O'REILLY®

Wydanie II

Język R w data science

Importowanie, porządkowanie, przekształcanie, wizualizowanie i modelowanie danych

Hadley Wickham
Mine Çetinkaya-Rundel
Garrett Grolemund

Helion

Tytuł oryginału: R for Data Science: Import, Tidy, Transform, Visualize, and Model Data, 2nd Edition

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-289-0653-2

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/jerda2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Wizualizowanie danych

Wprowadzenie

> „Zwykły wykres dostarcza umysłom analityków danych więcej informacji niż jakiekolwiek inne narzędzie”. — John Tukey

W R dostępnych jest kilka systemów generowania wykresów, jednak pakiet *ggplot2* jest jednym z najbardziej eleganckich i wszechstronnych spośród nich. W pakiecie *ggplot2* zaimplementowana jest *gramatyka grafiki*. Jest to spójny system opisywania i tworzenia wykresów. Za pomocą pakietu *ggplot2* możesz szybciej wykonywać liczne zadania dzięki opanowaniu jednego systemu i stosowaniu go w wielu miejscach.

Z tego rozdziału dowiesz się, jak wizualizować dane za pomocą pakietu *ggplot2*. Zaczniemy od utworzenia prostego wykresu punktowego, po czym dodamy do niego odwzorowania właściwości estetycznych i obiekty geometryczne (geomy). Są to podstawowe elementy składowe w pakiecie *ggplot2*. Następnie przeprowadzimy Cię przez wizualizowanie rozkładów pojedynczych zmiennych, a także wizualizowanie relacji między dwiema zmiennymi (lub większą ich liczbą). Zakończymy omówieniem zapisywania wykresów i wskazówkami dotyczącymi rozwiązywania problemów.

Wymagania wstępne

W tym rozdziale skupiamy się na *ggplot2*, jednym z podstawowych pakietów z biblioteki *tidyverse*. Aby uzyskać dostęp do zbiorów danych, stron pomocy i funkcji używanych w tym rozdziale, wczytaj bibliotekę *tidyverse* za pomocą poniższej instrukcji:

```
library(tidyverse)
#> ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
#> ✔ dplyr 1.1.0.9000  ✔ readr 2.1.4
#> ✔ forcats 1.0.0     ✔ stringr 1.5.0
#> ✔ ggplot2 3.4.1     ✔ tibble 3.1.8
#> ✔ lubridate 1.9.2   ✔ tidyr 1.3.0
#> ✔ purrr 1.0.1
#> ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
```

```
#> ✖ dplyr::lag() masks stats::lag()
#> ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
#>   conflicts to become errors
```

Ten jeden wiersz kodu wczytuje rdzeń biblioteki *tidyverse*, czyli pakiety, które będą używane w prawie każdej analizie danych. Informuje również, które funkcje z biblioteki *tidyverse* kolidują z funkcjami z podstawowego języka R (lub z innych pakietów, które zostały wczytane)[1].

Jeśli uruchomisz ten kod i otrzymasz komunikat o błędzie informującym, że nie ma pakietu o nazwie *tidyverse* (`there is no package called 'tidyverse'`), musisz najpierw zainstalować ten pakiet, a następnie ponownie uruchomić funkcję `library()`:

```
install.packages("tidyverse")
library(tidyverse)
```

Pakiet należy zainstalować tylko raz, ale trzeba go wczytać za każdym razem, gdy rozpoczynasz nową sesję.

Oprócz biblioteki *tidyverse* będziemy używać pakietu *palmerpenguins*, który zawiera zbiór danych na temat pingwinów (obejmuje on pomiary ciała pingwinów na trzech wyspach Archipelagu Palmera), a także pakietu *ggthemes*, który udostępnia bezpieczną paletę kolorów dla osób z zaburzeniami rozpoznawania barw.

```
library(palmerpenguins)
library(ggthemes)
```

Pierwsze kroki

Czy pingwiny z dłuższymi skrzydłami ważą więcej czy mniej niż pingwiny z krótszymi skrzydłami? Prawdopodobnie znasz już odpowiedź, ale postaraj się ją doprecyzować. Jak wygląda zależność między długością płetw a masą ciała? Czy jest dodatnia? Ujemna? Liniowa? Nieliniowa? Czy występują różnice w zależności od gatunku pingwina? A może w zależności od wyspy, na której żyje pingwin? Stworzymy wizualizacje, których można użyć, aby odpowiedzieć na te pytania.

Ramka danych penguins

Możesz przetestować swoje odpowiedzi na te pytania za pomocą ramki danych `penguins` znajdującej się w pakiecie *palmerpenguins* (czyli w ramce `palmerpenguins::penguins`). Ramka danych to „prostokątny" zbiór zmiennych (w kolumnach) i obserwacji (w wierszach). Ramka danych `penguins` zawiera 344 obserwacje zebrane i udostępnione przez dr Kristen Gorman z jednostki Palmer Station, Antarctica LTER[2].

[1] Możesz wyeliminować ten komunikat i wymusić rozwiązywanie konfliktów na żądanie za pomocą pakietu *conflicted*, który staje się tym ważniejszy, im więcej pakietów wczytujesz. Więcej informacji na temat pakietu *conflicted* znajdziesz na jego stronie (*https://oreil.ly/01bKz*).

[2] A.M. Horst, A.P. Hill i K.B. Gorman, *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020, pakiet dla R, wersja 0.1.0, *https://oreil.ly/ncwc5*, doi: 10.5281/zenodo.3960218.

Aby ułatwić dyskusję, warto zdefiniować kilka terminów:

Zmienna

Ilość, jakość lub właściwość, którą można zmierzyć.

Wartość

Stan zmiennej w momencie jej pomiaru. Wartość zmiennej w poszczególnych pomiarach może się zmieniać.

Obserwacja

Zestaw pomiarów wykonanych w podobnych warunkach (zazwyczaj wszystkie pomiary w ramach obserwacji są wykonywane w tym samym czasie i na tym samym obiekcie). Obserwacja obejmuje kilka wartości, z których każda jest powiązana z inną zmienną. Czasami obserwację nazywamy **punktem danych**.

Dane tabelaryczne

Zestaw wartości, z których każda powiązana jest ze zmienną i obserwacją. Dane tabelaryczne są *uporządkowane*, jeśli każda wartość jest umieszczona we własnej „komórce", każda zmienna we własnej kolumnie, a każda obserwacja we własnym wierszu.

W tym kontekście zmienna oznacza atrybut wszystkich pingwinów, a obserwacja reprezentuje wszystkie atrybuty pojedynczego pingwina.

Wpisz nazwę ramki danych w konsoli, a R wyświetli podgląd jej zawartości. Zauważ, że na górze tego podglądu znajduje się napis *tibble*. W bibliotece *tidyverse* używane są specjalne ramki danych nazywane *tibble*, które omawiamy dalej.

```
penguins
#> # A tibble: 344 × 8
#>   species island    bill_length_mm bill_depth_mm flipper_length_mm
#>   <fct>   <fct>     <dbl>          <dbl>         <int>
#> 1 Adelie  Torgersen 39.1           18.7          181
#> 2 Adelie  Torgersen 39.5           17.4          186
#> 3 Adelie  Torgersen 40.3           18            195
#> 4 Adelie  Torgersen NA             NA            NA
#> 5 Adelie  Torgersen 36.7           19.3          193
#> 6 Adelie  Torgersen 39.3           20.6          190
#> # … with 338 more rows, and 3 more variables: body_mass_g <int>, sex <fct>,
#> #   year <int>
```

Ta ramka danych zawiera osiem kolumn. Aby uzyskać inny widok, w którym widoczne są wszystkie zmienne i kilka pierwszych obserwacji każdej z nich, użyj funkcji `glimpse()`. Ponadto jeśli używasz RStudio, możesz uruchomić wywołanie `View(penguins)`, by otworzyć interaktywną przeglądarkę danych.

```
glimpse(penguins)
#> Rows: 344
#> Columns: 8
#> $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, A…
#> $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torge…
#> $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.…
#> $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.…
#> $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, …
```

```
#> $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 347…
#> $ sex               <fct> male, female, female, NA, female, male, female, m…
#> $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2…
```

Oto zmienne z ramki danych `penguins`:

`species`
: Gatunek pingwina (`Adelie`, `Chinstrap` lub `Gentoo`).

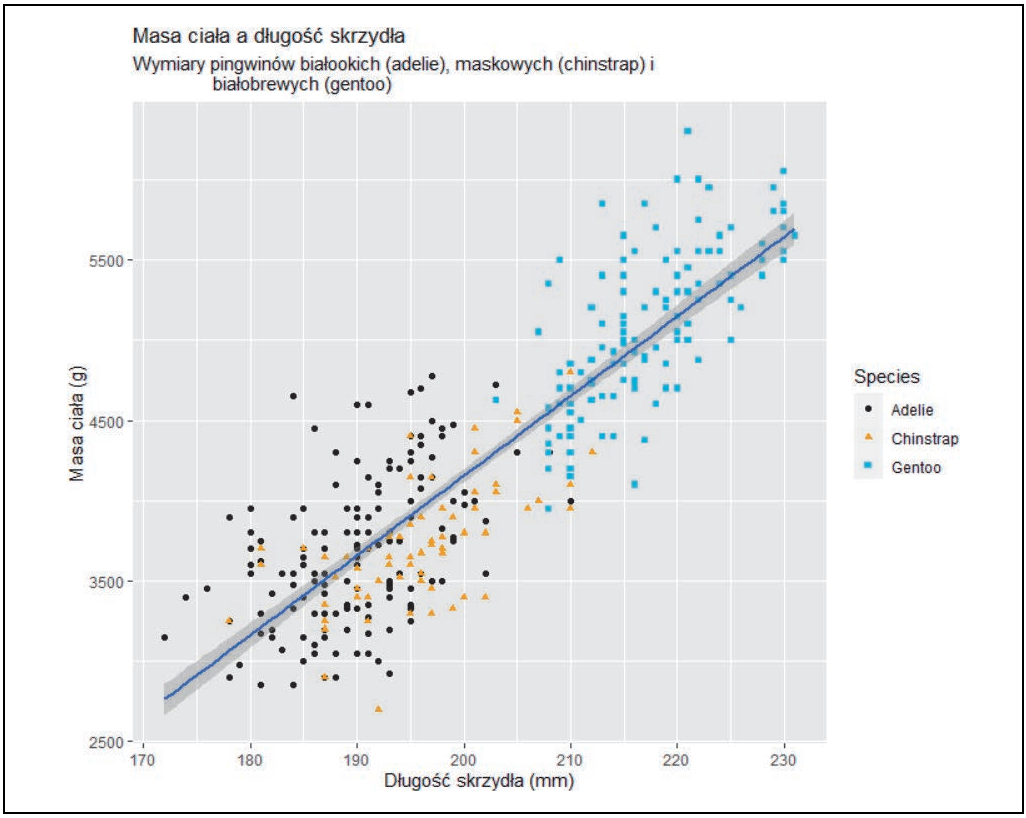
`flipper_length_mm`
: Długość skrzydła pingwina w milimetrach.

`body_mass_g`
: Masa ciała pingwina w gramach.

Aby dowiedzieć się więcej o ramce danych `penguins`, otwórz stronę pomocy, korzystając z instrukcji `?penguins`.

Docelowa wizualizacja

Naszym ostatecznym celem w tym rozdziale jest odtworzenie wizualizacji z rysunku 1.1, przedstawiającej zależność między długością skrzydeł a masą ciała pingwinów z uwzględnieniem gatunku pingwina.



Rysunek 1.1. Docelowa wizualizacja

Tworzenie wykresu za pomocą pakietu ggplot2

Odtwórzmy ten wykres krok po kroku.

W pakiecie *ggplot2* tworzenie wykresu rozpoczyna się od użycia funkcji `ggplot()`, która definiuje obiekt wykresu. Następnie do tego obiektu dodawane są warstwy. Pierwszym argumentem funkcji `ggplot()` jest zestaw danych używanych na wykresie. Tak więc wywołanie `ggplot(data = penguins)` tworzy pusty wykres, który jest przygotowany do wyświetlania danych z ramki danych `penguins`. Jednak ponieważ nie określiliśmy jeszcze, jak ma wyglądać wizualizacja, na razie wykres jest pusty. Nie jest on specjalnie interesujący, ale możesz myśleć o nim jak o pustym płótnie, na którym namalujesz pozostałe warstwy wykresu (rysunek 1.2).

```
ggplot(data = penguins)
```

Rysunek 1.2. Pierwszy, pusty wykres

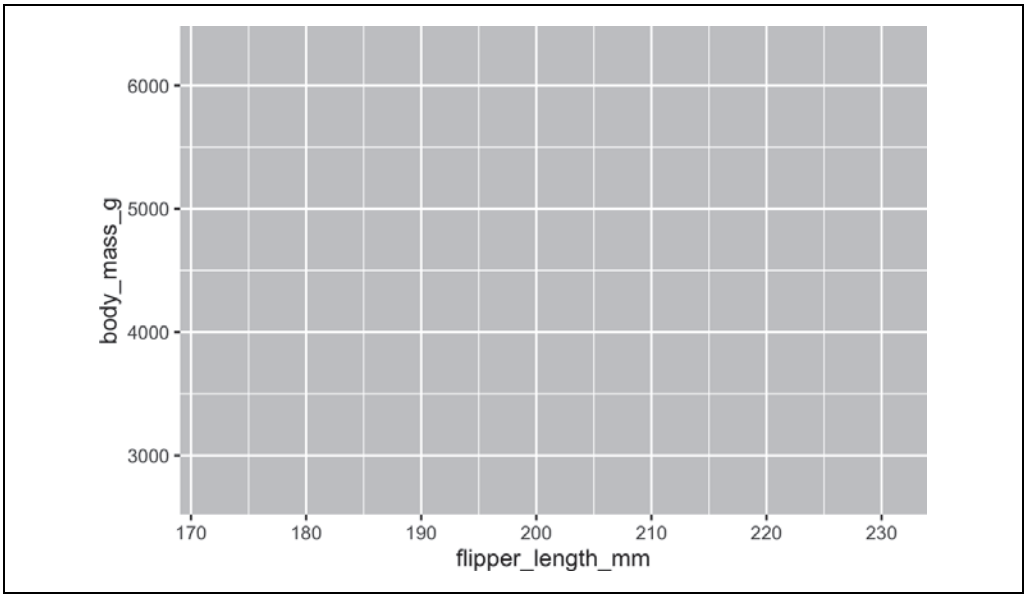
Następnie należy poinformować funkcję `ggplot()`, w jaki sposób informacje z danych mają być reprezentowane wizualnie. Argument `mapping` funkcji `ggplot()` definiuje sposób odwzorowania zmiennych ze zbioru danych na właściwości wizualne (*właściwości estetyczne*) wykresu. Argument `mapping` zawsze jest definiowany w funkcji `aes()`, a argumenty `x` i `y` funkcji `aes()` określają, które zmienne mają być odwzorowane na osiach `x` i `y`. Na razie odwzorujemy tylko długość skrzydeł na osi `x` i masę ciała na osi `y`. Pakiet *ggplot2* szuka odwzorowywanych zmiennych w argumencie `data`, który tu ma wartość `penguins`.

Wykres z rysunku 1.3 pokazuje wynik dodania tych odwzorowań.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
)
```

Puste płótno ma teraz lepiej określoną strukturę — wiadomo, gdzie będą wyświetlane długości skrzydeł (na osi x), a gdzie masa ciała (na osi y). Ale danych samych pingwinów nie ma jeszcze na wykresie. Dzieje się tak dlatego, że nie określiliśmy jeszcze w kodzie, jak obserwacje z ramki danych mają być reprezentowane na wykresie.

Aby to zrobić, trzeba zdefiniować *geom*, czyli obiekt geometryczny, którego wykres używa do reprezentowania danych. Takie obiekty geometryczne są dostępne w pakiecie *ggplot2* za pomocą funkcji o nazwach zaczynających się od członu `geom_`. Użytkownicy często opisują wykresy na podstawie



Rysunek 1.3. Wykres po dodaniu pierwszych odwzorowań

typu geomu używanego na wykresie. Na przykład na wykresach słupkowych używane są geomy słupkowe (`geom_bar()`), na wykresach liniowych stosuje się geomy liniowe (`geom_line()`), na wykresach pudełkowych używane są geomy pudełkowe (`geom_boxplot()`), na wykresach punktowych stosowane są geomy punktowe (`geom_point()`) i tak dalej.

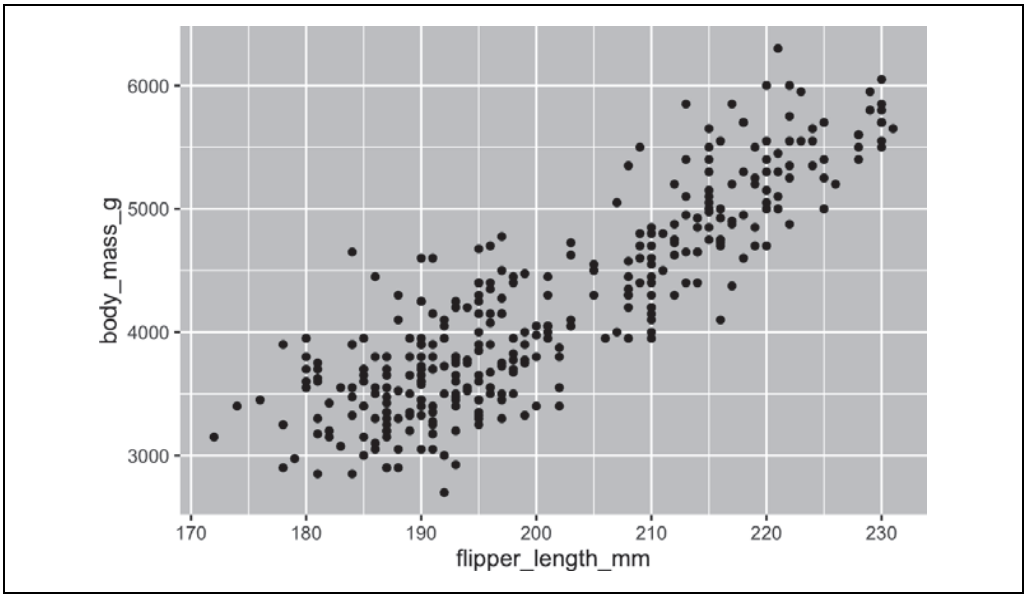
Funkcja `geom_point()` dodaje do wykresu warstwę punktów, która powoduje utworzenie wykresu punktowego (rysunek 1.4). Pakiet *ggplot2* zawiera wiele funkcji z rodziny `geom_`, a każda z nich dodaje do wykresu warstwę innego typu. W tej książce (zwłaszcza w rozdziale 9.) poznasz wiele funkcji tego typu.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point()
#> Warning: Removed 2 rows containing missing values (`geom_point()`).
```

Uzyskaliśmy coś, co wygląda jak wykres punktowy. Nie jest on jeszcze identyczny z wykresem docelowym, ale za pomocą obecnej wersji można zacząć szukać odpowiedzi na początkowe pytanie: jak wygląda związek między długością skrzydeł a masą ciała? Zależność wydaje się być dodatnia (wraz ze wzrostem długości skrzydeł rośnie masa ciała), dość liniowa (punkty są skupione wokół linii, a nie wokół krzywej) i umiarkowanie silna (nie ma zbyt dużego rozrzutu wokół tej linii). Pingwiny z dłuższymi skrzydłami generalnie mają większą masę ciała.

Zanim dodamy do tego wykresu więcej warstw, zatrzymajmy się na chwilę i zastanówmy nad otrzymanym komunikatem ostrzegawczym:

```
Removed 2 rows containing missing values (geom_point())
```



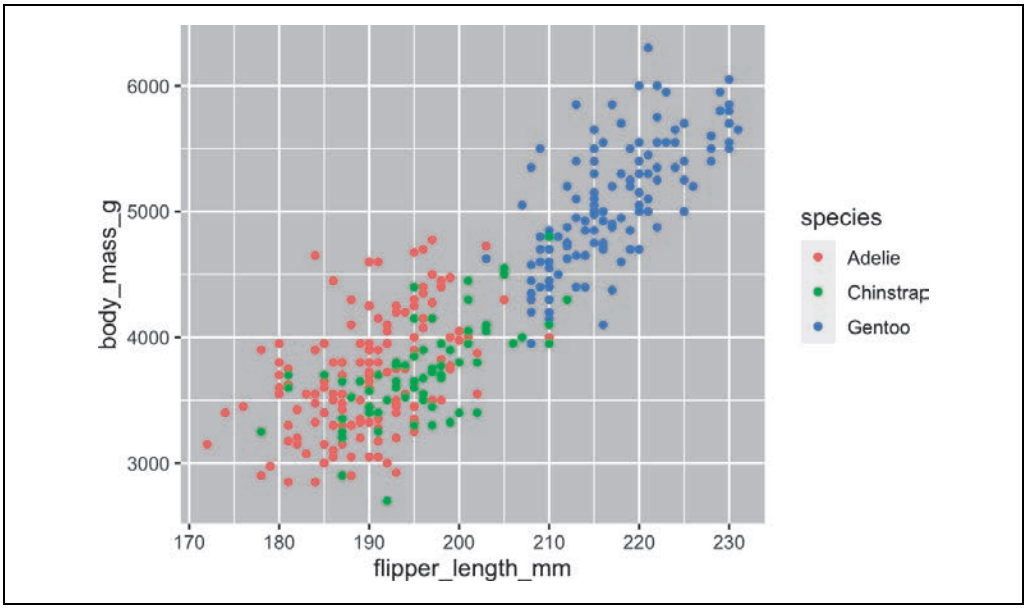
Rysunek 1.4. Wykres po dodaniu punktów danych

Ten komunikat pojawia się, ponieważ w zbiorze danych występują dwa pingwiny z brakującymi wartościami masy ciała i/lub długości skrzydła, a pakiet *ggplot2* nie potrafi przedstawić danych na wykresie bez obu tych wartości. Podobnie jak w języku R tak i w pakiecie *ggplot2* obowiązuje podejście, że brakujące wartości nigdy nie powinny być ignorowane bez informowania o nich użytkownika. Ten typ ostrzeżenia jest prawdopodobnie jednym z najczęściej wyświetlanych komunikatów, jakie można zobaczyć podczas pracy z rzeczywistymi danymi. Brakujące wartości są częstym problemem i dowiesz się o nich więcej w całej książce, a przede wszystkim w rozdziale 18. Dla pozostałych wykresów w tym rozdziale pomijamy to ostrzeżenie, aby nie powtarzać go przy każdym tworzonym wykresie.

Dodawanie właściwości estetycznych i warstw

Wykresy punktowe są przydatne do wyświetlania relacji między dwiema zmiennymi liczbowymi, ale zawsze warto sceptycznie traktować każdą zaobserwowaną relację między zmiennymi i zastanowić się, czy mogą istnieć inne zmienne, które wyjaśniają lub zmieniają charakter tej zaobserwowanej zależności. Na przykład czy związek między długością skrzydeł a masą ciała różni się w zależności od gatunku? Teraz uwzględnimy na wykresie gatunki i sprawdzimy, czy ujawni to dodatkowe informacje na temat zaobserwowanego związku między zmiennymi. W tym celu gatunki będziemy reprezentować za pomocą różnokolorowych punktów (rysunek 1.5).

Aby to osiągnąć, trzeba zmodyfikować właściwość estetyczną czy może geom? Jeśli udzielona odpowiedź to „trzeba odwzorować dane na właściwość estetyczną za pomocą funkcji `aes()`", to już wiesz, jak tworzyć wizualizacje danych za pomocą pakietu *ggplot2*! Jeżeli Twoja odpowiedź była inna, nie martw się. W trakcie lektury utworzysz wiele innych wykresów za pomocą pakietu *ggplot2* i będziesz mieć o wiele więcej okazji do sprawdzenia swojej intuicji podczas ich konstruowania.



Rysunek 1.5. Teraz gatunki są reprezentowane przez różnokolorowe punkty

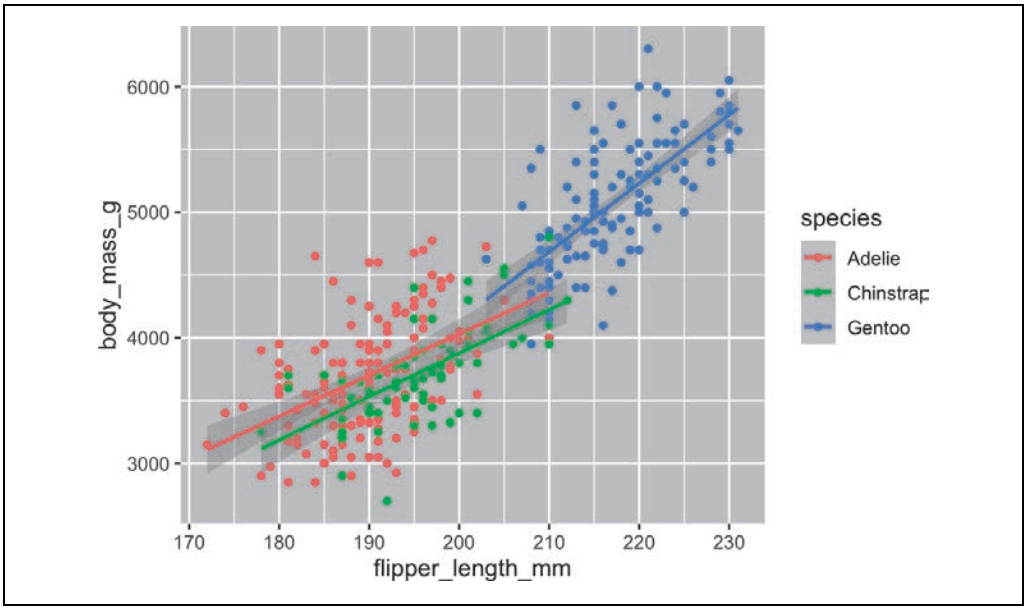
```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)
) +
  geom_point()
```

Gdy zmienna kategorialna jest odwzorowywana na właściwość estetyczną, *ggplot2* automatycznie przypisuje unikatową wartość właściwości estetycznej (tutaj jest to inny kolor) do każdego unikatowego poziomu zmiennej (do każdego z trzech gatunków). Ten proces jest nazywany **skalowaniem**. *ggplot2* dodaje również legendę wyjaśniającą, które wartości odpowiadają poszczególnym poziomom.

Teraz dodajmy jeszcze jedną warstwę: gładką krzywą przedstawiającą zależność między masą ciała a długością skrzydeł. Zanim przejdziesz dalej, zapoznaj się z poprzednim kodem i zastanów, jak dodać taką krzywą do istniejącego wykresu.

Ponieważ ta krzywa to nowy obiekt geometryczny reprezentujący dane, należy dodać nowy geom (`geom_smooth()`) w warstwie nad geomem punktów. Za pomocą opcji `method = "lm"` określ, że chcesz narysować linię najlepszego dopasowania w oparciu o model liniowy (rysunek 1.6).

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")
```



Rysunek 1.6. Geom geom_smooth() pozwala dodać krzywą reprezentującą dane

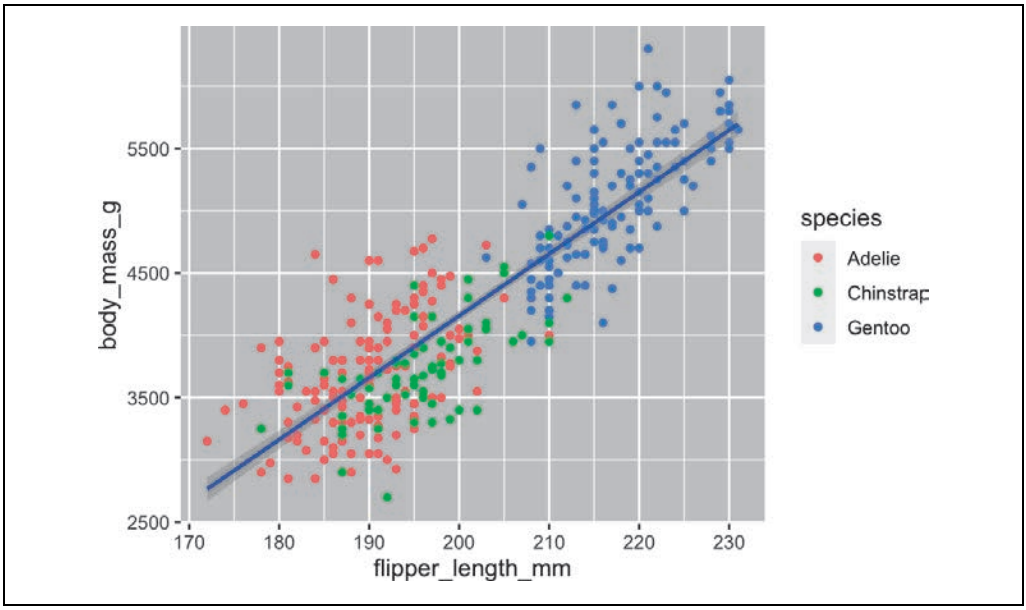
Udało się dodać linie, ale wykres wciąż nie wygląda jak wykres z docelowej wizualizacji z początku rozdziału, który ma tylko jedną linię dla całego zbioru danych. W obecnej wersji widoczne są oddzielne linie dla każdego z gatunków pingwinów.

Gdy w funkcji `ggplot()` odwzorowania na właściwości estetyczne są zdefiniowane na poziomie *globalnym*, są one przekazywane niżej do każdej z kolejnych warstw geomów na wykresie. Jednak każda funkcja geomów w *ggplot2* może również przyjmować argument `mapping`, co pozwala definiować odwzorowania na właściwości estetyczne na poziomie *lokalnym*. Są one dodawane do odwzorowań odziedziczonych z poziomu globalnego.

Ponieważ chcemy, aby kolory punktów były określone na podstawie gatunku, ale nie chcemy tworzyć oddzielnych linii dla każdego gatunku (rysunek 1.7), argument `color = species` należy podać tylko w wywołaniu `geom_point()`.

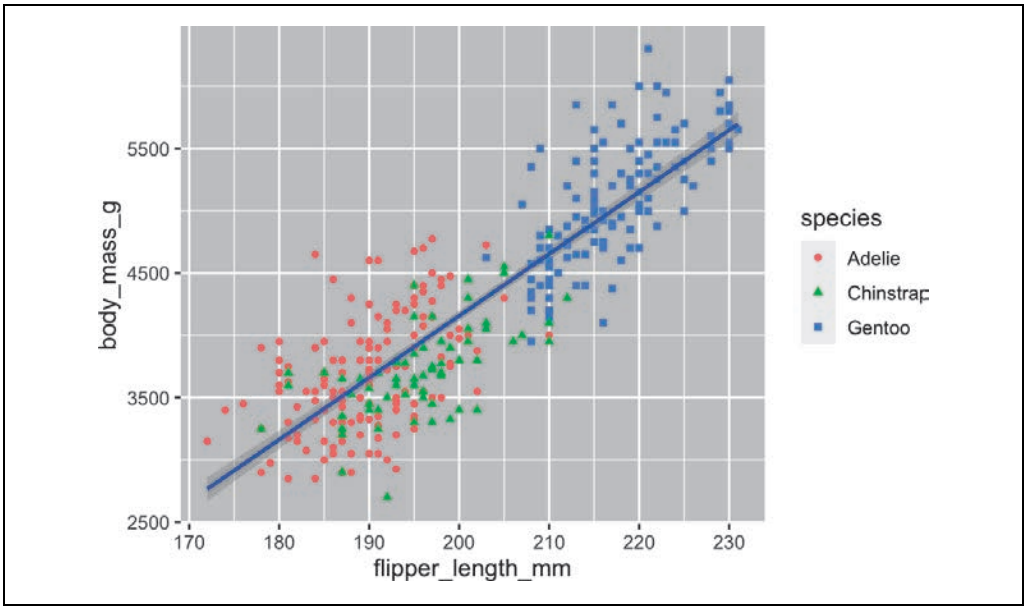
```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(mapping = aes(color = species)) +
  geom_smooth(method = "lm")
```

I gotowe! Uzyskaliśmy wykres, który wygląda bardzo podobnie do docelowej wizualizacji, choć nie jest jeszcze idealny. Nadal trzeba jeszcze wprowadzić różne kształty dla poszczególnych gatunków pingwinów i poprawić etykiety.



Rysunek 1.7. Teraz widoczna jest tylko jedna krzywa

Zasadniczo nie jest dobrym pomysłem przedstawianie informacji na wykresie za pomocą samych kolorów, ponieważ ludzie różnie postrzegają kolory ze względu na ślepotę barw lub inne różnice w widzeniu kolorów. Dlatego oprócz barw możemy odwzorować gatunki na właściwość estetyczną `shape` (rysunek 1.8).

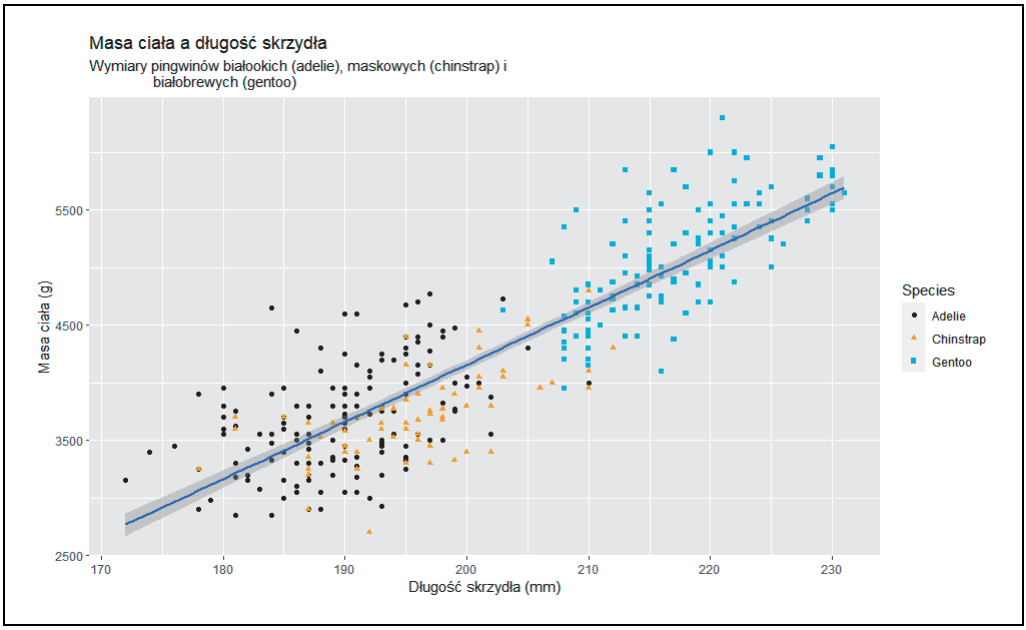


Rysunek 1.8. Poszczególne gatunki są wyróżnione nie tylko kolorem, ale i kształtami

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(mapping = aes(color = species, shape = species)) +
  geom_smooth(method = "lm")
```

Warto zauważyć, że legenda jest automatycznie aktualizowana i uwzględnia teraz różne kształty punktów.

Na koniec można poprawić etykiety na wykresie za pomocą funkcji `labs()` dla nowej warstwy. Niektóre argumenty tej funkcji są oczywiste: `title` pozwala dodać tytuł, a `subtitle` określa podtytuł wykresu. Inne argumenty odpowiadają odwzorowaniom na właściwości estetyczne: `x` to etykieta osi x, `y` to etykieta osi y, a `color` i `shape` definiują etykiety w legendzie. Ponadto można zmienić paletę kolorów, aby była bezpieczna dla osób ze ślepotą kolorów. Służy do tego funkcja `scale_color_colorblind()` z pakietu *ggthemes*. Efekt jest pokazany na rysunku 1.9.



Rysunek 1.9. Gotowy wykres

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Masa ciała a długość skrzydła",
    subtitle = "Wymiary pingwinów białookich (adelie), maskowych (chinstrap) i
                białobrewych (gentoo)",
    x = "Długość skrzydła (mm)", y = "Masa ciała (g)",
```

```
    color = "Species", shape = "Species"
  ) +
  scale_color_colorblind()
```

W końcu uzyskaliśmy wykres, który idealnie odpowiada docelowej wizualizacji!

Ćwiczenia

1. Ile wierszy ma ramka danych `penguins`? Ile kolumn?
2. Co opisuje zmienna `bill_depth_mm` w ramce danych `penguins`? Aby się tego dowiedzieć, zapoznaj się z pomocą, korzystając z instrukcji `?penguins`.
3. Utwórz wykres punktowy zmiennej `bill_depth_mm` względem zmiennej `bill_length_mm`. Należy więc skonstruować wykres punktowy ze zmienną `bill_depth_mm` na osi y i zmienną `bill_length_mm` na osi x. Opisz zależność między tymi dwiema zmiennymi.
4. Co się stanie, jeśli utworzysz wykres punktowy zmiennej `species` względem zmiennej `bill_depth_mm`? Jaki lepszy geom możesz wybrać?
5. Dlaczego poniższy kod wykresu jest błędny i jak można go poprawić?

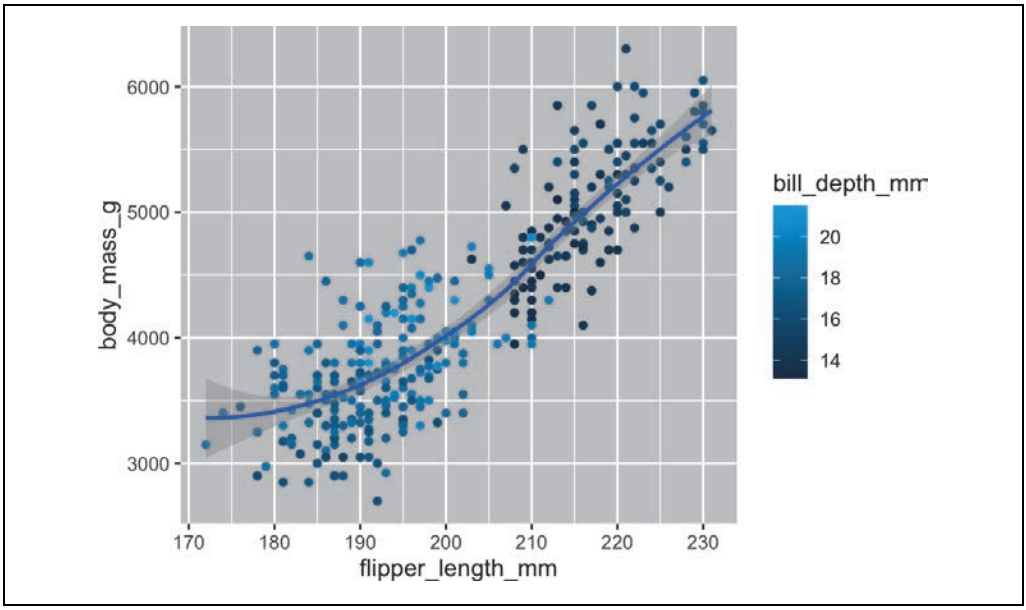
   ```
   ggplot(data = penguins) +
     geom_point()
   ```

6. Jak działa argument `na.rm` w funkcji `geom_point()`? Jaka jest wartość domyślna tego argumentu? Utwórz wykres punktowy, w którym zastosujesz ten argument o wartości `TRUE`.
7. Dodaj następujący podpis do wykresu utworzonego w poprzednim ćwiczeniu: „Dane pochodzą z pakietu palmerpenguins”. Wskazówka: zapoznaj się z dokumentacją funkcji `labs()`.
8. Odtwórz wizualizację z rysunku 1.10. Na jaką właściwość estetyczną należy odwzorować zmienną `bill_depth_mm`? Należy zastosować odwzorowania na poziomie globalnym czy na poziomie geomów?
9. „Uruchom” ten kod w głowie i spróbuj przewidzieć, jak będą wyglądać dane wyjściowe. Następnie uruchom kod w R i sprawdź, czy Twoje prognozy się potwierdziły:

   ```
   ggplot(
     data = penguins,
     mapping = aes(x = flipper_length_mm, y = body_mass_g, color = island)
   ) +
     geom_point() +
     geom_smooth(se = FALSE)
   ```

10. Czy wykresy uzyskane za pomocą dwóch poniższych fragmentów kodu będą różniły się od siebie? Dlaczego?

    ```
    ggplot(
      data = penguins,
      mapping = aes(x = flipper_length_mm, y = body_mass_g)
    ) +
      geom_point() +
      geom_smooth()
    ```



Rysunek 1.10. Odtwórz tę wizualizację

```
ggplot() +
  geom_point(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
  ) +
  geom_smooth(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
  )
```

Wywołania w pakiecie ggplot2

Po wprowadzających podrozdziałach przejdziemy do bardziej zwięzłego zapisu kodu opartego na pakiecie *ggplot2*. Do tej pory bezpośrednio zapisywaliśmy cały kod, co jest pomocne w trakcie nauki:

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point()
```

Zazwyczaj pierwszy argument (lub dwa argumenty) funkcji jest na tyle ważny, że należy znać go na pamięć. Pierwsze dwa argumenty funkcji `ggplot()` to `data` i `mapping`. W dalszej części książki nie będziemy podawać tych nazw. Oszczędza to pisania i, dzięki zmniejszeniu ilości dodatkowego tekstu, ułatwia dostrzeżenie różnic między kodem wykresów. Jest to naprawdę ważne zagadnienie, do którego wrócimy w rozdziale 25.

Oto bardziej zwięzła wersja kodu poprzedniego wykresu:

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

Dalej poznasz operator potoku, |>, który umożliwia zapis kodu wykresu w następujący sposób:

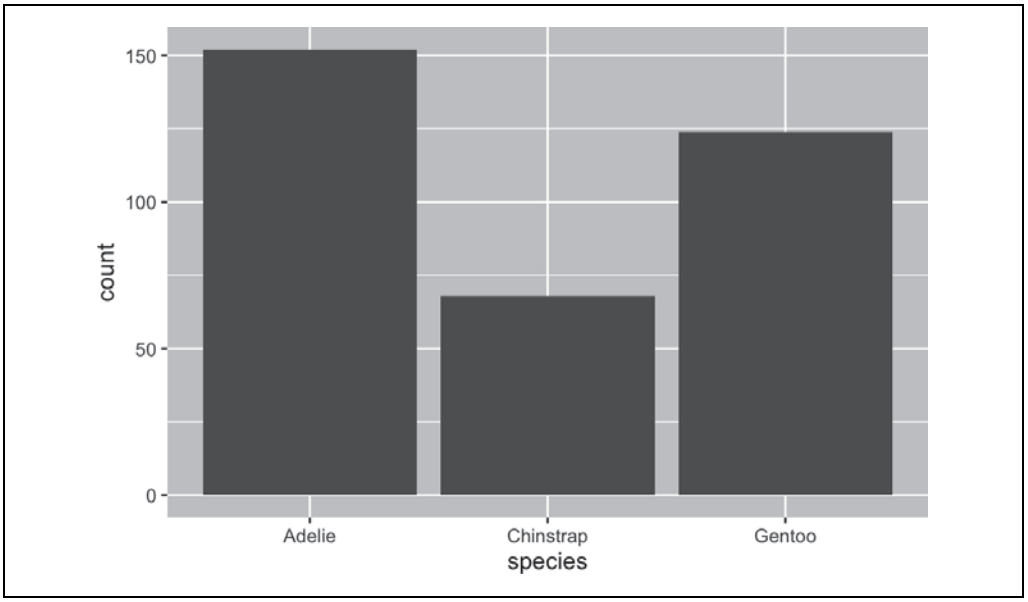
```
penguins |>
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

Wizualizacje rozkładu

Sposób wizualizacji rozkładu zmiennej zależy od typu zmiennej. Istnieją dwa typy zmiennych: kategorialne i liczbowe.

Zmienne kategorialne

Zmienna *kategorialna* może przyjmować tylko jedną z niewielkiego zestawu wartości. Do zbadania rozkładu zmiennej kategorialnej można użyć wykresu słupkowego. Wysokość słupków obrazuje liczbę obserwacji z każdą wartością zmiennej x (rysunek 1.11).



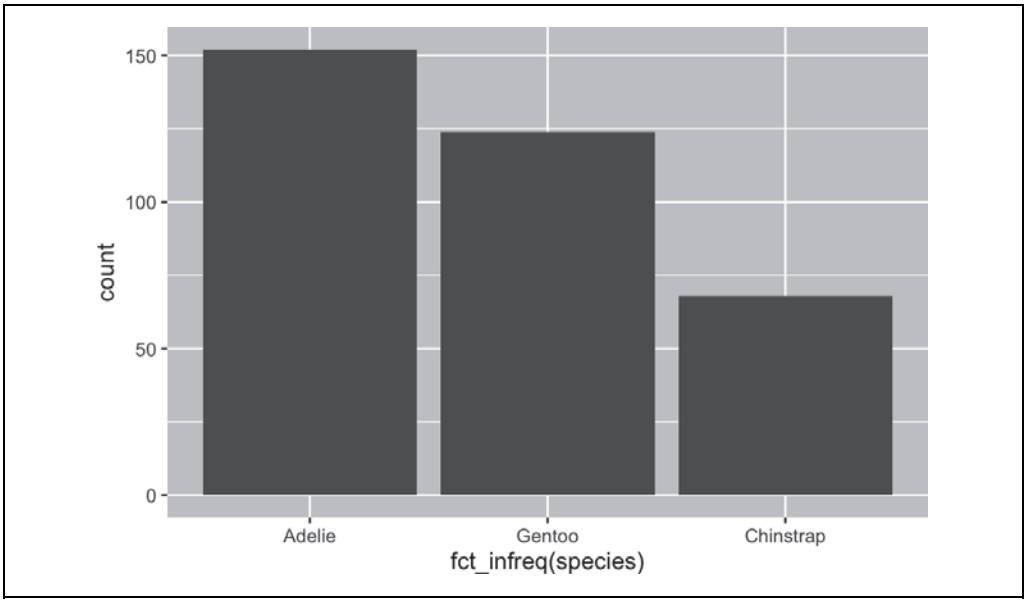
Rysunek 1.11. Wykres słupkowy

```
ggplot(penguins, aes(x = species)) +
  geom_bar()
```

Na wykresach słupkowych zmiennych kategorialnych bez określonego uporządkowania, na przykład zmiennej `species`, często zaleca się zmianę kolejności słupków na podstawie częstotliwości występowania poszczególnych wartości. Wymaga to przekształcenia zmiennej w czynnik (używany w R

do obsługi danych kategorialnych), a następnie zmiany kolejności poziomów tego czynnika. Efekt pokazany jest na rysunku 1.12.

```
ggplot(penguins, aes(x = fct_infreq(species))) +
  geom_bar()
```



Rysunek 1.12. Ten sam wykres po uporządkowaniu kolejności słupków

Więcej informacji na temat czynników i funkcji do ich obsługi (takich jak `fct_infreq()`) znajdziesz w rozdziale 16.

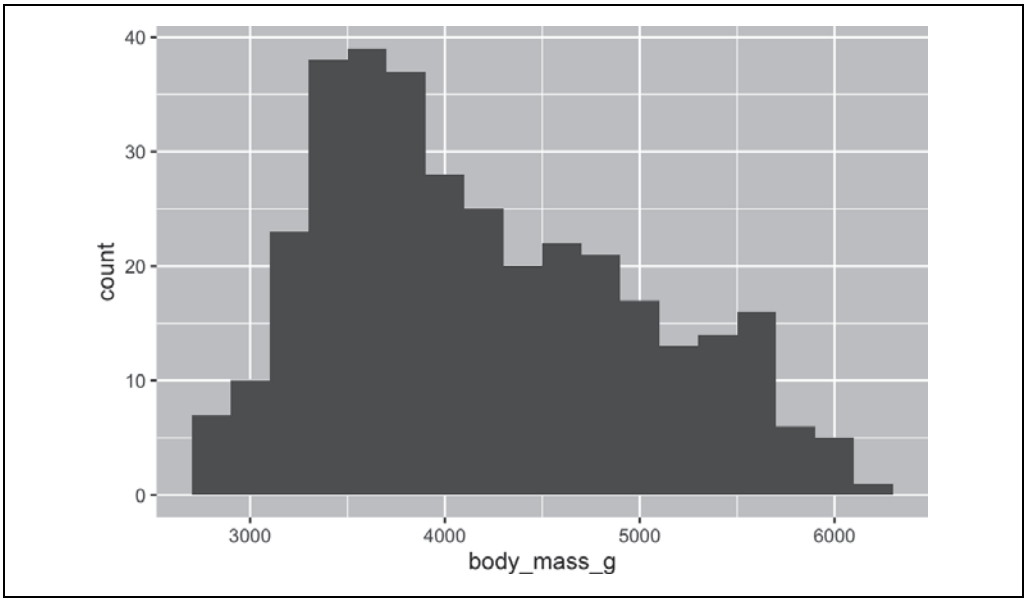
Zmienne liczbowe

Zmienna *liczbowa* (inaczej ilościowa) może przyjmować szeroki zakres wartości liczbowych i sensowne są operacje dodawania, odejmowania lub uśredniania takich wartości. Zmienne liczbowe mogą być ciągłe lub dyskretne.

Jedną z powszechnie stosowanych wizualizacji rozkładów zmiennych ciągłych jest histogram (rysunek 1.13).

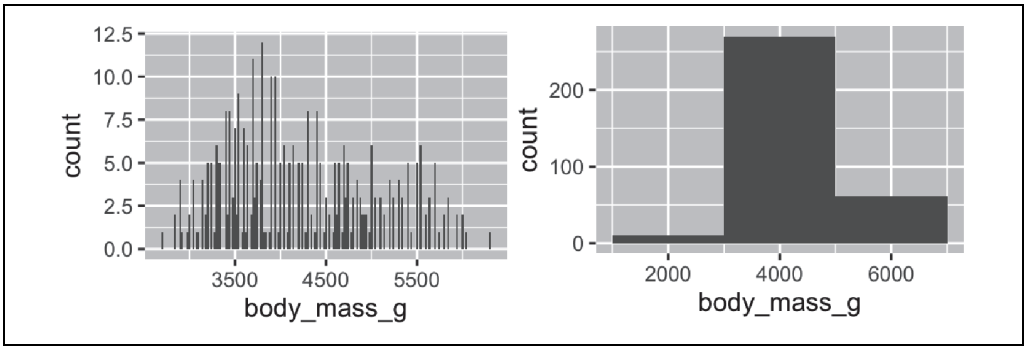
```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 200)
```

Histogram dzieli oś x na równo rozmieszczone przedziały. Wysokość słupka określa liczbę obserwacji, które mieszczą się w każdym przedziale. Na wykresie z rysunku 1.13 najwyższy słupek pokazuje, że 39 obserwacji ma wartość zmiennej `body_mass_g` między 3500 a 3700 gramów, które to wartości wyznaczają lewą i prawą krawędź słupka.



Rysunek 1.13. Histogram

Szerokość przedziałów w histogramie można określić za pomocą argumentu `binwidth`, który jest mierzony w jednostkach zmiennej `x`. Podczas pracy z histogramami należy zawsze sprawdzić różne wartości szerokości przedziału, ponieważ mogą one ujawniać różne wzorce. Na poniższych wykresach wartość `binwidth` wynosząca 20 skutkuje za wąskimi przedziałami, co prowadzi do zbyt dużej liczby słupków i utrudnia określenie kształtu rozkładu (rysunek 1.14). Podobnie wartość `binwidth` wynosząca 2000 jest zbyt duża, co powoduje, że wszystkie dane są podzielone na tylko trzy słupki, co także utrudnia określenie kształtu rozkładu. Wartość 200 zapewnia rozsądną równowagę.

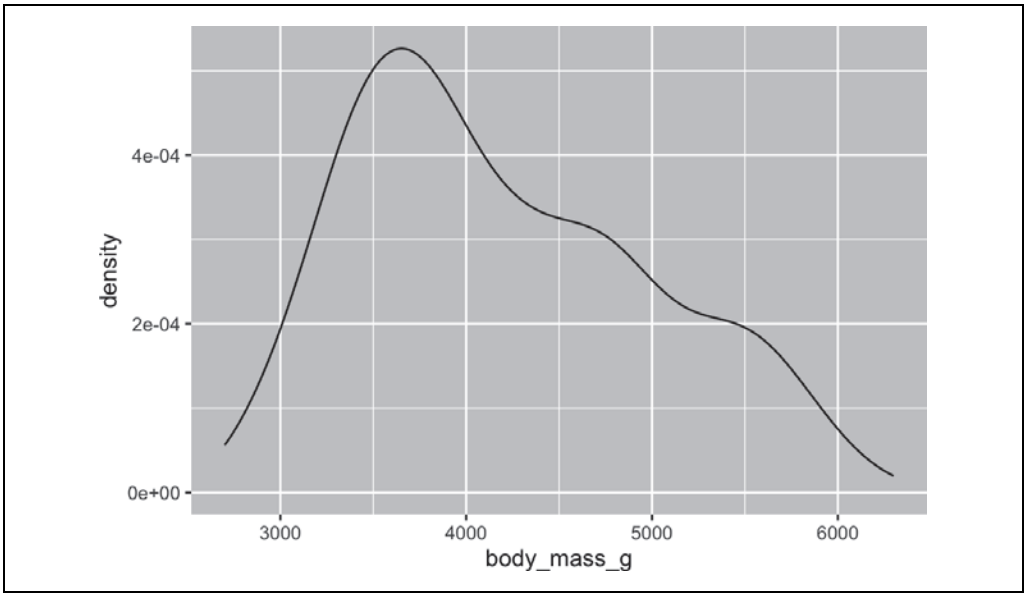


Rysunek 1.14. Za mała i za duża wartość argumentu binwidth

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 20)
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 2000)
```

Inną wizualizacją rozkładu zmiennych liczbowych jest wykres gęstości (rysunek 1.15). Wykres gęstości jest wygładzoną wersją histogramu i stanowi praktyczną alternatywę, szczególnie gdy używane są dane ciągłe o gładkim rozkładzie. Nie będziemy zagłębiać się w to, w jaki sposób funkcja `geom_density()` oblicza gęstość (więcej na ten temat przeczytasz w dokumentacji tej funkcji), warto jednak wyjaśnić, w jaki sposób rysowana jest krzywa gęstości. Posłużymy się do tego analogią. Wyobraź sobie histogram wykonany z drewnianych klocków. Następnie wyobraź sobie, że upuszczasz na niego ugotowane spaghetti. Kształt, jaki przybierze spaghetti położone na klocki, można uznać za kształt krzywej gęstości. Pokazuje ona mniej szczegółów niż histogram, ale może ułatwić szybkie poznanie kształtu rozkładu, a przede wszystkim ustalenie wartości modalnej i skośności.

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_density()
#> Warning: Removed 2 rows containing non-finite values (`stat_density()`).
```



Rysunek 1.15. Wykres gęstości

Ćwiczenia

1. Sporządź wykres słupkowy gatunków pingwinów (zmienna `species`), w którym przypiszesz zmienną `species` do właściwości estetycznej `y`. Czym ten wykres różni się od poprzednich?
2. Czym różnią się poniższe dwa wykresy? Która właściwość estetyczna, `color` czy `fill`, jest bardziej przydatna do zmiany koloru słupków?

```
ggplot(penguins, aes(x = species)) +
  geom_bar(color = "red")
ggplot(penguins, aes(x = species)) +
  geom_bar(fill = "red")
```

3. Do czego służy argument `bins` w funkcji `geom_histogram()`?

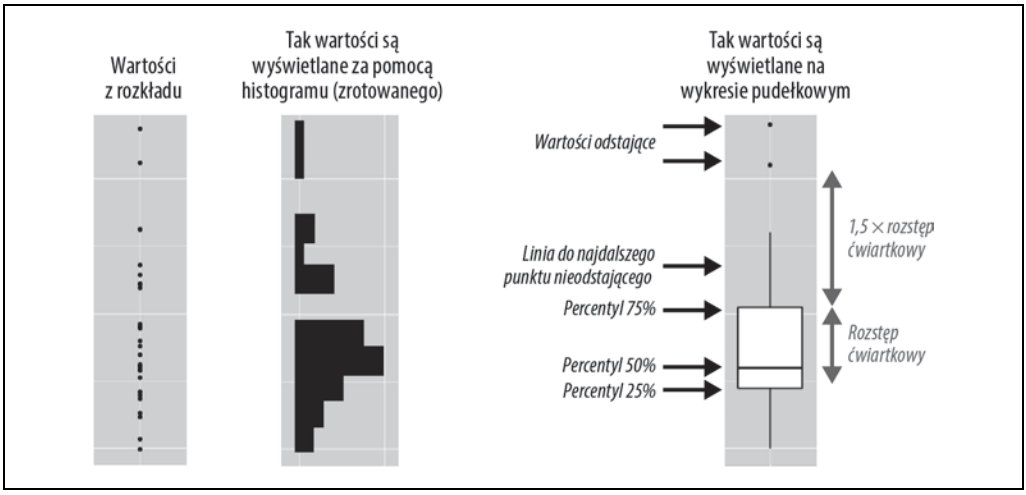
4. Utwórz histogram zmiennej `carat` ze zbioru danych `diamonds`, który jest dostępny po wczytaniu pakietu *tidyverse*. Poeksperymentuj z różnymi wartościami argumentu `binwidth`. Jaka wartość tego argumentu ujawnia najbardziej interesujące wzorce?

Wizualizacje relacji

Aby zwizualizować relację, potrzebne są co najmniej dwie zmienne odwzorowane na właściwości estetyczne wykresu. W następnych punktach poznasz wykresy często używane do wizualizowania relacji między dwiema zmiennymi lub większą ich liczbą oraz geomy stosowane do tworzenia takich wykresów.

Zmienne liczbowe i kategorialne

Aby zwizualizować związek między zmienną liczbową i kategorialną, można użyć wykresów pudełkowych jeden obok drugiego. **Wykres pudełkowy** to rodzaj wizualnego skrótu dla miar pozycji (percentyli), które opisują rozkład. Jest on również przydatny do identyfikowania ewentualnych wartości odstających. Jak pokazaliśmy na rysunku 1.16, każdy wykres pudełkowy składa się z następujących elementów:



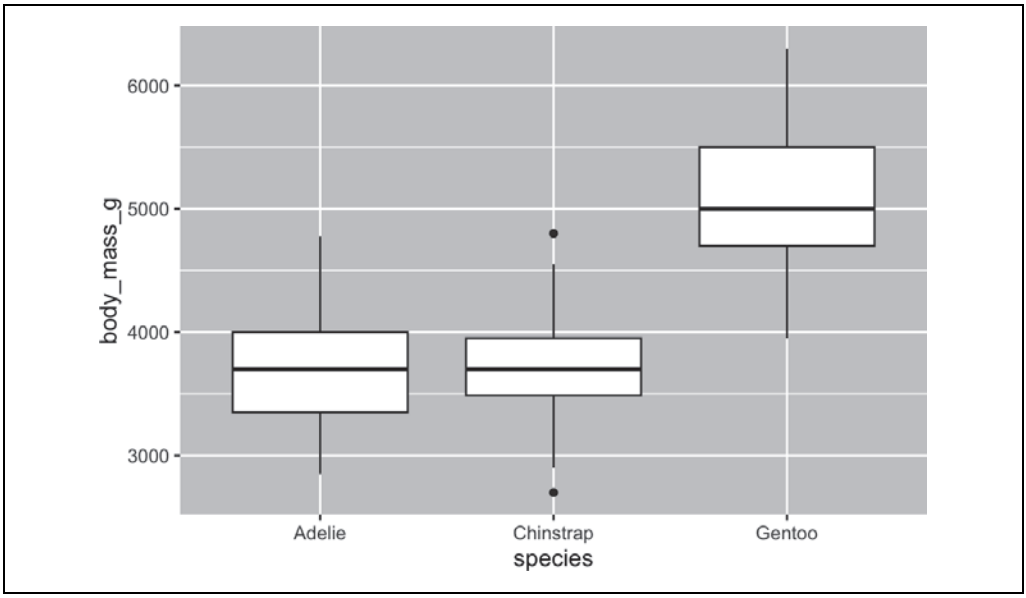
Rysunek 1.16. Rysunek ilustrujący, jak tworzony jest wykres pudełkowy

- „Pudełko", które określa zakres środkowej połowy danych. Jest to tak zwany **rozstęp ćwiartkowy** (ang. *interquartile range* — IQR). Rozciąga się on od percentyla 25% rozkładu do percentyla 75%. W środku pudełka znajduje się linia, która reprezentuje medianę, czyli percentyl 50% rozkładu. Te trzy linie dają wyobrażenie o rozrzucie rozkładu i o tym, czy rozkład jest symetryczny względem mediany, czy też skośny w którąś ze stron.
- Punkty, które reprezentują obserwacje wykraczające o więcej niż 1,5 × rozstęp ćwiartkowy od dowolnej krawędzi pudełka. Te odstające punkty są nietypowe, dlatego są wyświetlane pojedynczo.

- Linia (lub „wąs”), która rozciąga się od każdego końca pudełka i dochodzi do najdalszego nieodstającego punktu w rozkładzie.

Przyjrzyj się teraz rozkładowi masy ciała poszczególnych gatunków (rysunek 1.17). Użyj funkcji `geom_boxplot()`:

```
ggplot(penguins, aes(x = species, y = body_mass_g)) +
  geom_boxplot()
```



Rysunek 1.17. Rozkład masy ciała poszczególnych gatunków pingwinów

Możesz też wygenerować wykresy gęstości za pomocą funkcji `geom_density()` (rysunek 1.18).

```
ggplot(penguins, aes(x = body_mass_g, color = species)) +
  geom_density(linewidth = 0.75)
```

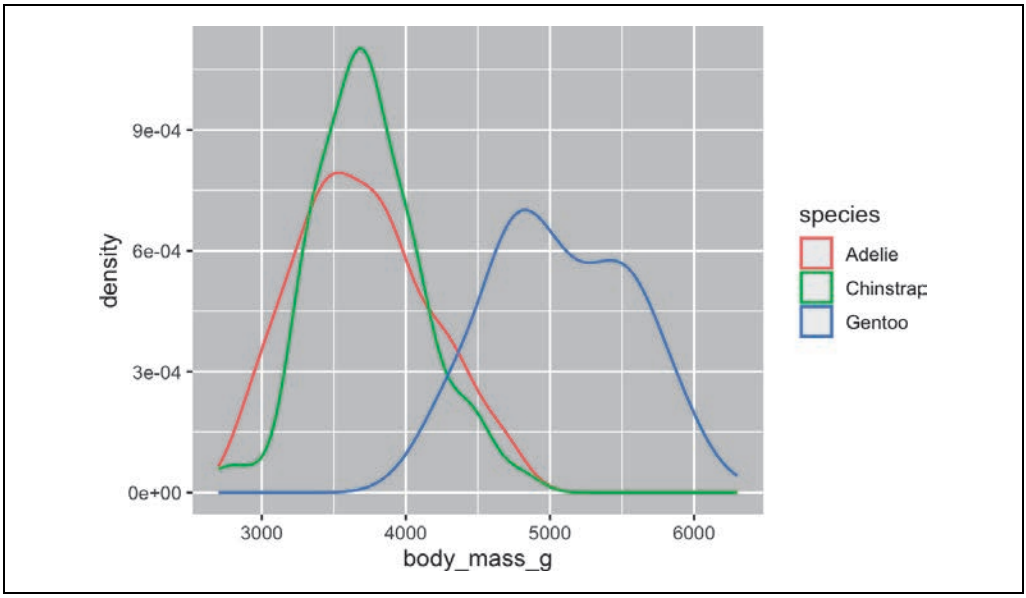
Dostosowaliśmy również grubość linii za pomocą argumentu `linewidth`, aby były lepiej widoczne.

Dodatkowo można odwzorować zmienną `species` na właściwości estetyczne `color` i `fill` oraz użyć właściwości estetycznej `alpha`, aby dodać przezroczystość do wypełnionych krzywych gęstości. Właściwość estetyczna `alpha` przyjmuje wartości od 0 (całkowicie przezroczysty) do 1 (całkowicie nieprzezroczysty). Na poniższym wykresie została użyta wartość 0,5 (rysunek 1.19).

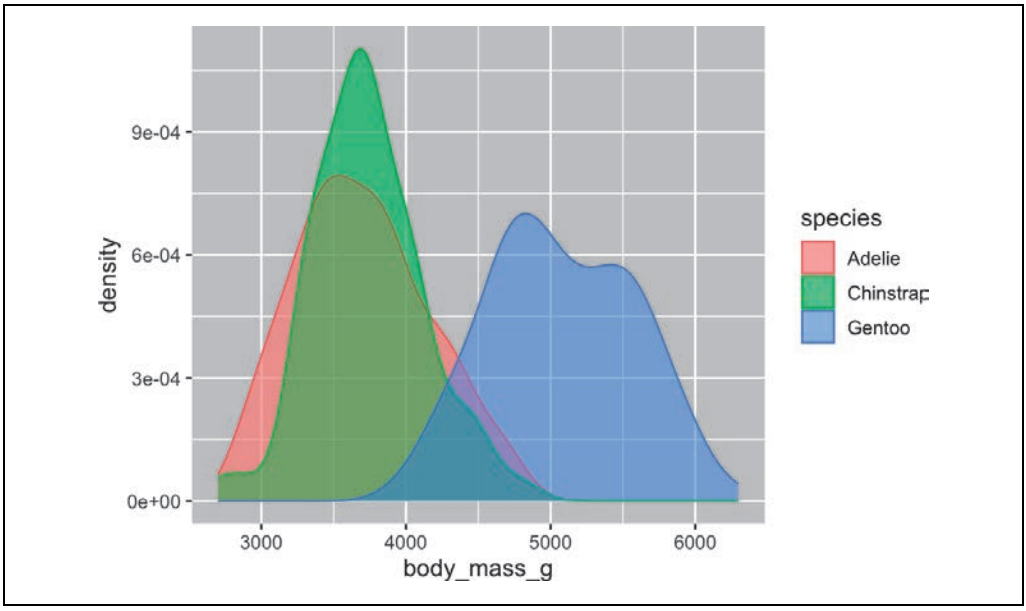
```
ggplot(penguins, aes(x = body_mass_g, color = species, fill = species)) +
  geom_density(alpha = 0.5)
```

Zwróć uwagę na terminologię, której tu używamy:

- *Odwzorowujemy* zmienne na właściwości estetyczne, jeśli chcemy, aby atrybut wizualny powiązany z daną właściwością estetyczną zmieniał się w zależności od wartości tej zmiennej.
- W przeciwnym razie *ustawiamy* wartość właściwości estetycznej.



Rysunek 1.18. Wykres gęstości



Rysunek 1.19. Wykres z częściowo przezroczystymi obszarami

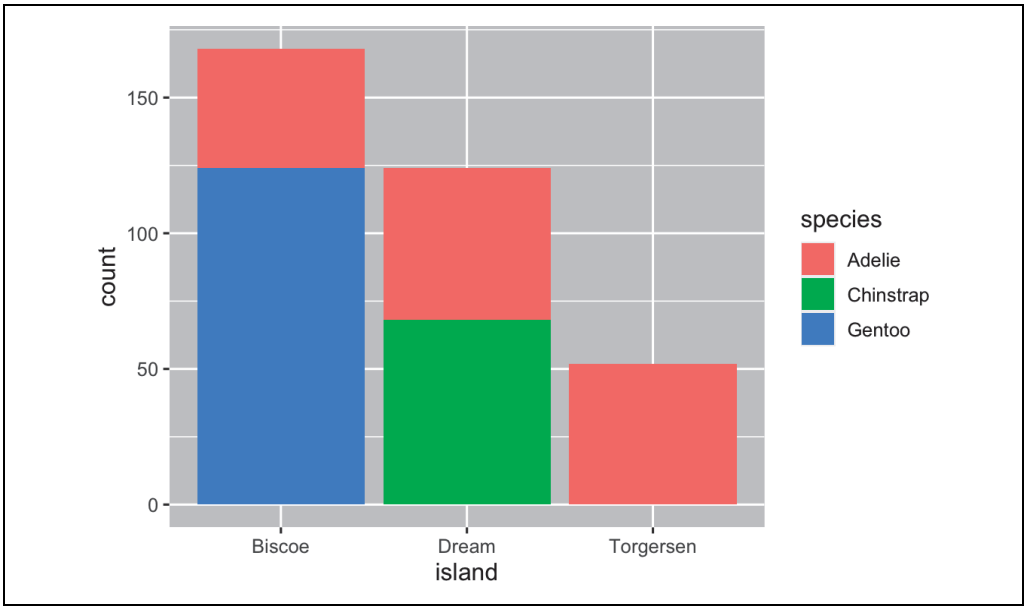
Dwie zmienne kategorialne

Do wizualizacji relacji między dwiema zmiennymi kategorialnymi można użyć skumulowanych wykresów słupkowych. Na przykład poniższe dwa wykresy słupkowe przedstawiają związek między

wyspą (zmienna `island`) a gatunkiem (zmienna `species`), a konkretnie wizualizują rozkład gatunków na każdej wyspie.

Pierwszy wykres przedstawia częstotliwość występowania każdego gatunku pingwinów na każdej wyspie. Ten wykres pokazuje, że na każdej wyspie występuje równa liczba pingwinów białookich, ale nie pozwala precyzyjnie ocenić procentowego rozkładu gatunków na każdej wyspie (rysunek 1.20).

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar()
```

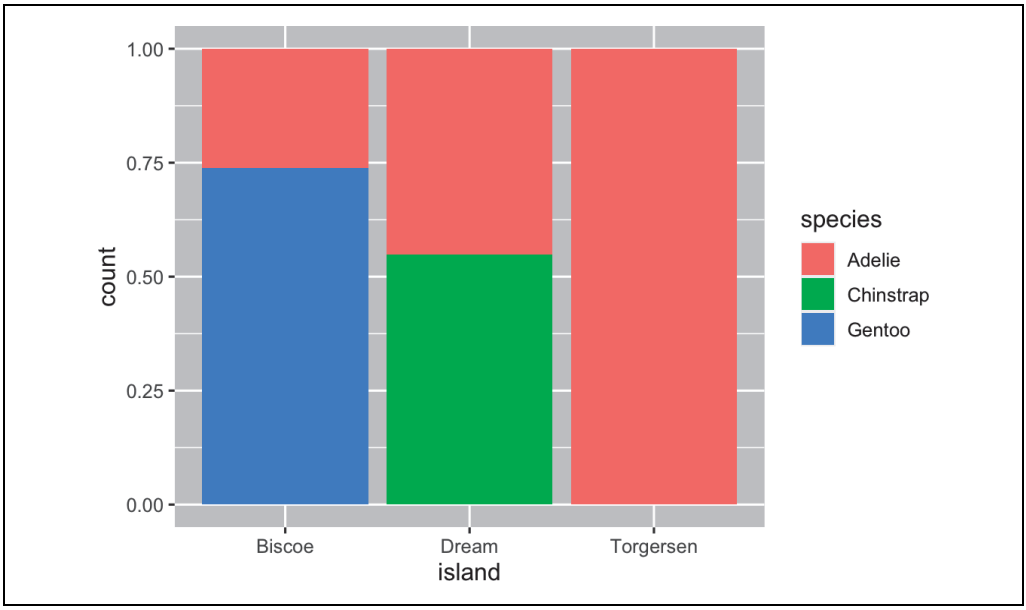


Rysunek 1.20. Na tym wykresie trudno ocenić procentowy rozkład gatunków na poszczególnych wyspach

Drugi wykres (rysunek 1.21) ilustruje względną częstotliwość i został utworzony za pomocą opcji `position = "fill"` dla geomu. Ten wykres jest bardziej przydatny do porównywania rozkładu gatunków na wyspach, ponieważ nie ma na niego wpływu różna liczba pingwinów na wyspach. Na podstawie tego wykresu można stwierdzić, że wszystkie pingwiny białobrewe żyją na wyspie Biscoe i stanowią tam około 75% wszystkich pingwinów, wszystkie pingwiny maskowe żyją na wyspie Dream i stanowią tam około 50% pingwinów, a pingwiny białookie żyją na wszystkich trzech wyspach i jako jedyne zamieszkują wyspę Torgersen.

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(position = "fill")
```

Podczas tworzenia wykresów słupkowych należy odwzorować zmienną, która zostanie rozdzielona między słupki, na właściwość estetyczną `x`, a zmienną, która wpływa na kolory wewnątrz słupków, na właściwość estetyczną `fill`.



Rysunek 1.21. Teraz procentowy rozkład jest lepiej widoczny

Dwie zmienne liczbowe

Do tej pory w kontekście wizualizowania relacji między dwiema zmiennymi liczbowymi opisaliśmy wykresy punktowe (tworzone za pomocą funkcji `geom_point()`) i gładkie krzywe (tworzone przy użyciu funkcji `geom_smooth()`). Wykres punktowy jest prawdopodobnie najczęściej używanym wykresem do wizualizowania relacji między dwiema zmiennymi liczbowymi (rysunek 1.22).

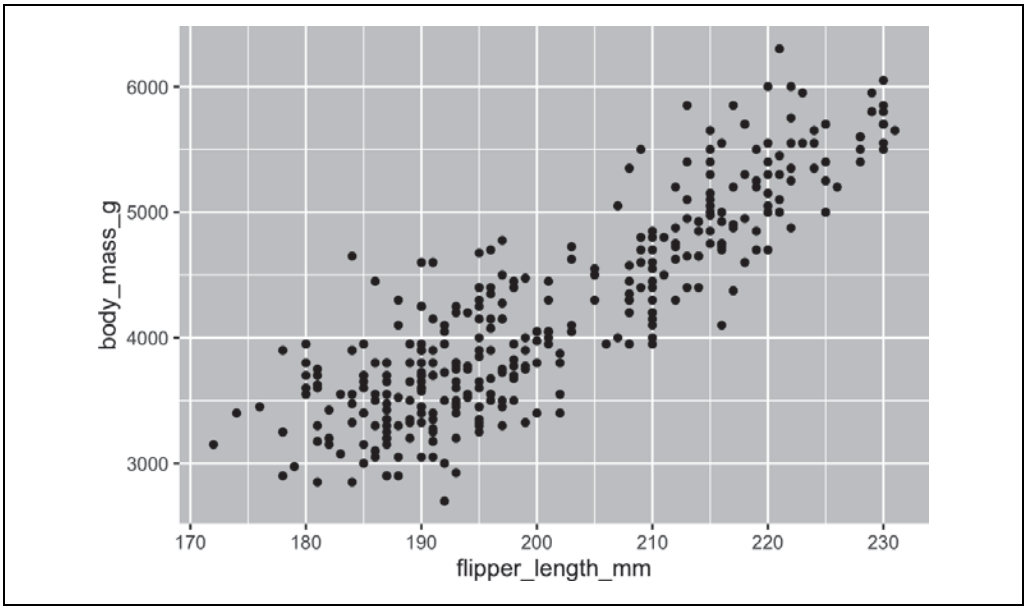
```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

Trzy zmienne lub większa ich liczba

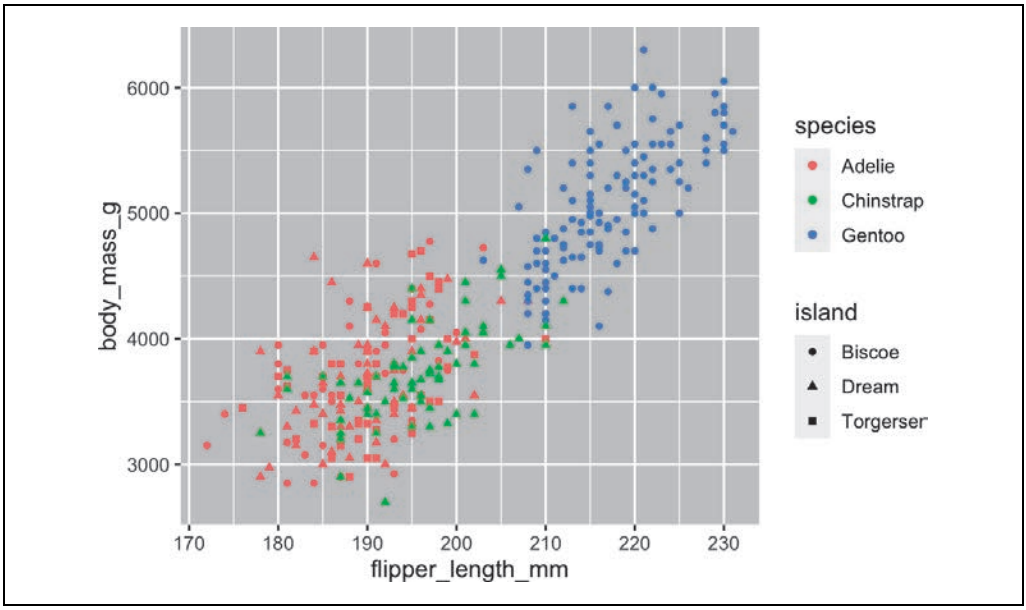
Jak wyjaśniliśmy w podrozdziale „Dodawanie właściwości estetycznych i warstw", można uwzględnić na wykresie więcej zmiennych dzięki odwzorowaniu ich na dodatkowe właściwości estetyczne. Na przykład na poniższym wykresie punktowym kolory punktów reprezentują gatunki, a kształty punktów reprezentują wyspy (rysunek 1.23).

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = island))
```

Jednak dodanie do wykresu zbyt wielu odwzorowań na właściwości estetyczne sprawia, że staje się on przeładowany i mało zrozumiały. Inną opcją, która jest przydatna zwłaszcza dla zmiennych kategorialnych, jest podzielenie wykresu na *fasety*, czyli podwykresy, z których każdy wyświetla jeden podzbiór danych.



Rysunek 1.22. Wykres punktowy



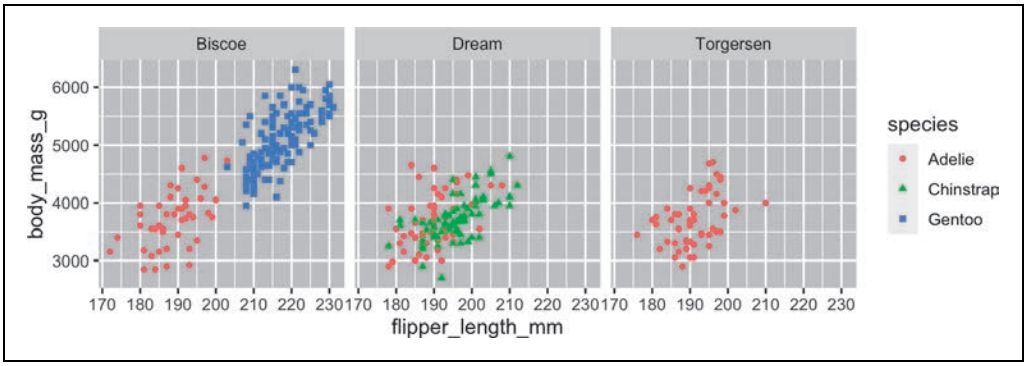
Rysunek 1.23. Różne właściwości estetyczne pozwalają zilustrować dodatkowe informacje

Aby podzielić wykres na fasety (rysunek 1.24) według pojedynczej zmiennej, należy użyć funkcji `facet_wrap()`. Pierwszym argumentem tej funkcji jest formuła[3] dodawana za pomocą znaku ~,

[3] W tym miejscu słowo „formuła" oznacza konstrukcję tworzoną za pomocą znaku ~, a nie synonim słowa „równanie".

po którym następuje nazwa zmiennej. Zmienna przekazywana do funkcji `facet_wrap()` powinna być zmienną kategorialną.

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  facet_wrap(~island)
```



Rysunek 1.24. Podwykresy uzyskane na podstawie jednej zmiennej

W rozdziale 9. poznasz wiele innych geomów służących do wizualizowania rozkładów zmiennych i relacji między nimi.

Ćwiczenia

1. Ramka danych `mpg`, która jest dołączona do pakietu *ggplot2*, zawiera 234 obserwacje zebrane przez amerykańską Agencję Ochrony Środowiska na temat 38 modeli samochodów. Które zmienne w ramce danych `mpg` są kategorialne? Które zmienne są liczbowe? Wskazówka: wpisz instrukcję `?mpg`, aby zapoznać się z dokumentacją zbioru danych. Jak można znaleźć te informacje po wpisaniu instrukcji `mpg`?
2. Utwórz wykres punktowy zmiennej `hwy` względem zmiennej `displot` na podstawie ramki danych `mpg`. Następnie odwzoruj trzecią zmienną liczbową na właściwość `color`, potem na właściwość `size`, potem jednocześnie na właściwości `color` i `size`, a potem na właściwość `shape`. Jakie są różnice w działaniu tych właściwości estetycznych dla zmiennych kategorialnych i liczbowych?
3. Co się stanie, jeśli na wykresie punktowym zmiennej `hwy` względem zmiennej `displot` trzecia zmienna zostanie odwzorowana na właściwość `linewidth`?
4. Co się stanie, jeśli ta sama zmienna zostanie odwzorowana na wiele właściwości estetycznych?
5. Utwórz wykres punktowy zmiennej `bill_depth_mm` względem zmiennej `bill_length_mm` i przypisz punktom kolory na podstawie gatunków (zmienna `species`). Co dodanie kolorów na podstawie gatunków ujawnia na temat zależności między tymi dwiema zmiennymi? Czego dowiesz się po podziale wykresu na fasety na podstawie gatunków?
6. Dlaczego poniższy kod generuje dwie odrębne legendy? Jak poprawisz kod, aby połączyć te dwie legendy?

```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm, y = bill_depth_mm,
    color = species, shape = species
  )
) +
  geom_point() +
  labs(color = "Species")
```

7. Utwórz dwa skumulowane wykresy słupkowe. Na jakie pytanie można odpowiedzieć za pomocą pierwszego wykresu? Na jakie pytanie można odpowiedzieć na podstawie drugiego wykresu?

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(position = "fill")
ggplot(penguins, aes(x = species, fill = island)) +
  geom_bar(position = "fill")
```

Zapisywanie wykresów

Po utworzeniu wykresu możesz chcieć usunąć go z R i zapisać jako grafikę, którą możesz wykorzystać w innym miejscu. Służy do tego funkcja `ggsave()`, która zapisuje ostatnio utworzony wykres na dysku:

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
ggsave(filename = "penguin-plot.png")
```

Spowoduje to zapisanie wykresu w katalogu roboczym, o czym więcej dowiesz się z rozdziału 6.

Jeśli nie określisz szerokości i wysokości (argumenty `width` i `height`), zostaną one ustalone na podstawie wymiarów bieżącego urządzenia. Jeśli chcesz, by kod działał powtarzalnie, należy określić te argumenty. Więcej informacji na temat funkcji `ggsave()` znajdziesz w dokumentacji.

Zalecamy jednak generowanie raportów końcowych za pomocą Quarto, systemu do powtarzalnego tworzenia materiałów, który umożliwia przeplatanie kodu z tekstem oraz automatyczne dołączanie wykresów. Więcej informacji na temat Quarto znajdziesz w rozdziale 28.

Ćwiczenia

1. Uruchom poniższe wiersze kodu. Który z dwóch wykresów zostanie zapisany w pliku *mpg-plot.png*? Dlaczego?

```
ggplot(mpg, aes(x = class)) +
  geom_bar()
ggplot(mpg, aes(x = cty, y = hwy)) +
  geom_point()
ggsave("mpg-plot.png")
```

2. Co należy zmienić w tym kodzie, aby zapisać wykres w formacie PDF zamiast w formacie PNG? Jak można sprawdzić, jakie typy plików graficznych są obsługiwane przez funkcję `ggsave()`?

Typowe problemy

Gdy zaczniesz uruchamiać kod w języku R, prawdopodobnie napotkasz problemy. Nie przejmuj się — to zdarza się każdemu. Wszyscy piszemy kod w języku R od lat, ale każdego dnia zdarza nam się napisać fragment, który nie działa przy pierwszej próbie uruchomienia!

Zacznij od dokładnego porównania kodu, który uruchamiasz, z kodem w książce. Język R jest niezwykle drobiazgowy, a źle umieszczony znak może wpłynąć na działanie kodu. Upewnij się, że każdy nawias (jest dopasowany do nawiasu), a każdy cudzysłów " ma parę w postaci drugiego cudzysłowu ". Czasami uruchomisz kod i nic się nie stanie. Sprawdź wtedy lewą stronę konsoli. Jeśli widzisz symbol +, oznacza to, że według języka R nie wpisałeś kompletnego wyrażenia, dlatego system czeka, aż je dokończysz. W takiej sytuacji zwykle łatwo jest zacząć pracę od nowa przez wciśnięcie klawisza *Escape*, aby przerwać przetwarzanie bieżącego polecenia.

Jednym z częstych problemów podczas tworzenia grafiki za pomocą pakietu *ggplot2* jest umieszczenie znaku + w niewłaściwym miejscu. Musi on znajdować się na końcu wiersza, a nie na jego początku. Innymi słowy, upewnij się, że przypadkowo nie napisałeś takiego kodu:

```
ggplot(data = mpg)
+ geom_point(mapping = aes(x = displ, y = hwy))
```

Jeśli nadal nie możesz zrobić postępów, spróbuj skorzystać z pomocy. Aby uzyskać pomoc na temat dowolnej funkcji języka R, uruchom polecenie `?nazwa_funkcji` w konsoli lub zaznacz nazwę funkcji i wciśnij klawisz *F1* w środowisku RStudio. Nie przejmuj się, jeśli pomoc nie wydaje się zbyt przydatna. Przejdź wtedy do przykładów i poszukaj kodu, który odpowiada zadaniu, jakie próbujesz wykonać.

Jeśli to nie pomoże, uważnie przeczytaj komunikat o błędzie. Czasami znajdziesz w nim odpowiedź! Ale jeśli dopiero zaczynasz przygodę z językiem R, to nawet jeżeli odpowiedź znajduje się w komunikacie o błędzie, możesz jeszcze nie wiedzieć, jak ją zrozumieć. Innym świetnym narzędziem jest Google. Spróbuj wygooglować komunikat o błędzie, ponieważ jest wysoce prawdopodobne, że ktoś inny miał ten sam problem i uzyskał pomoc w internecie.

Podsumowanie

W tym rozdziale omówiliśmy podstawy wizualizowania danych za pomocą pakietu *ggplot2*. Zaczęliśmy od podstawowej idei działania tego pakietu — wizualizacja to odwzorowanie zmiennych z danych na właściwości estetyczne takie jak położenie, kolor, rozmiar czy kształt. Następnie pokazaliśmy, jak zwiększyć złożoność wykresu i poprawić jego wygląd przez dodawanie kolejnych warstw. Omówiliśmy również wykresy często używane do wizualizacji rozkładu pojedynczej zmiennej, a także opisaliśmy wizualizację relacji między dwiema zmiennymi lub większą ich liczbą z wykorzystaniem dodatkowych odwzorowań na właściwości estetyczne i/lub podziału wykresu na mniejsze części za pomocą fasetów.

W tej książce wielokrotnie będziemy korzystać z wizualizacji. Tam, gdzie będzie to potrzebne, będziemy wprowadzać nowe techniki, a ponadto w rozdziałach od 9. do 11. zagłębimy się w tworzenie wizualizacji za pomocą pakietu *ggplot2*.

Teraz, gdy znasz już podstawy wizualizacji, w następnym rozdziale zamierzamy nieco zmienić temat i przedstawić kilka praktycznych porad dotyczących przepływu pracy. W tej części książki przeplatamy porady dotyczące przepływu pracy z omawianiem narzędzi do danologii, ponieważ pomoże Ci to zachować porządek podczas pisania coraz większej ilości kodu w języku R.

To zaskakująco dobra aktualizacja światowej klasy przewodnika po danologii z użyciem języka R

Emma Rand, University of York

Aby w pełni wykorzystać potencjał danych i przekształcać je w wartościową wiedzę, musisz się posługiwać odpowiednimi narzędziami. Szczególnie przyda Ci się znajomość języka R, który pozwala na efektywne wykonywanie zadań, od importowania surowych danych po komunikowanie uzyskanych wyników w zrozumiały sposób.

Oto drugie, zaktualizowane wydanie znakomitego przewodnika dla analityków danych. Dzięki niemu dowiesz się, w jaki sposób używać języka R do importowania, przekształcania i wizualizowania danych, a także do przekazywania uzyskanych wyników analizy. Nauczysz się też rozwiązywać najczęściej występujące problemy, a liczne ćwiczenia ułatwią Ci utrwalenie zdobytej wiedzy. Omówiono tu najnowsze funkcje języka i najlepsze praktyki w data science. Zaprezentowano również zasady korzystania z wielu bibliotek języka R, na przykład tidyverse, służącej do pobierania informacji z różnych źródeł.

Dzięki tej książce nauczysz się:

- wizualizować, czyli tworzyć wykresy na potrzeby eksploracji danych
- przekształcać, czyli pracować z różnymi typami zmiennych
- importować, czyli pobierać dane w formie wygodnej do analizy
- programować, czyli rozwiązywać problemy z danymi za pomocą języka R
- przekazywać informacje, czyli pracować z użyciem Quarto

Hadley Wickham jest głównym specjalistą naukowym w firmie Posit i członkiem R Foundation. Tworzy narzędzia ułatwiające pracę z danymi.

Mine Çetinkaya-Rundel jest dyrektorem studiów licencjackich na Wydziale Nauk Statystycznych Duke University i trenerem programistów w firmie Posit.

Garrett Grolemund jest autorem książki *Hands-On Programming with R* i dyrektorem do spraw nauczania w firmie Posit.

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 978-83-289-0653-2
9 788328 906532
Cena: 129,00 zł