Podstawy grafiki w R

Statystyka I z R

Bartosz Maćkiewicz

Metafora płótna

W R dostępnych jest wiele bibliotek do tworzenia wykresów. My w tej części kursu zajmiemy się podstawową, wbudowaną w R biblioteką nazwaną `graphics`. O rysowaniu wykresów w R możemy myśleć jak o rysowaniu wykresów na płótnie. Płótno podzielone jest na kilka obszarów. Obszary rysowania wyglądają mniej więcej tak:

Zwykle rysować będziemy na środkowym polu, a martwienie o pozostałe pola, takie zostawimy R.

Funkcje służące do rysowania w R możemy podzielić z grubsza na dwa typy. Pierwszy to funkcje wysokiego poziomu, drugi to funkcje niskiego poziomu.

Funkcje wysokiego poziomu.

Funkcje wysokiego poziomu co do zasady rysują cały wykres, to znaczy zarówno same kształty na wykresie jak i legendę, podpisy, ramki, osie, oznaczenia osi, tytuły, podtytuły itp. Przykładami funkcji wysokiego poziomu są funkcje `plot` (rysuje różne wykresy w zależności od przekazanych do funkcji danych) i `hist` (rysuje histogram).

Funkcje niskiego poziomu.

Funkcje nieskiego poziomu rysują określone kształty, najczęściej na już narysowanych wykresach. Przydają się do ulepszania i modyfikowania wykresów już stworzonych za pomoca funkcji wysokiego poziomu, ale mogą też służyć do narysowania czegoś "od zera". Przykłady takich funkcji to `abline` (nakłada na wykres linię), `points` (nakłada na wykres punkty) lub `text` (nakłada na wykres tekst).

Funkcja `plot`

Spróbujmy stworzyć najprostszy wykres punktowy (*scatterplot*, czasami po polsku również *wykres rozrzutu*). Użyjemy do tego znanego już z ćwiczeń domowych zbioru danych z bobrami i spróbujemy na wykresie zilustrować to, jak zmieniała się temperatura bobra w czasie. Zobaczymy również różne rodzaje wykresów (argument `type` - za pomocą `?plot` można dowiedzieć się, jakie inne wartości może przyjmować) oraz jak określić tytuł wykresu (argument `main`).

W tym ćwiczeniu skorzystamy jedynie z podzbioru ramki danych `beaver1`. Chcemy użyć obserwacji pochodzącego z pierwszego dnia pomiarów.

```
data(beavers)
beav <- beaver1[beaver1$day == 346, ] # wybieramy tylko dzień 346
# Zobaczymy jak wyglądają dane, które przekazujemy funkcji `plot`
head(beav)
```

```
##   day time  temp activ
## 1 346  840 36.33     0
```

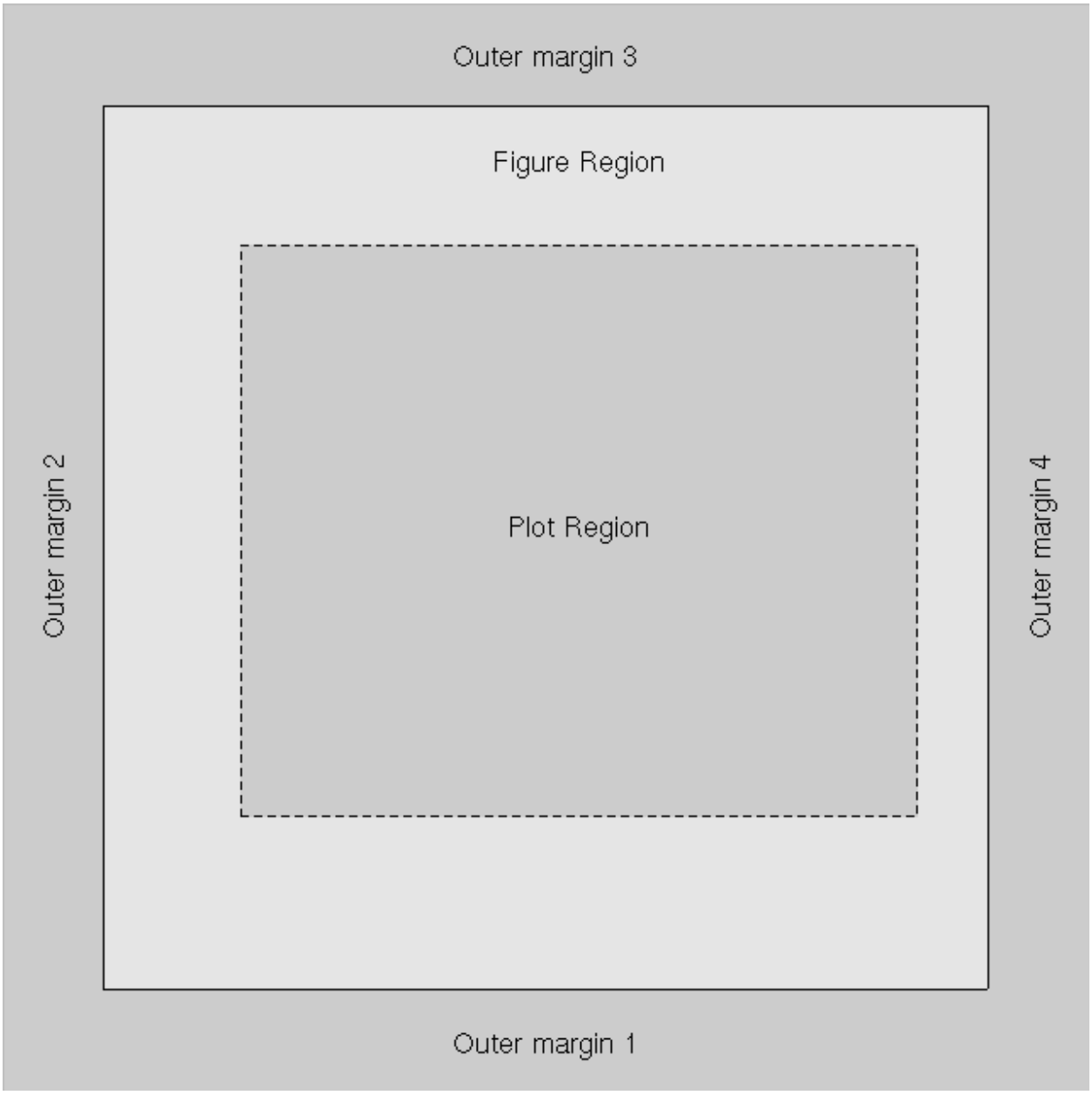


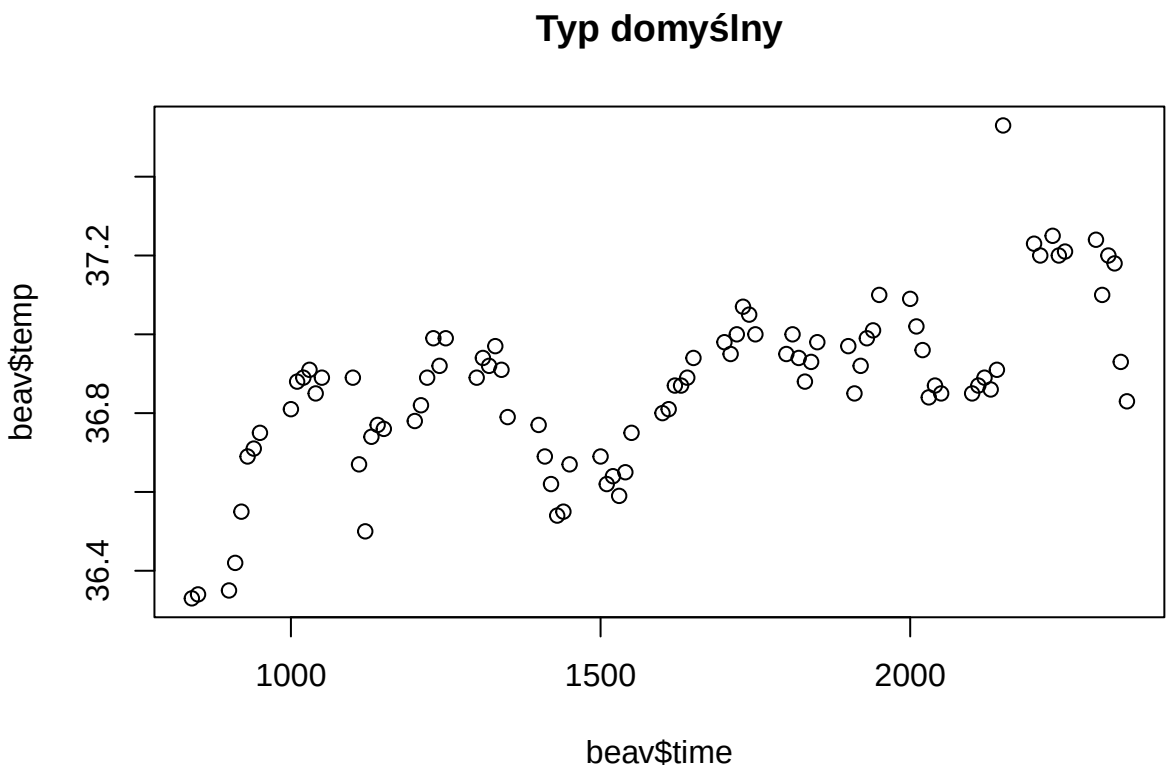
Figure 1: Obszary rysowania

```
## 2 346  850 36.34     0
## 3 346  900 36.35     0
## 4 346  910 36.42     0
## 5 346  920 36.55     0
## 6 346  930 36.69     0
```

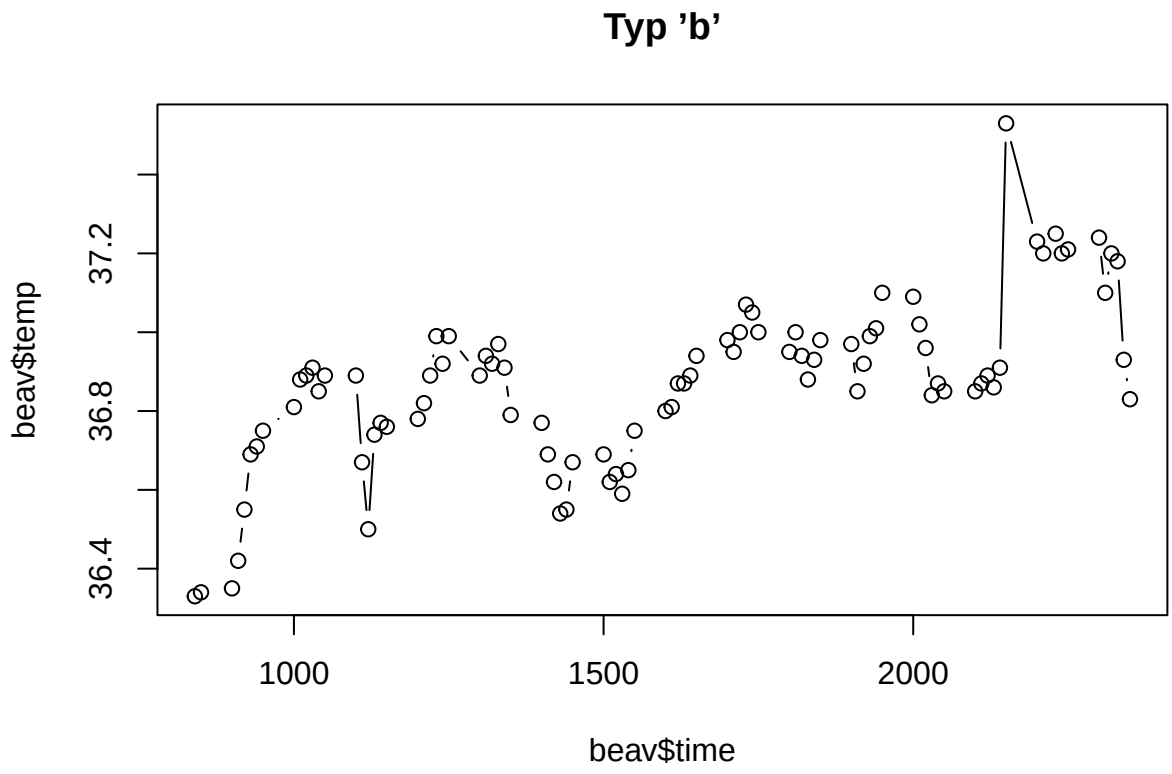
Funkcja `plot` w podstawowej formie przyjmuje dwa argumenty. Pierwszy to wektor ze współrzędnymi punktów na osi X, drugi to wektor ze współrzednymi punktów na osi Y.

Poniżej znajdują się przykłady różnych typów wykresów.

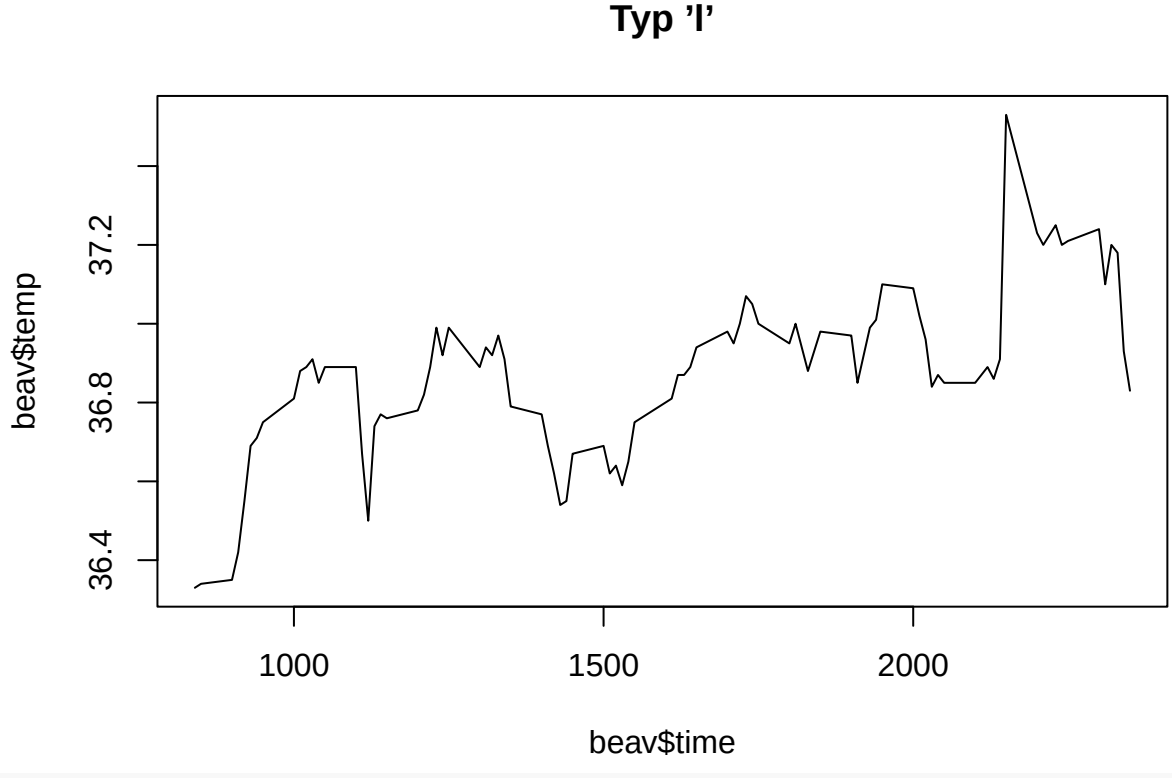
```
plot(beav$time, beav$temp, main = "Typ domyślny")
```



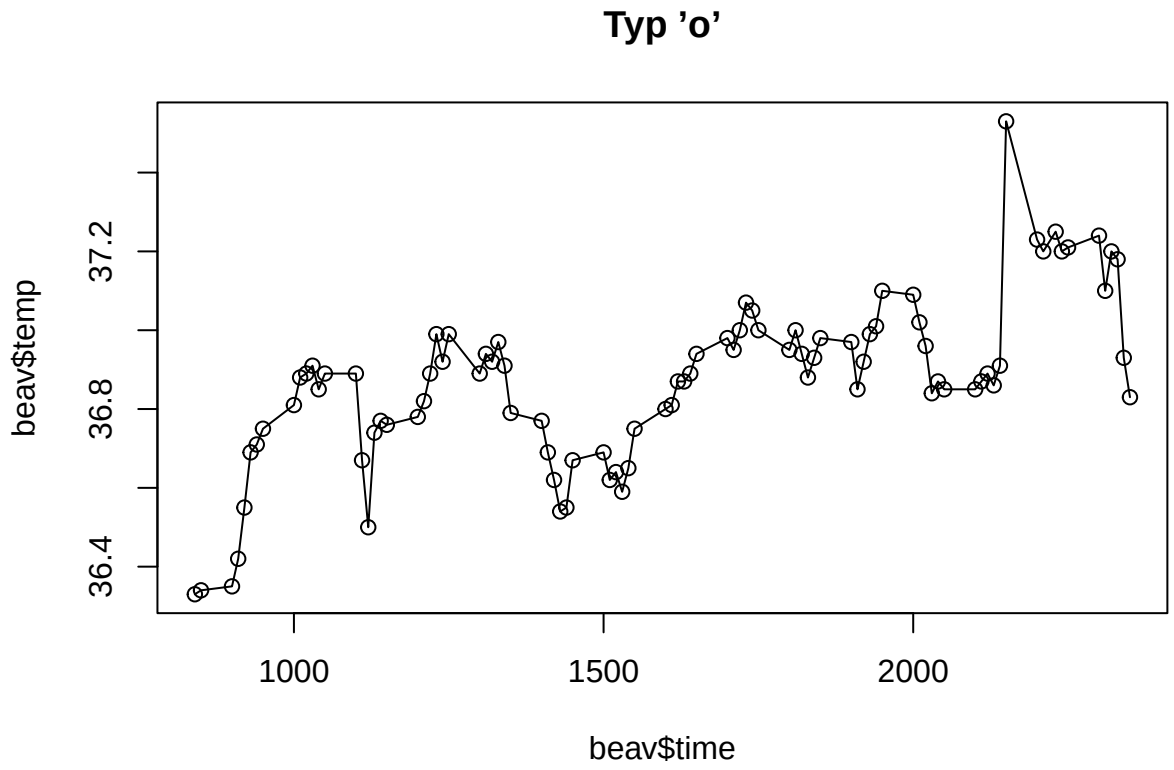

```
plot(beav$time, beav$temp, type='b', main = "Typ 'b'")
```



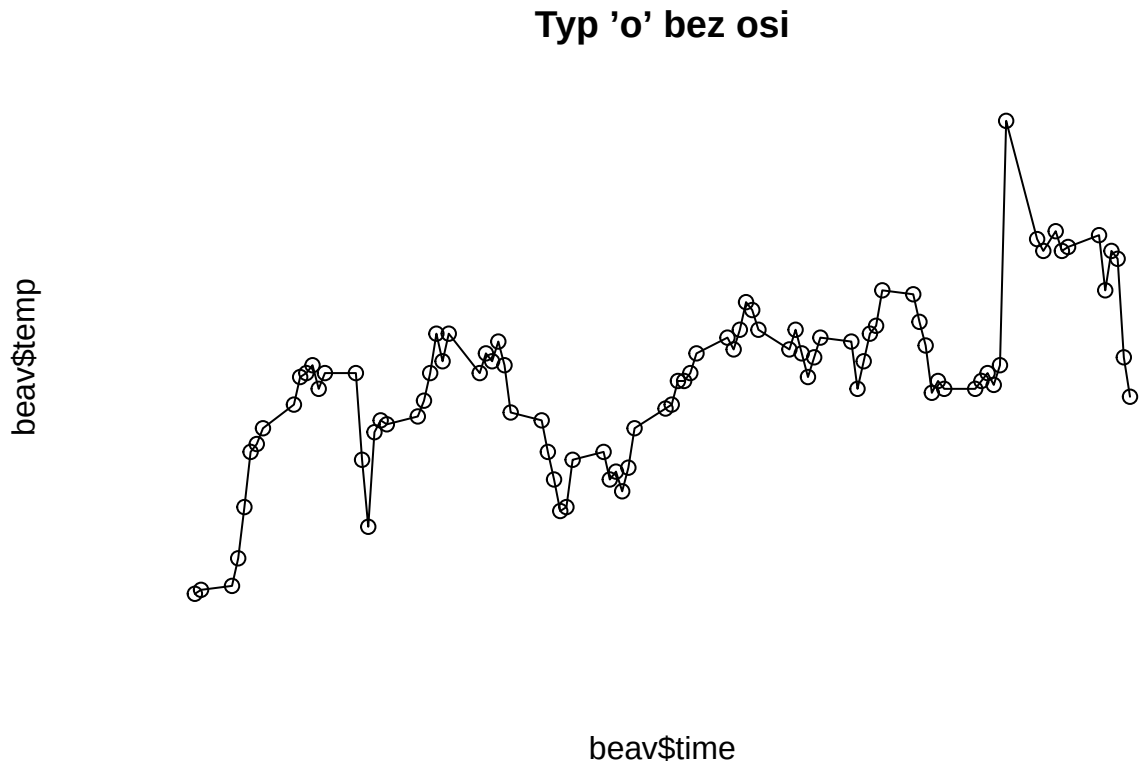

```
plot(beav$time, beav$temp, type='l', main = "Typ 'l'")
```




```
plot(beav$time, beav$temp, type='o', main = "Typ 'o'")
```

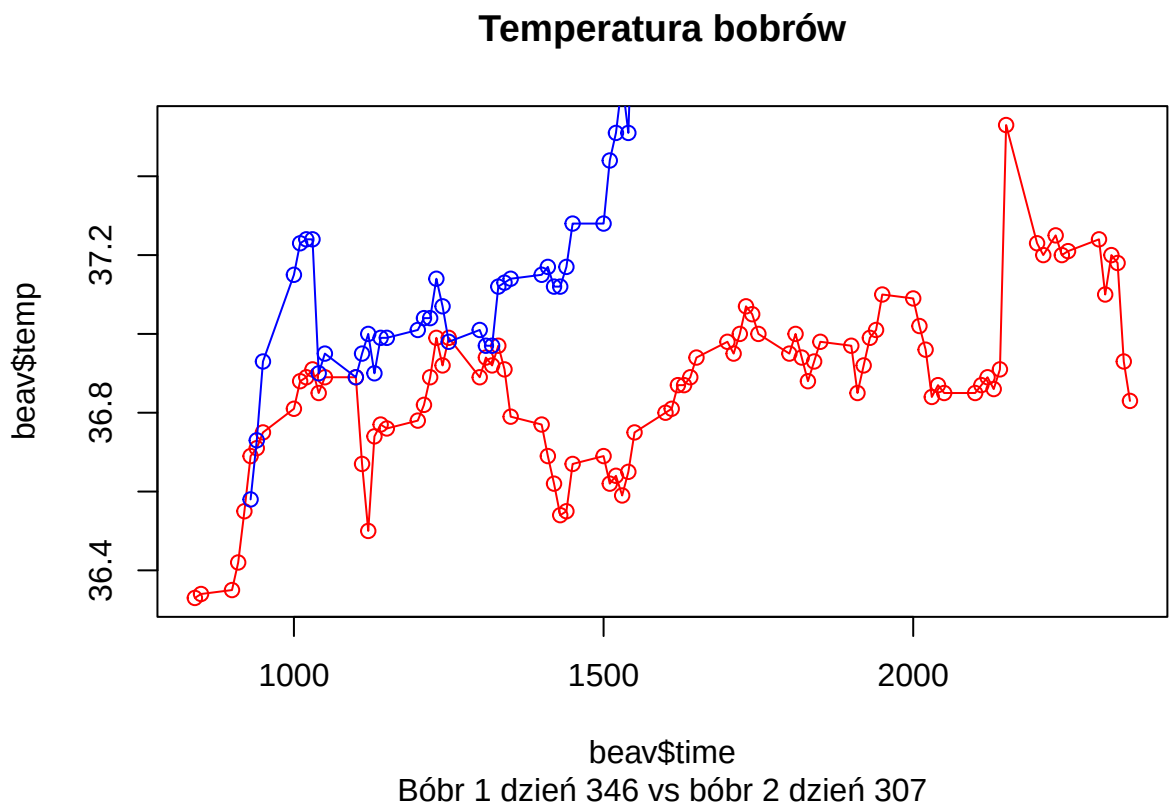



```
plot(beav$time, beav$temp, type='o', axes = F, main = "Typ 'o' bez osi")
```



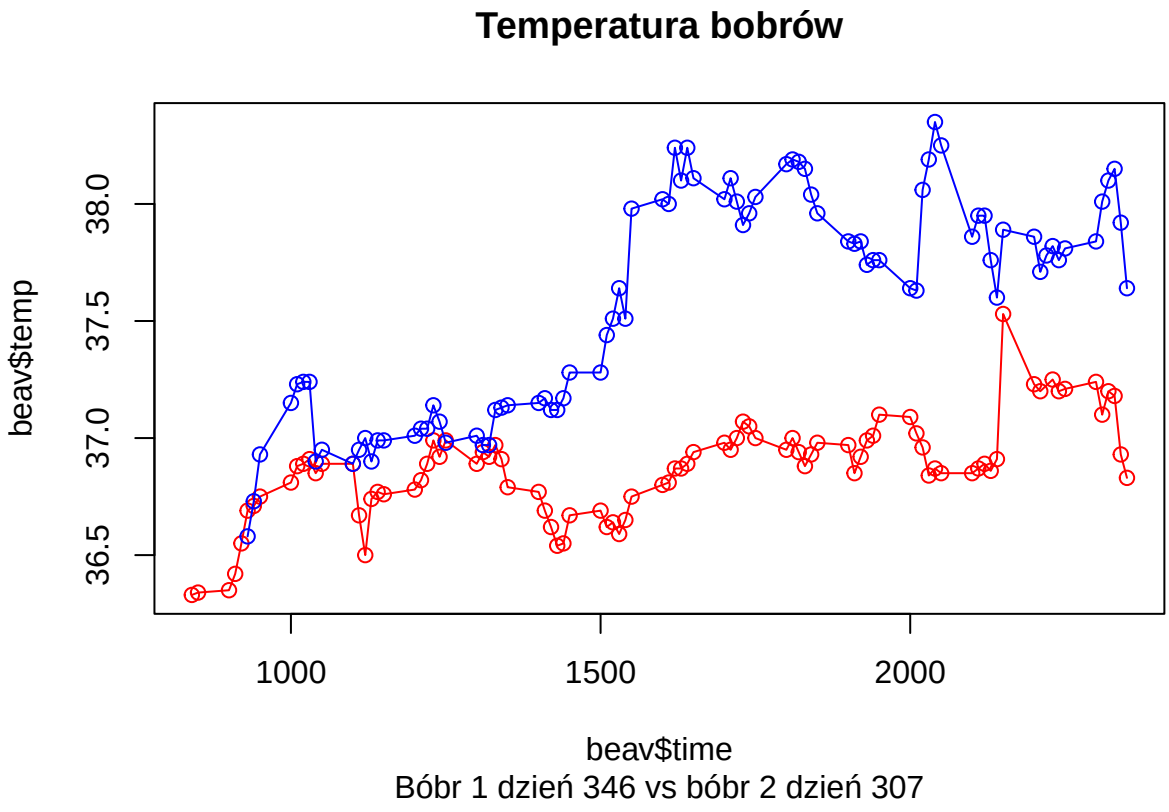
Spróbujmy nałożyć oba bobry na jeden wykres. Zrobimy to tworząc pierwszy z nich za pomocą funkcji wysokiego `plot`, a drugi za pomocą funkcji niskiego poziomu `points`. Funkcja `points` przyjmuje w zasadzie te same argumenty co `plot` ale rysuje wyłącznie punkty (a nie np. osie, podpisy itp.). W przypadku danych ze zbioru `beaver2` również skorzystamy z jednego dnia pomiarów.

```r
# wybieramy dane drugiego bobra z jednego dnia pomiarów
beav2 <- beaver2[beaver2$day == 307, ]

plot(beav$time, beav$temp, type='o',
     main='Temperatura bobrów', # główny tytuł wykresu
     sub='Bóbr 1 dzień 346 vs bóbr 2 dzień 307', # podtytuł wykresu
     col='Red', # kolor punktów
     )
points(beav2$time, beav2$temp, type='o', col='Blue')
```



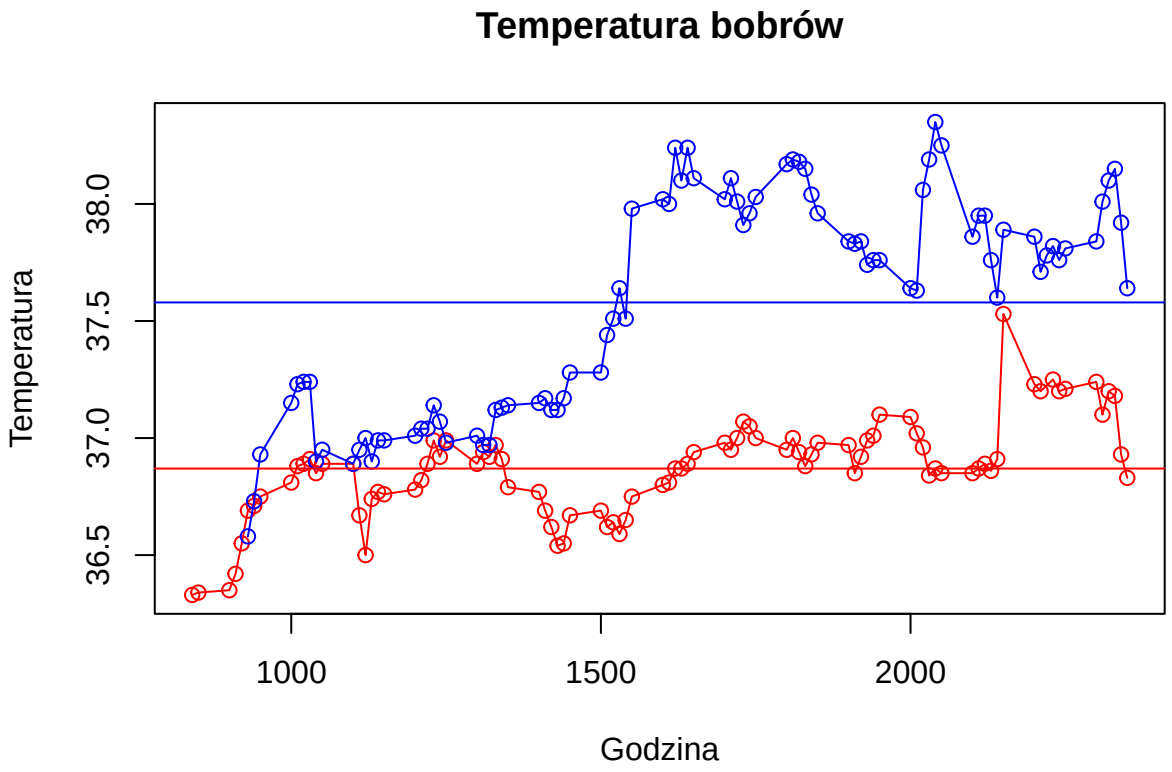
Niestety, niebieski bóbr wychodzi poza marginesy rysunku. Aby temu zapobiec, musimy ustawić skalę na osi Y tak, by obejmowała minimalne i maksymalne wartości obu bobór. W naszym przypadku R automatycznie dobiera wartości na Y tak, by mieścił się tylko pierwszy rysowany bóbr. Spróbujemy zmienić to zachowanie za pomocą argumenty `ylim`.

```r
plot(beav$time, beav$temp, type='o',
     main='Temperatura bobrów',
     sub='Bóbr 1 dzień 346 vs bóbr 2 dzień 307',
     col='Red',
     ylim = c(min(c(beav$temp, beav2$temp)), # najmniejsza obserwacja dla obu bobrów
              max(c(beav$temp, beav2$temp))) # największa obserwacja dla obu bobrów
     )

points(beav2$time, beav2$temp, type='o', col='Blue')
```



Funkcja `points` to nie jedyna funkcja niskiego poziomu, której możemy użyć. Za pomocą funkcji niskiego poziomu `abline` dodajmy na rysunku średnią temperatury obu bobrów. Zmienimy także podpisy osi na bardziej adekwatne.

```
plot(beav$time, beav$temp,
     type='o',
     main='Temperatura bobrów',
     col='Red',
     ylim = c(min(c(beav$temp, beav2$temp)),
              max(c(beav$temp, beav2$temp))),
     # podpisy osi ustawia się za pomocą argumentów ylab i xlab
     ylab = 'Temperatura',
     xlab = 'Godzina')
points(beav2$time, beav2$temp, type='o', col='Blue')

# abline przyjmuje różne argumenty
# z argumentem `h` rysuje linię poziomą (horizontal) na określonej wysokości
abline(h = mean(beav$temp), col = 'Red')
abline(h = mean(beav2$temp), col = 'Blue')
```



Graficzne funkcje niskiego poziomu

Poniżej znajduje się lista funkcji niskiego poziomu. Nie jest wyczerpująca, ale są to najbardziej przydatne funkcje. O każdej z tych funkcji można doczytać w dokumentacji.

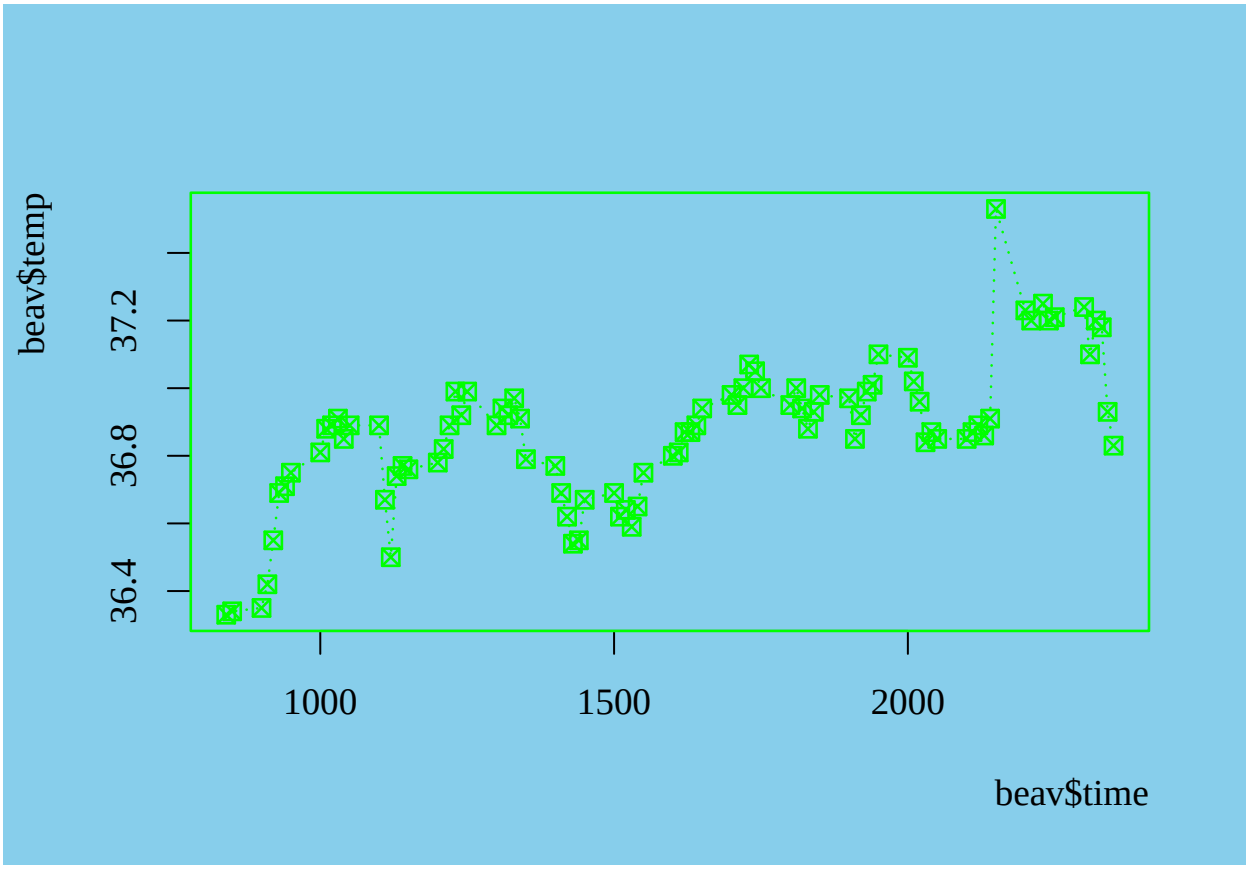
- `abline` - rysuje prostą
- `arrows` - dodaje strzałkę
- `axis` - dodaje dodatkowe osie
- `legend` - dodaje legendę
- `points` - rysuje punkty
- `lines` - rysuje linie
- `poly` - rysuje wielokąt
- `rect` - rysuje prostokąt
- `segments` - rysuje odcinki
- `text` - dodaje tekst
- `title` - dodaje tytuł

Parametry graficzne

Możliwości personalizacji wykresów w R są niemal nieograniczone. Parametry graficzne rysowania można zmienić za pomocą funkcji `par`. Kilka przykładowych widzimy niżej. Uwaga! Należy pamiętać, że bardzo dużo rzeczy, które możemy ustawić w funkcji `par` możemy także przekazać jako argument naszej funkcji rysującej.

```
par(adj = 1, # 0 - tekst wyrównany do lewej, 0.5 - wycentrowany, 1 - tekst wyrównany do prawej
    bg = 'skyblue', # kolor tła
    cex = 1.2, # powiększenie tekstu
    col = 'Green', # kolor wykresu i ramek
    family = 'serif', # rodzina czcionki
    lty=3, # typ linii
```

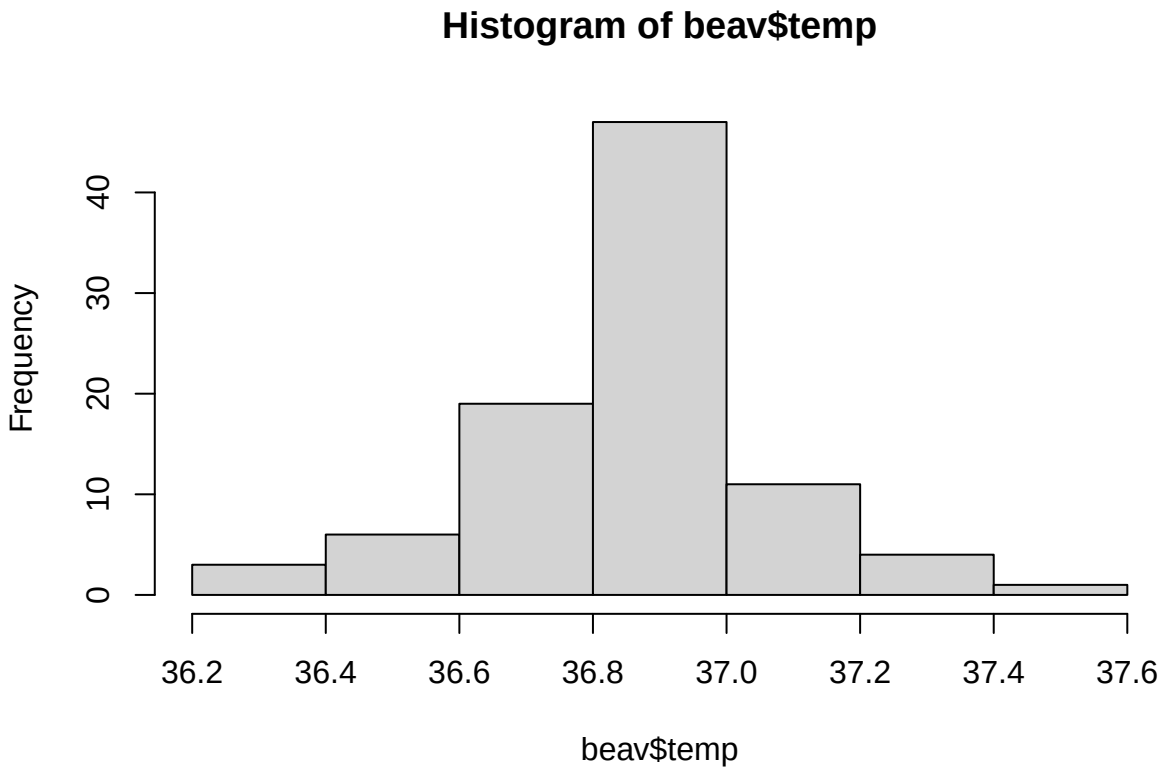
```
    lwd=1.3, # szerokość linii
    pch = 8) # symbol używany do oznaczania punktów

plot(beav$time, beav$temp, type='o', pch = 7) # symbol numer 8 to gwiazdki, ale my daliśmy 7 przy wywoł
```



`hist` - histogram

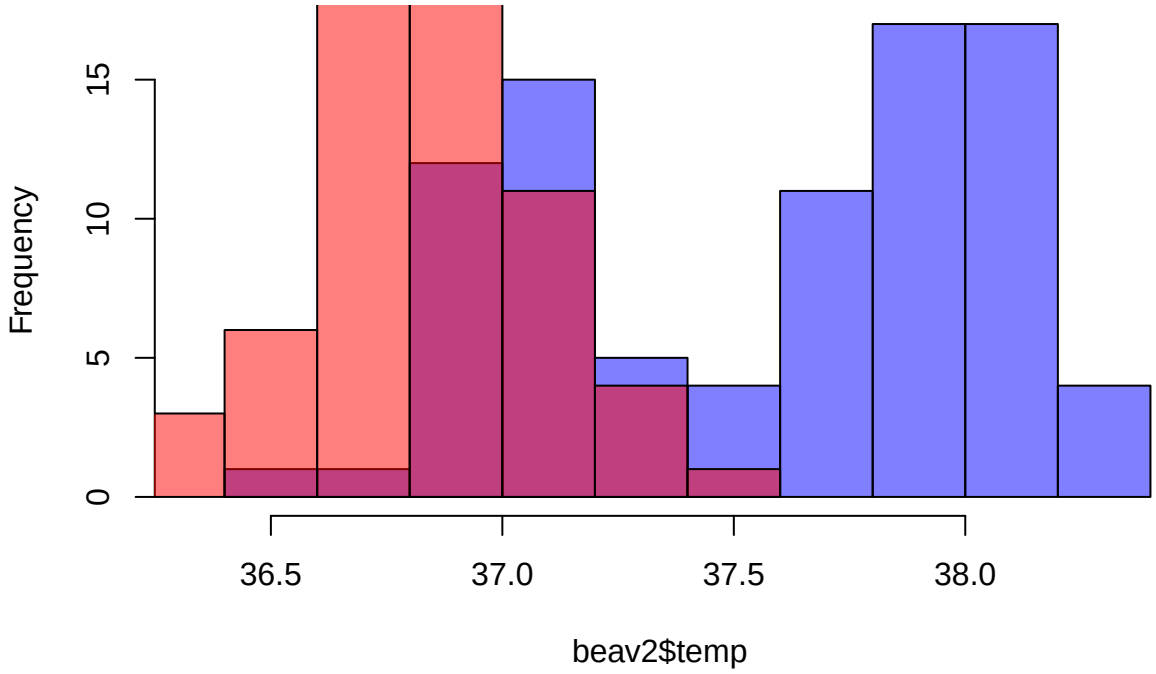
Histogram to typ wykresu, który bardzo często jest wykorzystywany do obrazowania rozkładu danych. Możemy go stworzyć za pomocą funkcji wysokiego poziomu `hist`.

```
hist(beav$temp)
```



Spróbujmy wykonać trick polegający na naniesieniu dwóch histogramów na jeden wykres. Proszę uważnie prześledzić kod z komentarzami.

```
# Tutaj stworzymy swoje "przezroczyste" kolory za pomocą funkcji adjustcolor.
# Bierzemy domyślne kolory R i ustawiamy kanał alfa (przezroczystosć) na 0.5
# Dzięki czemu stają się półprzezroczyste

t_red = adjustcolor('Red', alpha.f=0.5)
t_blue = adjustcolor('Blue', alpha.f=0.5)

# Tak jak przy wykresach punktowych musieliśmy ustawić skalę na osi Y,
# tak tutaj musimy zrobić to na osi X

hist(beav2$temp,
     col = t_blue,
     xlim = c(min(c(beav$temp, beav2$temp)),
              max(c(beav$temp, beav2$temp))))

# Przekazując argumentowi `add` wartość TRUE mówimy Rowi,
# żeby drugi histogram narysował na pierwszym.

hist(beav$temp,
     col = t_red,
     add = TRUE)
```

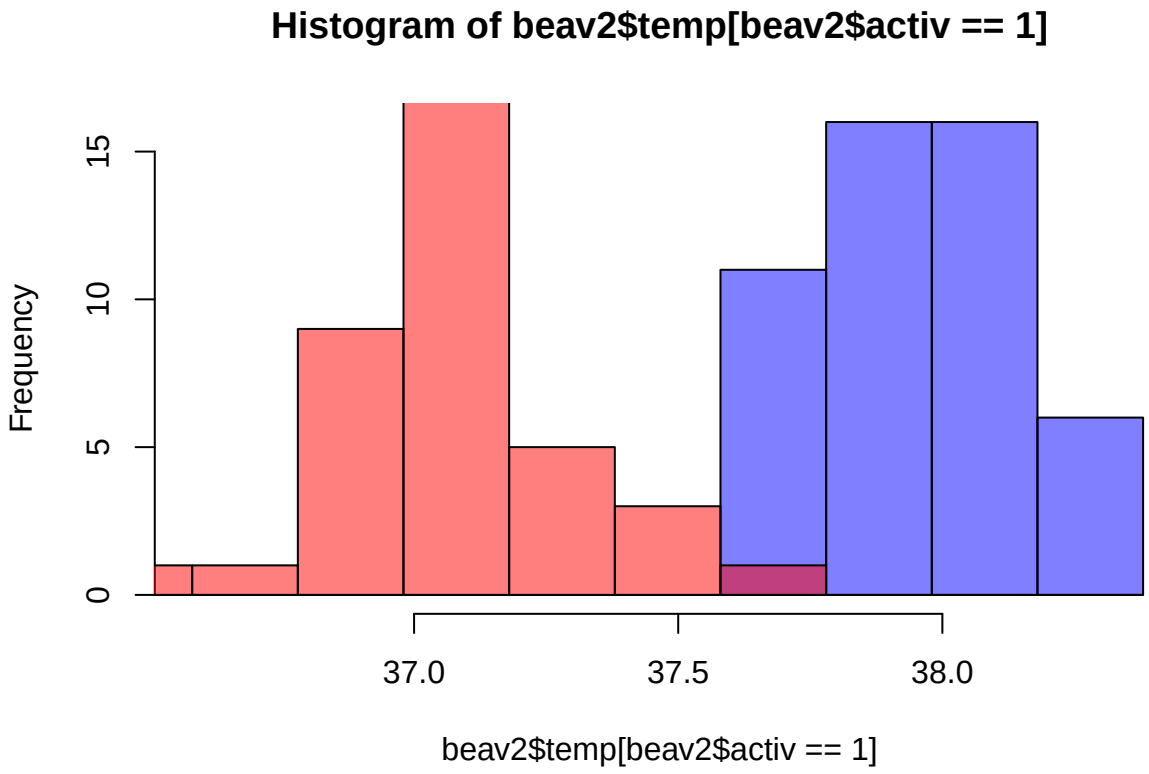


W kolejnym przykładzie robimy w zasadzie to samo, ale musimy sobie dodatkowo poradzić z jeszcze jednym problemem, polegajacym na doborze szerokości "koszyków" histogramu. W tym celu tworzymy zmienną `bins`, w której ustawiamy takie wartości koszyków, żeby oba wykresy mieściły się. Przekazujemy ją w argumencie `breaks`.

```
bins <- seq(min(beav2$temp)-0.2, # początek pierwszego koszyka
            max(beav2$temp)+0.2, # koniec ostatniego koszyka
            by=0.2 # szerokość koszyka
            )
bins
```

```
##  [1] 36.38 36.58 36.78 36.98 37.18 37.38 37.58 37.78 37.98 38.18 38.38
```

```
hist(beav2$temp[beav2$activ == 1], # tylko okresy aktywne
     col = t_blue,
     xlim = c(min(beav2$temp), max(beav2$temp)),
     breaks = bins)

hist(beav2$temp[beav2$activ == 0], # tylko okresy nieaktywne
     col = t_red,
     breaks = bins,
     add = TRUE
     )
```



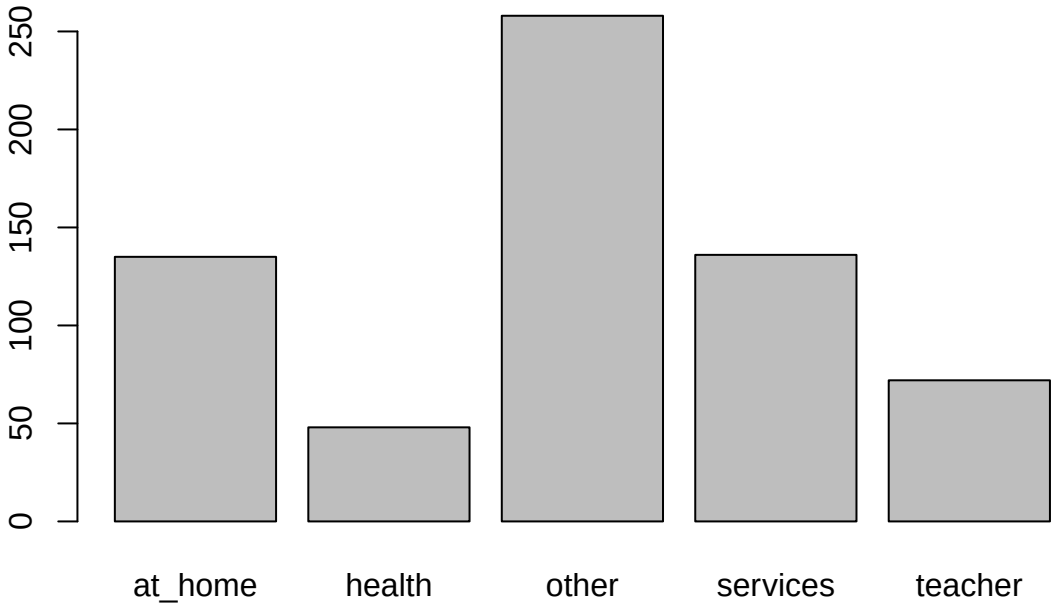
barplot - wykres słupkowy

Wykres słupkowy to chyba najczęściej wykorzystywany w praktyce typ wykresu. Nie przekazuje za dużo informacji, ale często może się przydać. Często można go spotkać w mediach, na plakatach, na infografikach itp. Jest zdecydowanie lepszy, niż wykres kołowy, którego w zasadzie nigdy nie powinniśmy używać.

```
df <- read.csv('student-por.csv', sep=";", header = TRUE)
table(df$Mjob) # tabela z której tworzymy wykres
```

```
##
##  at_home   health    other services  teacher
##      135       48      258      136       72
```

```
# wykres, jako argument przyjmuje twór "macierzopodobny" - w tym wypadku tabelę
barplot(table(df$Mjob))
```

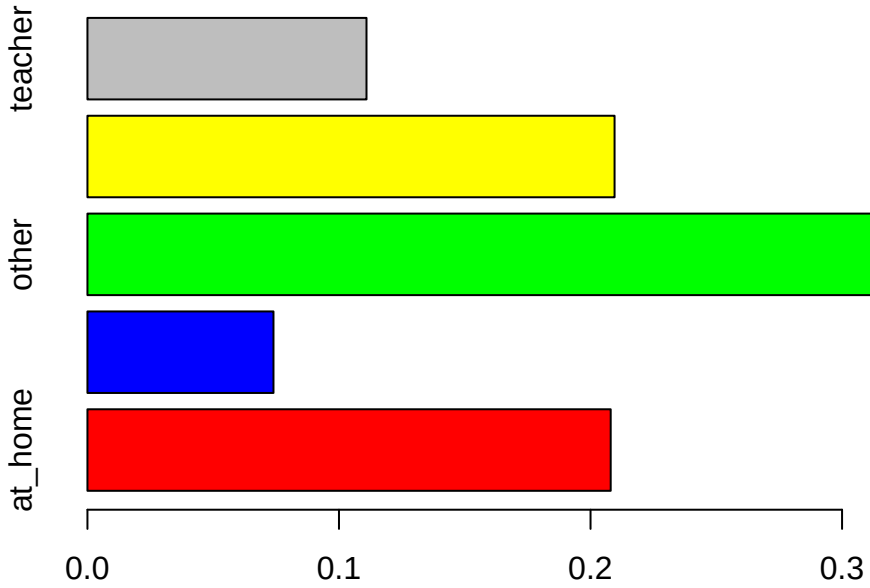


Wykres możemy upiększać ustawiając mu kolory oraz przerzucając go na drugi bok.

```
df <- read.csv('student-por.csv', sep=";", header = TRUE)

# za pomocą prop.table możemy stworzyć wykres procentowy
prop.table(table(df$Mjob))
```

```
##
##    at_home     health      other   services    teacher
## 0.20801233 0.07395994 0.39753467 0.20955316 0.11093991
```

```
barplot(prop.table(table(df$Mjob)),
        col = c('Red', 'Blue', 'Green', 'Yellow', 'Grey'), # kolory
        horiz = TRUE) # orientacja pozioma
```



Jak działają wykresy procentowe z nakładającymi się kolumnami?

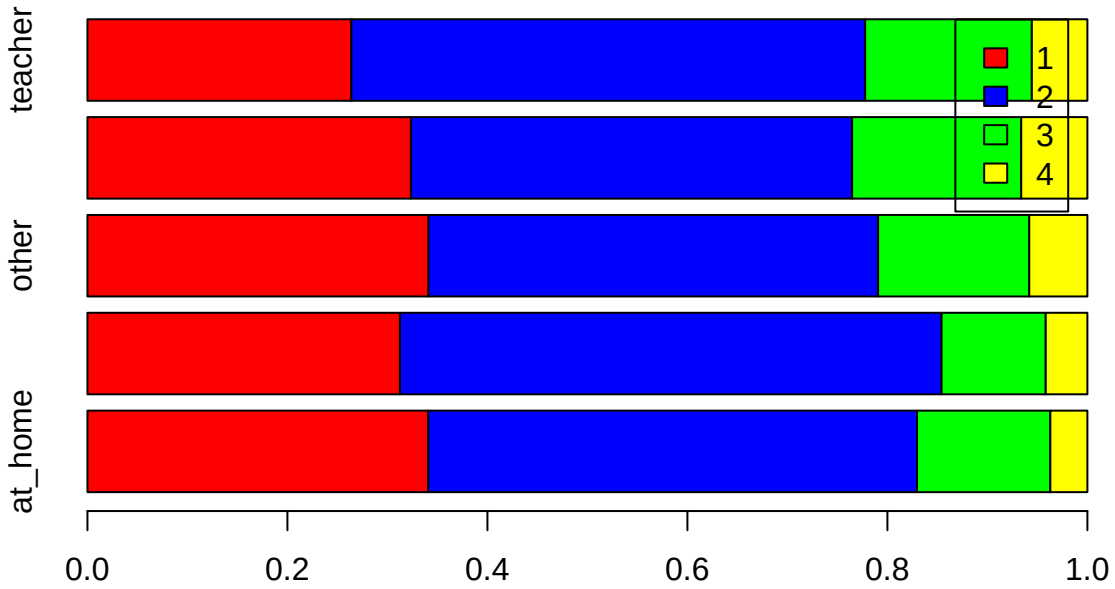
1. Musimy mieć tabelę procentową. Uzyskujemy taką za pomocą funkcji `prop.table`. Musimy pamiętać,

by przekazać odpowiedni wymiar (`margin`)
2. Funkcja barplot kiedy przekażemy jej argument w postaci dwuwymiarowej tabeli, traktowac będzie kolumny jako kolumny na wykresie, a poszczególne wiersze jako kolejne segmenty.
3. Jeżeli ustawili Państwo dobrze wymiary, to wykresy powinny mieć tę samą wysokość.

```
# za pomocą prop.table możemy stworzyć wykres procentowy
prop.table(table(df$studytime, df$Mjob),
           2 # proporcje sumują się do 1 w kolumnach
           )
```

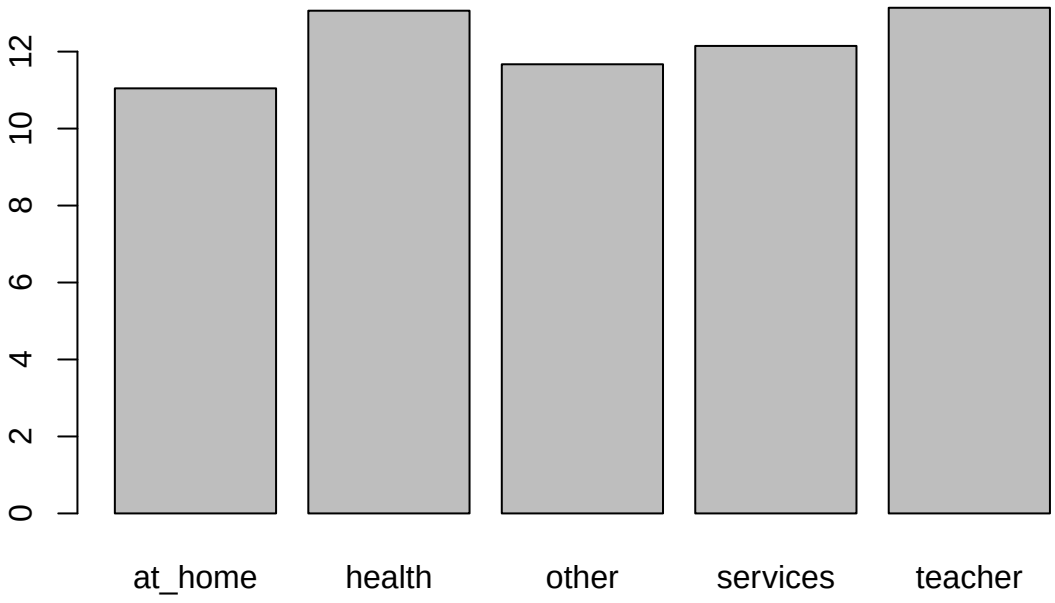
```
##
##        at_home     health      other   services    teacher
##   1 0.34074074 0.31250000 0.34108527 0.32352941 0.26388889
##   2 0.48888889 0.54166667 0.44961240 0.44117647 0.51388889
##   3 0.13333333 0.10416667 0.15116279 0.16911765 0.16666667
##   4 0.03703704 0.04166667 0.05813953 0.06617647 0.05555556
```

```
barplot(prop.table(table(df$studytime, df$Mjob), 2),
        col = c('Red', 'Blue', 'Green', 'Yellow'), # kolory
        horiz = TRUE, # orientacja pozioma
        legend = TRUE) # legenda
```



Tutaj szybki sposób na naniesienie na wykres jakiegoś parametru naszych danych za pomocą funkcji `tapply` i `barplot`.

```
barplot(tapply(df$G3, df$Mjob, mean))
```



`boxplot` - wykres pudełkowy

Proszę nauczyć się odczytywać dane z wykresów pudełkowych.

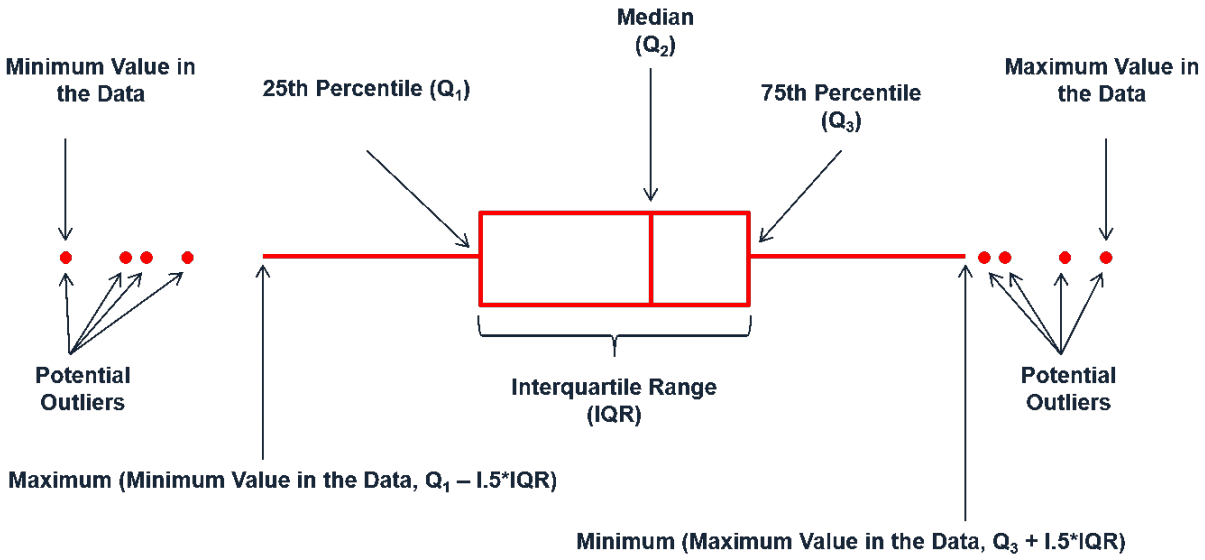
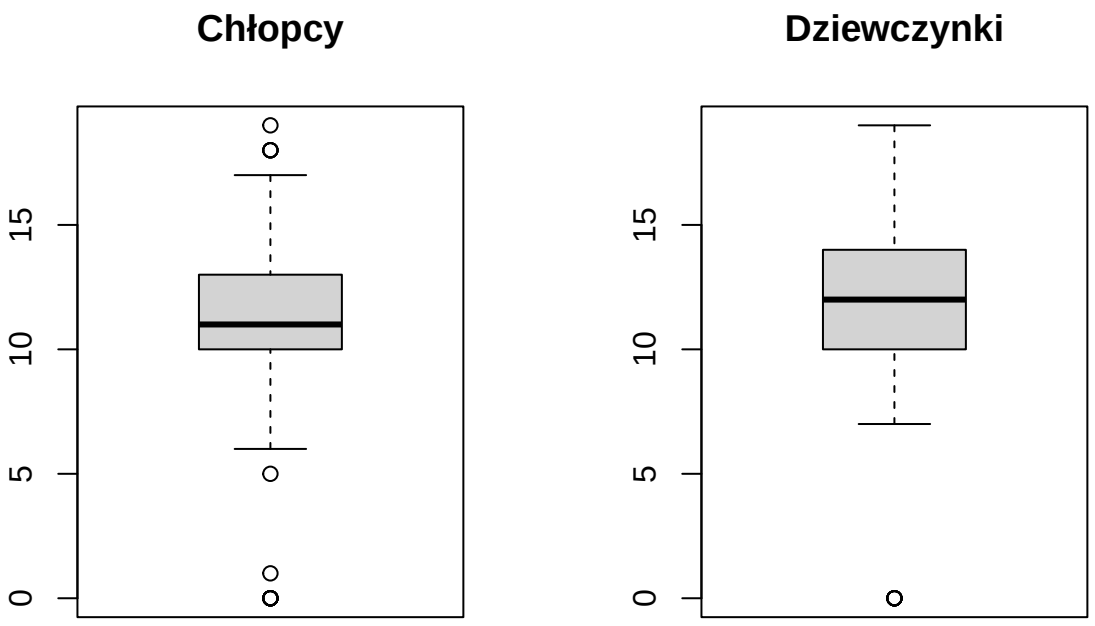


Figure 2: Boxplot

Oraz przeczytać dowolny artykuł gdziekolwiek na ten temat. Moze być ten:

http://onlinestatbook.com/2/graphing_distributions/boxplots.html

```
par(mfrow = c(1,2))
boxplot(split(df, df$sex)[[2]]$G3, main = 'Chłopcy')
boxplot(split(df, df$sex)[[1]]$G3, main = 'Dziewczynki')
```



`vioplot` - wykres skrzypcowy

Wykres skrzypcowy to bardziej nowoczesny odpowiednik wykresu pudełkowego. Można o nim myśleć jako o połączeniu histogramu (lub wykresu ilustrującego estymowaną z rozkładu empirycznego gęstość prawdopodobieństwa) oraz wykresu pudełkowego. Niektórym jego wygląd się kojarzy:

Poniżej znajduje się wyjaśnienie, co oznaczają poszczególne elementy wykresu skrzypcowego:

Więcej o wykresie skrzypcowym można przeczytać na przykład na Wikipedii: https://en.wikipedia.org/wiki/Violin_plot

```
# musimy zainstalować pakiet `vioplot` tworzący wykres skrzypcowy
# install.packages("vioplot")

library(vioplot)
```

```
## Ładowanie wymaganego pakietu: sm
```

```
## Package 'sm', version 2.2-5.7: type help(sm) for summary information
```

```
## Ładowanie wymaganego pakietu: zoo
```

```
##
## Dołączanie pakietu: 'zoo'
```

```
## Następujące obiekty zostały zakryte z 'package:base':
##
##     as.Date, as.Date.numeric
```

```
par(mfrow = c(1,2))
vioplot(split(df, df$sex)[[2]]$G3)
vioplot(split(df, df$sex)[[1]]$G3)
```

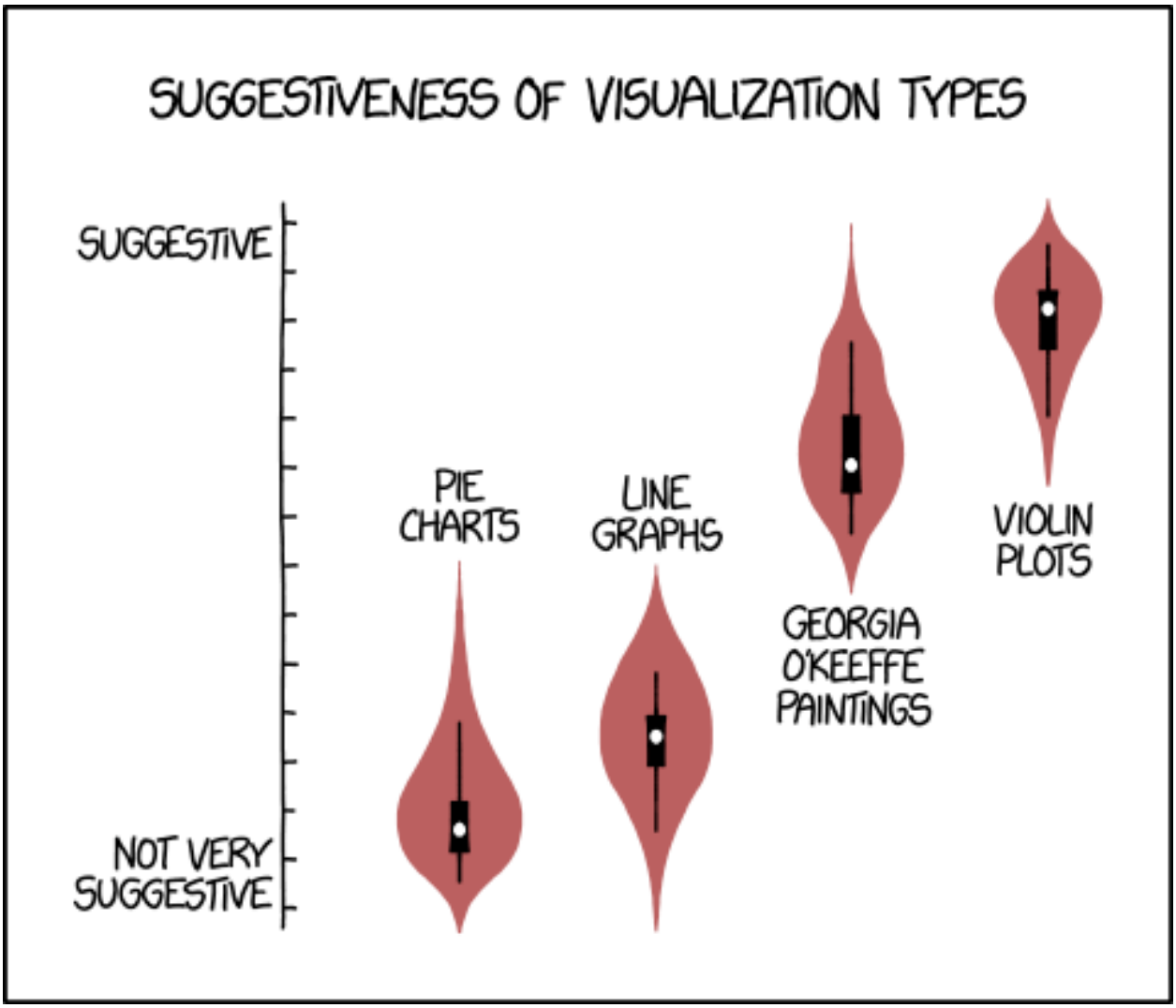


Figure 3: Obrazek z XKCD, https://xkcd.com/1967/

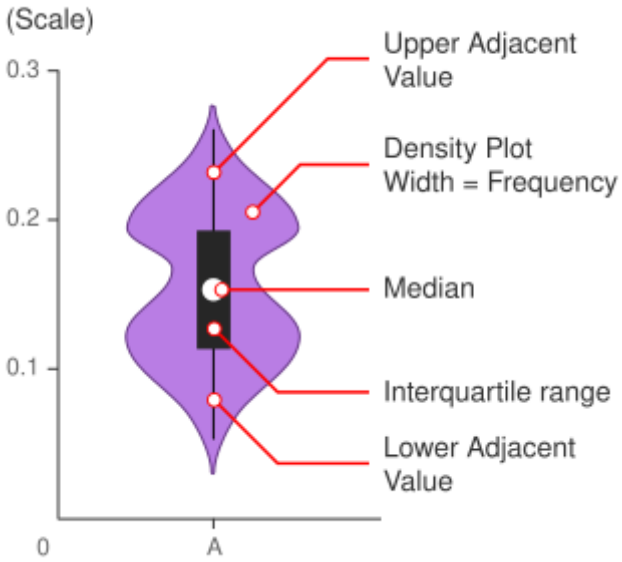


Figure 4: Wykres skrzypcowy