oraz 6, 7 itd. Jak widać, naszym zadaniem jest pominięcie w obliczeniach określonej wartości w wektorze, co w tym wypadku oznacza odwoływanie się do dwóch fragmentów tego samego wektora. Na podstawie przykładowych danych dla Gdańska stwórzmy wektor o nazwie *zgony* i wykonajmy na nim opisane działanie:

```
> zgony <- zgonyGdansk[,2]
> zgony
 [1]  1361 16919  1531  2030  2255  2903  2170  1970  2782  2495
> sredniaZfragmentow <- c()
> for (i in 1 : (length(zgony)-4)) {
+ sredniaZfragmentow[i] <- mean(c(zgony[i:(i+1)], zgony[(i+3):(i+4)]))
+ }
> sredniaZfragmentow
[1] 5641.25 5902.00 2158.50 2106.25 2477.50 2587.50
>
```

W tym wypadku najważniejszy jest dla nas fragment kodu:

```
c(zgony[i:(i+1)], zgony[(i+3):(i+4)])
```

w którym wykonywane jest działanie na różnych częściach tego samego wektora *zgony*. W celu wykonania stosownych obliczeń części te zostały połączone w jeden nowy wektor za pomocą funkcji c(), co można schematycznie zapisać:

```
> nowyWektor <- c(wektorFragment1, wektorFragment2, ...)
```

Innymi słowy, obliczenia średniej arytmetycznej wykonywane są na wartościach „jednolitego" wektora.

Aby móc zrozumieć fragmenty przedstawionych dalej kodów, niezbędne jest jeszcze posługiwanie się funkcją nieobciążonego odchylenia standardowego, tj. sd():

```
> sd(zgony)
[1] 4690.7
>
```

oraz funkcją służącą do zaokrągleń wyników: round(*zaokraglane*, *zaokrąglenie*)

```
> round(2.53467, 2)
[1] 2.53
>
```

której argumentem może być wektor z liczbami zmiennoprzecinkowymi (uwaga: w środowisku **R** przecinek ma funkcję separatora).

Dla przypomnienia, opis innych nieomówionych wyżej funkcji i instrukcji, znajdzie czytelnik w pierwszym artykule poświęconym programowaniu w środowisku **R**[13].

[13] D.K. Chojecki, *O programowaniu w środowisku R* [5].

Opis kodu programowego dla formuły Jacques'a Dupâquiera (*kryzysyDem1*, *kryzysyDem2*, *kryzysyDem3* – zob. końcową część artykułu po komentarzach)

Ogólne informacje o przygotowanej funkcji:
A.

> *kryzysyDem1*(*nazwa ramki*, *granica dolna stopnia kryzysu*, *granica górna stopnia kryzysu*)

1. (Punkty odnoszą się do wierszy kodu programowego)

Funkcja oblicza wskaźniki kryzysu dla poszczególnych lat według niezmodyfikowanej formuły. Działa na dwukolumnowej ramce posiadającej nagłówki (zobacz ramkę *zgonyGdansk*). W pierwszej kolumnie rzeczonej ramki muszą znajdować się dane o okresach, w drugiej – o poziomie zjawiska, czyli zdarzeniach. Nagłówki kolumn mogą mieć dowolne nazwy. Wymagany jest liczbowy typ danych dla drugiej kolumny ramki.
Funkcja posiada trzy argumenty, przy czym podstawienie danych do pierwszego jest konieczne (nazwa ramki), do pozostałych zaś nie. Drugi i trzeci argument funkcji, tj. dolna i górna granica stopnia kryzysu, są ustawione domyślnie na minus nieskończoność (-Inf) i plus nieskończoność (Inf), w związku z czym w wypadku niepodania dla nich wartości wyświetlane są wszystkie wyniki obliczeń.
Przykłady wprowadzania danych do funkcji:

```
> kryzysyDem1(mojaRamka)
> kryzysyDem1(mojaRamka, )
> kryzysyDem1(mojaRamka, , )
> kryzysyDem1(mojaRamka, 4, )
> kryzysyDem1(mojaRamka, , 2)
> kryzysyDem1(mojaRamka, -1, 1)
```

UWAGA: dany wektor ramki musi mieć długość równą przynajmniej 11. Ramka nie może zawierać niekompletnych danych.

2–3

Utworzenie wektorów *czas* i *zdarzenia* na podstawie odpowiednich kolumn ramki.

4

Wyznaczenie ostatniej pozycji wektora „iteracyjnego”.

5–9

Utworzenie wektora jednostek czasu (pierwsza kolumna zwracanej przez funkcję ramki).

10–14

Utworzenie wektora danych poziomu zjawiska – zdarzeń (we wzorze: Z_{x+10}).

15–19

Utworzenie wektora z wynikami zwykłej dziesięcioletniej ruchomej średniej arytmetycznej dla zdarzeń (we wzorze: $\overline{Z}_{x-x+9}$).

20–24

Utworzenie wektora z wynikami dziesięcioletniego ruchomego odchylenia standardowego dla zdarzeń (we wzorze: $S(z)_{x-x+9}$).

25

Utworzenie wektora z wynikami otrzymanymi na podstawie formuły Dupâquiera: $I_{k1} = \frac{Z_{x+10} - \overline{Z}_{x-x+9}}{S(z)_{x-x+9}}$.

26

Utworzenie wektora z wynikami wyświetlanymi w zależności od podanych wartości granic w drugim i trzecim argumencie funkcji oraz zaokrąglenie otrzymanych wyników do trzech miejsc po przecinku.
UWAGA: chcąc zaprezentować wyniki na wykresie – np. funkcją plot() – w wypadku, gdy nie są one ukazywane dla każdego okresu, co wynika z określonego wyboru granic, w ostatnim elemencie funkcji ifelse() należy zamienić wartość pustą"" (dwa cudzysłowy) na NA (ang. *not available*).

27

Utworzenie wynikowej ramki zawierającej wektor jednostek czasu (pierwsza kolumna) i przyporządkowane im wyniki otrzymane na podstawie formuły Dupâquiera.

28

Zamknięcie ciała funkcji *kryzysyDem1*.

B.

> *kryzysyDem2*(*nazwa ramki*, *granica dolna stopnia kryzysu*, *granica górna stopnia kryzysu*)

1

Funkcja oblicza wskaźniki kryzysu dla poszczególnych okresów według zmodyfikowanej formuły, tzn. uwzględnia trzy perspektywy czasu: przeszłą, „teraźniejszą" i przyszłą. Końcowe wyniki są średnią arytmetyczną z wyników uzyskanych dla trzech perspektyw.

Reszta jak w wypadku funkcji *kryzysyDem1*().

2–3

Utworzenie wektorów *czas* i *zdarzenia* na podstawie odpowiednich kolumn ramki.

4

Wyznaczenie długości wektora *zdarzenia*.

5–9

Utworzenie wektora jednostek czasu (pierwsza kolumna zwracanej przez funkcję ramki).

10–14

Utworzenie wektora danych poziomu zjawiska – zdarzeń (we wzorze: Z_{x+10}).

15–19

Perspektywa przeszła. Utworzenie wektora z wynikami zwykłej dziesięcioletniej ruchomej średniej arytmetycznej dla zdarzeń (we wzorze: $\overline{Z}_{x-x+9}$).

20–24

Perspektywa przeszła. Utworzenie wektora z wynikami dziesięcioletniego ruchomego odchylenia standardowego dla zdarzeń (we wzorze: $S(z)_{x-x+9}$).

25–29

Perspektywa „teraźniejsza". Utworzenie wektora z wynikami zwykłej dziesięcioletniej ruchomej średniej arytmetycznej dla zdarzeń (we wzorze: $\overline{Z}_{(x+5-x+9),\ (x+11-x+15)}$).

30–34

Perspektywa „teraźniejsza". Utworzenie wektora z wynikami dziesięcioletniego ruchomego odchylenia standardowego dla zdarzeń (we wzorze: $S(z)_{(x+5-x+9),\ (x+11-x+15)}$).

35–39

Perspektywa przyszła. Utworzenie wektora z wynikami zwykłej dziesięcioletniej ruchomej średniej arytmetycznej dla zdarzeń (we wzorze: $\overline{Z}_{x+11-x+20}$).

40–44

Perspektywa przyszła. Utworzenie wektora z wynikami dziesięcioletniego ruchomego odchylenia standardowego dla zdarzeń (we wzorze: $S(z)_{x+11-x+20}$).

45

Perspektywa przeszła. Utworzenie wektora z wynikami otrzymanymi na podstawie formuły Dupâquiera: $I_{k1} = \dfrac{Z_{x+10} - \overline{Z}_{x-x+9}}{S(z)_{x-x+9}}$.

46

Perspektywa „teraźniejsza". Utworzenie wektora z wynikami otrzymanymi na podstawie formuły: $I_{k2} = \dfrac{Z_{x+10} - \overline{Z}_{(x+5-x+9),\ (x+11-x+15)}}{S(z)_{(x+5-x+9),\ (x+11-x+15)}}$.

47

Perspektywa przyszła. Utworzenie wektora z wynikami otrzymanymi na podstawie formuły: $I_{k3} = \frac{Z_{x+10} - \overline{Z}_{x+11-x+20}}{S(z)_{x+11-x+20}}$.

48–52

Utworzenie wektora z uśrednionymi wynikami $\overline{I}_k = \frac{I_1 + I_2 + I_3}{3}$.

53–54

Utworzenie wektora z uśrednionymi wynikami wyświetlanymi w zależności od podanych wartości granic w drugim i trzecim argumencie funkcji oraz zaokrąglenie otrzymanych wyników do trzech miejsc po przecinku.
UWAGA: chcąc zaprezentować wyniki na wykresie – np. funkcją plot() – w wypadku, gdy nie są one ukazywane dla każdego okresu, co wynika z określonego wyboru granic, w ostatnim elemencie funkcji ifelse() należy zamienić wartość pustą"" (dwa cudzysłowy) na NA (ang. *not available*).

55

Utworzenie wynikowej ramki zawierającej wektor jednostek czasu (pierwsza kolumna) i przyporządkowane im wyniki otrzymane na podstawie formuły uwzględniającej trzy perspektywy czasu (wartości uśrednione).

56

Zamknięcie ciała funkcji *kryzysyDem2*.

C.

> *kryzysyDem3*(*nazwa ramki, granica dolna stopnia kryzysu, granica górna stopnia kryzysu*)

1

Funkcja oblicza wskaźniki kryzysu dla poszczególnych okresów według zmodyfikowanej formuły, tzn. uwzględnia trzy perspektywy czasu: przeszłą, „teraźniejszą" i przyszłą. Końcowe wyniki są średnią arytmetyczną z wyników uzyskanych dla trzech perspektyw, ale ich wyświetlenie następuje tylko wtedy, gdy

w każdej perspektywie czasu uzyskane wyniki spełniają określone warunki, tj. kryteria graniczne podane w drugim lub trzecim argumencie funkcji.
Reszta jak w wypadku funkcji *kryzysyDem1*.

2–52

Tak samo jak w funkcji: *kryzysyDem2*.

53–55

Utworzenie wektora z uśrednionymi wynikami – wyświetlanymi w zależności od podanych wartości granic w drugim i trzecim argumencie funkcji, ale tylko wtedy, gdy **w każdej** perspektywie czasu wyniki lokują się w tym samym przedziale podanych wartości granic – oraz zaokrąglenie otrzymanych wyników do trzech miejsc po przecinku.
UWAGA: chcąc zaprezentować wyniki na wykresie – np. funkcją plot() – w wypadku, gdy nie są one ukazywane dla każdego okresu, co wynika z określonego wyboru granic, w ostatnim elemencie funkcji ifelse() należy zamienić wartość pustą "" (dwa cudzysłowy) na NA (ang. *not available*).

56

Utworzenie wynikowej ramki zawierającej wektor jednostek czasu (pierwsza kolumna) i przyporządkowane im wyniki otrzymane na podstawie formuły uwzględniającej trzy perspektywy czasu (wartości uśrednione).

57

Zamknięcie ciała funkcji *kryzysyDem3*.

```
> kryzysyDem1 <- function(ramka, stopienGrDolna = -Inf, stopienGrGorna = Inf) {
+ czas <- ramka[,1]
+ zdarzenia <- ramka[,2]
+ koniecWektoraIter <- length(zdarzenia)-10
+ jednostkiCzasuBez10pierwszych <- c()
+ for (i in 1 : koniecWektoraIter) {
+ jednostkiCzasuBez10pierwszych[i] <- czas[i+10]
+ }
+ jednostkiCzasuBez10pierwszych
+ daneRok11iNastepneLata <- c()
+ for (i in 1 : koniecWektoraIter) {
+ daneRok11iNastepneLata[i] <- zdarzenia[i+10]
+ }
+ daneRok11iNastepneLata
+ wynikiSredniejAryt <- c()
+ for (i in 1 : koniecWektoraIter) {
+ wynikiSredniejAryt[i] <- mean(zdarzenia[i:(i+9)])
+ }
+ wynikiSredniejAryt
+ wynikiOdchyleniaStand <- c()
+ for (i in 1 : koniecWektoraIter) {
+ wynikiOdchyleniaStand[i] <- sd(zdarzenia[i:(i+9)])
+ }
+ wynikiOdchyleniaStand
+ wynikiDane <- (daneRok11iNastepneLata - wynikiSredniejAryt) / wynikiOdchyleniaStand
+ wynikiWybor <- ifelse(wynikiDane >= stopienGrDolna & wynikiDane < stopienGrGorna, round(wynikiDane,3),"")
+ data.frame(jednostkiCzasuBez10pierwszych, wynikiWybor)
+ }
>
```

```
1)  > kryzysyDem2 <- function(ramka, stopienGrDolna = -Inf, stopienGrGorna = Inf) {
2)  + czas <- ramka[,1]
3)  + zdarzenia <- ramka[,2]
4)  + dlugoscWektoraZdarzenia <- length(zdarzenia)
5)  + jednostkiCzasuBez10pierwszychI10ostatnich <- c()
6)  + for (i in 1 : (dlugoscWektoraZdarzenia-20)) {
7)  + jednostkiCzasuBez10pierwszychI10ostatnich[i] <- czas[i+10]
8)  + }
9)  + jednostkiCzasuBez10pierwszychI10ostatnich
10) + daneRok11iNastepneLataBez10ostatnich <- c()
11) + for (i in 1 : (dlugoscWektoraZdarzenia-20)) {
12) + daneRok11iNastepneLataBez10ostatnich[i] <- zdarzenia[i+10]
13) + }
14) + daneRok11iNastepneLataBez10ostatnich
15) + wynikiSredniejAryt1 <- c()
16) + for (i in 1 : (dlugoscWektoraZdarzenia-20)) {
17) + wynikiSredniejAryt1[i] <- mean(zdarzenia[i:(i+9)])
18) + }
19) + wynikiSredniejAryt1
20) + wynikiOdchyleniaStand1 <- c()
21) + for (i in 1 : (dlugoscWektoraZdarzenia-20)) {
22) + wynikiOdchyleniaStand1[i] <- sd(zdarzenia[i:(i+9)])
23) + }
24) + wynikiOdchyleniaStand1
25) + wynikiSredniejAryt2 <- c()
26) + for (i in 1 : (dlugoscWektoraZdarzenia-15)) {
```

```
27) + wynikiSredniejAryt2[i-5] <- mean(c(zdarzenia[i:(i+4)], zdarzenia[(i+6):(i+10)]))
28) + }
29) + wynikiSredniejAryt2
30) + wynikiOdchyleniaStand2 <- c()
31) + for (i in 1 : (dlugoscWektoraZdarzenia-15)) {
32) + wynikiOdchyleniaStand2[i-5] <- sd(c(zdarzenia[i:(i+4)],zdarzenia[(i+6):(i+10)]))
33) + }
34) + wynikiOdchyleniaStand2
35) + wynikiSredniejAryt3 <- c()
36) + for (i in 1 : (dlugoscWektoraZdarzenia-9)) {
37) + wynikiSredniejAryt3[i-11] <- mean(zdarzenia[i:(i+9)])
38) + }
39) + wynikiSredniejAryt3
40) + wynikiOdchyleniaStand3 <- c()
41) + for (i in 1 : (dlugoscWektoraZdarzenia-9)) {
42) + wynikiOdchyleniaStand3[i-11] <- sd(zdarzenia[i:(i+9)])
43) + }
44) + wynikiOdchyleniaStand3
45) + wynikiDane1 <- (daneRok11iNastepneLataBez10ostatnich - wynikiSredniejAryt1) / wynikiOdchyleniaStand1
46) + wynikiDane2 <- (daneRok11iNastepneLataBez10ostatnich - wynikiSredniejAryt2) / wynikiOdchyleniaStand2
47) + wynikiDane3 <- (daneRok11iNastepneLataBez10ostatnich - wynikiSredniejAryt3) / wynikiOdchyleniaStand3
48) + wynikiDane123SredniaAryt <- c()
49) + for (i in 1 : length(daneRok11iNastepneLataBez10ostatnich)) {
50) + wynikiDane123SredniaAryt[i] <- mean(c(wynikiDane1[i], wynikiDane2[i], wynikiDane3[i]))
51) + }
52) + wynikiDane123SredniaAryt
```

```
53) + wynikiWybor <- ifelse(wynikiDane123SredniaAryt >= stopienGrDolna & wynikiDane123SredniaAryt < stopienGrGorna,
54)   ↑round(wynikiDane123SredniaAryt, 3), " ")
55) + data.frame(jednostkiCzasuBez10pierwszychI10ostatnich, wynikiWybor)
56) + }
57) >

1)  > kryzysyDem3 <- function(ramka, stopienGrDolna = -Inf, stopienGrGorna = Inf) {
    od punktu 2 do 52 jak powyższym kodzie
53) + wynikiWybor <- ifelse(wynikiDane1 >= stopienGrDolna & wynikiDane1 < stopienGrGorna &
54) + wynikiDane2 >= stopienGrDolna & wynikiDane2 < stopienGrGorna &
55) + wynikiDane3 >= stopienGrDolna & wynikiDane3 < stopienGrGorna, round(wynikiDane123SredniaAryt, 3)," ")
56) + data.frame(jednostkiCzasuBez10pierwszychI10ostatnich, wynikiWybor)
57) + }
58) >
```

Tabela 2. Liczba zgonów w Gdańsku w latach 1601–1846 i wyniki formuł do wykrywania kryzysów demograficznych

Rok	Liczba zgonów	W1	W2/W3	Rok	Liczba zgonów	W1	W2/W3
1	2	3	4	5	6	7	8
1601	1361	•	•	1724	1872	0,941	0,980
1602	16919	•	•	1725	1678	0,628	0,307
1603	1531	•	•	1726	1548	–0,230	–0,353
1604	2030	•	•	1727	1617	0,160	–0,133
1605	2255	•	•	1728	1415	–1,369	–1,365
1606	2903	•	•	1729	1842	1,804	0,407
1607	2170	•	•	1730	1595	0,009	–0,319
1608	1970	•	•	1731	1713	0,731	–0,044
1609	2782	•	•	1732	1605	–0,107	–0,429
1610	2495	•	•	1733	1599	–0,272	–0,489
1611	2316	–0,283	–0,491	1734	5843	30,956	**13,707**
1612	2846	–0,192	0,030	1735	1799	–0,184	–0,287
1613	3080	1,718	0,901	1736	2474	0,312	0,368
1614	3297	2,059	1,264	1737	3944	1,350	3,411
1615	2198	–0,953	–0,687	1738	2104	–0,196	0,423
1616	2671	0,149	–0,208	1739	1732	–0,518	–0,266
1617	3123	1,264	0,258	1740	1640	–0,574	–0,440
1618	2378	–0,694	–0,559	1741	1966	–0,344	0,109
1619	2181	–1,442	–0,913	1742	1659	–0,588	–0,428
1620	11936	22,731	9,741	1743	1266	–0,879	–1,605
1621	2165	–0,486	–0,608	1744	1164	–0,911	–2,548
1622	2452	–0,383	–0,497	1745	1855	–0,152	0,074
1623	2908	–0,215	–0,306	1746	1737	–0,309	–0,420
1624	10535	2,350	3,583	1747	1917	0,013	0,074
1625	4197	–0,016	0,355	1748	1743	0,132	–0,273
1626	2253	–0,605	–0,526	1749	1906	0,910	0,257
1627	3278	–0,310	0,038	1750	2040	1,304	0,648
1628	2607	–0,498	–0,268	1751	2223	1,697	1,014
1629	4285	–0,046	0,327	1752	1846	0,291	–0,363
1630	5039	0,106	1,273	1753	1752	–0,054	–0,646
1631	1879	–0,836	–0,625	1754	1574	–0,891	–1,241
1632	1660	–0,904	–0,758	1755	2095	1,308	0,270
1633	1352	–0,970	–0,932	1756	2475	3,036	1,397
1634	2966	–0,275	–0,023	1757	2599	2,453	1,343
1635	1730	–0,987	–0,716	1758	3311	3,896	**3,509**
1636	1596	–0,919	–0,748	1759	1983	–0,393	–0,348
1637	1984	–0,524	–0,479	1760	2029	–0,320	–0,247
1638	2142	–0,295	–0,358	1761	2084	–0,208	–0,104
1639	7466	4,004	**11,438**	1762	2896	1,432	2,589
1640	2493	–0,147	–0,122	1763	1888	–0,731	–0,565

1	2	3	4	5	6	7	8
1641	2371	–0,087	–0,294	1764	1683	–1,168	–1,398
1642	2118	–0,257	–0,692	1765	1730	–1,134	–1,302
1643	2122	–0,283	–0,448	1766	1854	–0,772	–0,879
1644	2563	–0,079	–0,238	1767	2082	–0,227	–0,118
1645	2244	–0,241	–0,600	1768	2188	0,064	0,618
1646	2927	0,128	0,482	1769	1882	–0,469	–0,583
1647	2685	–0,096	–0,159	1770	1870	–0,470	–0,666
1648	2563	–0,216	–0,354	1771	2111	0,274	0,215
1649	2878	–0,048	–0,218	1772	2279	0,748	0,852
1650	2883	1,350	0,225	1773	1972	0,077	–0,234
1651	2452	–0,268	–0,421	1774	1996	0,158	0,001
1652	3666	3,649	1,142	1775	2390	2,325	1,867
1653	11616	20,573	**10,019**	1776	2455	2,156	2,367
1654	2243	–0,497	–0,466	1777	2185	0,307	1,120
1655	2301	–0,463	–0,456	1778	1901	–1,137	–0,267
1656	3180	–0,156	–0,075	1779	1650	–2,111	–1,380
1657	7569	1,385	2,374	1780	1614	–1,858	–1,330
1658	1796	–0,764	–0,554	1781	1750	–1,068	–0,566
1659	2305	–0,563	–0,209	1782	2092	0,242	1,155
1660	5515	0,482	4,600	1783	1837	–0,568	0,076
1661	1988	–0,723	–0,039	1784	1683	–1,039	–0,597
1662	1575	–0,831	–0,545	1785	1633	–1,048	–0,771
1663	1361	–0,805	–0,837	1786	1568	–1,110	–0,966
1664	1305	–0,840	–1,016	1787	1938	0,696	0,836
1665	1740	–0,558	–0,306	1788	2030	1,551	1,670
1666	1788	–0,502	–0,117	1789	2138	1,934	3,151
1667	1952	–0,353	0,340	1790	1602	–1,077	–0,359
1668	1461	–0,548	–0,760	1791	1551	–1,305	–0,603
1669	1677	–0,340	–0,336	1792	1571	–1,035	–0,396
1670	2371	0,269	1,621	1793	1398	–1,660	–1,343
1671	1474	–0,765	–1,057	1794	1609	–0,426	–0,028
1672	1430	–0,756	–1,250	1795	1806	0,422	1,800
1673	1700	0,135	–0,324	1796	1491	–0,949	–0,655
1674	1702	0,039	–0,471	1797	1544	–0,680	–0,295
1675	2052	1,160	0,599	1798	1385	–1,201	–1,046
1676	2647	2,991	**3,123**	1799	1511	–0,447	–0,469
1677	1813	–0,082	–0,175	1800	1557	0,086	–0,227
1678	1783	–0,122	–0,287	1801	1448	–0,802	–0,771
1679	2265	1,036	1,241	1802	1470	–0,512	–0,562
1680	2005	0,204	0,301	1803	1571	0,403	–0,297
1681	1982	0,257	0,190	1804	1853	2,747	0,572
1682	1751	–0,550	–0,505	1805	1926	2,401	0,505
1683	1482	–1,631	–1,571	1806	1985	2,337	0,519
1684	2149	0,613	0,498	1807	7207	26,082	12,377

1	2	3	4	5	6	7	8
1685	1611	–1,191	–1,193	1808	3114	0,520	0,420
1686	1637	–0,913	–1,029	1809	2495	0,074	0,037
1687	1712	–0,546	–0,752	1810	1802	–0,378	–0,370
1688	1913	0,298	–0,302	1811	2101	–0,223	–0,154
1689	2259	1,617	0,518	1812	3465	0,536	0,736
1690	2117	1,062	0,163	1813	5592	1,692	7,475
1691	2728	3,316	1,695	1814	1807	–0,733	1,426
1692	3362	3,758	**3,063**	1815	1025	–1,153	–0,811
1693	3100	1,727	2,597	1816	987	–1,075	–0,928
1694	2199	–0,097	0,354	1817	825	–1,060	–1,536
1695	2107	–0,255	0,114	1818	1002	–0,909	–1,095
1696	1824	–0,852	–0,560	1819	1077	–0,700	–0,748
1697	1518	–1,474	–1,329	1820	1188	–0,519	–0,286
1698	2131	–0,314	0,413	1821	1280	–0,412	–0,025
1699	1930	–0,714	–0,272	1822	1225	–0,391	–0,354
1700	2214	–0,151	0,354	1823	1107	–0,346	–0,941
1701	1552	–1,313	–0,978	1824	1213	0,229	–0,528
1702	1535	–1,092	–0,915	1825	1324	1,671	0,151
1703	1691	–0,677	–0,610	1826	1318	1,272	–0,029
1704	1958	0,310	–0,096	1827	1479	2,063	0,493
1705	2158	1,189	0,219	1828	1481	1,874	0,428
1706	1745	–0,395	–0,345	1829	1903	4,621	2,168
1707	1727	–0,430	–0,358	1830	1824	2,075	1,160
1708	1956	0,368	–0,068	1831	2266	3,237	**3,007**
1709	24533	96,637	**96,029**	1832	1556	0,114	–0,012
1710	1784	–0,323	–0,073	1833	1498	–0,138	–0,210
1711	1763	–0,320	–0,059	1834	1395	–0,595	–0,620
1712	1587	–0,348	–0,291	1835	1330	–0,907	–0,887
1713	1567	–0,351	–0,308	1836	1161	–1,470	–1,513
1714	2299	–0,247	1,448	1837	1964	1,164	•
1715	1712	–0,334	0,219	1838	1319	–0,939	•
1716	1406	–0,370	–0,995	1839	2137	1,466	•
1717	1605	–0,337	–0,128	1840	1683	0,100	•
1718	1579	–0,339	–0,177	1841	1406	–0,601	•
1719	1711	–0,315	0,218	1842	1459	–0,282	•
1720	1592	–0,459	–0,160	1843	1567	0,104	•
1721	1435	–1,036	–1,128	1844	1357	–0,606	•
1722	1442	–0,834	–1,204	1845	1605	0,217	•
1723	1495	–0,545	–0,780	1846	2077	1,709	•

Objaśnienia: W1 – wyniki funkcji *kryzysyDem1*; W2/3 – wyniki funkcji *kryzysyDem2* oraz *kryzysyDem3* (pogrubione wartości dla przedziału wskaźnika od 2 do mniej niż 4 oraz 4 i więcej).

Źródło: obliczenia własne na podstawie: Jan Baszanowski, *Przemiany demograficzne w Gdańsku w latach 1601–1846 w świetle tabel ruchu naturalnego*, Gdańsk 1995, s. 348–354 (aneks).

Tabela 3. Liczba udzielonych ślubów na Pomorzu Zachodnim (Brandenburskim) w latach 1726–1805 i wyniki formuł do wykrywania kryzysów demograficznych

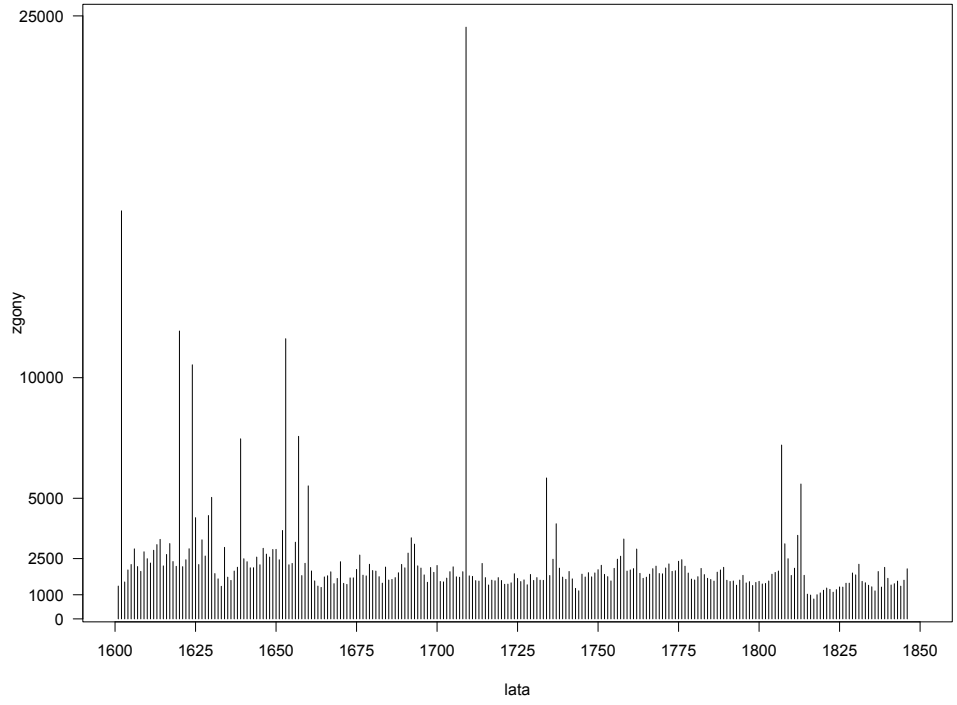
Rok	Liczba ślubów	W1	W2/3	Rok	Liczba ślubów	W1	W2/3
1	2	3	4	5	6	7	8
1726	2742	•	•	1766	3466	0,343	0,783
1727	2722	•	•	1767	2959	–0,462	–0,512
1728	2822	•	•	1768	2838	–0,837	–0,919
1729	2818	•	•	1769	2908	–0,800	–0,822
1730	3508	•	•	1770	2807	–0,848	–1,207
1731	3246	•	•	1771	2624	–1,200	–2,581
1732	3327	•	•	1772	3092	–0,308	–0,784
1733	3538	•	•	1773	3279	0,208	–0,004
1734	3422	•	•	1774	3496	0,666	1,040
1735	3090	•	•	1775	3304	0,666	0,288
1736	2917	–0,633	–0,896	1776	3278	0,674	0,170
1737	2983	–0,514	–0,663	1777	3352	1,066	0,460
1738	3199	0,115	0,007	1778	3421	1,124	0,774
1739	3251	0,185	0,219	1779	3165	0,031	–0,706
1740	2848	–1,909	**–1,314**	1780	3643	1,680	2,661
1741	3303	0,544	0,503	1781	3329	0,231	0,367
1742	3668	2,137	1,977	1782	3410	0,470	0,827
1743	3338	0,431	0,361	1783	3399	0,235	0,629
1744	2878	–1,294	–1,128	1784	3083	–2,288	–1,653
1745	2688	–1,794	–1,823	1785	3107	–1,519	**–1,411**
1746	2761	–1,175	**–1,374**	1786	3203	–0,684	–0,680
1747	3174	0,266	–0,025	1787	3328	0,097	0,035
1748	3910	2,593	2,184	1788	3228	–0,469	–0,594
1749	3332	0,376	0,260	1789	3166	–0,731	–0,922
1750	3390	0,498	0,406	1790	3221	–0,406	–0,816
1751	3332	0,227	0,277	1791	3309	0,536	–0,450
1752	3513	0,687	0,646	1792	3434	1,661	–0,002
1753	3551	0,860	0,787	1793	3509	2,221	0,135
1754	3451	0,516	0,445	1794	3689	3,143	0,569
1755	3026	–0,780	–0,400	1795	3452	0,744	–0,766
1756	2412	–2,988	–1,787	1796	4063	4,283	•
1757	2187	–2,853	–2,114	1797	4161	2,678	•
1758	2602	–1,146	–1,089	1798	4349	2,372	•
1759	4373	2,590	2,223	1799	4648	2,432	•
1760	2793	–0,606	–0,678	1800	3713	–0,144	•
1761	2642	–0,740	–0,909	1801	4311	1,067	•

1	2	3	4	5	6	7	8
1762	3863	1,218	1,179	1802	4353	0,977	•
1763	3321	0,331	0,207	1803	4237	0,519	•
1764	4294	1,790	2,493	1804	4246	0,401	•

Objaśnienia: W1 – wyniki funkcji *kryzysyDem1*; W2/3 – wyniki funkcji *kryzysyDem2* oraz *kryzysyDem3* (pogrubione wartości dla przedziału wskaźnika od –2 do –1).

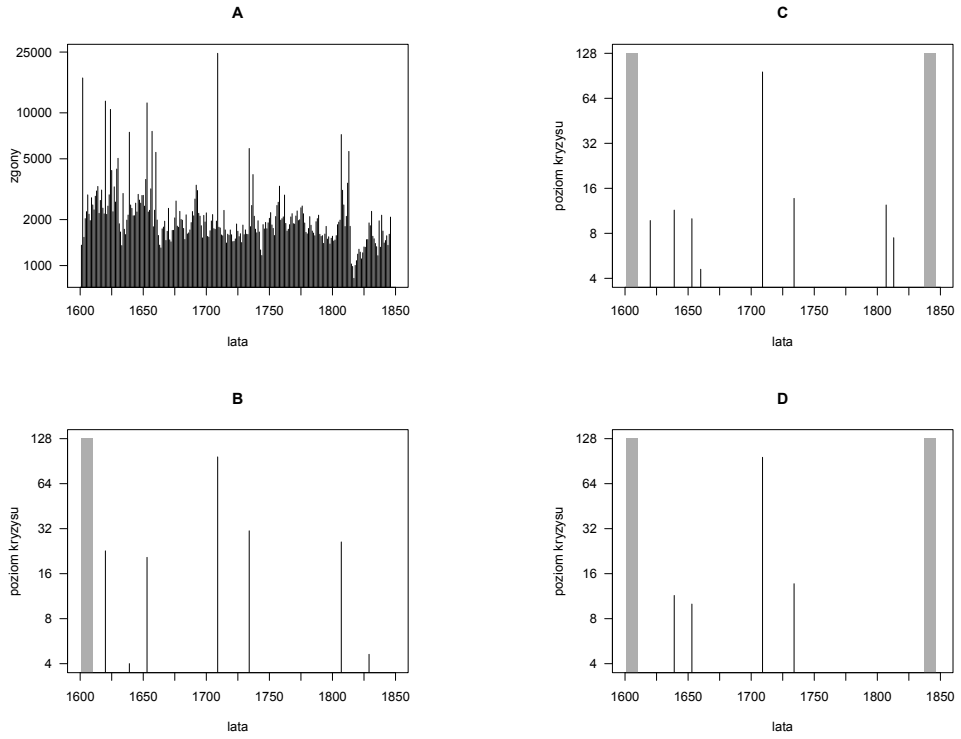
Źródło: obliczenia własne na podstawie: Zygmunt Szultka, *Rozwój demograficzny w drugiej połowie XVII i początkach XVIII wieku*, [w:] *Historia Pomorza*, t. II: *do roku 1815*, część 3: *Pomorze Zachodnie w latach 1648–1815*, opracowali Zygmunt Szultka i Henryk Lesiński przy współudziale Dariusza Lukasiewicza i Alfreda Wielopolskiego, Poznań 2003, s. 55–56, tenże, *Dynamika rozwoju i struktura społeczno–zawodowa ludności Pomorza pruskiego w XVIII i początkach XIX wieku*, tamże, s. 481–484.

Wykres 1. Liczba zgonów w Gdańsku w latach 1601–1846



Źródło: opracowanie własne na podstawie danych z tabeli 2.

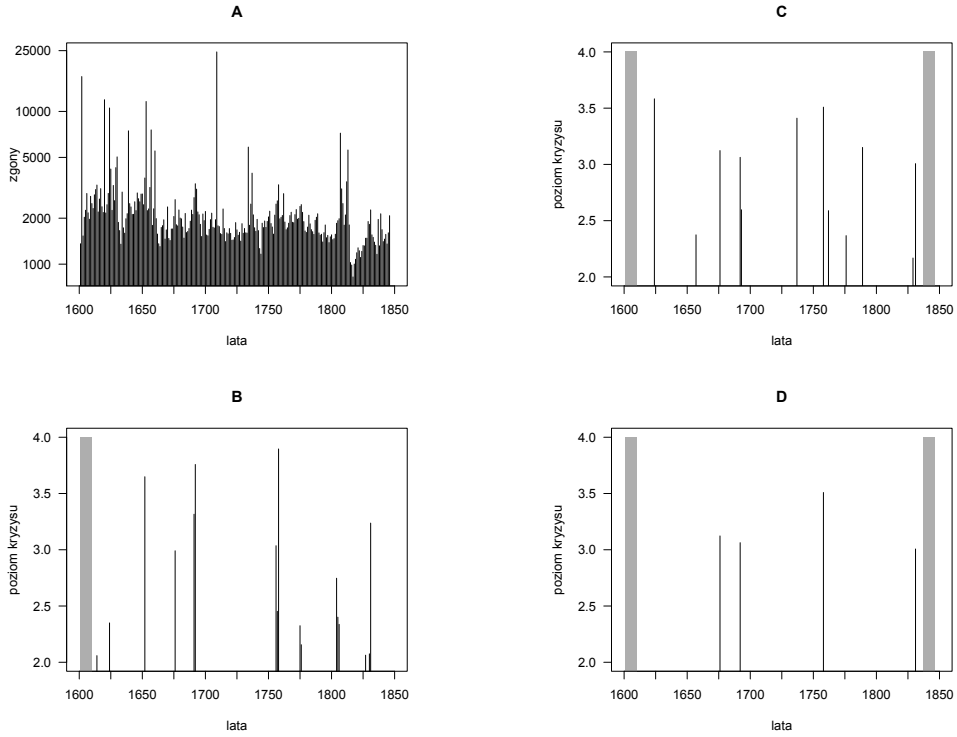
Wykres 2. Liczba zgonów w Gdańsku w latach 1601–1846 i wyniki formuł wykrywających silne i większe kryzysy demograficzne (wartość wskaźnika: 4 i więcej)



Objaśnienia:
A – liczba zgonów; B – wyniki funkcji *kryzysyDem1*, C – *kryzysyDem2*, D – *kryzysyDem3*.
Szare pola – lata, dla których nie można wyliczyć wartości wskaźnika na podstawie danych z tabeli 2.

Źródło: opracowanie własne na podstawie danych z tabeli 2.

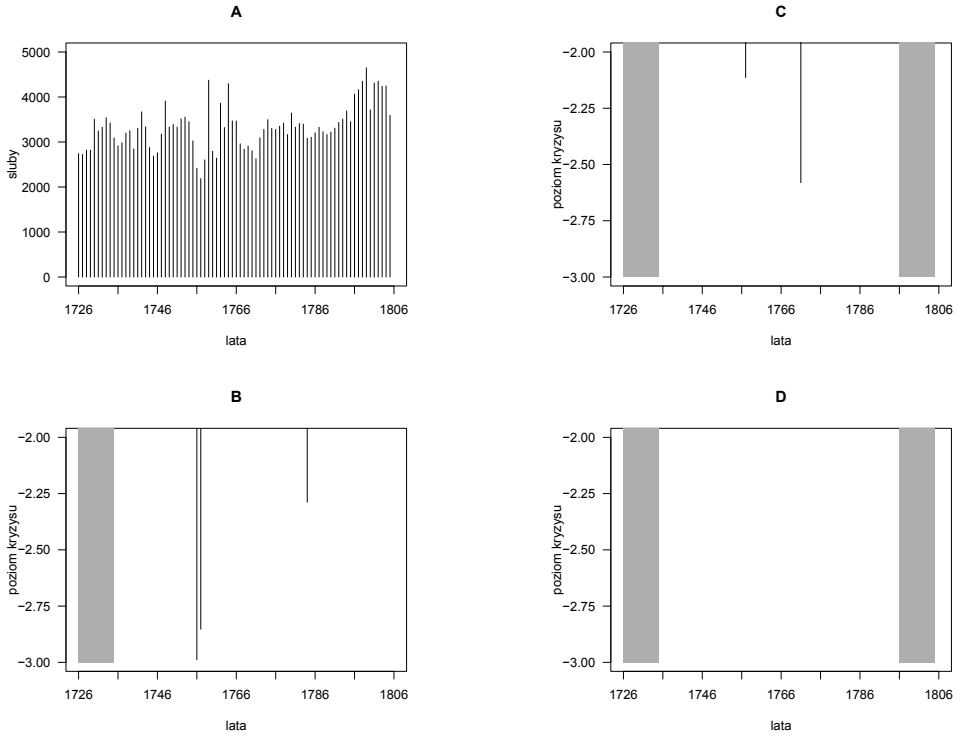
Wykres 3. Liczba zgonów w Gdańsku w latach 1601–1846 i wyniki formuł wykrywających średnie kryzysy demograficzne (wartości wskaźnika: od 2 do mniej niż 4)



Objaśnienia:
A – liczba zgonów; B – wyniki funkcji *kryzysyDem1*, C – *kryzysyDem2*, D – *kryzysyDem3*.
Szare pola – lata, dla których nie można wyliczyć wartości wskaźnika na podstawie danych z tabeli 2.

Źródło: opracowanie własne na podstawie danych z tabeli 2.

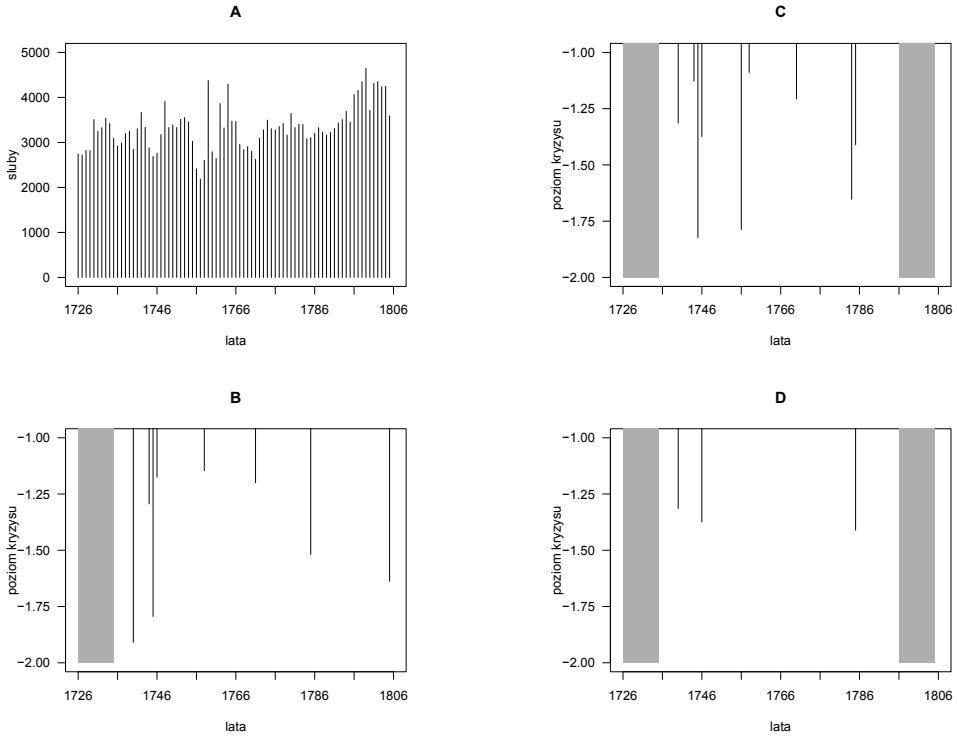
Wykres 4. Liczba udzielonych ślubów na Pomorzu Zachodnim (Brandenburskim) w latach 1726–1805 i wyniki formuł wykrywających średnie kryzysy demograficzne (wartości wskaźnika: od –3 do mniej niż –2)



Objaśnienia:
A – liczba udzielonych ślubów; B – wyniki funkcji *kryzysyDem1*, C – *kryzysyDem2*, D – *kryzysyDem3*.
Szare pola – lata, dla których nie można wyliczyć wartości wskaźnika na podstawie danych z tabeli 3.

Źródło: opracowanie własne na podstawie danych z tabeli 3.

Wykres 5. Liczba udzielonych ślubów na Pomorzu Zachodnim (Brandenburskim) w latach 1726–1805 i wyniki formuł wykrywających lekkie kryzysy demograficzne (wartości wskaźnika: od –2 do mniej niż –1)



Objaśnienia:
A – liczba udzielonych ślubów; B – wyniki funkcji *kryzysyDem1*, C – *kryzysyDem2*, D – *kryzysyDem3*.
Szare pola – lata, dla których nie można wyliczyć wartości wskaźnika na podstawie danych z tabeli 3.

Źródło: opracowanie własne na podstawie danych z tabeli 3.

On Programming in Software Environment R Exemplified with a Computational Algorithm for the Formula to Detect Demographic Crises

Summary

The article presents Jacques Dupâquier's method to detect demographic crises on the basis of the temporal distribution of deaths in its classical and modified form, popular in historical demography. The presentation of the method based on the standardised data is a pretext for studying the programming in R (language), which is a continuation of the previous article. The element that has been highlighted is the for-loop, presented step by step. An extensive introduction is supposed to familiarise the reader with the three programming codes of the functions that compute indicators of crisis for particular years: *kryzysyDem1* (the classical form of the formula), *kryzysyDem2* and *kryzysyDem3*, a modified form that takes into consideration three temporal perspectives of the development of the phenomenon, i.e. the past, present and future ones; the final results, which satisfy the determined limit criteria, are the average value of the results obtained for the three perspectives. The presented codes of the variants of Jacques Dupâquier's formula should – to a significant extent – facilitate the computation of indicators of demographic crises including various limit criteria.

Keywords: demographic crises, formula, Jacques Dupâquier, R (programming language), the rudiments of programming, for-loop