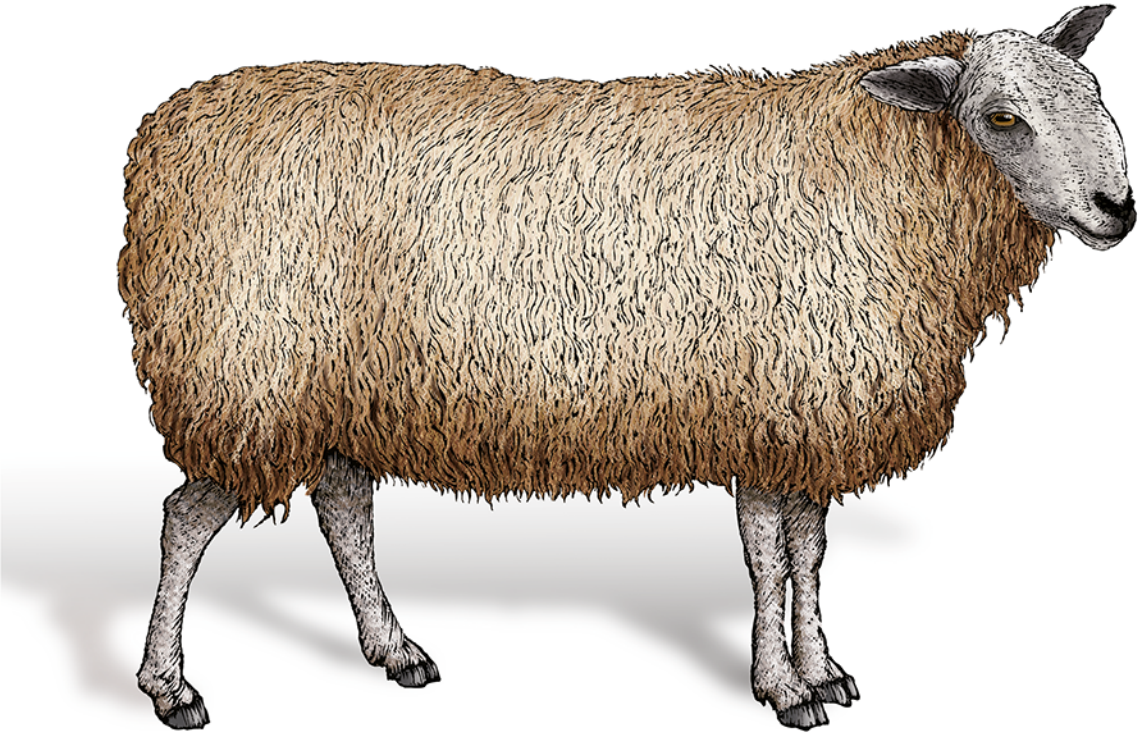
O'REILLY®

Szeregi czasowe

Praktyczna analiza i predykcja z wykorzystaniem statystyki i uczenia maszynowego



Helion

Aileen Nielsen

Tytuł oryginału: Practical Time Series Analysis: Prediction with Statistics and Machine Learning

Tłumaczenie: Filip Kamiński

ISBN: 978-83-283-6721-0

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/szecza.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/szecza
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 3.

Metody eksplorowania danych czasowych

Omówienie metod eksploracji danych czasowych składać się będzie z dwóch części. W pierwszej przedstawię zastosowanie popularnych metod eksploracji danych do analizy szeregów czasowych. Szczególną uwagę zwrócimy na rysowanie wykresów i histogramów oraz grupowanie danych.

W drugiej części skupimy się na metodach opracowanych specjalnie na potrzeby eksploracji szeregów czasowych. Metody te bazują na zależnościach czasowych pomiędzy punktami danych i nie mogą być zastosowane w przypadku braku takich relacji.

Metody ogólnego przeznaczenia

Rozpoczniemy od zastanowienia się, w jaki sposób możemy zastosować popularne metody eksploracji danych do szeregów czasowych. Na początku proces ten będzie przebiegał identycznie jak w przypadku danych innego typu.

Zanim zaczniesz analizę, powinieneś najpierw dowiedzieć się więcej na temat samych danych. Musisz ustalić, jakie kolumny znajdują się w Twoich danych, jakie są zakresy przechowywanych w nich wartości i które spośród jednostek miary sprawdzą się najlepiej. Pytania, jakie Ci się nasuną, nie różnią się od tych, które powinieneś postawić, widząc po raz pierwszy jakikolwiek nowy zbiór danych:

- Czy w zbiorze istnieją kolumny silnie skorelowane z innymi?
- Jaka jest średnia wartości zmiennej w danej kolumnie? Jaka jest jej wariancja?

Aby odpowiedzieć na te pytania, możesz zastosować znane Ci techniki, takie jak parametry statystyczne, histogramy, odpowiednio przygotowane wykresy punktowe lub liniowe. Pracując z szeregami czasowymi, dodatkowo należy ustalić:

- Jaki jest zakres obserwowanych wartości? Czy zakres ten zmienia się w czasie lub innej jednostce podlegającej analizie?
- Czy dane wyglądają na spójne i zostały zmierzone w jednolity sposób? Czy pojawiły się jakieś zmiany w zachowaniu lub sposobie pomiaru danych w czasie?

Chcąc odpowiedzieć na te pytania, musisz uwzględnić w wymienionych wcześniej metodach aspekt temporalny. Aby wziąć pod uwagę upływ czasu, możemy uwzględnić go w obliczaniu statystyk,

umieścić na jednej z osi na naszych wykresach lub dokonać grupowania na jego podstawie (na przykład w histogramach lub wykresach punktowych).

W pozostałej części tego podrozdziału przyjrzymy się przykładom zastosowania tradycyjnych metod eksploracji danych do problemów czasowych, a także sposobom ich przystosowania do zmian czasu. Do analizy wykorzystamy zbiór danych o europejskich giełdach dostępny w środowisku R.

Operacje grupowania

Podobnie jak w przypadku innych danych, analiza oparta na grupowaniu danych może być przydatna również w problemach dotyczących szeregów czasowych. Podczas gdy w klasycznych problemach dane grupujemy na podstawie parametrów takich jak wiek, płeć czy sąsiedztwo, w analizie szeregów czasowych zastosowanie może znaleźć grupowanie w oparciu o czas (na przykład obliczanie miesięcznych średnich lub tygodniowych median). Możesz także spróbować połączyć grupy bazujące na klasycznych i czasowych charakterystykach, na przykład obliczyć miesięczne średnie spożycie kalorii w podziale na płeć lub medianę liczby przespanych godzin przez pacjentów jakiegoś szpitala. Istnieje wiele sposobów pozwalających na interakcję osi czasowych z innymi parametrami:

- Jeżeli masz powody sądzić, że w danych zachodzi jakaś istotna zmiana ich charakteru, to możesz spróbować przeanalizować różne okresy czasu oddzielnie. Do znalezienia rozsądnych rozgraniczeń pomiędzy okresami możesz wykorzystać narzędzia analizy eksploracyjnej.
- Możesz zauważyć, że dane zawarte w jednym szeregu (jak w przykładzie z darowiznami) prezentują się lepiej, gdy rozdzielimy je na kilka równoległych do siebie ciągów (na przykład jeden darczyńca — jeden szereg).

Wykresy liniowe

Przyjrzyjmy się zbiorowi *EuStockMarkets*, który jest dostępny w środowisku R. Aby się zorientować, czego dotyczą zawarte w nim dane, możemy przyjrzeć się jego początkowemu fragmentowi:

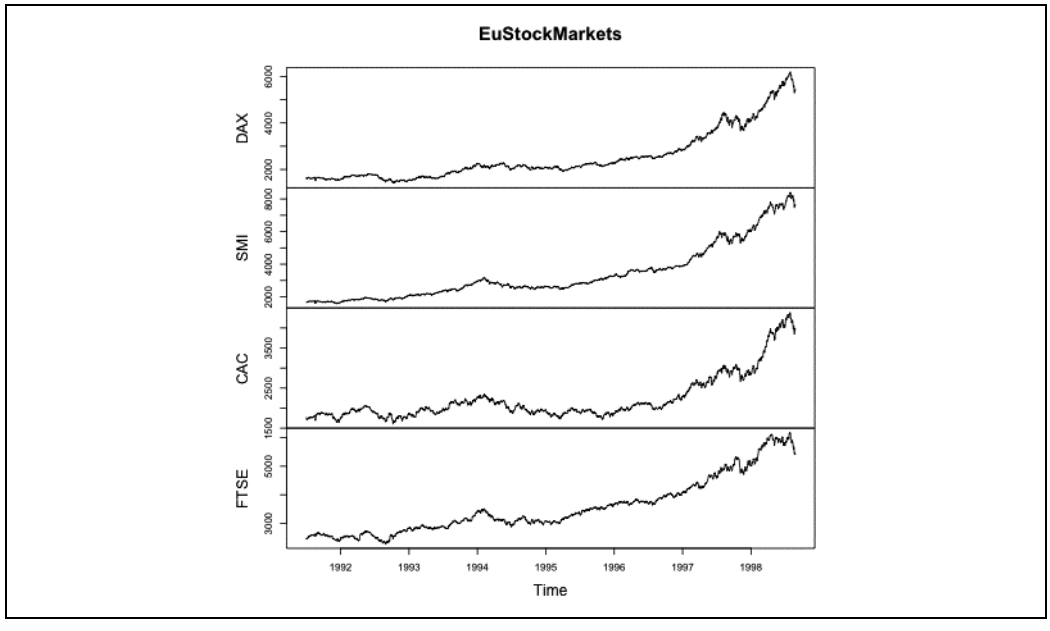
```
## R
> head(EuStockMarkets)
         DAX    SMI    CAC   FTSE
[1,] 1628.75 1678.1 1772.8 2443.6
[2,] 1613.63 1688.5 1750.5 2460.2
[3,] 1606.51 1678.6 1718.0 2448.2
[4,] 1621.04 1684.1 1708.1 2470.4
[5,] 1618.16 1686.6 1723.1 2484.7
[6,] 1610.61 1671.6 1714.3 2466.8
```

Ten wbudowany w R zestaw danych zawiera dzienne ceny zamknięcia notowań pochodzące z czterech największych giełd w Europie (lata 1991 – 1998). Dane rejestrowane były tylko w dni robocze.

Spośród analizowanych przez nas do tej pory zbiorów zestaw ten jest najlepiej przygotowany do dalszej pracy (ma elegancki format i został odpowiednio spróbkowany). W tym przypadku nie musimy się martwić o brakujące dane, strefy czasowe i błędy pomiarowe, możemy od razu przejść do dalszej analizy.

Naszym pierwszym krokiem będzie spojrzenie na ten szereg w sposób bardzo podobny do tego, w jaki patrzymy na dane pozbawione informacji czasowej (aczkolwiek w przypadku szeregów czasowych możemy skorzystać jedynie z prostszych metod ogólnego przeznaczenia). Wykreślając każdy indeks giełdowy w osobnym układzie współrzędnych, otrzymujemy całkiem sensowne rezultaty. W odróżnieniu w analizie przekrojowej wykresy relacji pomiędzy pojedynczą cechą i jej indeksem nie dostarczają żadnych informacji na temat sytuacji w układzie i pozwalają jedynie na ustalenie arbitralnego porządku danych (występującego w tabeli lub wyniku zapytania SQL). W przypadku szeregów czasowych (jak widać na rysunku 3.1) wykresy takie są źródłem sporej ilości informacji:

```
## R
> plot(EuStockMarkets)
```



Rysunek 3.1. Prosty wykres szeregu czasowego

Zauważ, że stosując wywołanie funkcji `plot()`, otrzymujesz wykres, który został automatycznie podzielony tak, aby prezentował każdy szereg osobno. Ma to miejsce dlatego, że, tak jak pokazano poniżej, dane przechowywane są w obiekcie typu `mts` (z obiektów typu `ts` korzystalibyśmy, gdyby dane zamiast kilku równoległych do siebie szeregów zawierały tylko jeden):

```
## R
> class(EuStockMarkets)
[1] "mts"    "ts"     "matrix"
```

Wiele popularnych pakietów intensywnie wykorzystuje typ `ts` i jego pochodne. Użycie tych typów pozwala łatwo stworzyć automatycznie opisane wykresy prezentujące dane w kilku panelach. Ponadto obiekty typu `ts` umożliwiają skorzystanie z wielu wygodnych funkcji:

- `frequency` — pozwala ustalić roczną częstotliwość danych:

  ```
  ## R
  > frequency(EuStockMarkets)
  [1] 260
  ```

- `start` i `end` — pozwalają ustalić daty pojawienia się pierwszej i ostatniej informacji:

```
## R
> start(EuStockMarkets)
[1] 1991  130
> end(EuStockMarkets)
[1] 1998  169
```

- `window` — umożliwia wyodrębnienie fragmentu danych na podstawie czasu:

```
## R
> window(EuStockMarkets, start = 1997, end = 1998)
Time Series:
Start = c(1997, 1)
End = c(1998, 1)
Frequency = 260
             DAX    SMI    CAC   FTSE
1997.000 2844.09 3869.8 2289.6 4092.5
1997.004 2844.09 3869.8 2289.6 4092.5
1997.008 2844.09 3869.8 2303.8 4092.5
...
1997.988 4162.92 6115.1 2894.5 5168.3
1997.992 4055.35 5989.9 2822.9 5020.2
1997.996 4125.54 6049.3 2869.7 5018.2
1998.000 4132.79 6044.7 2858.1 5049.8
```

Klasa `ts` ma zarówno wady, jak i zalety. Jak wspomniałam wcześniej, `ts` i jej pochodne występują w wielu pakietach związanych z analizą szeregów czasowych, a ich zastosowanie umożliwia stworzenie wykresów w prosty i szybki sposób. Z drugiej strony, sposoby indeksowania danych oraz działanie funkcji `window` mogą z czasem sprawiać problemy. W książce będziesz mieć okazję przyjrzeć się różnym sposobom przechowywania danych i uzyskiwania dostępu do nich. Decyzję dotyczącą wyboru najlepszego z nich pozostawiam Tobie.

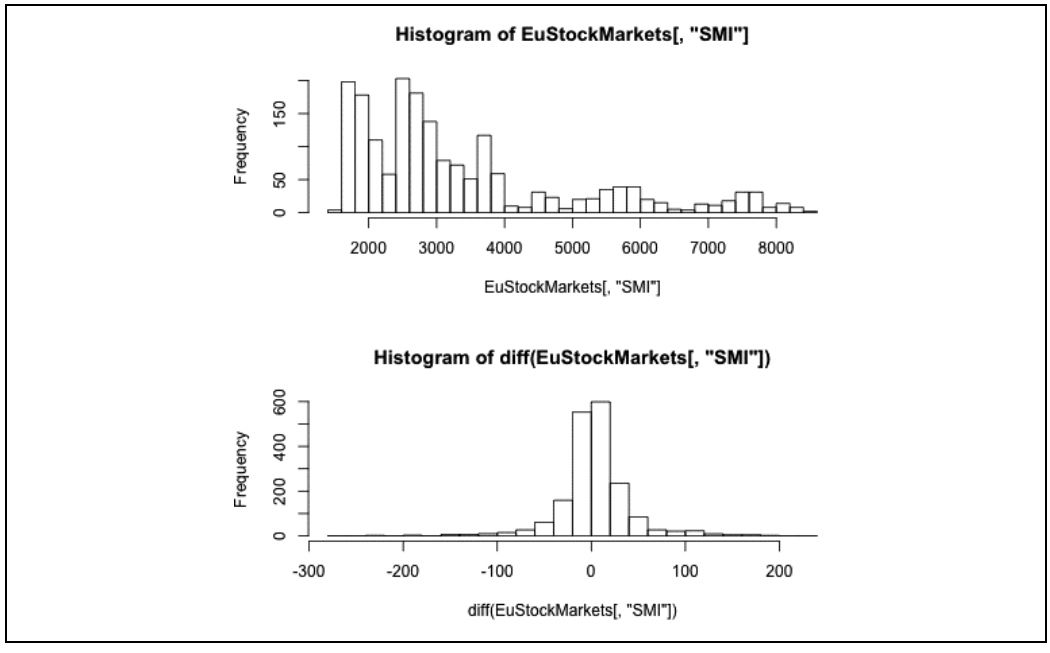
Histogramy

Kontynuujemy nasze porównanie metod eksploracji danych. Chcielibyśmy, podobnie jak czyni się to w przypadku zwyczajnych danych, narysować histogram szeregu. Uwzględnienie na histogramie osi czasu wymaga trochę zachodu — można to zrobić, obliczając kolejne różnice pomiędzy cenami (*różnicując szereg* — rysunek 3.2):

```
## R
> hist(EuStockMarkets[, "SMI"], 30)
> hist(diff(EuStockMarkets[, "SMI"], 30))
```

W kontekście szeregów czasowych histogram danych różnicowych jest często bardziej interesujący od otrzymanego z danych nieprzekształconych. W końcu w kontekście szeregów czasowych (zwłaszcza w obszarze finansów) najbardziej interesuje nas nie wartość, lecz jej zmiana pomiędzy kolejnymi punktami w czasie. W przypadku wykresów cecha ta jest szczególnie istotna, a histogram danych zawierających trend nie prezentuje praktycznie żadnych wartościowych informacji.

Przyjrzyjmy się teraz nowym informacjom, jakie możemy odczytać z histogramu danych różnicowych. Podczas gdy oryginalne wykresy w poprzedniej sekcji zawierały idealną wizję praktycznie nieprzerwanie rosnących cen akcji, histogram pokazuje nam codzienne doświadczenia gracza giełdowego. Zróżnicowany histogram mówi nam, że wartości w szeregu rosły zarówno w górę (dodatnie



Rysunek 3.2. Dane pokazane na histogramie otrzymanym przed przekształceniem (górny rysunek) nie mają rozkładu normalnego. Oznacza to, że w danych znajduje się jakiś trend. Obliczając różnice cen akcji, usuwamy trend i przekształcamy dane tak, aby ich rozkład zbliżył się do normalnego (dolny rysunek)

różnice), jak i w dół (ujemne różnice) o mniej więcej taką samą wartość w czasie. Oczywiście ponieważ wiemy, że indeksy giełdowe charakteryzują się tendencją wzrostową, różnice te nie były dokładnie takie same. Na podstawie histogramu możemy zobaczyć, że aby osiągnąć dodatni trend, wystarczą tylko niewielkie różnice na korzyść dodatnich wartości.

Dane te są dobrym przykładem tego, dlaczego musimy zwracać uwagę na skalę czasową podczas próbkowania, podsumowywania i zadawania pytań o dane. To, czy zachowanie układu wygląda na wykresie bardzo dobrze (zobacz wykres długoterminowy), czy tak sobie (histogram zróżnicowanych danych), będzie zależeć od przyjętej skali czasu. Musimy odpowiedzieć sobie na pytanie, czy zależy nam na codziennych danych, czy raczej interesuje nas perspektywa długoterminowa. Jeśli pracujemy w firmie zajmującej się handlem akcjami w celu wypracowania rocznego zysku, raportując szefowi straty wynikające z ujemnych różnic pomiędzy kolejnymi dniami, musimy brać pod uwagę perspektywy krótkoterminowe. Jeżeli jednak jesteśmy dużym inwestorem instytucjonalnym (na przykład uniwersytetem lub szpitalem), możemy pozwolić sobie na długoterminowe spojrzenie na inwestycję, w którym będziemy oczekiwali trendu wzrostowego. Drugi scenariusz wiąże się również ze specyficznymi wyzwaniami: musimy na przykład ustalić, kiedy należy sprzedać posiadane akcje. Chcemy zmaksymalizować zysk, ale musimy wziąć pod uwagę również możliwość kryzysu. Pytania tego typu są źródłem utrzymania branży finansowej i rozwoju badań na temat szeregów czasowych i metod prognozowania.

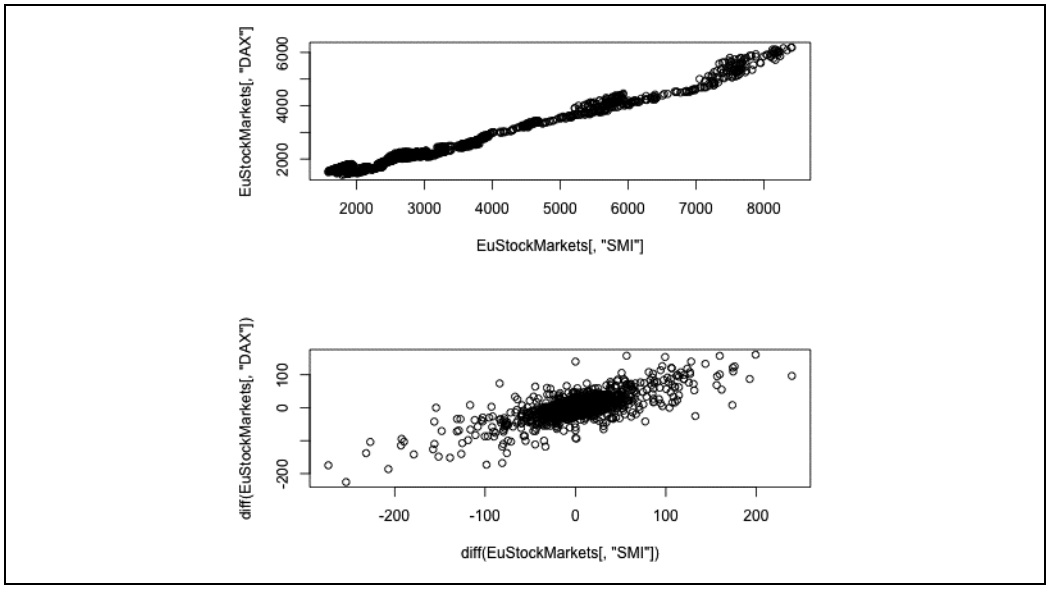
Wykresy punktowe

Wykresy punktowe (ang. *scatter plot*) sprawdzają się równie dobrze w przypadku szeregów czasowych i innych danych. Możemy z nich skorzystać, aby ustalić, w jaki sposób w czasie są ze sobą powiązane dwa indeksy giełdowe, oraz określić związek pomiędzy zachodzącymi na nich zmianami cen a upływem czasu.

Aby to ustalić, możemy stworzyć dwa wykresy (rysunek 3.3):

- wykres wartości z obu giełd w czasie,
- wykres prezentujący dzienne zmiany cen na obu parkietach (otrzymany przy użyciu funkcji `diff()`):

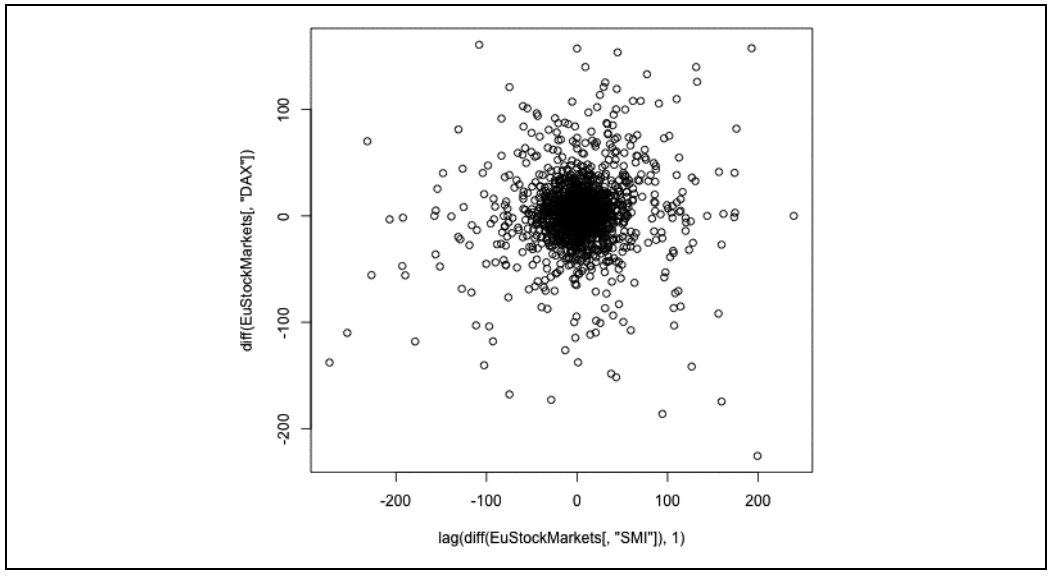
```
## R
plot(EuStockMarkets[, "SMI"], EuStockMarkets[, "DAX"])
plot(diff(EuStockMarkets[, "SMI"]), diff(EuStockMarkets[, "DAX"]))
```



Rysunek 3.3. Proste wykresy punktowe dwóch indeksów giełdowych wskazują na dużą korelację pomiędzy danymi. Istnieją jednak powody, dla których powinniśmy przyglądać się tym wykresom z pewną dozą podejrzliwości

Podobnie jak wcześniej, wykres zawierający wartości daje nam mniej informacji niż wykres wartości zróżnicowanych. Pomimo tego, że na drugim wykresie dane wyglądają na silnie skorelowane, relacje pomiędzy nimi nie są aż tak silne, jak to wygląda (aby dowiedzieć się więcej na ten temat, zajrzyj do sekcji „Korelacje pozorne" w tym rozdziale).

Pozorne korelacje występujące na rysunku 3.3 są interesujące, ale nawet gdybyśmy nie mieli powodów, aby w nie wątpić, to niestety nie są to korelacje, na których dałoby się zarobić. Związek pomiędzy cenami akcji na obu parkietach zauważony na tym wykresie dotyczy jedynie tych samych punktów w czasie. Aby zarobić, musimy się dowiedzieć, czy wcześniejsza w czasie zmiana ceny akcji na pierwszym parkiecie pozwoli nam przewidzieć późniejszą zmianę na drugim. Aby to zrobić, dokonamy cofnięcia jednego z szeregów o jeden dzień. Spójrz uważnie na poniższy kod. Zauważ, że wciąż różnicujemy oba szeregi, ale dodatkowo przesuwamy również jeden z nich w czasie (rysunek 3.4):



Rysunek 3.4. Związek pomiędzy cenami akcji zniknął zaraz po uwzględnieniu różnicy w czasie. Jak widać, ceny na parkiecie SMI nie wpływają na ceny w indeksie DAX

```
## R
> plot(lag(diff(EuStockMarkets[, "SMI"]), 1),
           diff(EuStockMarkets[, "DAX"]))
```

Linie w tym kodzie zostały dopasowane tak, aby ułatwić jego czytanie. Tworzenie długich wierszy nieczytelnego kodu może być kuszące, szczególnie w funkcyjnych języka programowania, takich jak R lub Python, ale należy tego unikać zawsze, gdy jest to możliwe.

Oto co możemy zauważyć na podstawie otrzymanych wyników:

- Do eksploracji szeregów czasowych możemy zastosować te same metody co do innych danych, ale musimy pamiętać, że ich bezmyślna aplikacja nie przyniesie dobrych rezultatów. W celu poprawy otrzymywanych rezultatów powinniśmy się zastanowić nad wykorzystaniem tych metod na przekształconych danych.
- Zazwyczaj na zachowanie danych największy wpływ ma związek pomiędzy danymi lub ich ewolucja w czasie.
- Rysunek 3.4 dowodzi, dlaczego bycie inwestorem giełdowym jest takie trudne. Jeżeli stosujesz taktykę pasywnego inwestowania i czekasz na efekty długoterminowego trendu wzrostowego, możesz zarobić, ale wiedz, że przewidywanie przyszłości nie jest łatwe!

Dostępna w R funkcja `lag()` może nie przesuwać danych w kierunku, w którym tego oczekujesz. Aby uniknąć przesunięcia danych w niewłaściwym kierunku, pamiętaj, że funkcja `lag()` pozwala na przesunięcie danych jedynie „do przodu”. W zależności od przeznaczenia analizy dopuszczalne mogą być przesunięcia czasowe zarówno w przód, jak i w tył.

Metody przeznaczone do eksploracji szeregów czasowych

Część metod przeznaczonych do analizy szeregów czasowych koncentruje się na relacjach pomiędzy wartościami w różnych punktach czasu. Jeżeli nie miałeś wcześniej styczności z danymi czasowymi, metody te prawdopodobnie nie będą Ci znane. Resztę tego rozdziału poświęcę omówieniu różnych pojęć i technik związanych z klasyfikacją szeregów czasowych.

Przyjrzymy się bliżej następującym pojęciom:

Stacjonarność
: Dowiesz się, co oznacza stacjonarność w kontekście szeregów czasowych, i zapoznasz się z testami statystycznymi pozwalającymi ją zbadać.

Autokorelacja
: Zastanowimy się, co oznacza, że szereg czasowy charakteryzuje się wysokim wskaźnikiem autokorelacji, oraz co mówi nam ona na temat leżącej u jego podstaw dynamiki.

Pozorna korelacja
: Dowiesz się, co oznacza pozorny charakter korelacji i kiedy należy spodziewać się jego wystąpienia.

Nauczysz się korzystać z następujących metod:

- ruchome i rozszerzające się okna czasowe,
- autokorelacja:
- funkcja autokorelacji,
- funkcja autokorelacji cząstkowej.

Omówię wszystkie pojęcia i wynikające z nich metody, stosując kolejność od stacjonarności poprzez autokorelacje do korelacji pozornych. Zanim przejdziemy do szczegółów, zobaczmy, skąd się wzięła ta kolejność.

Pierwszym pytaniem, które prawdopodobnie sobie zadasz, widząc szereg czasowy, jest to, czy odzwierciedla on system stabilny, czy raczej taki, który ulega ciągłym zmianom. Znajomość poziomu „stabilności" (nie chodzi tu o stabilność w technicznym tego słowa znaczeniu), nazywanego inaczej poziomem **stacjonarności**, jest przydatna w ocenie, na ile długookresowe zachowanie systemu w przeszłości wpływa na jego zachowanie w przyszłości. Zazwyczaj po dokonaniu oceny stacjonarności próbujemy ustalić, czy w danym szeregu występuje jakaś dynamika wewnętrzna (na przykład zmiany sezonowe). Wykrycie takich zmian sprowadza się do zbadania autokorelacji — czyli odpowiedzi na pytanie, jak ściśle dane z bliższej lub dalszej przeszłości wpływają na wyniki w przyszłości. Wreszcie, gdy wydaje się nam, że znaleźliśmy pewną dynamikę zachowania naszego systemu, musimy się upewnić, że znalezione związki mają charakter przyczynowo-skutkowy. Jeżeli nie uda nam się tego dowieść, będziemy mieli do czynienia z korelacjami pozornymi.

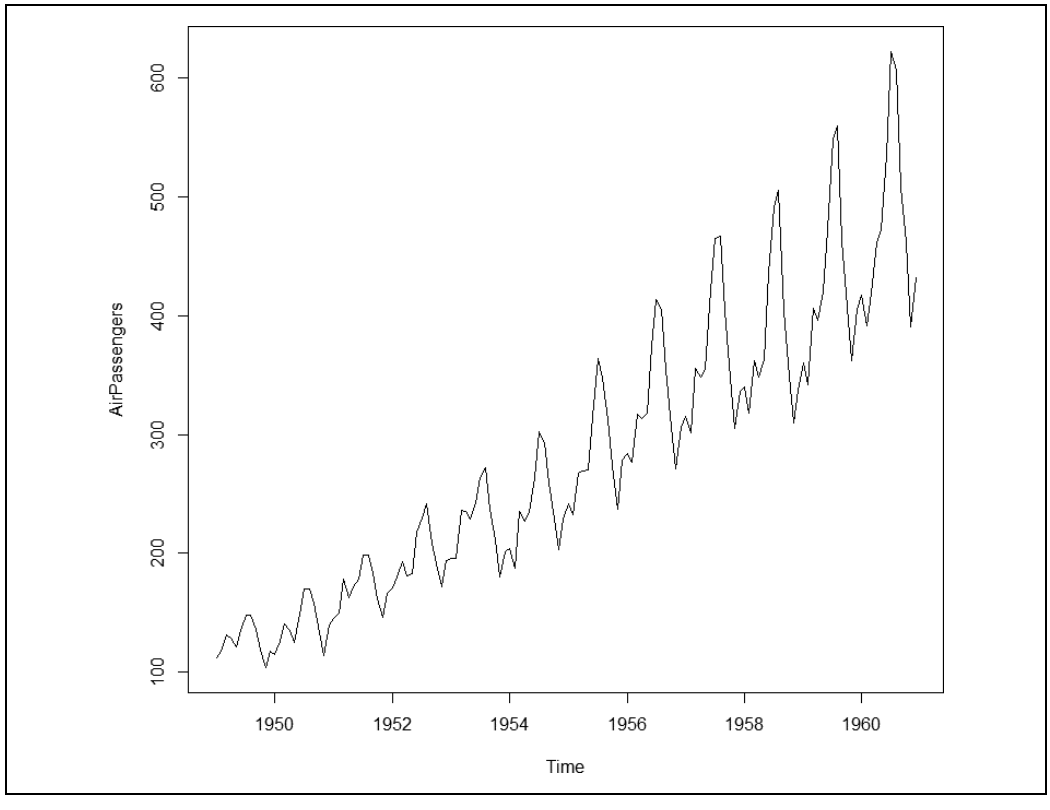
O stacjonarności słów kilka

Wiele tradycyjnych statystycznych modeli szeregów czasowych opiera się na założeniu stacjonarności. Ogólnie rzecz biorąc, stacjonarny szereg czasowy to szereg opisywany przez dość stałe w czasie parametry statystyczne (szczególnie średnią i wariancję).

Chociaż definicja ta wydaje się całkiem prosta, to stacjonarność rzeczywistego szeregu czasowego może być trudna do określenia. Opierając się na własnej intuicji, można łatwo dać się oszukać, dlatego zanim przejdziemy do szczegółów implementacyjnych i testów statystycznych, omówię pojęcie stacjonarności zarówno od intuicyjnej, jak i od formalnej strony.

Intuicje

Stacjonarny szereg czasowy to taki, w którym kolejne wartości odzwierciedlają stabilny stan układu. Czasami trudno jest stwierdzić, co to dokładnie oznacza, i łatwiej jest wykluczyć stacjonarność niż ją udowodnić. Prostym przykładem danych niestacjonarnych jest badany przez nas w rozdziale 2. zestaw danych dotyczący pasażerów linii lotniczych, którego wykres przedstawiono na rysunku 3.5 (dla przypomnienia: zestaw ten jest dostępny do pobrania w internecie, a w środowisku R można go znaleźć pod nazwą *AirPassenger*).



Rysunek 3.5. Zestaw danych dotyczących pasażerów linii lotniczych jest przykładem szeregu niestacjonarnego. Wraz z upływem czasu wzrasta zarówno średnia, jak i wariancja, widoczny jest także sezonowy charakter danych będący naturalnym przeciwieństwem stacjonarności

Istnieje kilka cech, które pokazują, że proces ten nie jest stacjonarny. Po pierwsze, średnia, zamiast utrzymywać się na stałym poziomie, rośnie wraz z czasem. Po drugie, z roku na rok wzrasta różnica między rocznymi minimami i maksimami (wzrost wariancji z czasem). Po trzecie, proces wykazuje silną sezonowość — zaprzeczenie stacjonarności.

Definicja pojęcia stacjonarności i rozszerzony test Dickeya-Fullera

Według jednej z prostszych definicji proces stacjonarny to taki, w którym dla wszystkich możliwych opóźnień k rozkład $y_t, y_{t+1},\ldots, y_{t+k}$ nie zależy od wartości t.

Testy statystyczne sprawdzające stacjonarność często sprowadzają się do pytania, czy jedynka jest pierwiastkiem równania charakterystycznego opisującego ten proces[1]. Jeżeli jedynka jest pierwiastkiem równania charakterystycznego opisującego liniowy szereg czasowy, możemy być pewni, że nie jest on stacjonarny. Brak tego pierwiastka natomiast nie jest warunkiem wystarczającym stacjonarności. W ogólności problem stacjonarności jest trudny do rozwiązania, a próby znalezienia pierwiastków jednostkowych stanowią aktywny obszar badań.

Niemniej jednak prostą intuicję dotyczącą tego, czym jest pierwiastek jednostkowy, można uzyskać, przyglądając się procesowi błądzenia losowego:

$$y_t = \phi \cdot y_{t-1} + e_t$$

W tym procesie wartość w danym punkcie czasu zależy od wartości bezpośrednio ją poprzedzającej oraz pewnego losowego błędu. Jeżeli ϕ jest równe 1, to proces ten ma pierwiastek jednostkowy i nie jest procesem stacjonarnym (jego wartości dążą do nieskończoności). Warto w tym miejscu zauważyć, że szereg, który nie jest stacjonarny, niekoniecznie musi zawierać jakiś trend. Błądzenie losowe jest dobrym przykładem niestacjonarnego szeregu czasowego, w którym nie występuje żaden trend[2].

Stosowanie testów sprawdzających, czy dany proces jest stacjonarny, nazywa się fachowo weryfikacją hipotezy stacjonarności szeregu. **Rozszerzony test Dickeya-Fullera** (ADF) to najpopularniejsze narzędzie służące do badania problemu stacjonarności. Hipoteza zerowa w tym teście zakłada istnienie pierwiastka jednostkowego. W zależności od wyników testu hipotezę zerową można odrzucić z przyjętym poziomem istotności, czyli, innymi słowy, stwierdzić nieobecność pierwiastka z danym prawdopodobieństwem.

Zauważ, że testy stacjonarności koncentrują się na zmianie średniej szeregu. Zmiany wariancji nie są formalnie testowane, lecz jedynie uwzględnione w przekształceniach. Test stacjonarności szeregu sprowadza się do przetestowania jego stopnia zintegrowania. Szereg o stopniu zintegrowania d to taki, na którym w celu uzyskania szeregu stacjonarnego musimy wykonać różnicowanie co najmniej d razy.

Test Dickeya-Fullera wygląda następująco:

$$\Delta y_t = y_t - y_{t-1} = (\phi - 1) \cdot y_{t-1} + \varepsilon_t$$

Weryfikacja, czy $\phi = 1$, to rozszerzony test t-Studenta sprawdzający, czy parametr opóźnionego członu y_{t-1} jest równy 0. Różnica między prostym a rozszerzonym testem Dickeya-Fullera polega na wzięciu pod uwagę większej liczby opóźnień, tak aby leżący u jego podstaw model mógł uwzględniać bardziej złożoną dynamikę zachowań. Test rozszerzony możemy przedstawić jako:

$$Y_t - \phi_1 \cdot y_{t-1} - \phi_1 \cdot y_{t-2} \ldots = \varepsilon_t$$

[1] Jeżeli nie wiesz, co to oznacza, nic nie szkodzi. Na końcu rozdziału, na liście polecanych pozycji, znajdziesz prace, z których możesz dowiedzieć się więcej na ten temat.

[2] Może się wydawać, że w tym szeregu czasowym (otrzymanym na podstawie procesu błądzenia losowego) występuje jakiś trend. Wrażenie to inspiruje wiele debat — zwłaszcza w temacie analizy cen akcji na giełdach.

Przeprowadzenie testu ADF wymaga nieco więcej przekształceń algebraicznych, tak aby można było przedstawić dany szereg jako serię zróżnicowanych opóźnień. Również rozkład prawdopodobieństwa, względem którego testowana jest hipoteza zerowa, różni się nieco od obecnego w pierwotnym teście Dickeya-Fullera. Test ADF jest najpopularniejszym prezentowanym w literaturze testem stacjonarności szeregów czasowych.

Niestety testy nie są panaceum na problemy związane ze stacjonarnością. Powody są następujące:

- Mają problemy z odróżnieniem pierwiastka jednostkowego od pierwiastków niewiele różniących się od tej wartości.
- W przypadku niewielkich próbek często pojawiają się fałszywie dodatnie błędy I rodzaju (FP).
- Większość testów nie sprawdza wszystkich czynników wpływających na niestacjonarność procesu. Działanie niektórych testów będzie polegało jedynie na sprawdzeniu, czy średnia lub wariancja (ale nie obie) są stacjonarne. Inne testy będą koncentrowały się raczej na ogólnym rozkładzie wartości. Ważne jest, aby podczas korzystania z danego testu rozumieć granice jego zastosowania oraz upewnić się, że występujące w nim ograniczenia są spójne z wyobrażeniem na temat badanych danych.

Alternatywna hipoteza zerowa — test KPSS

Podczas gdy test ADF zakłada w swojej hipotezie zerowej obecność pierwiastka jednostkowego, test Kwiatkowskiego-Phillipsa-Schmidta-Shina (KPSS) korzysta z przeciwnego założenia. W odróżnieniu od ADF test KPSS nie jest dostępny w standardowej dystrybucji R (ale jest sporo jego implementacji). Istnieje kilka (wykraczających poza zakres tej książki) niuansów dotyczących stosowania tych testów i ich poprawnego przeprowadzania, które są szeroko omawiane w sieci (https://perma.cc/D3F2-TATY).

W praktyce

W praktyce stacjonarny charakter analizowanego szeregu ma znaczenie z wielu powodów. Przede wszystkim duża liczba modeli, takich jak tradycyjne modele o znanych zaletach i modele statystyczne, zakłada stacjonarność procesu. Modele te omówię w rozdziale 6.

Z szerszej perspektywy istotne jest to, że w przypadku niestacjonarnego szeregu czasowego wraz z upływem czasu zmieniać się będzie dokładność modelu. Oznacza to, że w przypadku zastosowania zakładającego stacjonarność modelu do estymacji średniej w szeregu niestacjonarnym (czyli w takim, w którym średnia, podobnie jak wariancja, jest zmienna) możesz otrzymać rezultaty obarczone tak dużym błędem, że ich użyteczność będzie wątpliwa.

Często chcąc zmienić szereg niestacjonarny w stacjonarny, wystarczy zastosować kilka prostych przekształceń. Transformacja logarytmiczna lub pierwiastek kwadratowy to dwie popularne opcje — zwłaszcza gdy w szeregu zmienia się wariancja. W podobny sposób, poprzez jedno- lub wielokrotne zastosowanie operacji różnicowania, z szeregu możemy usunąć trend. Jeżeli jednak po wykonaniu więcej niż dwóch, trzech różnicowań w szeregu dalej występują trendy, mało prawdopodobne jest, że powtarzając tę operację, uda się otrzymać szereg stacjonarny.

Logarytm czy pierwiastek?

Chociaż obliczenie pierwiastka kwadratowego jest mniej złożone od logarytmowania, rozważ każdą z tych opcji. Zamiast od razu nadmiernie optymalizować kod i analizę najpierw zastanów się nad zakresem danych i sposobami jego ograniczenia.

Stacjonarność nie jest jedynym występującym w modelach służących do prognozowania założeniem. Innym powszechnym, ale odrębnym od stacjonarności założeniem jest normalny charakter rozkładu zmiennych wejściowych lub zmiennej przewidywanej. Aby go zapewnić, możesz wykorzystać inne przekształcenia. Jednym z powszechnie występujących jest transformacja Box Cox dostępna w pakietach *forecast* (R) i *scipy.stats* (Python). Transformacja ta modyfikuje dane o niestandardowym rozkładzie (ang. *skewed data*) tak, aby ich rozkład przypominał bardziej normalny. Pamiętaj, że to, iż możesz przekształcić dane, wcale nie oznacza, że powinieneś to zrobić. Zanim podejmiesz decyzję o modyfikacji, zastanów się dokładnie nad znaczeniem odległości między kolejnymi punktami w szeregu i upewnij się, niezależnie od powierzonego Ci zadania, że transformacje, które chcesz wykorzystać, zachowują najważniejsze informacje.

Transformacje też mają założenia

Może Ci się wydawać, że wybrane przez Ciebie transformacje nie bazują na żadnych założeniach (przecież logarytm i pierwiastek to takie proste operacje). Radzę jednak dobrze się zastanowić, czy na pewno jest to prawda.

Jak już wspominałam, transformacja logarytmiczna i pierwiastek kwadratowy są powszechnie stosowane w celu zmniejszenia wariancji danych w czasie. Jednak z ich zastosowaniem wiąże się szereg założeń, takich jak na przykład dodatniość danych. Jeżeli przed dokonaniem pierwiastkowania lub logarytmowania zdecydujesz się na dodanie do danych stałej wartości, to albo musisz założyć, że wartość ta nie ma znaczenia, albo wprowadzisz do danych nowe odchylenia. Wreszcie, pierwiastkując lub logarytmując dane, skutecznie zmniejszasz różnice pomiędzy wartościami — zwłaszcza dużymi — zmniejszając widoczność wartości odstających. Problem ten nie dotyczy małych wartości i w zależności od sytuacji może, ale nie musi, zależeć Ci na takim działaniu.

Stosowanie okien czasowych

Przyjrzymy się teraz najważniejszym pomocnym w eksploracji danych wykresom, których stworzenie jest zazwyczaj jednym z początkowych etapów przygotowywania analizy.

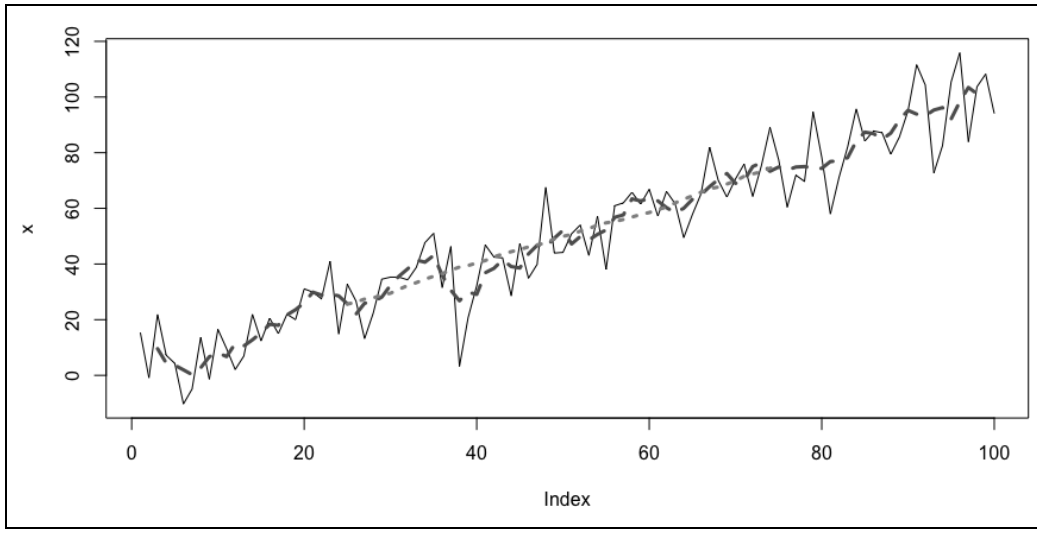
Okna czasowe

Popularną w pracy z szeregami czasowymi funkcją jest okno czasowe, czyli dowolna funkcja pozwalająca zagregować dane w celu ich kompresji (tak jak w omawianym w poprzednim rozdziale downsamplingu) lub wygładzenia (również omówionego w rozdziale 2.). Oprócz omówionych już wcześniej zastosowań dane przetworzone w ten sposób możemy wykorzystać do przygotowania wartościowych wizualizacji.

Przy użyciu dostępnej w R funkcji `filter()` możemy obliczyć średnią ruchomą lub inny liniowo związany z danymi parametr:

```
## R
## Do obliczenia średniej ruchomej można wykorzystać dostępną w R funkcję filter
x <- rnorm(n = 100, mean = 0, sd = 10) + 1:100
mn <- function(n) rep(1/n, n)
plot(x, type = 'l',                lwd = 1)
lines(filter(x, mn( 5)), col = 2, lwd = 3, lty = 2)
lines(filter(x, mn(50)), col = 3, lwd = 3, lty = 3)
```

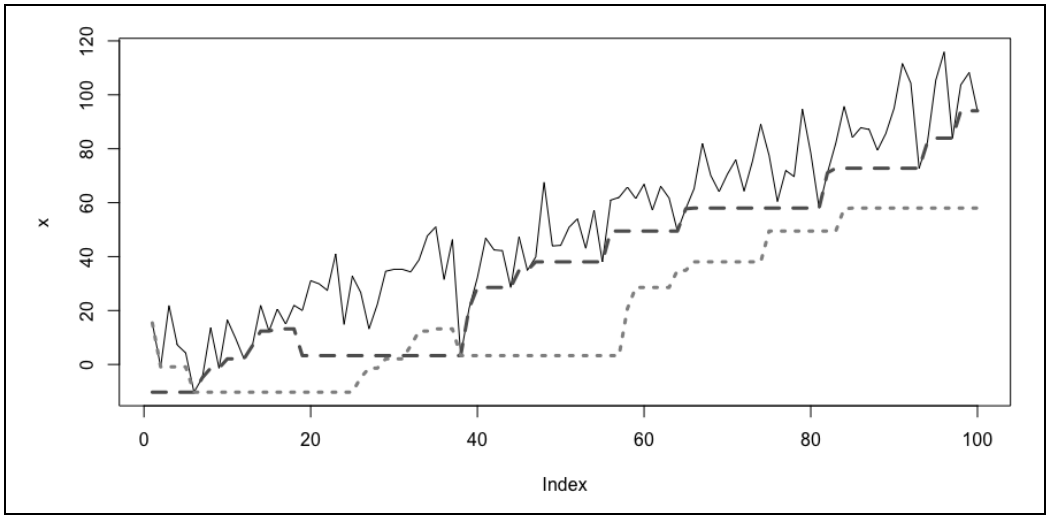
Rysunek 3.6 pokazuje rezultat uruchomienia powyższego kodu.



Rysunek 3.6. Dwie otrzymane w wyniku zastosowania średniej ruchomej krzywe, które można wykorzystać do eksploracji danych. Jeżeli dane są wyjątkowo zaszumione, tego typu krzywe mogą być przydatne w poszukiwaniu trendów lub rozróżnieniu interesujących odchyleń od zwykłego szumu

Ponieważ działanie funkcji `filter()` opiera się na liniowej transformacji danych w oknie, to w przypadkach, w których interesujący nas parametr nie jest kombinacją liniową danych znajdujących się w oknie, zamiast z `filter()` powinniśmy skorzystać z dostępnej w pakiecie *zoo* funkcji `rollapply()` (rysunek 3.7.):

```
## R
## rollapply() pozwala dostosować działanie do potrzeb użytkownika
require(zoo)
f1 <- rollapply(zoo(x), 20, function(w) min(w),align = "left", partial = TRUE)
f2 <- rollapply(zoo(x), 20, function(w) min(w), align = "right", partial = TRUE)
plot (x,           lwd = 1,         type = 'l')
lines(f1, col = 2, lwd = 3, lty = 2)
lines(f2, col = 3, lwd = 3, lty = 3)
```



Rysunek 3.7. Dwie krzywe otrzymane w wyniku zastosowania ruchomego minimum z wyrównaniem do lewej (ciemniejsza linia przerywana) lub prawej (jaśniejsza linia przerywana). Wyrównanie do lewej bazuje na danych z przyszłości, a wyrównanie do prawej opiera się jedynie na przeszłości (ma to znaczenie w przypadku zapobiegania zjawisku lookahead). Zastosowanie wyrównania do lewej może być użyteczne w odpowiedzi na pytania dotyczące przydatności informacji z przyszłości w przeszłości. Jeżeli uzyskane w ten sposób informacje z przyszłości nie są przydatne, związana z nimi zmienna nie jest wartościowa w kontekście analizy szeregów czasowych

Korzystając z pakietu *zoo*, stosuj obiekty typu `zoo`. W przypadku bezpośredniego przekazania do funkcji `rollapply()` wektora danych numerycznych argument `align` nie zadziała. Możesz to sprawdzić, usuwając wrapper `zoo()` w powyższym przykładzie — w rezultacie otrzymasz identyczne krzywe, a R nie poinformuje Cię o zaistnieniu problemu (który w przypadku szeregów czasowych może spowodować niezamierzone wystąpienie zjawiska *lookahead*). Przykład ten nie jest odosobniony, a tego typu „błędy" pojawiają się w wielu popularnych pakietach do R i innych języków skryptowych. Dlatego pamiętaj o częstym sprawdzaniu kodu i miej się na baczności!

Jeżeli chcesz ograniczyć liczbę wykorzystywanych pakietów, możesz również zaimplementować własną wersję okna czasowego. W takim przypadku zalecam najpierw zapoznanie się z kodem źródłowym podobnej metody w jakimś popularnym pakiecie (na przykład w *zoo*, *https://perma.cc/5LTP-Q45T*). Istnieje wiele problemów związanych z implementacją okna, pojawiających się nawet w jednowymiarowych szeregach czasowych. Należy zastanowić się nad tym, jak traktować wartości `NA`, oraz opracować działanie metody w miejscach, w których dostępna będzie mniejsza od rozmiaru okna ilość danych (początek i koniec szeregu).

Różne typy danych służące do przechowywania szeregów czasowych w R

We wcześniejszych fragmentach tego rozdziału korzystaliśmy z typu `ts`, dostępnego w R. W tej części stosujemy obiekty typu `zoo`. Zawsze zwracaj uwagę na obiekty typu `xts`. Oto krótkie podsumowanie sposobów, w jakie typy `zoo` i `xts` rozszerzają typ `ts`:

- Obiekty typu `ts` zakładają, że przechowywany szereg ma stałą częstotliwość próbkowania i zawiera komplet informacji. Z tego powodu w typie `ts` nie są przechowywane znaczniki czasu dla każdej wartości, lecz jedynie informacje o początku, końcu i częstotliwości danego szeregu.
- Typ `ts` wspiera cykle rekurencyjne takie jak lata czy miesiące.
- Obiekty typu `zoo` przechowują znaczniki czasu i nie wymagają, aby pomiędzy danymi występowały regularne odstępy czasu.
- Obiekty typu `zoo` mogą zostać wyświetlone zarówno horyzontalnie, jak i wertykalnie.
- Dane w obiekcie `zoo` mogą być przechowywane jako wektor lub macierz.
- Typ `xts` jest rozszerzeniem typu `zoo` i oferuje jeszcze więcej możliwości.

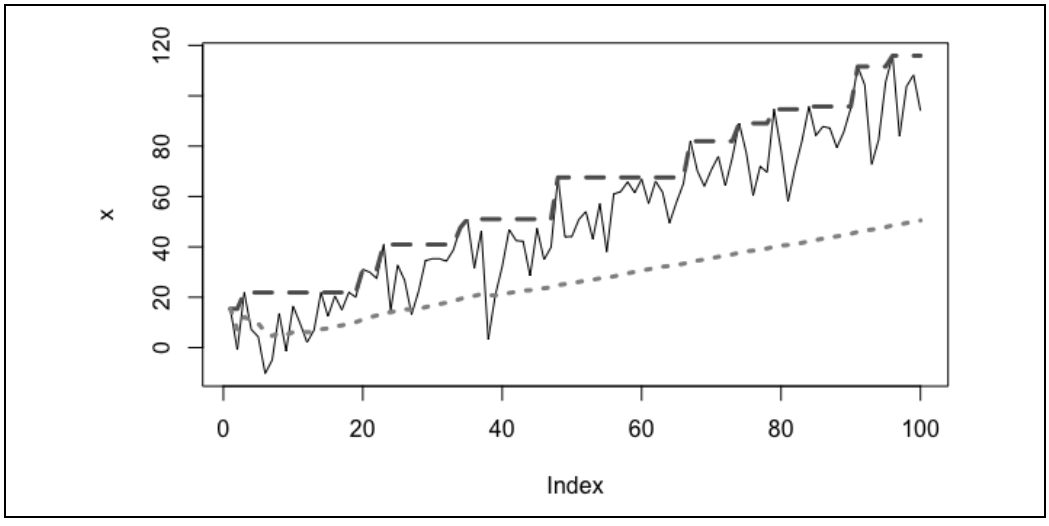
Rozszerzające się okna

Ze względu na ograniczone zastosowanie w analizie szeregów czasowych okna o zmiennej, rosnącej szerokości (ang. *expanding windows*) są mniej popularne od zwykłych okien czasowych. Ich użycie ma sens jedynie wtedy, gdy chcesz oszacować jakąś statystykę o stabilnym (a nie zmiennym lub oscylacyjnym) charakterze. Na początku swojego działania okno takie przyjmuje określony początkowy rozmiar, a następnie wraz z upływem czasu, zamiast przesuwać się i zachowywać swój stały rozmiar, rozszerza się, obejmując kolejne punkty.

Ponieważ rozszerzające się okno umożliwia spojrzenie w głębszą przeszłość szeregu, otrzymane w wyniku jego zastosowania szacunki są zazwyczaj dokładniejsze. Jego działanie sprawdza się tylko w przypadku założenia stacjonarności obliczanego parametru. Okno takie może pomóc w przygotowywaniu statystyk „online", w których kolejne wartości pojawiają się w czasie rzeczywistym.

Dostępne w pakiecie standardowym R funkcje takie jak `cummax` i `cummin` są implementacjami okien o zwiększającej się szerokości. Z łatwością możesz zmodyfikować działanie funkcji `cumsum` tak, aby obliczała średnią. Rysunek 3.8 prezentuje rezultaty obliczania średniej i maksimum przy użyciu okien tego typu.

```
## R
# Rozszerzające się okna
plot(x, type = 'l', lwd = 1)
lines(cummax(x),            col = 2, lwd = 3, lty = 2) # maksimum
lines(cumsum(x)/1:length(x), col = 3, lwd = 3, lty = 3) # średnia
```



Rysunek 3.8. Linią przerywaną z długimi kreskami oznaczono maksimum, a linią z krótkimi kreskami — średnią. W przypadku maksimum zastosowanie okna o rozszerzającej się szerokości powoduje odkreślenie największej wartości, jaka pojawiła się do danego momentu w czasie, i czyni z maksimum funkcję monotoniczną. Dzięki możliwości uwzględnienia wszystkich danych z przeszłości ogólny trend zmian jest mniej widoczny, a uzyskana średnia jest niższa niż w przypadku zwykłego okna czasowego (rysunek 3.7). W zależności od charakteru badanego zjawiska założenia te mogą być dobre lub złe

Podobnie jak w przypadku okna czasowego, jeżeli chcesz zastosować własną funkcję w oknie rozszerzającym, możesz użyć do tego celu `rollapplly()`. Ponieważ rozmiar okna będzie się zmieniać, tym razem zamiast jego rozmiar musisz podać ciąg kolejnych szerokości. Poniższy kod spowoduje powstanie identycznego jak na rysunku 3.8 wykresu. Tym razem jednak zamiast wbudowanych w R funkcji wykorzystaliśmy `rollapply()`:

```
## R
plot(x, type = 'l', lwd = 1)
lines(rollapply(zoo(x), seq_along(x), function(w) max(w),
                        partial = TRUE, align = "right"),
         col = 2, lwd = 3, lty = 2)
lines(rollapply(zoo(x), seq_along(x), function(w) mean(w),
                        partial = TRUE, align = "right"),
         col = 3, lwd = 3, lty = 3)
```

Niestandardowe funkcje w oknach

W dziedzinach, w których znane są prawa rządzące zachowaniem danego zjawiska, lub w analizie, w której kierujemy się jakąś heurystyką, o możliwości obliczenia wartości jakiejś niestandardowej funkcji w danym oknie zależy nam jedynie w pewnych przypadkach.

Możemy na przykład chcieć uwzględnić w oknie jakieś szczególne, wynikające z charakterystyki obszaru, z którego pochodzi szereg, cechy. Być może chcemy ustalić, kiedy w naszym szeregu pojawiają się jakieś wartościowe, monotoniczne trendy (na przykład wzrost stężenia cukru we krwi), a kiedy

mamy do czynienia jedynie z wahaniami góra/dół, spowodowanymi szumem w urządzeniu pomiarowym. W tym celu możemy opracować własną funkcją i zastosować ją w oknie rozszerzającym się lub czasowym.

Związki pomiędzy wartościami w szeregu

U podstaw autokorelacji leży idea mówiąca, że między dwiema różniącymi się w czasie wartościami w jednym szeregu może istnieć jakiś związek. Zauważ, że w tym znaczeniu termin „autokorelacja" odnosi się do pewnej ogólnej koncepcji, a nie określenia technicznego, którego znaczenie jest inne.

Jako przykład takich związków rozważmy szereg czasowy zawierający codzienne pomiary temperatury. Powiedzmy, że możesz w nim zauważyć istnienie pewnej korelacji pomiędzy temperaturami zmierzonymi 15 maja i 15 sierpnia każdego roku, na przykład im wyższą temperaturę odnotowano 15 maja, tym cieplej/zimniej było 15 sierpnia. Może Ci się wydawać, że odkrywszy tę zależność, poznałeś interesującą własność klimatu dotyczącą pewnej długoterminowej przewidywalności. Z drugiej strony, być może ustaliłeś coś przeciwnego, a mianowicie, że związek pomiędzy tymi wartościami jest bliski zeru, czyli, innymi słowy, że znajomość temperatury z 15 maja nie pozwala przewidzieć jej wartości z 15 sierpnia. Myślę, że przykład ten pokazuje w uproszczeniu, czym jest autokorelacja.

Pojęcie autokorelacji jest rozszerzeniem idei pochodzących z powyższego przykładu i zakłada istnienie związków pomiędzy wartościami, które nie muszą być zakotwiczone w konkretnych punktach czasu. Autokorelacja stawia bardziej ogólne pytanie o istnienie związku między dwoma dowolnymi punktami w szeregu czasowym, które znajdują się w określonej od siebie odległości.

Funkcja autokorelacji

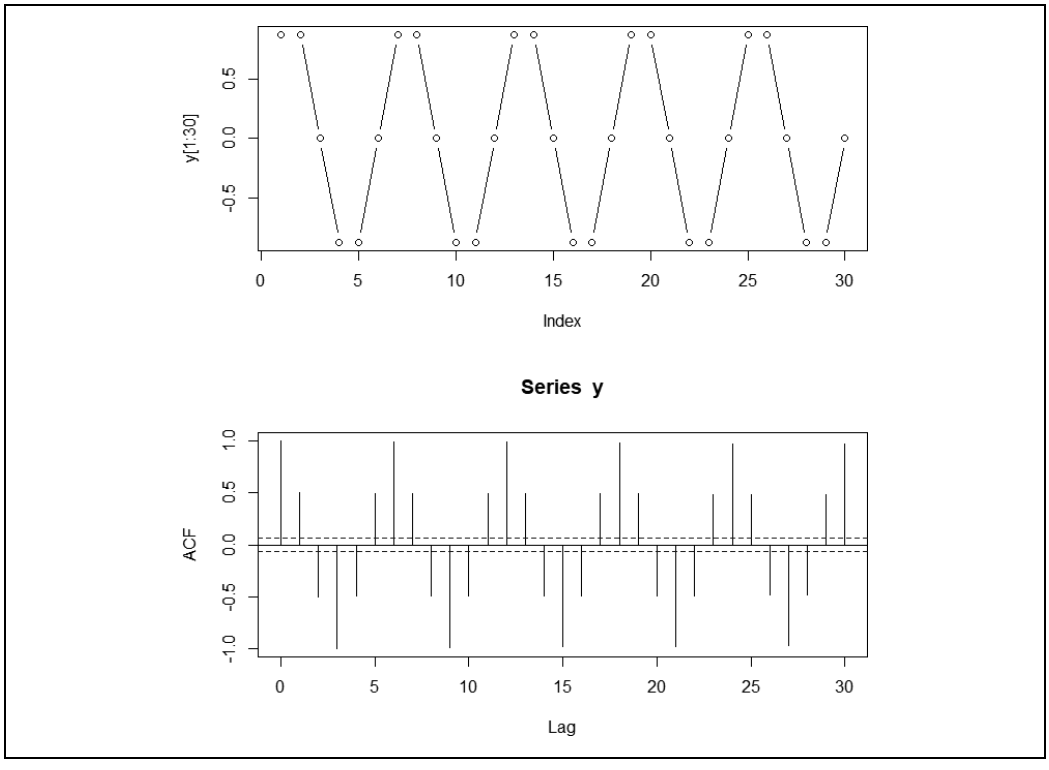
Zacznijmy od zapoznania się z doskonałą definicją autokorelacji zamieszczoną w anglojęzycznej Wikipedii (*https://perma.cc/U8JY-QD7U*):

> **Autokorelacja** to związek pomiędzy sygnałem a jego opóźnioną kopią w funkcji opóźnienia. Nieformalnie jest to podobieństwo między obserwacjami w funkcji opóźnienia czasowego między nimi.

Mówiąc prościej, autokorelacja daje nam wyobrażenie o liniowym powiązaniu pomiędzy danymi znajdującymi się w szeregu w zależności od odległości pomiędzy nimi.

Funkcję autokorelacji (ang. *autocorrelation function* — ACF) można zrozumieć za pomocą wykresu stworzonego na przykład w R (rysunek 3.9):

```
## R
x <- 1:100
y <- sin(x * pi /3)
plot(y, type = "b")
acf(y)
```



Rysunek 3.9. Wykres funkcji sinus i jej funkcji autokorelacji

Korelacja w systemie deterministycznym

Zachowanie opisanego znaną i prostą funkcją sinus szeregu z rysunku 3.9 przy znajomości danych wejściowych jest w pełni deterministyczne. Zastanówmy się więc, dlaczego korelacja nie jest równa 1. Spróbuj przyjrzeć się szeregowi i zastanów się, co właściwie oblicza funkcja ACF. Po chwili zdasz sobie sprawę, że dla wielu wartości kolejna wartość może wzrosnąć lub zmniejszyć się, w zależności od tego, gdzie w cyklu występuje dany punkt. Jeżeli znasz kilka kolejnych wartości, możesz ustalić kierunek, w którym zmierza proces, ale znając tylko jedną — nie (a to właśnie sprawdza funkcja autokorelacji). Z tego powodu, ponieważ po większości wartości nie występuje jedna, unikalna wartość, współczynnik autokorelacji jest mniejszy od 1. Pamiętaj, że współczynnik autokorelacji mniejszy od jedności nie oznacza, że masz do czynienia z zaszumionym lub statystycznym szeregiem czasowym.

Z wykresu funkcji autokorelacji możemy odczytać, że wartość współczynnika korelacji wynosi 1 w przypadku, gdy odległość pomiędzy punktami wynosi 0 (zależność ta występuje w każdym szeregu czasowym). Punkty różniące się od siebie o 1 mają korelację równą 0,5. Dla punktów różniących się o 2 korelacja jest równa –0,5 itd.

Obliczenie autokorelacji jest bardzo proste, możemy zrobić to sami, korzystając z dostępnej w pakiecie *data.table* funkcji `shift()`:

```
## R
> cor(y, shift(y, 1), use = "pairwise.complete.obs")
[1] 0.5001531
> cor(y, shift(y, 2), use = "pairwise.complete.obs")
[1] -0.5037152
```

Otrzymane wyniki są zbliżone do zaprezentowanych na rysunku 3.9[3]. Chociaż samodzielne zaimplementowanie autokorelacji nie jest trudne, zwykle lepiej jest skorzystać z gotowych rozwiązań, takich jak na przykład funkcja `acf()`. W odróżnieniu od napisanych samodzielnie mają one wiele zalet:

- Automatycznie tworzą wykres z odpowiednio dobranymi osiami i opisami.
- Ograniczają (przynajmniej zazwyczaj, parametr ten można modyfikować) maksymalną liczbę opóźnień branych pod uwagę przy obliczaniu wartości ACF.
- Obsługują szeregi wielu zmiennych.

Funkcja ACF ma też kilka interesujących właściwości matematycznych:

- Autokorelacja funkcji okresowej ma taki sam okres jak funkcja pierwotna (co widać na poprzednich wykresach).
- Autokorelacja sumy funkcji okresowych jest sumą autokorelacji każdej funkcji (możesz to łatwo sprawdzić za pomocą kilku linii kodu).
- W każdym szeregu czasowym dla zerowego opóźnienia autokorelacja jest równa 1.
- Autokorelacja próbki białego szumu jest równa 0 dla każdego opóźnienia różnego od 0.
- Funkcja ACF jest symetryczna względem dodatnich i ujemnych opóźnień — wystarczy więc rozpatrywać tylko dodatnie wartości. Aby to udowodnić, możesz spróbować narysować wykres ACF, korzystając z ręcznie obliczonych wartości.
- Reguła statystyczna pozwalająca na ocenę, czy występująca w szeregu autokorelacja jest znacząca, mówi, że „obszar krytyczny", w którym należałoby uznać autokorelację za nieznaczącą, ograniczony jest przez $\pm 1{,}96 \cdot \sqrt{n}$. Formuła ta wymaga wystarczająco dużej próby i skończonej wartości wariancji badanego procesu.

Funkcja autokorelacji cząstkowej

Funkcja autokorelacji cząstkowej (ang. *partial autocorrelation function* — PACF) może być trudniejsza do zrozumienia niż ACF. Cząstkowa autokorelacja szeregu czasowego dla danego opóźnienia jest cząstkową korelacją szeregu czasowego z samym sobą, która uwzględnia wszystkie informacje pojawiające się między analizowanymi punktami w czasie.

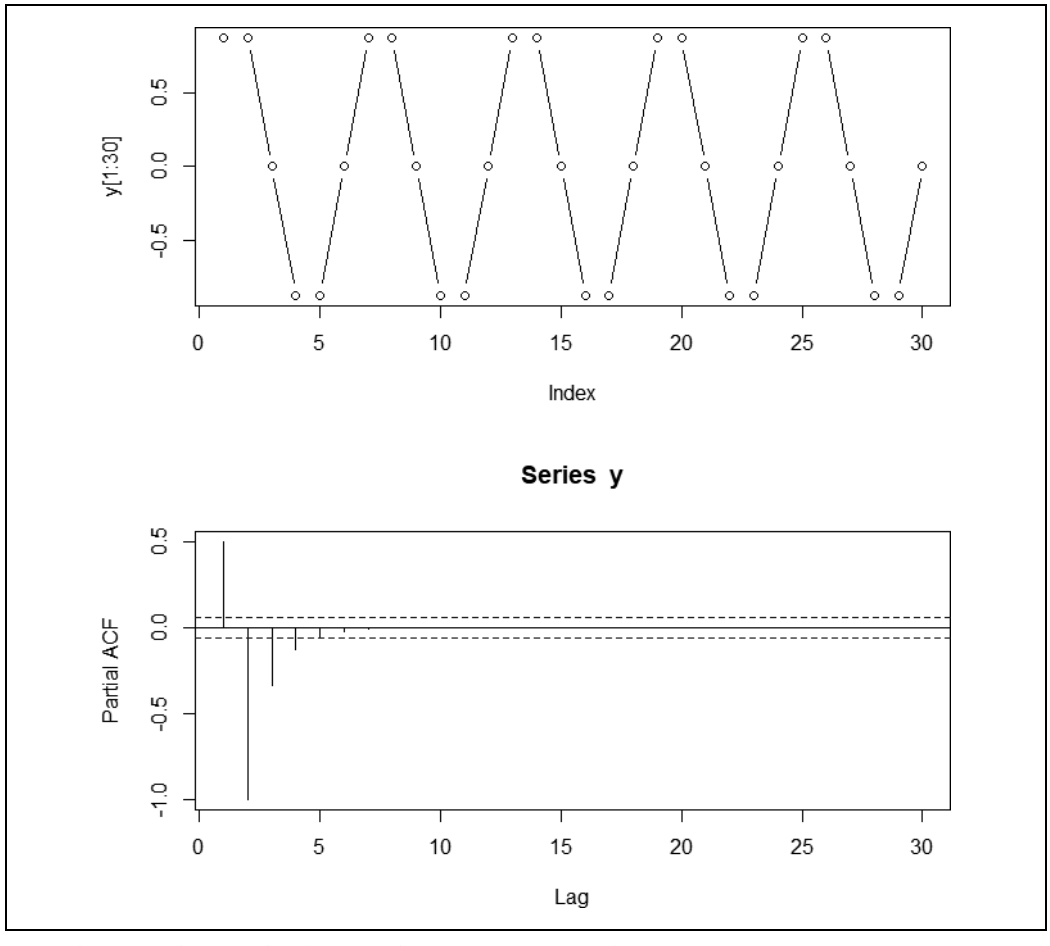
Zamiast machnąć ręką i stwierdzić, że brzmi to rozsądnie, warto zastanowić się, co dokładnie oznacza uwzględnienie wszystkich informacji między dwoma punktami w czasie. Wykonując obliczenia, musisz ustalić wartości wielu korelacji warunkowych, a następnie odjąć je od korelacji

[3] Otrzymane wyniki nie są dokładnie takie same, ponieważ funkcje `cor()` i `acf()` wykorzystują w trakcie obliczeń inne dzielniki. Jeżeli chcesz dowiedzieć się więcej na ten temat, zajrzyj do dyskusji w serwisie StackExchange (*https://perma.cc/M7V6-HN5Y*).

całkowitej. Ponieważ obliczenie wartości PACF nie jest proste, powstało wiele metod jej szacowania. Nie będę ich tutaj omawiać, osoby zainteresowane odsyłam do dokumentacji R i Pythona.

Ponieważ od strony teoretycznej autokorelacja cząstkowa jest skomplikowana, łatwiej jest zrozumieć jej działanie i uświadomić sobie użyteczność, obserwując wykresy (rysunek 3.10):

```
## R
y <- sin(x * pi /3)
plot(y[1:30], type = "b")
pacf(y)
```



Rysunek 3.10. Wykres autokorelacji cząstkowej sezonowego, pozbawionego szumu procesu

W przypadku szeregu wartości sinusa rezultat obliczeń autokorelacji cząstkowej znacząco różni się od klasycznej autokorelacji. Wykres PACF pokazuje, które punkty mają charakter informacyjny, a które stanowią jedynie kolejne harmoniczne krótszych okresów czasowych.

W przypadku sezonowego, pozbawionego szumu procesu, takiego jak funkcja sinus o okresie T, wartości funkcji ACF będą się powtarzały w kolejnych okresach: T, $2T$, $3T$, …, aż do nieskończoności (ACF nie wyeliminuje zbędnych korelacji). W odróżnieniu od niej autokorelacja cząstkowa ujawnia,

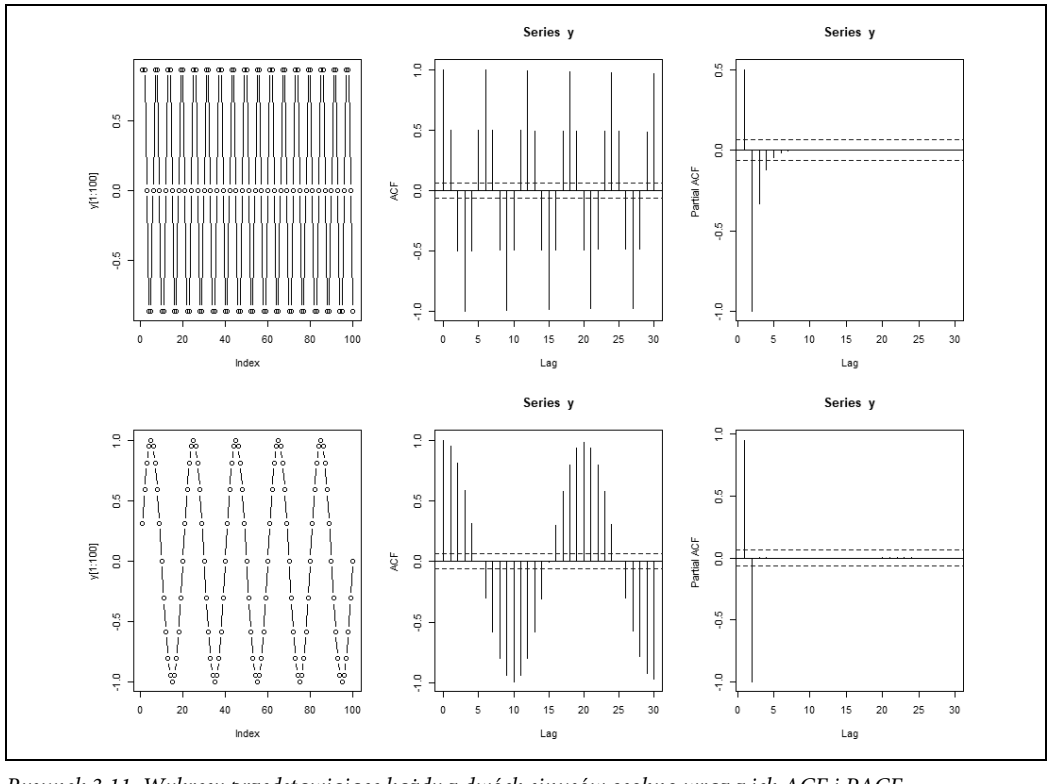
które z korelacji dla danego opóźnienia są tymi „prawdziwymi”, i eliminuje redundancje. Informacja ta jest nie do przecenienia, ponieważ dzięki niej wiemy, ile danych musimy zebrać, aby móc zastosować odpowiednie do naszej skali czasowej okno.

Podobnie jak w przypadku ACF, o znaczącej wartość PACF możemy mówić, jeżeli jej wartość znajduje się poza zakresem $\pm 1{,}96 \cdot \sqrt{n}$. Wszelkie opóźnienia z wartościami PACF mieszczącymi się w tym zakresie są nieznaczące.

Dotychczas przyglądaliśmy się tylko przykładom danych całkowicie pozbawionych szumów, w których występowała tylko jedna zmienność okresowa. Przeanalizujmy teraz bardziej skomplikowany przykład. Przyjrzyjmy się sumie dwóch krzywych sinusoidalnych kolejno przy braku szumów, niskim poziomie szumów i wysokim poziomie szumów.

Najpierw spójrzmy na pozbawione szumu wykresy każdego z dwóch szeregów osobno (rysunek 3.11):

```
## R
y = sin(x * pi / 4)
plot(y, type = 'b')
acf(y)
pacf(y)
y = sin(x * pi / 10)
plot(y, type = 'b')
acf(y)
pacf(y)
```

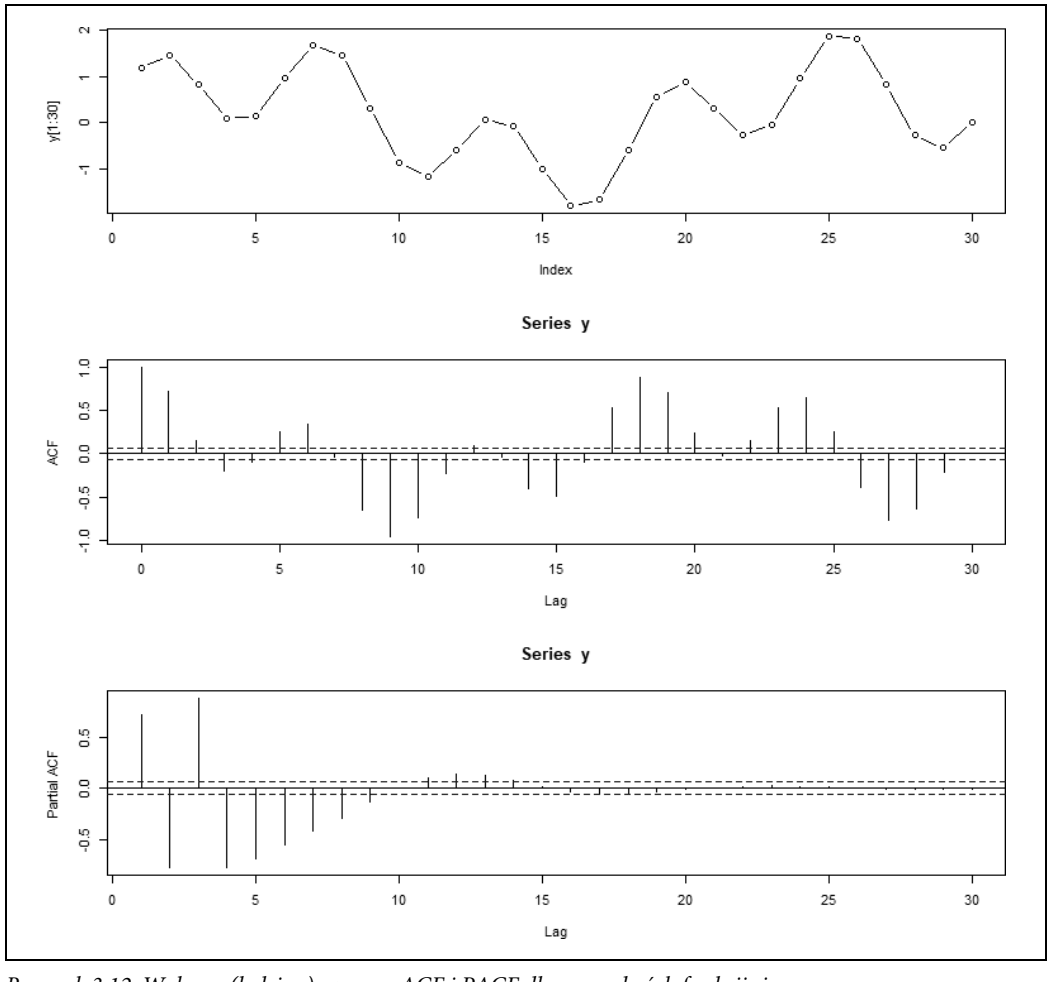


Rysunek 3.11. Wykresy przedstawiające każdy z dwóch sinusów osobno wraz z ich ACF i PACF

Wartość ACF w przypadku szeregu stacjonarnego powinna szybko spadać do zera. W przypadku niestacjonarnym wartość przy opóźnieniu równym 1 jest dodatnia i duża.

Łączymy ze sobą oba szeregi, sumując ich wartości i tworząc te same wykresy dla otrzymanego rezultatu (rysunek 3.12):

```
## R
y <- y1 + y2
plot(y, type = "b")
acf (y)
pacf(y)
```

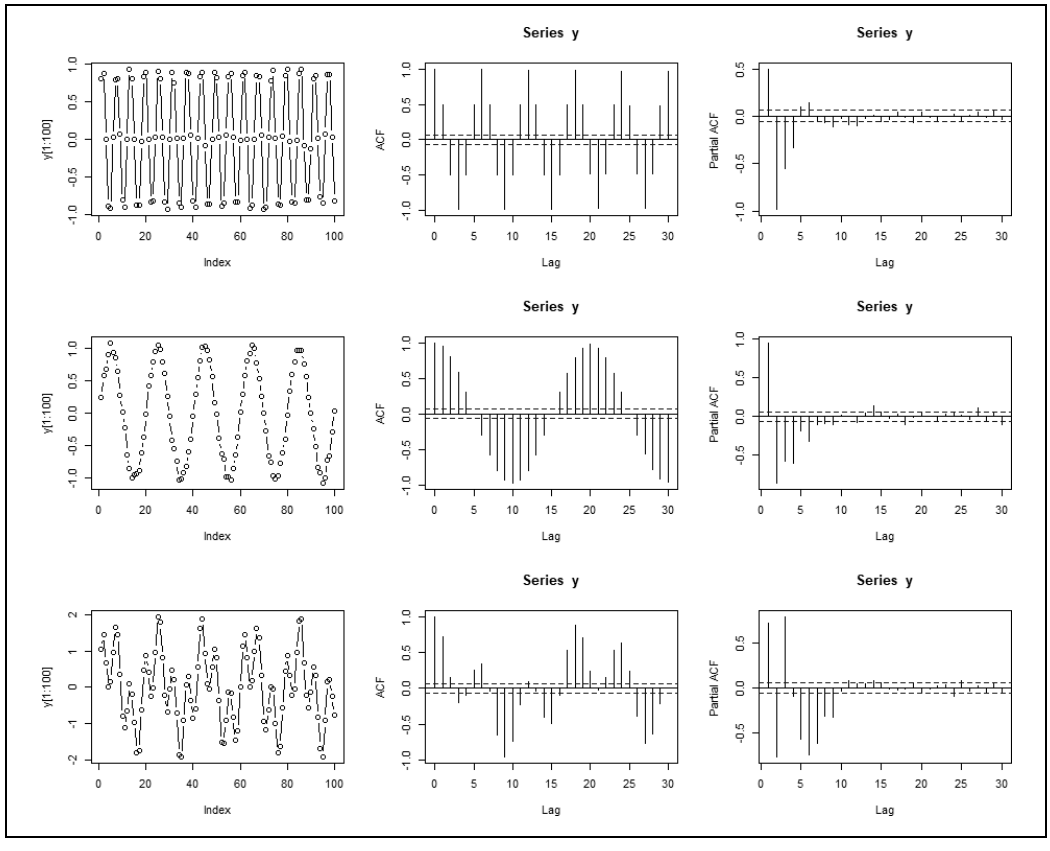


Rysunek 3.12. Wykresy (kolejno) szeregu, ACF i PACF dla sumy dwóch funkcji sinus

Jak widzimy, otrzymany wykres potwierdza omawianą wcześniej własność: ACF sumy dwóch szeregów okresowych jest sumą poszczególnych wartości ACF — najłatwiej zauważyć to, obserwując sekwencje dodatnie → ujemne → dodatnie → ujemne.

Wartość PACF nie jest prostą sumą wartości autokorelacji cząstkowych poszczególnych szeregów. Rezultat otrzymany z wywołania PACF jest wystarczająco prosty do zrozumienia po obliczeniu, ale nie jest go łatwo wygenerować lub przewidzieć. Wynik otrzymany dla sumy sinusów pokazuje nam, że cząstkowa autokorelacja odgrywa w zsumowanym szeregu większą rolę niż w każdej z jego składowych osobno. Wiąże się to z obecnością w szeregu dwóch różnych okresów, które wprowadzają dodatkowe oscylacje wpływające na cykliczność i powodujące zmniejszenie wpływu otoczenia na wartości w punktach.

Spójrzmy na uwzględniający średni szum wariant tej samej sytuacji (rysunek 3.13):

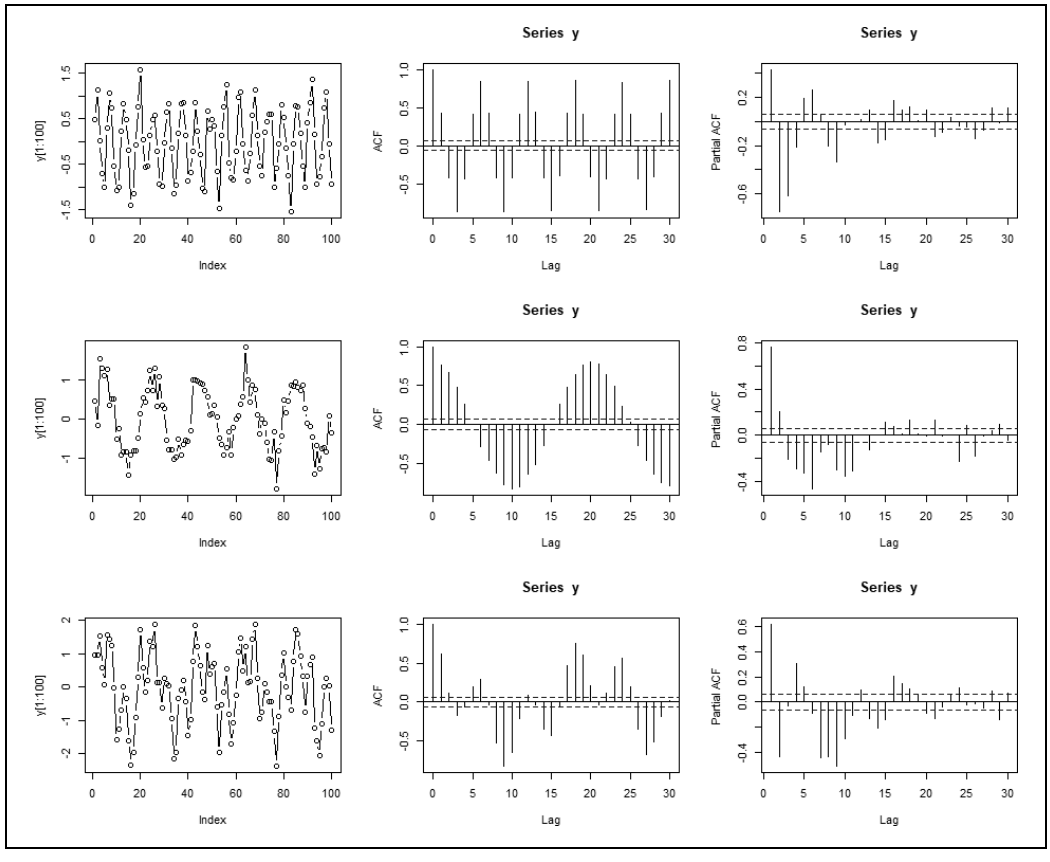


Rysunek 3.13. Wykresy (kolejno) szeregu, ACF i PACF dla sumy dwóch funkcji sinus z uwzględnieniem szumu

```
## R
noise1 <- rnorm(100, sd = 0.05)
noise2 <- rnorm(100, sd = 0.05)
y1 <- y1 + noise1
y2 <-y2 + noise2
y <- y1 + y2
plot(y1, type = 'b')
acf (y1)
pacf(y1)
plot(y2, type = 'b')
acf (y2)
```

```
pacf(y2)
plot(y, type = 'b')
acf (y)
pacf(y)
```

W ostatnim przypadku dodajemy do szeregu jeszcze więcej szumu, tak aby początkowe dane nie przypominały krzywej sinusoidalnej. Pomijamy przykładowy kod, ponieważ jest on, poza większymi wartościami parametrów `sd` w funkcji `rnorm`, identyczny z tym w powyższym przykładzie. Jak widzimy, dodatkowe szumy znacząco utrudniają interpretację wykresów — zwłaszcza w przypadku autokorelacji cząstkowej. Jedyną różnicą pomiędzy wykresami 3.13 i 3.14 jest większa wartość parametru `sd` w szumie.

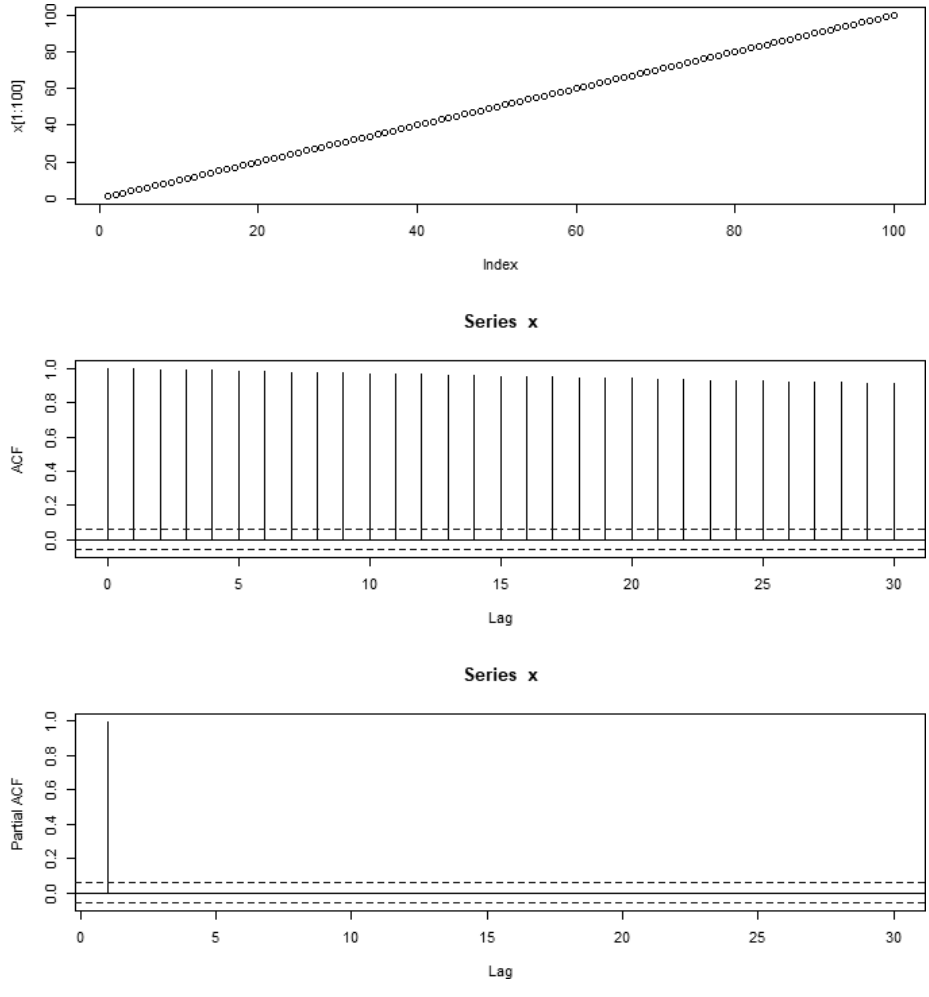


Rysunek 3.14. Wykresy (kolejno) szeregu, ACF i PACF dla sumy dwóch funkcji sinus z uwzględnieniem bardzo dużego szumu

Dane niestacjonarne

Zastanówmy się, jak będą wyglądały wykresy ACF i PACF w przypadku szeregu acyklicznego o wyraźnym trendzie (rysunek 3.15):

```
## R
x <- 1:100
plot(x)
acf (x)
pacf(x)
```

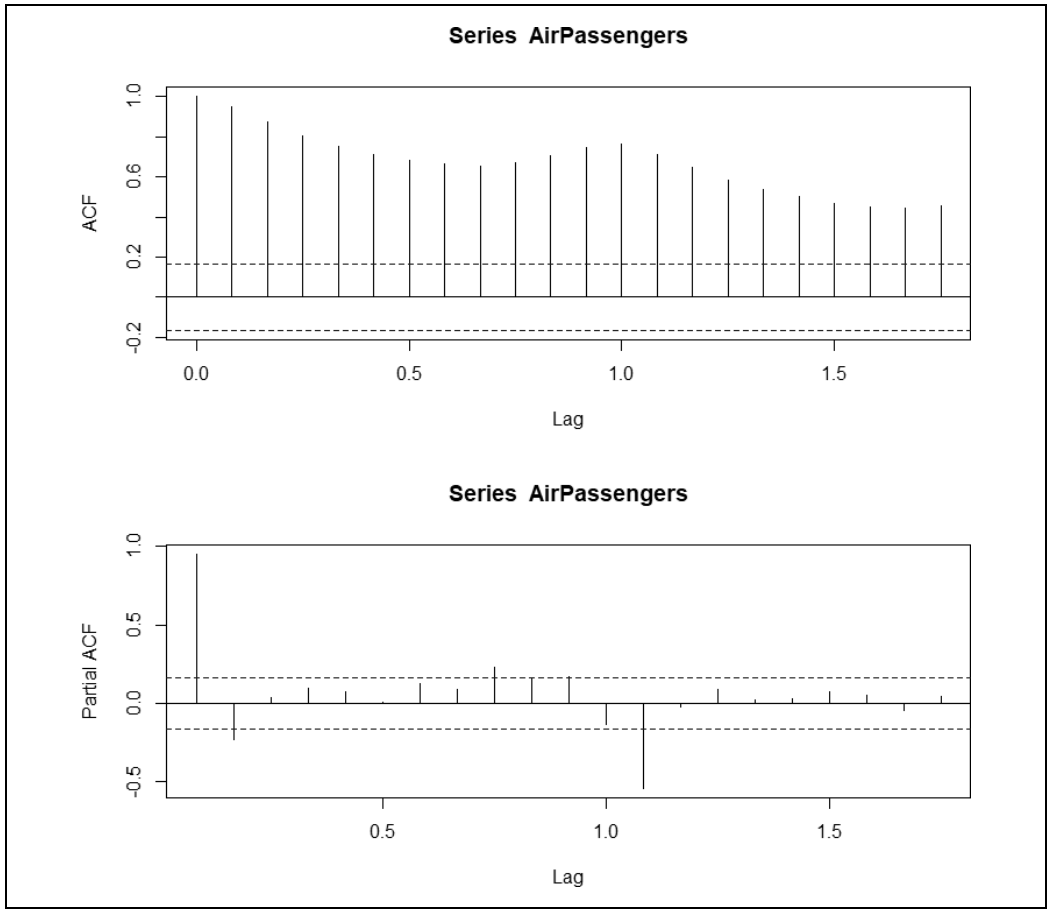


Rysunek 3.15. Wykresy (kolejno) szeregu, ACF i PACF dla szeregu z liniowym trendem

Otrzymany wykres ACF nie dostarcza zbyt wielu informacji, a wartość korelacji dla poszczególnych opóźnień jest taka sama. Wskazuje to jednakową korelację wszystkich opóźnień z danymi (trudno powiedzieć, co to znaczy). Jak widać na załączonym obrazku, o sensowności obliczeń decyduje nie sama możliwość ich wykonania, lecz informacje, jakie można na ich podstawie uzyskać.

Na szczęście otrzymany wykres PACF jest prostszy w analizie i mówi nam, że jedyna istotna korelacja występuje dla opóźnienia jednostkowego. Jest tak, ponieważ znając wartość w punkcie bezpośrednio poprzedzającym analizowany, mamy wszystkie informacje niezbędne do ustalenia wartości w kolejnym. Innymi słowy, w szeregu tym wartość w punkcie jest równa wartości w poprzednim punkcie plus 1.

Na zakończenie przyjrzyjmy się wykresom PACF i ACF otrzymanym z danych rzeczywistych. Na rysunku 3.16 przedstawiono wyniki dla zestawu danych o pasażerach linii lotniczych (`AirPassengers`). Biorąc pod uwagę to, co zobaczyłeś dotychczas, zastanów się, dlaczego w ACF występuje tyle wartości „krytycznych" (odpowiedź: dane zawierają trend) i dlaczego na wykresie PACF wartości „krytyczne" pojawiają się dla dużych opóźnień (odpowiedź: obecność cyklu rocznego, który jest widoczny nawet pomimo obecności trendu).



Rysunek 3.16. Wykresy ACF i PACF dla zestawu danych o pasażerach linii lotniczych. Opóźnienia widoczne na osi x nie są liczbami całkowitymi, ponieważ reprezentują fragmenty kolejnych lat (dane te zapisane są jako obiekt typu ts zawierający informacje o częstotliwości i pozwalający na jej wykorzystanie m.in. do rysowania wykresów)

Korelacje pozorne

Osoby, dla których analiza szeregów czasowych jest czymś nowym, często zaczynają pracę od zastosowania standardowych metod eksploracji takich jak na przykład wykreślanie dwóch zmiennych względem siebie i obliczanie ich korelacji. Nowi, niedoświadczeni analitycy będą zapewne bardzo podekscytowani, gdy już na wczesnym etapie procesu eksploracji zauważą coś, co wydaje się bardzo silną korelacją między danymi wskazującą na związek pomiędzy nimi. Z każdym kolejnym krokiem poszukiwań osoby takie będą znajdywały inne, zaskakująco wysokie korelacje. W końcu, żałując, że prace z szeregami czasowymi rozpoczęły tak późno, i myśląc, że wszystko, czego dotkną, zamienia się w złoto, osoby takie zaczną opowiadać, jaki to niesamowity system trafił się im do analizy.

Niestety w kolejnym kroku osoba taka straci entuzjazm. W najlepszym przypadku wskutek lektury na temat analizy szeregów czasowych, a w najgorszym po tym, jak przedstawi swoje odkrycia komuś innemu i zda sobie sprawę, że otrzymane wyniki nie mają sensu. Każdy sceptyk powie takiemu analitykowi, że otrzymane przez niego wartości korelacji są zbyt wysokie i że na ich podstawie można właściwie powiedzieć, iż każde dowolne dwie wartości są ze sobą powiązane. Kolejne problemy pojawią się, gdy analityk zabierze się za następny zestaw danych i — ku własnemu zaskoczeniu — odkryje, że i on ma zadziwiająco wysoką korelację. W pewnym momencie stanie się jasne, że tak wysokie korelacje pojawiają się nawet pomimo braku hipotezy pozwalającej je wyjaśnić.

Ta historia bardzo przypomina wczesną historię ekonometrii. W XIX wieku, kiedy ekonomiści zaczęli myśleć o idei cyklu koniunkturalnego, niektórzy badacze rozpoczęli poszukiwania zewnętrznych czynników napędzających zmiany na rynkach, upatrując ich przykładowo w plamach słonecznych (cykl 11-letni) lub różnych cyklach meteorologicznych (na przykład przewidywany 4-letni cykl opadów). W każdym z tych przypadków uzyskiwali oni bardzo pozytywne i silnie skorelowane wyniki, nawet jeśli brakowało im hipotezy, która by je wyjaśniała.

Wielu ekonomistów i statystyków pozostawało jednak słusznie sceptycznymi wobec tych rewelacji. Udny Yule formalnie zbadał ten problem w artykule zatytułowanym *Why do we Sometimes get Nonsense-Correlations?* (*https://www.jstor.org/stable/2341482*). Wkrótce powstał również nowy obszar prowadzenia badań, który po dziś dzień naukowcom zarówno sprawia przyjemność, jak i przysparza kłopotów. Unikanie pozornych korelacji pozostaje ważnym problemem, a ich obecność stanowi przedmiot żarliwych dyskusji i sporów, w których jedna strona próbuje wykazać ich obecność, a druga jej zaprzeczyć. Podobnie jeden z argumentów zaprzeczających zmianom klimatu opiera się na argumencie, że korelacja między wzrostem emisji dwutlenku węgla a globalnym wzrostem temperatur jest pozorna i wynika z trendów występujących w tych danych (osobiście nie uważam tego argumentu za przekonujący).

Z biegiem lat ekonomiści odkryli, że dane zawierające trend mogą się przyczyniać do występowania pozornych korelacji. Rozumowanie jest dość proste: skoro szeregi czasowe z trendem zawierają więcej informacji niż ich stacjonarne odpowiedniki, to istnieje w nich więcej czynników wpływających na wartości w punktach.

Oprócz obecności trendów niektóre inne popularne cechy szeregów czasowych mogą sprzyjać powstawaniu pozornych korelacji:

- Sezonowość — pomyśl o występującej w lecie pozornej korelacji pomiędzy spożyciem hot doga a śmiercią wskutek utonięcia.

- Zmiany poziomu lub nachylenia danych wynikające ze zmian reżimu czasowego (ich rezultatem jest powstanie rozkładu przypominającego kształtem hantle do ćwiczeń, który charakteryzuje się nieistotnie wysoką wartością korelacji).
- Obliczanie skumulowanych sum (jest to sztuczka stosowana w niektórych branżach, aby modele lub korelacje wyglądały na lepsze niż w rzeczywistości).

Kointegracja

Kointegracja odnosi się do relacji pomiędzy dwoma szeregami czasowymi. Często stosowanym przykładem jest pijany pieszy i jego pies. Jeżeli dokonamy pomiaru ścieżki spaceru każdego z nich osobno, to otrzymamy pozornie nie związane ze sobą dane, których wartości nie różnią się jednak znacząco.

W przypadku kointegracji tych danych zauważysz wysoką korelację. Trudność polegać będzie na ocenie, czy te dwa procesy są ze sobą powiązane, czy raczej widzisz jedynie korelację pozorną wynikającą z wysokich wartości autokorelacji w obu szeregach. Istotna różnica polega na tym, że w przypadku pozornej korelacji nie istnieją żadne związki pomiędzy danymi, podczas gdy skointegrowane szeregi czasowe są ze sobą silnie powiązane.

Istnieje dobrze znany blog (a teraz także książka) pełen wspaniałych przykładów pozornych korelacji. Jeden z nich zamieściłam na rysunku 3.17. Ilekroć kusi Cię myśl, że znalazłeś szczególny i bardzo silny związek między wartościami, sprawdź swoje dane pod kątem obecności oczywistych przyczyn pozornych korelacji (takich jak trendy).

*Rysunek 3.17. Niektóre z pozornych korelacji mogą być bardzo przekonywujące. Ten wykres pochodzi ze strony Tylera Vigensa zawierającej przykłady pozornych korelacji (*https://perma.cc/6UYH-FPBX*)*

Przegląd użytecznych metod wizualizacji

Wykresy stanowią podstawę dokładnej analizy eksploracyjnej danych czasowych. Z pewnością będziesz chciał wizualizować dane w odniesieniu do osi czasowej — najlepiej w sposób, który odpowiadać będzie na pytania dotyczące zestawu danych takie jak zachowanie konkretnej zmiennej lub ogólny rozkład danych w czasie.

We wcześniejszych podrozdziałach przyjrzeliśmy się niektórym powszechnym technikom wizualizacji danych, takim jak wykreślanie wartości w funkcji czasu lub wykresy punktowe różnych kolumn w czasie. W końcowej części rozdziału poświęconego eksploracyjnej analizie danych omówię kilka metod wizualizacji szczególnie pomocnych w badaniu zachowania szeregów czasowych.

Przyjrzymy się wizualizacjom o różnym stopniu złożoności:

- Jednowymiarowej wizualizacji pozwalającej zrozumieć czasowy rozkład danych w „znalezionym" szeregu czasowym z rozdziału 2.
- Dwuwymiarowemu histogramowi umożliwiającemu zrozumienie typowej trajektorii wartości w czasie w danych pochodzących z wielu równoległych pomiarów danego zjawiska (na przykład dane z wieloletnich i wielokrotnych obserwacji jakiegoś zjawiska fizycznego).
- Trójwymiarowej wizualizacji, w której czas może występować niejawnie (zero wymiarów) lub pojawiać się nawet w dwóch wymiarach.

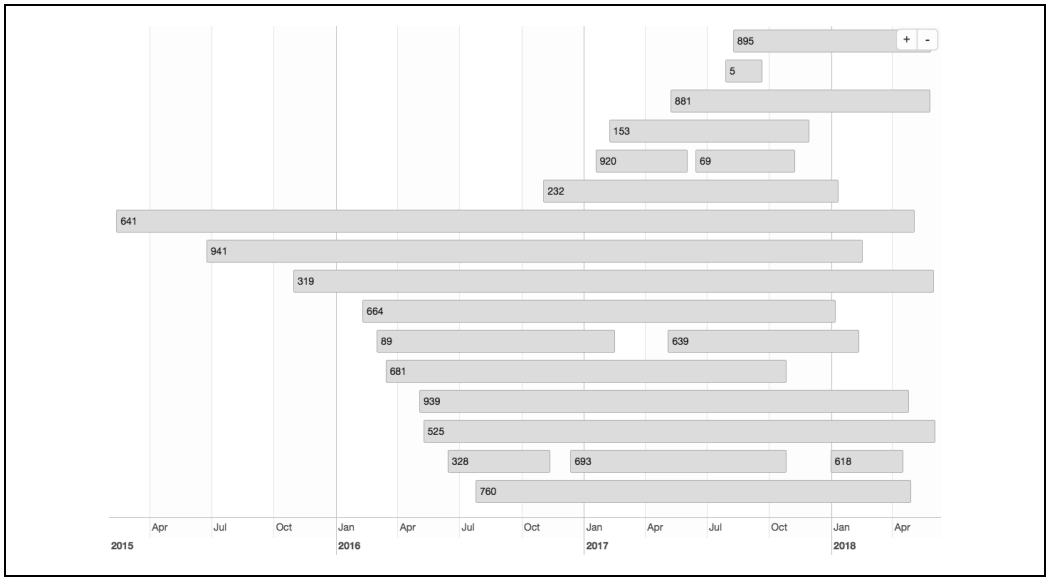
Wizualizacje w jednym wymiarze

W przypadku pomiaru danych dotyczących wielu jednostek (wielu użytkowników, członków itp.) powinniśmy analizować jednocześnie wiele równoległych szeregów czasowych. Przedstawienie ich na jednym wykresie pozwoli podkreślić indywidualny charakter każdej jednostki i odnoszących się do niej ram czasowych. W takim przypadku możemy zignorować zmierzone wartości i potraktować same zakresy czasowe, w których pojawiają się dane, jako interesującą nas informację. W ten sposób sam okres czasu staje się jednostką naszej analizy. Do tego przykładu spośród wielu dostępnych rozwiązań wybieramy dostępny dla środowiska R pakiet *timevis*. Przyjrzyjmy się niewielkiemu podzbiorowi danych o darowiznach z rozdziału 2. (rysunek 3.18):

```
## R
require(timevis)
donations <- fread("donations.csv")
d <- donations[, .(min(timestamp), max(timestamp)), user]
names(d) <- c("content", "start", "end")
d <- d[start != end]
timevis(d[sample(1:nrow(d), 20)])
```

Rysunek 3.18 pozwala dostrzec okresy, w których organizacja miała najwięcej członków. Z wykresu możemy odczytać także pewne wskazówki co do rozkładu darowizn w czasie członkostwa konkretnej osoby.

Będące w użyciu od ponad wieku diagramy Gantta stosowane są najczęściej w zarządzaniu projektami. Diagramy te powstały niezależnie w wielu różnych branżach i są bardzo łatwe w interpretacji.



Rysunek 3.18. Diagram Gantta dla losowej próbki danych może dać pewne wyobrażenie o rozkładzie „aktywności" pośród użytkowników/darczyńców

Pomimo związków z zarządzaniem projektami diagramy Gantta mogą być przydatne w analizach szeregów charakteryzujących się występowaniem pomiarów odnoszących się do wielu niezależnych od siebie podmiotów. Dane z rysunku 3.18 umożliwiły szybką odpowiedź na moje pytania o związki pomiędzy historią członkostwa i wartościami darowizn w całej bazie użytkowników, których odczytanie z tabeli mogłoby być bardzo trudne.

Wizualizacje w dwóch wymiarach

W tej sekcji wykorzystamy dane dotyczące pasażerów linii lotniczych do obserwacji sezonowości i trendów. Ponieważ zmiana czasu może zachodzić na więcej niż jednej osi, nie powinniśmy myśleć o czasie jako o parametrze liniowym. Istnieje oczywiście oś czasu, która przesuwa się liniowo z dnia na dzień i z roku na rok, ale możemy ją również wzbogacić o informacje w kolejnym wymiarze, takie jak zmiany godzin w ciągu dnia, dni w tygodniu itd. Jako że niektóre zachowania mogą się pojawiać o określonej porze dnia lub w określonym dniu miesiąca, podejście takie ułatwia nam analizę sezonowości. Powinniśmy szczególnie skupiać się na sezonowym aspekcie danych i nie myśleć o zmianie czasu jedynie jako o uporządkowanej chronologicznie liniowej osi w danych.

Zanim przejdziemy dalej, przekształcamy dane z typu `ts` na macierz:

```
> t(matrix(AirPassengers, nrow = 12, ncol = 12))
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
 [1,]  112  118  132  129  121  135  148  148  136   119   104   118
 [2,]  115  126  141  135  125  149  170  170  158   133   114   140
 [3,]  145  150  178  163  172  178  199  199  184   162   146   166
 [4,]  171  180  193  181  183  218  230  242  209   191   172   194
 [5,]  196  196  236  235  229  243  264  272  237   211   180   201
 [6,]  204  188  235  227  234  264  302  293  259   229   203   229
 [7,]  242  233  267  269  270  315  364  347  312   274   237   278
```

```
 [8,]  284  277  317  313  318  374  413  405  355  306  271  306
 [9,]  315  301  356  348  355  422  465  467  404  347  305  336
[10,]  340  318  362  348  363  435  491  505  404  359  310  337
[11,]  360  342  406  396  420  472  548  559  463  407  362  405
[12,]  417  391  419  461  472  535  622  606  508  461  390  432
```

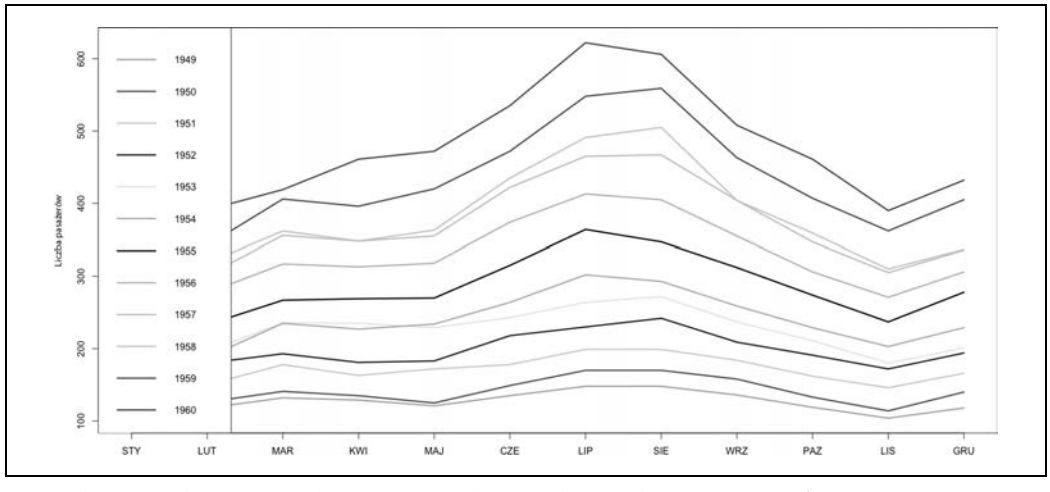
Zauważ, że musieliśmy dokonać transpozycji danych, aby zgadzały się z kolejnością występującą w obiekcie typu `ts`.

Sposób przechowywania danych w tabelach

Przy domyślnych ustawieniach środowiska R kolejne rekordy w danych przechowywane są w kolejnych kolumnach (*https://perma.cc/L4BH-DKB8*). Porządek ten różni się od występującego w bibliotece *NumPy* i większości baz danych SQL porządku wierszowego. Warto się dowiedzieć, jakie są domyślne ustawienia dla Twojego języka/środowiska, nie tylko na potrzeby wizualizacji, ale również w celu efektywnego zarządzania pamięcią.

Na osi *x* wykreślamy kolejne miesiące, dane dzielimy na kolejne lata i każdy rok prezentujemy za pomocą osobnej linii (rysunek 3.19):

```
## R
colors <- c("green", "red", "pink", "blue", "yellow","lightsalmon", "black", "gray",
            "cyan",  "lightblue",   "maroon", "purple")
matplot(matrix(AirPassengers, nrow = 12, ncol = 12),
        type = 'l', col = colors, lty = 1, lwd = 2.5,
        xaxt = "n", ylab = "Passenger Count")
legend("topleft", legend = 1949:1960, lty = 1, lwd = 2.5,
       col = colors)
axis(1, at = 1:12, labels = c("STY", "LUT", "MAR", "KWI",
                              "MAJ", "CZE", "LIP", "SIE",
                              "WRZ", "PAZ", "LIS", "GRU"))
```

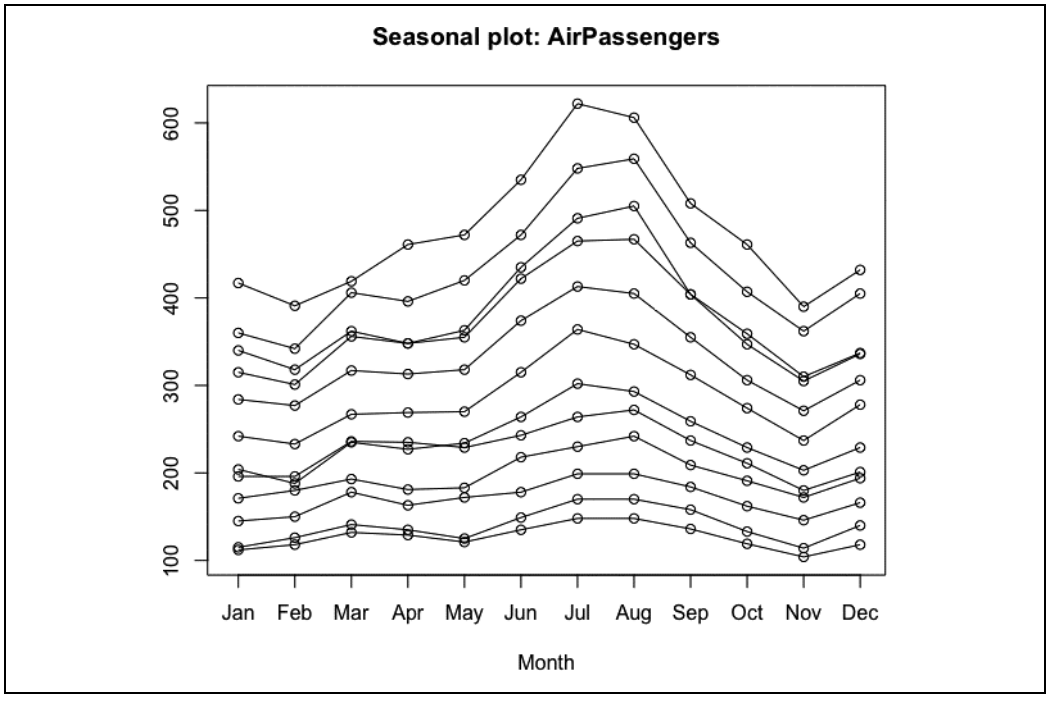


Rysunek 3.19. Liczba pasażerów linii lotniczej w danym roku w podziale na miesiące[4]

[4] Aby zobaczyć ten i kolejne wykresy, narysuj je samodzielnie lub zapoznaj się z materiałami dołączonymi do książki, które są dostępne na serwerze wydawnictwa Helion (*ftp://ftp.helion.pl/przyklady/XXX.zip*).

Ten sam wykres możemy otrzymać także w prostszy sposób, stosując funkcję z pakietu *forecast* (rysunek 3.20):

```
## R
require(forecast)
seasonplot(AirPassengers)
```



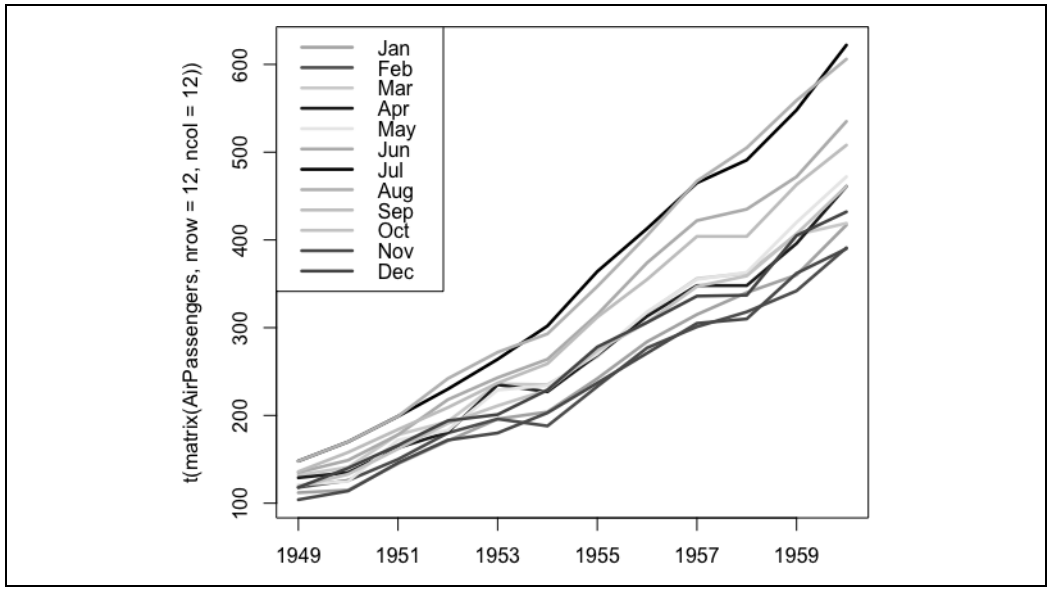
Rysunek 3.20. Podobny wykres możemy otrzymać, korzystając z funkcji seasonplot()

Na osi *x* odkreślamy kolejne miesiące dla wszystkich lat. Każdego roku najwięcej osób odbywało podróż w lipcu lub sierpniu. Lokalny wzrost liczby podróżnych widoczny jest też, przynajmniej w większości lat, w marcu.

Dane pochodzące z różnych lat rzadko się krzyżują. W tamtych latach rozwój lotnictwa był tak dynamiczny, że rzadko się zdarzało, że liczba pasażerów w tym samym miesiącu w różnych latach była taka sama (oprócz kilku wyjątków poza sezonem). Już na podstawie samych obserwacji możemy opracować porady dla przewoźników lotniczych dotyczące ich planów rozwoju.

Odwrotny do poprzedniego wykres krzywych miesięcznych względem rocznych jest rzadziej stosowany, ale także pomocny (rysunek 3.21):

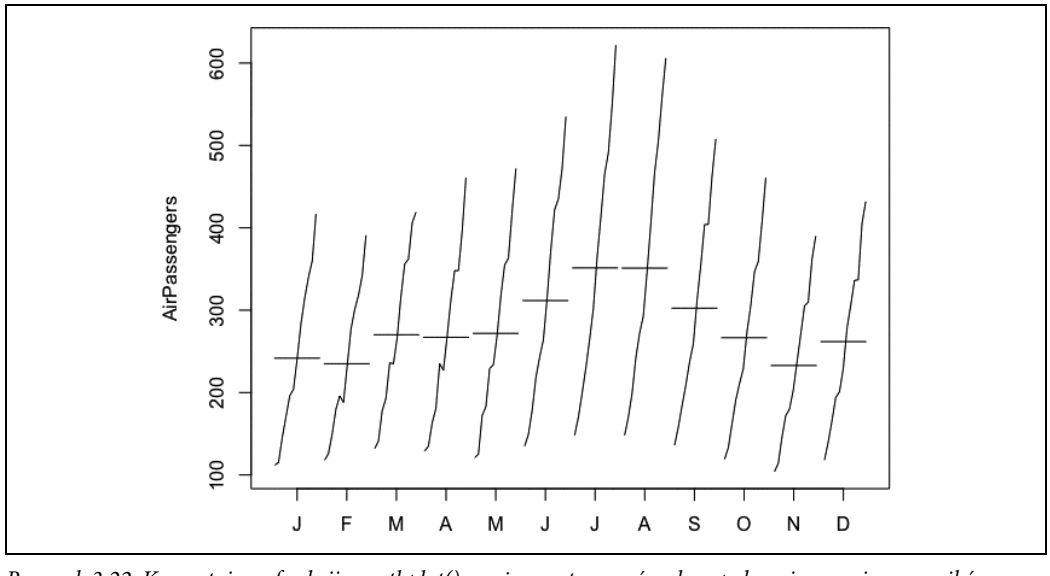
```
## R
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
matplot(t(matrix(AirPassengers, nrow = 12, ncol = 12)),
          type = 'l', col = colors, lty = 1, lwd = 2.5)
legend("left", legend = months,
                      col = colors, lty = 1, lwd = 2.5)
```



Rysunek 3.21. Zmiany w podziale miesięcznym względem lat

Z biegiem lat trend wzrostowy przyspiesza tak jak i samo tempo wzrostu. Ponadto liczba pasażerów w lipcu i sierpniu rośnie szybciej niż w pozostałych miesiącach. Podobne spostrzeżenia i podobny wykres możemy otrzymać także z prostej funkcji dostępnej w pakiecie *forecast* (rysunek 3.22):

```
## R
monthplot(AirPassengers)
```



Rysunek 3.22. Korzystając z funkcji monthplot(), możemy otrzymać wykres pokazujący zmianę wyników w poszczególnych miesiącach na przestrzeni lat

Istnieją dwa ogólne wnioski, jakie możemy sformułować na podstawie tych wykresów:

- W szeregach czasowych występują więcej niż jedna wartościowa oś czasowa, względem której możemy wykreślać wartości. Skorzystaliśmy zarówno z podziału lat względem miesięcy (od stycznia do grudnia), jak i miesięcy względem lat (od pierwszego do dwunastego roku pomiarów).
- W oparciu o wizualizacje z zagregowanym czasem możemy się uzyskać wiele ciekawych i przydatnych w predykcji informacji.

W następnym kroku zajmiemy się przygotowaniem dwuwymiarowego histogramu. W kontekście szeregów czasowych możemy myśleć o dwuwymiarowych histogramach jako połączeniach czasu (lub parametru z nim powiązanego) na jednej osi i interesującej nas wartości na drugiej. Stworzone przez nas wcześniej wykresy zagregowanych danych są na dobrej drodze do dwuwymiarowych histogramów. Przydałoby się im jednak jeszcze kilka modyfikacji:

- Musimy agregować dane zarówno na osi czasu, jak i na osi pasażerów.
- Potrzebujemy więcej danych. Histogram dwuwymiarowy ma sens dopiero w przypadku, gdy zagregowane krzywe nakładają się na siebie. W przeciwnym razie nie przekaże nam on żadnych nowych informacji.

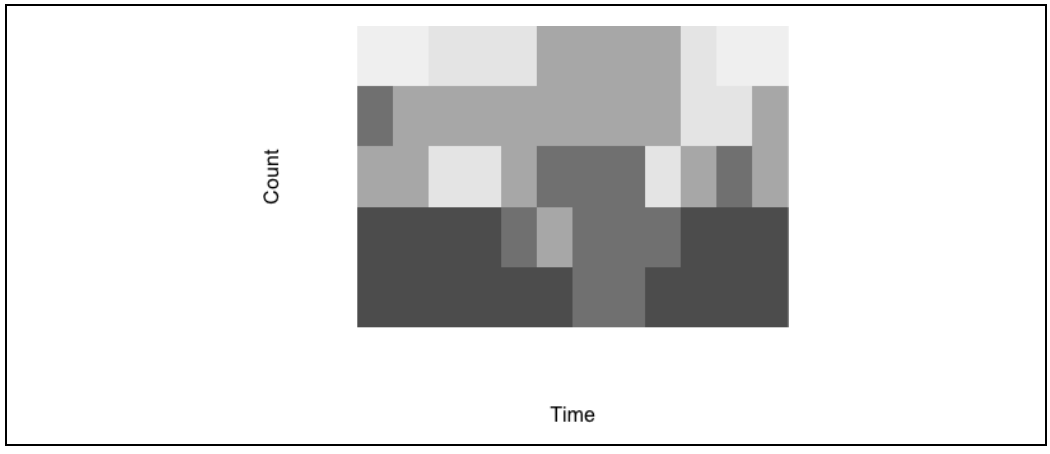
Zanim przejdziemy do poważniejszego przykładu, przyjrzyjmy się histogramowi dla małego zbioru danych. Funkcję odpowiedzialną za jego stworzenie napiszemy sami:

```
##R
hist2d <- function(data, nbins.y, xlabels) {
  ## Zapewniamy równomierne odstępy pomiędzy danymi w ybins,
  ## uwzględniamy również minimum i maksimum
  ymin = min(data)
  ymax = max(data) * 1.0001
  ## Łatwy sposób na poradzenie sobie z problemem przynależności wartości granicznych

  ybins = seq(from = ymin, to = ymax, length.out = nbins.y + 1 )

  ## Tworzenie macierzy zer o odpowiednim rozmiarze
  hist.matrix = matrix(0, nrow = nbins.y, ncol = ncol(data))

  ## Umieszczamy dane w macierzy zgodnie z zasadą, że każdy wiersz zawiera informację o jednym punkcie danych
  for (i in 1:nrow(data)) {
    ts = findInterval(data[i, ], ybins)
    for (j in 1:ncol(data)) {
      hist.matrix[ts[j], j] = hist.matrix[ts[j], j] + 1
    }
  }
  hist.matrix
}
```

Możemy teraz przedstawić dane jako tzw. *heat map*:

```
## R
h = hist2d(t(matrix(AirPassengers, nrow = 12, ncol = 12)), 5, months)
image(1:ncol(h), 1:nrow(h), t(h), col = heat.colors(5),
      axes = FALSE, xlab = "Time", ylab = "Passenger Count")
```

Niestety otrzymany wykres jest niezbyt udany (rysunek 3.23).



Rysunek 3.23. Heat map utworzona na podstawie funkcji generującej histogram dwuwymiarowy

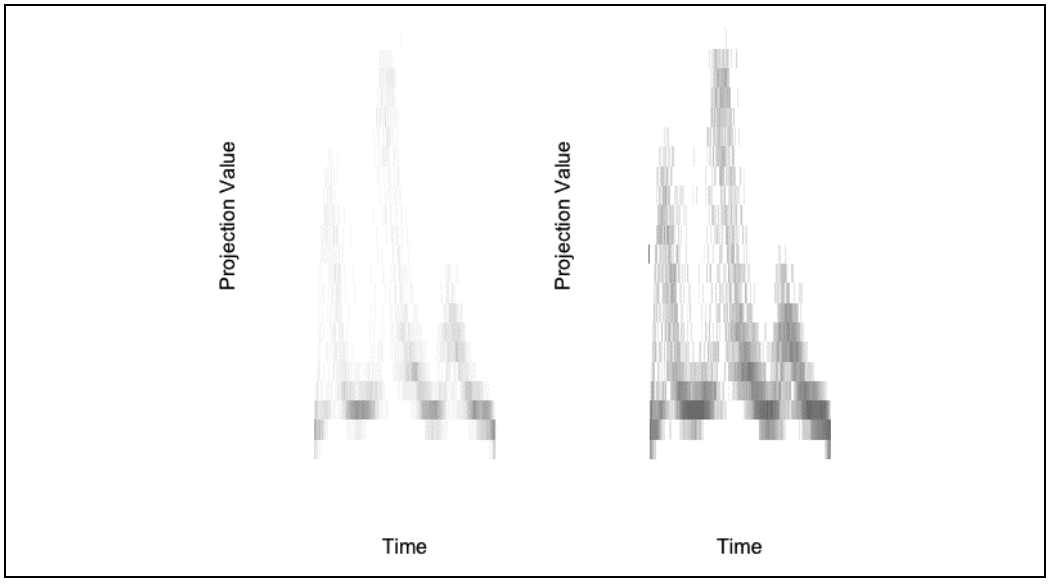
Ze względu na niewystarczającą ilość danych otrzymany wykres nie jest zbyt użyteczny (12 krzywych miesięcznych podzieliliśmy na 5 grup). Dodatkowo histogram wymaga, aby dane były stacjonarne (a nie są). W tym przykładzie w danych istnieje trend, który będzie nam przeszkadzał w obserwacji sezonowości.

Przyjrzymy się teraz kolejnemu, tym razem dużemu i niezanieczyszczonemu trendem zestawowi. Stanowi on podzbiór zestawu *FiftyWords* znajdującego się w *UCR Time Series Classification Archive* (*https://perma.cc/Y982-9FPS*). Zestaw zawiera powtarzające się dane dotyczące 50 różnych słów zapisanych w jednowymiarowych szeregach czasowych (przy czym każdy szereg ma jednakową długość). Zbiór danych użytych do stworzenia rysunku 3.24 pochodzi z prowadzonych kiedyś przeze mnie warsztatów na temat szeregów czasowych i można go pobrać pod adresem *https://oreil.ly/M6T-u* (możesz też do tego celu wykorzystać cały podzbiór):

```
## R
require(data.table)
words <- fread("50words_TEST.csv")
w1 <- words[V1 == 1]
h = hist2d(w1, 25, 1:ncol(w1))
colors <- gray.colors(20, start = 1, end = .5)
par(mfrow = c(1, 2))
image(1:ncol(h), 1:nrow(h), t(h),
      col = colors, axes = FALSE, xlab = "Time", ylab = "Projection Value")
image(1:ncol(h), 1:nrow(h), t(log(h)),
      col = colors, axes = FALSE, xlab = "Time", ylab = "Projection Value")
```

Wykres po prawej stronie rysunku 3.24 wygląda lepiej, ponieważ kolorowanie odbywa się zgodnie z logarytmiczną transformacją pomiarów, a nie ich bezpośrednią wartością.

Jest to zastosowanie identycznego pomysłu co przy logarytmowania szeregu czasowego w celu zmniejszenia wariancji, znaczenia i odległości pomiędzy wartościami odstającymi. Zastosowanie transformacji logarytmicznej poprawia wizualizację, ponieważ nie marnuje ono dużej części zakresu kolorystycznego na stosunkowo rzadkie wysokie wartości.



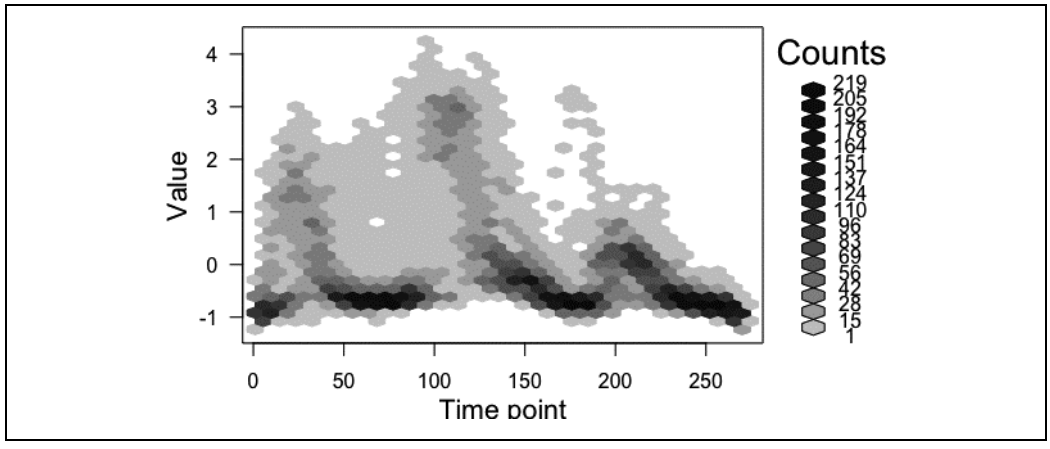
Rysunek 3.24. Dwuwymiarowy histogram wartości dźwięku dla pojedynczego słowa. Lewy wykres ma skalę liniową, a prawy logarytmiczną

Nasze autorskie rozwiązanie nie będzie tak atrakcyjne wizualnie jak rozwiązania wbudowane, ale warto się mu przyjrzeć, aby zobaczyć różnice. Aby skorzystać z zalet gotowych implementacji, musimy przekształcić dane. Funkcje wbudowane oczekują na wejściu par wartości *x,y*, które następnie zamienią w dwuwymiarowy histogram. W przeciwieństwie do naszego rozwiązania gotowe metody nie są zaprojektowane specjalnie dla szeregów czasowych, ale za to oferują świetnie wyglądające wizualizacje (rysunek 3.25):

```
## R
require(hexbin)
w1 <- words[V1 == 1]
## Zmiana danych na pary wartości x,y wymagane przez większość implementacji histogramów 2d
names(w1) <- c("type", 1:270)
w1 <- melt(w1, id.vars = "type")
w1 <- w1[, -1]
names(w1) <- c("Time point", "Value")
plot(hexbin(w1))
```

Wizualizacje w trzech wymiarach

Narzędzia służące do trójwymiarowej wizualizacji danych nie są dostępne w standardowej dystrybucji R, ale można je bez problemu znaleźć w wielu pakietach. W tej sekcji zamieściłam kilka uzyskanych z wykorzystaniem pakietu *plotly* wykresów. Wybrałam go ze względu na łatwość obsługi wykresów w RStudio i możliwość eksportu do formatów webowych. Ponadto zarówno pobranie, jak i instalacja *plotly* są proste i przyjemne (czego nie można powiedzieć o niektórych konkurencyjnych rozwiązaniach).



Rysunek 3.25. Alternatywne przedstawienie dwuwymiarowego histogramu dla dych samych danych

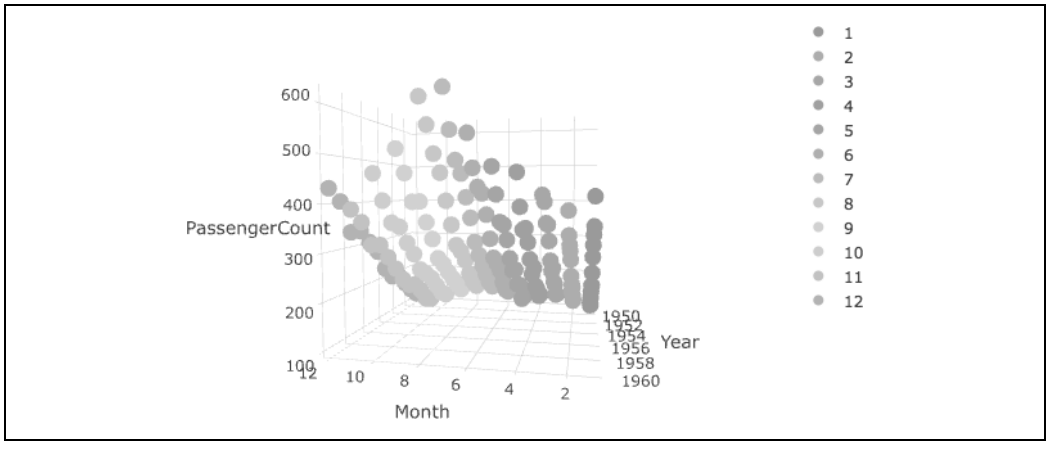
Ponownie weźmy pod uwagę zbiór danych o pasażerach linii lotniczych. Narysujmy go w trzech wymiarach, przeznaczając dwa wymiary na czas (miesiące i lata) oraz jeden wymiar na wartości w punktach:

```
## R
require(plotly)
require(data.table)
months = 1:12
ap = data.table(matrix(AirPassengers, nrow = 12, ncol = 12))
names(ap) = as.character(1949:1960)
ap[, month := months]
ap = melt(ap, id.vars = 'month')
names(ap) = c("month", "year", "count")
p <- plot_ly(ap, x = ~month, y = ~year, z = ~count,
             color = ~as.factor(month)) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Month'),
                      yaxis = list(title = 'Year'),
                      zaxis = list(title = 'PassengerCount')))
```

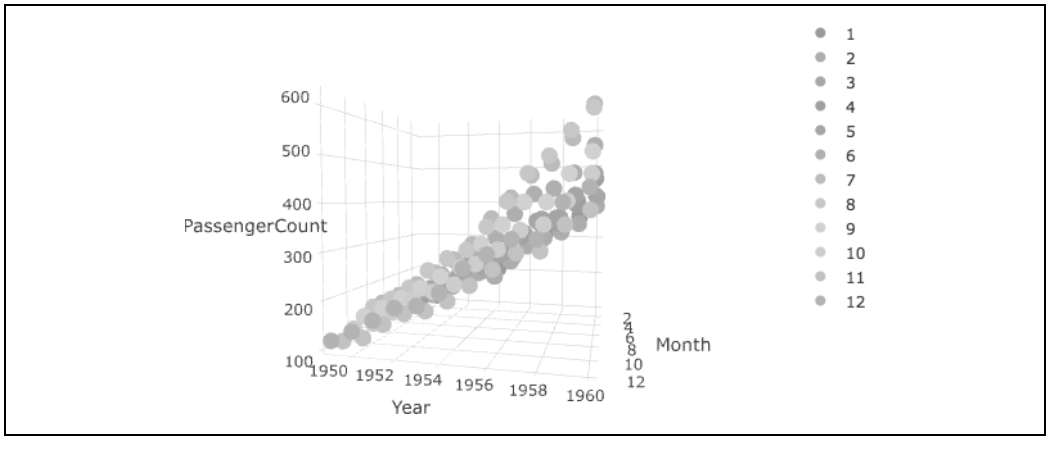
Ta trójwymiarowa wizualizacja pozwala nam zorientować się w ogólnym charakterze danych. Pomimo że zbiór ten widzieliśmy już wcześniej, wykres trójwymiarowy wygląda o wiele lepiej niż dwuwymiarowy histogram (rysunki 3.26 i 3.27).

Zamiast umieszczać czas na dwóch osiach możemy jedną z nich przeznaczyć na położenie. Zjawisko błądzenia losowego możemy zwizualizować za pomocą odrobinę zmodyfikowanego kodu demonstrującego możliwości pakietu *plotly* (rysunki 3.28 i 3.29):

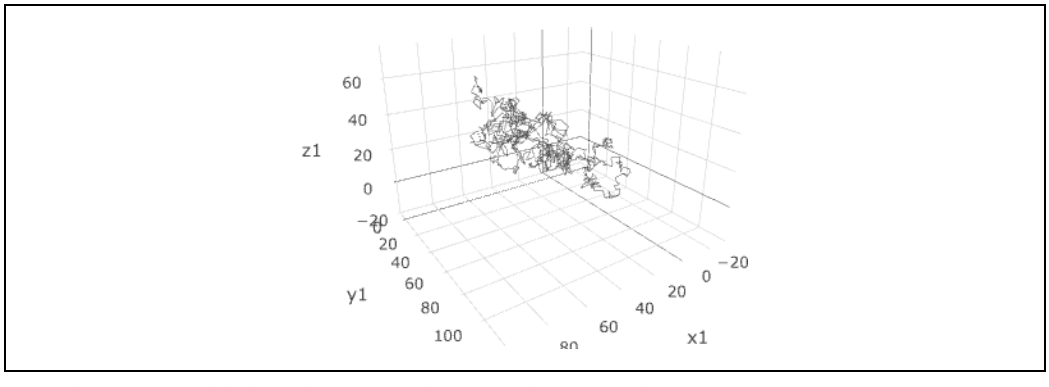
```
## R
file.location <- 'https://raw.githubusercontent.com/plotly/datasets/master/_3d-line-plot.csv'
data <- read.csv(file.location)
p <- plot_ly(data, x = ~x1, y = ~y1, z = ~z1,
             type = 'scatter3d', mode = 'lines',
             line = list(color = '#1f77b4', width = 1))
```



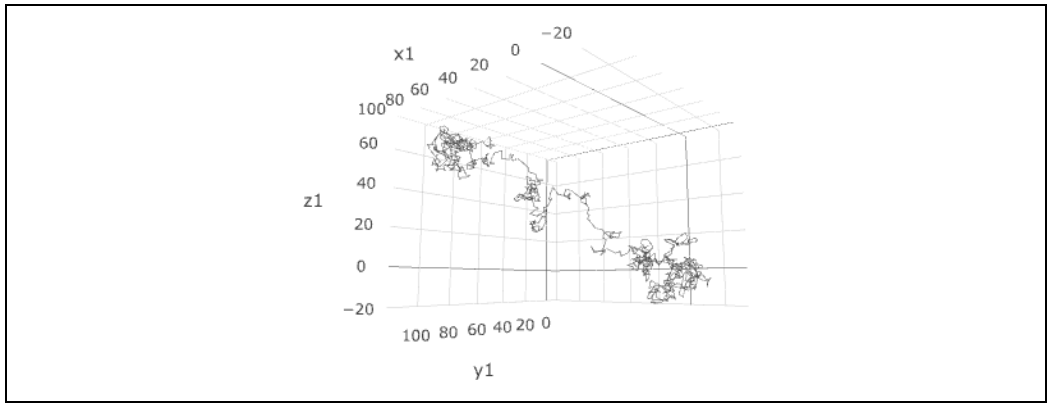
Rysunek 3.26. Trójwymiarowy wykres punktowy danych AirPassenger. Z tej perspektywy sezonowość jest doskonale widoczna



Rysunek 3.27. Ten sam wykres obserwowany z perspektywy uwidaczniającej tempo wzrostu trendu w kolejnych latach. Jeśli uruchomisz kod na własnym komputerze, będziesz mógł obracać wykresy samodzielnie



Rysunek 3.28. Proces błądzenia losowego widoczny z jednej z perspektyw



Rysunek 3.29. Z tej perspektywy na wykresie widać o wiele więcej. Jeszcze raz zachęcam do wypróbowania kodu samodzielnie!

W tym przykładzie kluczowa jest interaktywność wykresu, która umożliwia jego obrót. Statyczne zrzuty stworzone z trójwymiarowego wykresu mogą czasem być mylące lub uwypuklać informacje, które nas w danym momencie nie interesują.

Dobrym ćwiczeniem byłoby wygenerowanie zaszumionych dwuwymiarowych danych dotyczących jakichś zmian sezonowych i ich wizualizacja w identyczny jak wyżej sposób. W porównaniu do danych losowych otrzymany wykres powinien się istotnie różnić. Pakiety takie jak *plotly* pozwalają na szybkie eksperymentowanie z danymi i zapewniają kompleksową odpowiedź zwrotną.

Zobacz też

- O pozornych korelacjach:

  Ai Deng, *A Primer on Spurious Statistical Significance in Time Series Regressions*, „Economics Committee Newsletter, 14, nr 1, 2015, *https://perma.cc/9CQR-RWHC*.

  Ten branżowy artykuł na temat pozornych korelacji i ich wyglądu w danych może się przydać do opracowania praktycznych wskazówek dotyczących tego, kiedy i gdzie należy ich szukać we własnych danych. Materiał ten jest napisany w bardzo przystępnej formie.

  Tyler Vigen, *Spurious Correlations*, Hachette, New York 2015, *https://perma.cc/YY6R-SKWA*.

  Ten zbiór niedorzecznych korelacji pomiędzy szeregami czasowymi jest niezbędną lekturą nie tylko dla każdego analityka.

  Antonio Noriega i Daniel Ventosa-Santaulària, *Spurious Regression Under Broken-Trend Stationarity*, „Journal of Time Series Analysis”, 27, nr 5, 2006, s. 671 – 684, *https://perma.cc/V993-SF4F*.

  Autorzy opracowują zarówno dane teoretyczne, jak i symulacyjne, aby wykazać, że zmiany poziomu lub trendu w niezależnie i losowo generowanych danych wpływają na obecność korelacji pozornych.

C.W.J. Granger i P. Newbold, *Spurious Regressions in Econometrics*, „Journal of Econometrics" 2, nr 2, 1974, s. 111 – 20, *https://perma.cc/M8TE-AL6U*.

Ten artykuł ekonometryczny doprowadził jednego z autorów do nagrody Nobla za zidentyfikowanie trudności związanych z pozornymi korelacjami i popularyzację właściwego podejścia do identyfikowania korelacji w szeregach czasowych.

- O eksploracyjnej analizie danych:

David R. Brillinger i Mark A. Finney, *An Exploratory Data Analysis of the Temperature Fluctuations in a Spreading Fire*, „Environmetrics", 25, nr 6, 2014, s. 443 – 453, *https://perma.cc/QB3D-APKM*.

Dokładny opis tego, jak rzeczywiste dane laboratoryjne zawierające informacje geotemporalne zostały przeanalizowane przez badaczy akademickich i rządowych.

Robert H. Shumway i David S. Stoffer, „Time Series Regression and Exploratory Data Analysis", w: *Time Series Analysis and Its Applications with R Examples*, Springer, New York 2011, *https://perma.cc/UC5B-TPVS*.

Rozdział na temat eksploracyjnej analizy danych z klasycznego podręcznika do analizy szeregów czasowych dla studentów szkół wyższych.

- Więcej o wizualizacji:

Christian Tominski i Wolfgang Aigner, *The TimeViz Browser*, *https://perma.cc/94ND-6ZA5*.

Wspaniały katalog przykładów i kodów źródłowych wielu interesujących wizualizacji szeregów czasowych pochodzących zarówno z prac naukowych, jak i zastosowań przemysłowych.

Oscar Perpiñán Lamigueiro, *GitHub Repository for Displaying Time Series, Spatial, and Space-time Data with R*, *https://perma.cc/R69Y-5JPL*.

Repozytorium zawierające kody źródłowe różnych wizualizacji szeregów czasowych w R, w tym wizualizacje danych geoprzestrzennych.

Myles Harrison, *5 Ways to Do 2D Histograms in R*, R-bloggers, 1.09.2014, *https://perma.cc/ZCX9-FQQY*.

Jest to praktyczny przewodnik po możliwościach wizualizacji dwuwymiarowych histogramów w różnych pakietach stworzonych dla środowiska R. Oprócz podstawowego przeglądu pokrewny artykuł (*https://edav.info/tidyquant.html*) zapewnia także przegląd pakietu *tidyquant* służącego do wizualizacji danych giełdowych — ważnego źródła szeregów czasowych.

- O trendach:

Halbert White i Clive W.J. Granger, *Considerations of Trends in Time Series*, „Journal of Time Series Econometrics", 3, nr 1, 2011, *https://perma.cc/WF2H-TVTL*.

W tym dość „świeżym" artykule autorzy zauważają, że pomimo wszechobecności trendów nawet w tradycyjnej statystyce brakuje zestawu definicji pozwalających na opisanie ich różnych klas. W przystępnym stylu autorzy prezentują różne spostrzeżenia statystyczne umożliwiające określenie, kiedy w niestacjonarnych danych mogą występować trendy, a także proponują sposoby na poprawę istniejących już metod ich wykrywania.

Skutecznie analizuj szeregi czasowe i wydobywaj bezcenną wiedzę!

Analiza szeregów czasowych zyskuje na znaczeniu. Wraz z postępującą digitalizacją danych służby zdrowia, rozwojem inteligentnych miast czy upowszechniającym się internetem rzeczy staje się coraz bardziej potrzebna. Obiecującym rozwiązaniem jest analiza szeregów czasowych metodami wspomaganymi uczeniem maszynowym. Techniki te umożliwiają skuteczne monitorowanie i wykorzystywanie coraz większych zbiorów danych. Być może ich zastosowanie do pracy z szeregami czasowymi wydaje się nieoczywiste, jednak bez analiz szeregów czasowych nie można w pełni wykorzystać zebranych danych.

Ta książka jest szerokim, aktualnym i praktycznym przeglądem metod analizy szeregów czasowych, w którym ujęto pełny potok przetwarzania danych czasowych i modelowania. Zaprezentowano w niej rzeczywiste przypadki użycia tych metod i zilustrowano je obszernymi fragmentami znakomicie zaprojektowanego kodu w językach R i Python. Znalazły się tutaj praktyczne wskazówki ułatwiające rozwiązywanie najczęstszych problemów występujących w inżynierii danych czasowych i ich analizie. Ujęto tu zarówno konwencjonalne metody statystyczne, jak i nowoczesne techniki uczenia maszynowego. To bardzo przydatny przewodnik, dzięki któremu analitycy danych, inżynierowie oprogramowania i naukowcy będą mogli płynnie przejść od podstaw pracy z szeregami czasowymi do rozwiązywania konkretnych zagadnień na profesjonalnym poziomie.

Dzięki tej książce nauczysz się:

- pozyskiwać, przechowywać i przetwarzać szeregi czasowe
- eksplorować dane czasowe i je symulować
- wykonywać pomiary błędów
- pracować z szeregami czasowymi za pomocą uczenia maszynowego lub uczenia głębokiego
- oceniać dokładność i wydajność modeli

Aileen Nielsen jest inżynierem oprogramowania i analitykiem danych. Współpracuje ze start-upami, które korzystają z szeregów czasowych i sieci neuronowych. Wcześniej pracowała w kancelariach prawnych, laboratoriach badawczych i start-upach technologicznych. Interesuje się inżynierią oprogramowania obronnego oraz współdziałaniem prawa i technologii. Często występuje na konferencjach dotyczących uczenia maszynowego i predykcji za pomocą sieci neuronowych.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-6721-0

9 788328 367210

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 79,00 zł