Krzysztof Trajkowski

Analiza i wizualizacja danych naukowych

27 marca 2013

Spis treści

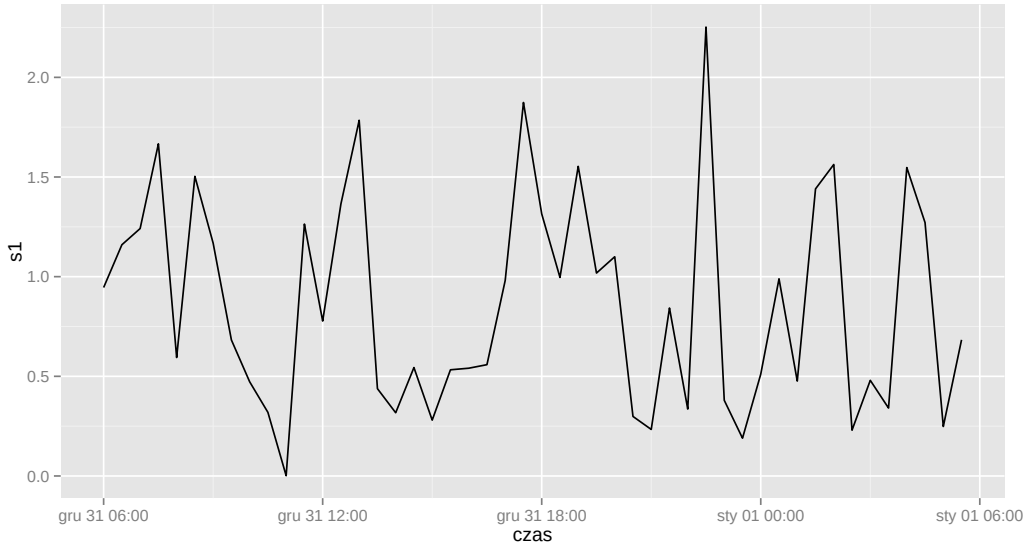
1. Wykresy liniowe

Mamy do dyspozycji pewne dane:

```
# dane - zmienna s1:
set.seed(3691)
s1= abs(rnorm(48))
```

Wiemy, że pomiary były wykonywane od godziny 06:00, 31 grudnia 2012 do godziny 05:30, 01 stycznia 2013 roku co 30 minut:

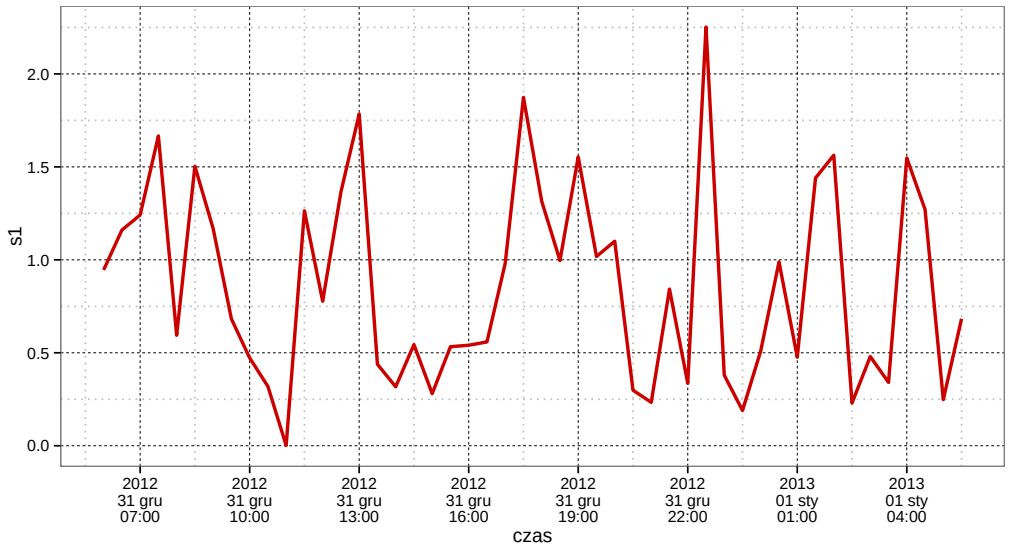
```
# dane - zmienna s1 i zmienna czas:
s1= data.frame(czas= seq(
as.POSIXct("2012-12-31␣06:00:00"),
as.POSIXct("2013-01-01␣05:30:00"),
by = "30␣min"), s1)
library(ggplot2)
# wykres:
ggplot(s1, aes(x= czas, y= s1)) + geom_line()
```



Rys. 1.1. Wykres liniowy.

Ustawienia domyślne można modyfikować w zależności od własnych potrzeb. Rysunek 1.2 to przykład modyfikacji kilku ustawień. Została dodana funkcja `theme_bw()` zatem tło wykresu zostało zamienione na białe, zmieniono także kolory i linie siatek, na osi x została zwiększona częstość podziałki (co 3 godziny) oraz zmodyfikowano zapis daty i czasu na osi x.

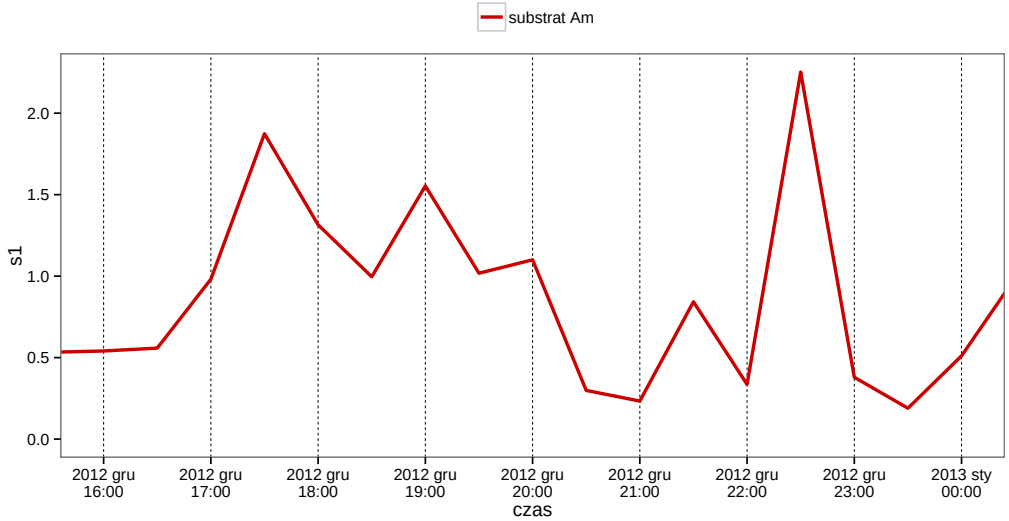
```
library(scales)
# wykres:
ggplot(s1, aes(x= czas, y= s1))+
geom_line(colour= "red3",size= 1)+ theme_bw()+
scale_x_datetime(breaks= date_breaks("3␣hours"),
labels= date_format("%Y\n%d␣%b\n%H:%M"))+
theme(
panel.grid.minor= element_line(colour= "gray", linetype = "dotted"),
panel.grid.major= element_line(colour= "black", linetype= "dashed"))
```



Rys. 1.2. Wykres liniowy.

Możemy także wyodrębnić interesujący nas okres oraz dodać legendę.

```
# wykres:
ggplot(s1)+ geom_line(aes(x=czas,y=s1,colour="substrat Am"),size=1)+
scale_colour_manual(values="red3")+labs(colour= "")+ theme_bw()+
scale_x_datetime(limits=c(
as.POSIXct("2012-12-31 16:00:00"),
as.POSIXct("2013-01-01 00:00:00")),
breaks=date_breaks("1 hours"),labels= date_format("%Y %b\n%H:%M"))+
theme(
legend.position= "top",
panel.grid.major.x=element_line(colour="black", linetype= "dashed"),
panel.grid.major.y=element_blank(),
panel.grid.minor=element_blank())
```



Rys. 1.3. Wykres liniowy.

W zależności od tego jaki chcemy uzyskać format daty/czasu na osi x możemy używać następujących kodów:

Tabela 1.1. Formatowanie daty i czasu

kod	oznaczenie	wynik
%S	sekunda	00-59
%M	minuta	00-59
%l	godzina	1-12
%I	godzina	01-12
%H	godzina	00-23
%a	dzień tygodnia - skrócona nazwa	pon-nie
%A	dzień tygodnia - cała nazwa	poniedziałek-niedziela
%e	dzień miesiąca	1-31
%d	dzień miesiąca	01-31
%m	miesiąc - numer	01-12
%b	miesiąc - skrócona nazwa	sty-gru
%B	miesiąc - cała nazwa	styczeń-grudzień
%y	rok - skrócony zapis	00-99
%Y	rok - pełny zapis	0000-9999

Środowisko R oraz pakiet `ggplot2` może być dobrym narzędziem do wykonywania wykresów funkcji matematycznych. Poniżej na rysunku (1.4) są trzy wykresy:

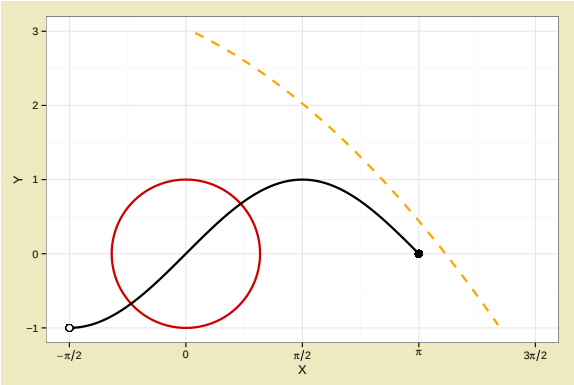
$$f(x) = \sin(x) \quad dla \quad x \in \left\langle -\frac{2}{\pi}, \pi \right\rangle \tag{1.1}$$

$$f(x) = -0,1157(x+2)^2 + 3,5 \tag{1.2}$$

oraz

$$x^2 + y^2 = 1 \tag{1.3}$$

```
library(shape)
X=getellipse(1,1,mid=c(0,0))[,1]
Y=getellipse(1,1,mid=c(0,0))[,2]
fun1= function(x) ifelse(x >= -pi/2 & x < pi, sin(x), NA)
fun2= function(x) -0.1157*(x+2)^2+3.5
ggplot(data.frame(X,Y),aes(X,Y))+
theme_bw()+
ylim(-1,3)+coord_fixed(ratio= 1)+
geom_path(col="red3",size=1)+
stat_function(fun= fun1, size=1)+
stat_function(fun= fun2, size=1,lty=2,col="orange")+
geom_point(aes(-pi/2,-1),factor=1,size=3,pch=21,col="black",
bg="white")+
geom_point(aes(pi,0),factor=1,size=3,pch=21,col="black",bg="black")+
scale_x_continuous(
limits=c(-pi/2,3*pi/2),
breaks=c(-pi/2, 0, pi/2, pi, 3*pi/2),
labels=c(expression(-pi/2),expression(0),expression(pi/2),
expression(pi),expression(3*pi/2)))+
opts(plot.background= theme_rect(fill= "lemonchiffon2", colour= NA))
```



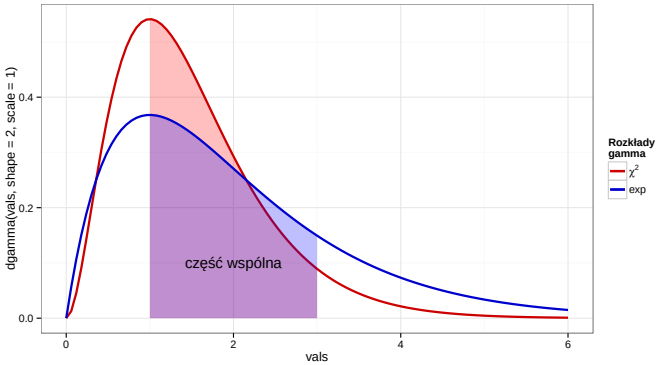
Rys. 1.4. Okrąg, sinusoida i funkcja kwadratowa.

Poniżej zaprezentujemy dwa wykresy funkcji prawdopodobieństwa (rozkład wykładniczy: $\mathcal{E}(\lambda = 2)$ oraz rozkład chi-kwadrat: $\chi^2(df = 6)$) z wykorzystaniem rozkładu gamma:

$$\Gamma(\lambda, k) = \frac{\lambda^k}{\Gamma(k)} x^{k-1} \exp(-x\lambda) \tag{1.4}$$

gdzie: $\lambda > 0$ – parametr skali, $k > 0$ – parametr kształtu.

```
vals= seq(1,3,.01)
ggplot(mapping= aes(x=vals, y=dgamma(vals,shape=2,scale=1)))+
xlim(0,6)+theme_bw()+
geom_area(mapping= (aes(x=vals,y=dgamma(vals,shape=6/2,scale=1/2))),
fill="red", alpha=1/4)+
stat_function(fun= dgamma, arg= list(shape = 6/2,scale= 1/2),
aes(colour= "a"), lwd=1)+ # a - funkcja przypisana do koloru: red3
geom_area(mapping= (aes(x=vals, y=dgamma(vals,shape=2,scale=1))),
fill="blue", alpha=1/4)+
stat_function(fun= dgamma, arg= list(shape = 2,scale= 1),
aes(colour= "b"), lwd=1)+ # b - funkcja przypisana do koloru: blue
scale_colour_manual(values= c("red3","blue"),
labels= c(bquote(chi^2),"exp"))+
labs(colour= "Rozkłady\ngamma")+
geom_text(aes(x= 2, y= 0.1), label= "część␣wspólna")
```



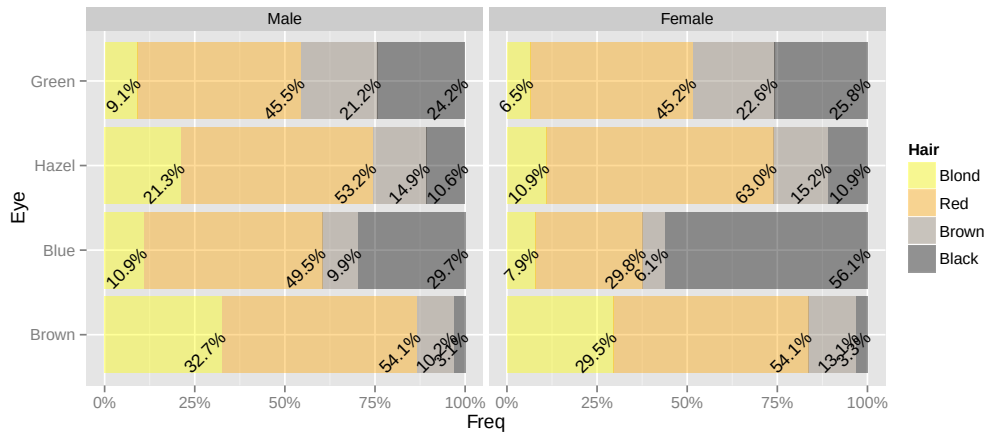
Rys. 1.5. Rozkład wykładniczy i chi-kwadrat.

2. Wykresy paskowe i histogramy

W tym rozdziale wykorzystamy dane `HairEyeColor`. W pierwszej kolejności przygotujemy dane tzn. przekształcimy tabelę kontyngencji `HairEyeColor` w ramkę danych `df`:

```
df= as.data.frame(HairEyeColor)
```

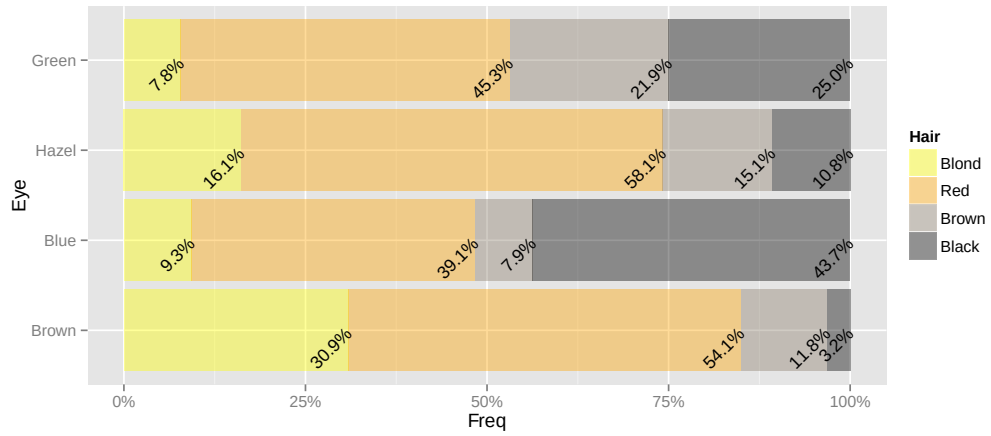
```
# wykres:
p=ggplot(df, aes(x= Eye, y=Freq,fill=Hair ))+
geom_bar(position= "fill",alpha=0.4)+facet_wrap(~ Sex)+
coord_flip()+
scale_fill_manual(
values = c("yellow","orange","bisque4","black"),
labels=c("Blond","Red","Brown","Black"))+
scale_y_continuous(labels = percent)
# parametry wykresu:
D=ggplot_build(p)$data[[1]]
# wykres + wartości procentowe pasków:
p+geom_text(aes(D$x,D$y-D$ymin,
label= percent(D$y-D$ymin),factor=1), size= 4,
hjust= 1, vjust= 0.5, position= "stack",angle=45)
```



Rys. 2.1. Frakcje kolorów włosów (Hair) z podziałem na kolor oczu i płeć.

```
ddf= aggregate(Freq ~ Hair+Eye, data=df, FUN=sum)
```

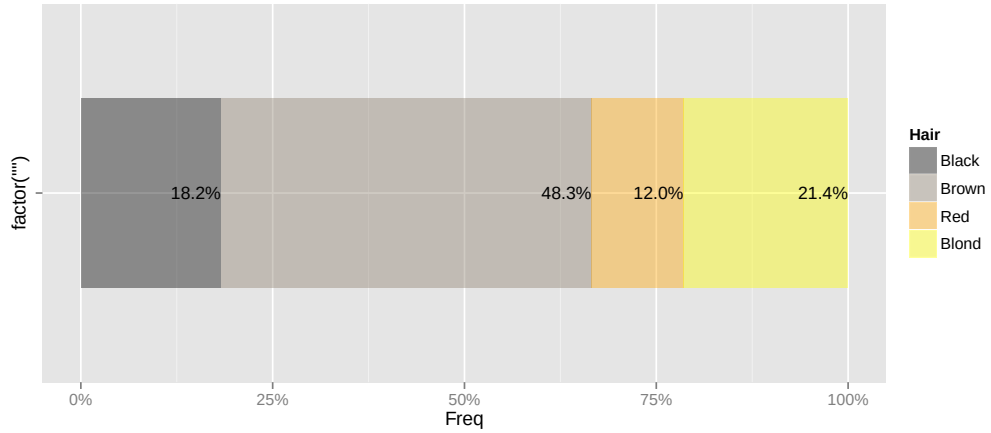
```
# wykres:
p=ggplot(ddf, aes(x= Eye, y=Freq,fill=Hair ))+
geom_bar(position= "fill",alpha=0.4)+
coord_flip()+
scale_fill_manual(
values = c("yellow","orange","bisque4","black"),
labels=c("Blond","Red","Brown","Black"))+
scale_y_continuous(labels = percent)
# parametry wykresu:
D=ggplot_build(p)$data[[1]]
# wykres + wartości procentowe pasków:
p+geom_text(aes(D$x,D$y-D$ymin,label= percent(D$y-D$ymin),factor=1),↩
    size= 4, hjust= 1, vjust= 0.5, position= "stack",angle=45)
```



Rys. 2.2. Frakcje kolorów włosów (Hair) z podziałem na kolor oczu.

```
dddf= aggregate(Freq ~ Hair, data=df, FUN=sum)
```

```
# wykres:
p=ggplot(dddf, aes(x= factor(""),y=Freq, fill= Hair) )+
geom_bar(position= "fill",alpha=0.4,width=0.5)+
coord_flip()+
scale_fill_manual(
values = c("black","bisque4","orange","yellow"),
labels=c("Black","Brown","Red","Blond"))+
scale_y_continuous(labels = percent)
# parametry wykresu:
D=ggplot_build(p)$data[[1]]
# wykres + wartości procentowe pasków:
p+geom_text(aes(D$x,D$y-D$ymin,
label= percent(round(D$y-D$ymin,4)),factor=1), size= 4,
hjust= 1, vjust= 0.5, position= "stack")
```



Rys. 2.3. Frakcje kolorów włosów (Hair).

Należy zaznaczyć, że w funkcji `geom_bar(ggplot2)` (rys. 2.1–2.3) wykorzystaliśmy opcję `position="fill"`. Do graficznej prezentacji liczebności można także użyć opcji `"dodge"` lub `"stack"`.

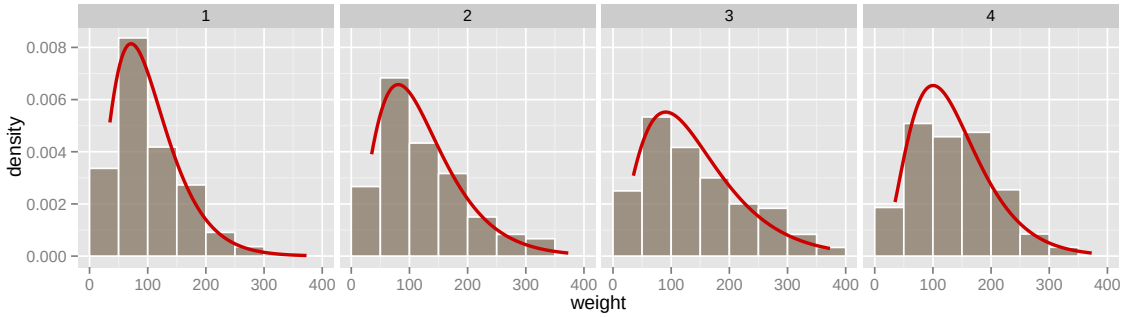
W tym rozdziale przedstawimy także wykresy zwane histogramami. Poniżej kilka przykładów z wykorzystaniem danych `ChickWeight` gdzie `weight` oznacza wagę kurczaka natomiast `Diet` to rodzaj diety. Oprócz wykonania histogramów wag kurczaków z podziałem na rodzaj diety, każdy histogram porównamy z rozkładem gamma (1.4). Parametry rozkładu gamma: $\Gamma(\lambda, k)$ zostały obliczone na podstawie wzorów:

$$\bar{x} = \frac{k}{\lambda} \tag{2.1}$$

$$\sigma^2 = \frac{k}{\lambda^2} \tag{2.2}$$

gdzie: $\bar{x}$ to średnia, natomiast σ^2 to wariancja.

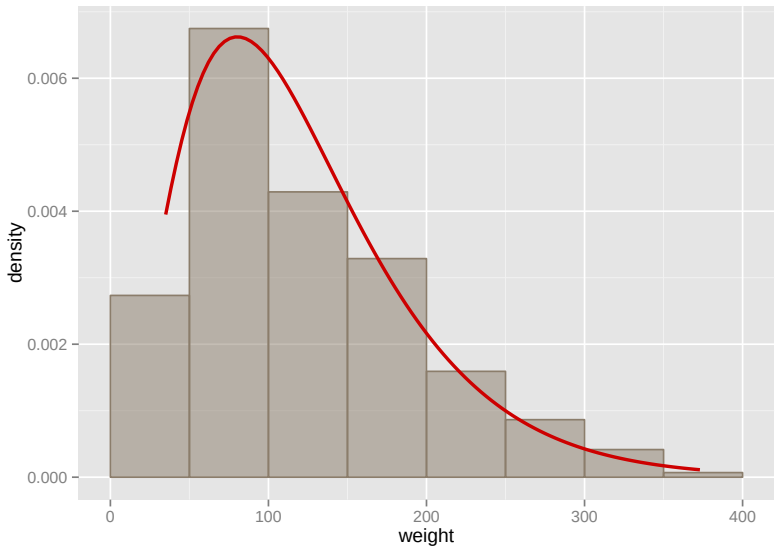
```
library(plyr)
grid= with(ChickWeight, seq(min(weight), max(weight), length= 100))
GAMMA= ddply(ChickWeight, "Diet", function(df) {
 data.frame(
 predicted= grid,
 density= dgamma(grid,
 shape=mean(df$weight)^2/var(df$weight),
 rate=mean(df$weight)/var(df$weight)))
})
# wykres:
ggplot(data = ChickWeight,aes(x = weight))+
facet_wrap(~Diet, nrow= 1, ncol= 4)+
geom_histogram(aes(y = ..density..),colour="white",fill="bisque4",
alpha=0.8,binwidth=20,breaks=seq(0,400,by=50)) +
geom_line(data= GAMMA, aes(x= predicted, y= density),
colour= "red3",size= 1)
```



Rys. 2.4. Rozkład wag kurczaków z podziałem na rodzaj diety.

Poniżej histogram bez podziału na stosowaną dietę. Zamiast wartości domyślnych dla parametrów `binwidth` i `breaks` wykorzystaliśmy wartości z funkcji `density(stats)` oraz `hist(graphics)`.

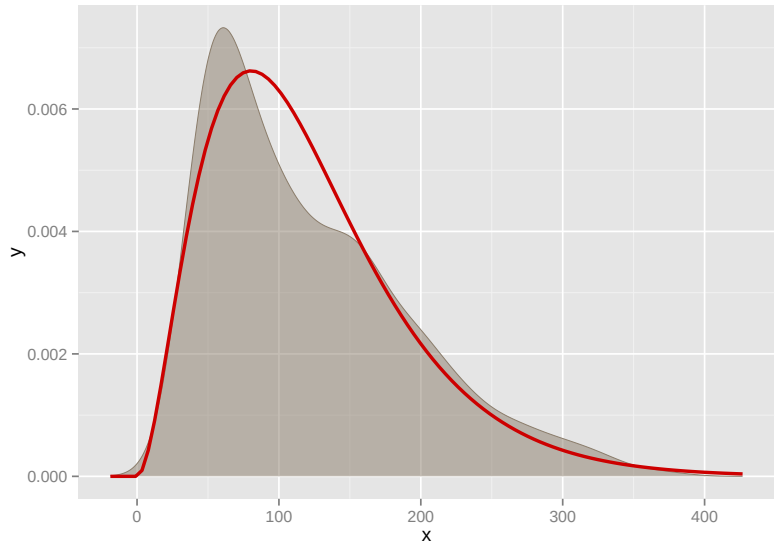
```
BW= density(ChickWeight$weight)$bw
BR= hist(ChickWeight$weight,plot=F)$breaks
# wykres:
ggplot(ChickWeight, aes(x = weight))+
geom_histogram(aes(y=..density..),colour="bisque4", fill="bisque4",
alpha=0.5, binwidth =BW, breaks= BR)+
stat_function(fun= dgamma, colour= "red3", size=1,
arg= list( shape=mean(weight)^2/var(weight),
rate=mean(weight)/var(weight)))
```



Rys. 2.5. Rozkład wag kurczaków.

Funkcja gęstości.

```
d2= with(density(weight),data.frame(x,y))
# wykres:
ggplot(data= d2, mapping= aes(x= x, y= y))+
geom_line(data= d2, mapping= aes(x= x, y= y),
colour="bisque4",lwd=0.2)+
geom_area(data=d2,mapping = aes(x= x, y= y),
fill="bisque4", alpha=0.5)+
stat_function(fun= dgamma, colour= "red3", lwd=1,
arg= list(shape=mean(weight)^2/var(weight),
rate=mean(weight)/var(weight)))
```



Rys. 2.6. Rozkład wag kurczaków.

3. Estymacja i wygładzanie

W tym rozdziale przedstawimy dopasowanie krzywych do danych doświadczalnych (tzw. fitowanie) z wykorzystaniem nieliniowej metody najmniejszych kwadratów: algorytm Gaussa-Newtona – `nls(stats)` i algorytm Levenberga-Marquardta – `nlsLM(minpack.lm)`. Na końcu zaprezentujemy także kilka metod wygładzania danych.

Na podstawie danych pomiarowych zamieszczonych poniżej:

```
# dane:
xs=c(0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0)
ys=c(
0.800,0.810,0.800,0.820,0.815,0.820,0.835,0.870,0.890,0.960,0.990)
ye=c(
0.013,0.017,0.014,0.014,0.014,0.011,0.015,0.014,0.013,0.017,0.013)
ds=data.frame(xs,ys,ye)
```

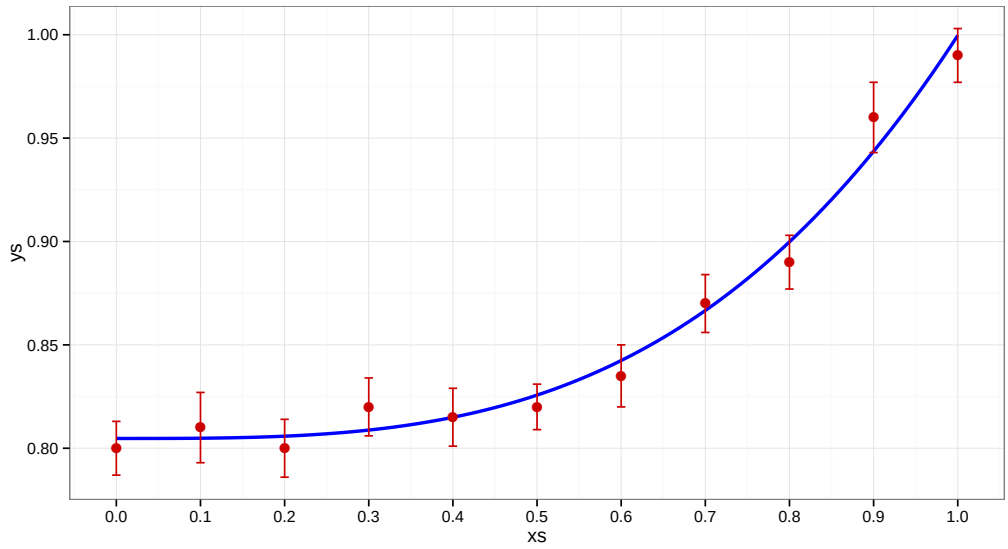
oszacujemy parametry modelu o postaci:

$$y = ax^b + c \tag{3.1}$$

z wykorzystaniem funkcji `nls(stats)`. W modelu uwzględnimy także błąd pomiaru.

```
# algorytm Gaussa-Newtona:
fit1=nls(ys=a*xs^b+c,
start=list(a=0.5,b=2,c=0.5), weights=ye, data=ds)
summary(fit1)

Formula: ys ~ a * xs^b + c

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 0.194929   0.009651  20.198 3.77e-08 ***
b 3.216612   0.366544   8.776 2.23e-05 ***
c 0.804732   0.004953 162.484 2.30e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.001191 on 8 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 2.454e-06
```

```
# oszacowana funkcja:
fun1=function(x) coef(fit1)[1]*x^coef(fit1)[2]+coef(fit1)[3]
# wykres:
ggplot(ds,aes(x=xs,y=ys))+
theme_bw()+
scale_x_continuous(breaks=seq(0,1,by=0.1))+
stat_function(fun=fun1,size=1,colour="blue")+
geom_errorbar(aes(x=xs,ymin=ys-ye,ymax=ys+ye),
width=0.01,size=0.5,colour="red3")+
geom_point(size=3,colour="red3")
```



Rys. 3.1. Fitowanie.

Tym razem wykorzystamy algorytm Levenberga-Marquardta do oszacowania modelu o postaci:

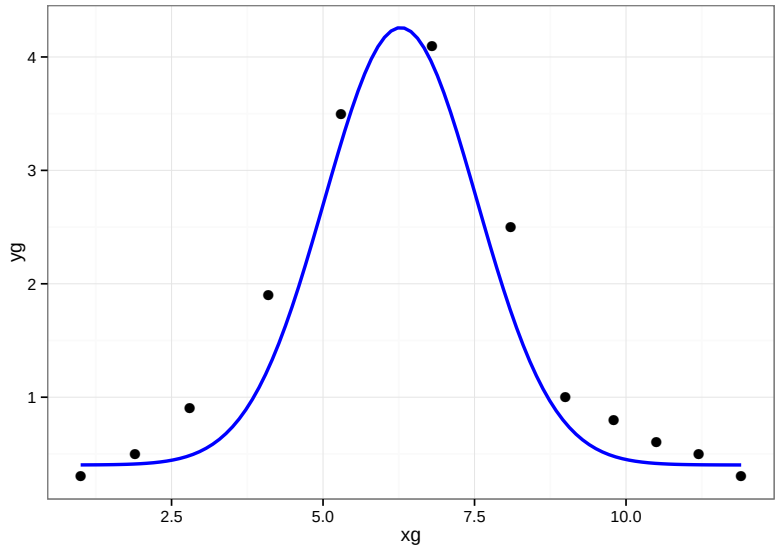
$$y = y_0 + A \cdot \exp\left(\frac{-(x - x_c)^2}{2 \cdot \omega^2}\right) \tag{3.2}$$

gdzie: y_0 – parametr przesunięcia, A – wysokość, ω – szerokość, x_c – środek wykresu.

```
# dane:
xg=c(1.0,1.9,2.8,4.1,5.3,6.8,8.1,9.0,9.8,10.5,11.2,11.9)
yg=c(0.3,0.5,0.9,1.9,3.5,4.1,2.5,1.0,0.8,0.6,0.5,0.3)
dg=data.frame(xg,yg)
```

```
# algorytm Levenberga-Marquardta:
library(minpack.lm)
fit2=nlsLM(yg~y0+A*exp((-1*(xg-xc)^2)/(2*w^2)),
start=list(y0=0.5, A=4, xc=6, w=2),data=dg)
summary(fit2)

Formula: yg ~ y0 + A * exp((-1 * (xg - xc)^2)/(2 * w^2))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
y0  0.40237    0.07127   5.646 0.000484 ***
A   3.85641    0.13352  28.882 2.23e-09 ***
xc  6.28066    0.05366 117.036 3.18e-14 ***
w   1.57771    0.06983  22.593 1.56e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1447 on 8 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.49e-08
```

```
# oszacowana funkcja:
fun2=function(x)
coef(fit2)[1]+coef(fit2)[2]*exp((-(x-coef(fit2)[3])^2)/(2*coef(fit2)↩
    [4]))
# wykres:
ggplot(dg,aes(x=xg,y=yg))+theme_bw()+geom_point(size=3)+
stat_function(fun=fun2,size=1,colour="blue")
```



Rys. 3.2. Fitowanie.

Dopasowywanie specyficznych krzywych jest często stosowane w takich naukach jak: biologia, medycyna, farmakologia czy toksykologia. Przykładem może być estymacja parametrów równania Michaelisa-Mentena:

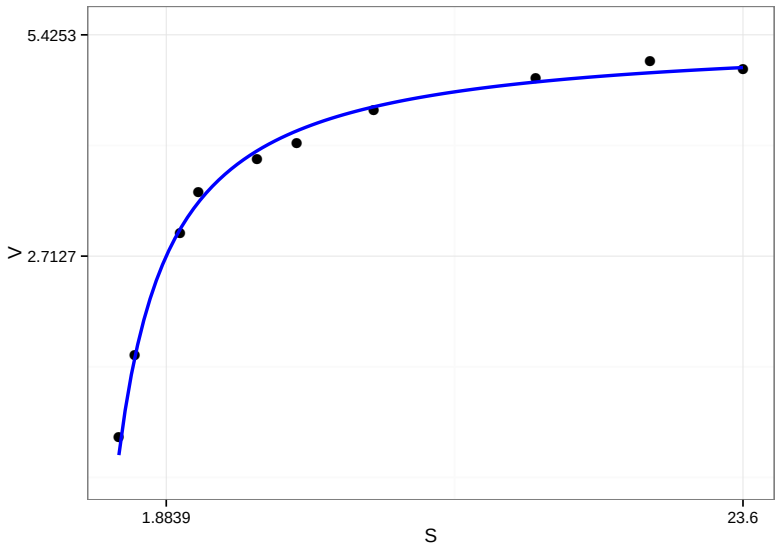
$$V = \frac{S \cdot V_{max}}{S + K_m} \tag{3.3}$$

```
# dane:
S=c(1.0,2.7,3.4,4.1,5.3,6.8,8.7,9.8,12.0,13.2)
V=c(2.1,2.5,3.8,3.5,4.9,6.1,7,7.9,8.5,8.6)
D=data.frame(S,V)
```

```
# algorytm Gaussa-Newtona:
fit3=nls(V~SSmicmen(S,Vmax,Km))
summary(fit3)

Formula: V ~ SSmicmen(S, Vmax, Km)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Vmax  5.42532    0.09705   55.90 1.16e-11 ***
Km    1.88394    0.14908   12.64 1.44e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1259 on 8 degrees of freedom
```

```
Number of iterations to convergence: 0
Achieved convergence tolerance: 9.707e-07
```

```
# oszacowana funkcja:
fun3=function(x) SSmicmen(x,coef(fit3)[1],coef(fit3)[2])
# wykres:
ggplot(D,aes(x=S,y=V))+theme_bw()+geom_point(size=3)+
stat_function(fun=fun3,size=1,colour="blue")+
scale_x_continuous(
breaks=c(coef(fit3)[2],23.6),
labels=c(round(coef(fit3)[2],4),23.6))+
scale_y_continuous(limits=c(0,5.5),
breaks=c(coef(fit3)[1]/2,coef(fit3)[1]),
labels=c(round(coef(fit3)[1]/2,4),round(coef(fit3)[1],4)))
```



Rys. 3.3. Zależność szybkości reakcji enzymatycznej (V) od stężenia substratu (S).

Także i w tym przypadku możemy wykorzystać algorytm Levenberga-Marquardta:

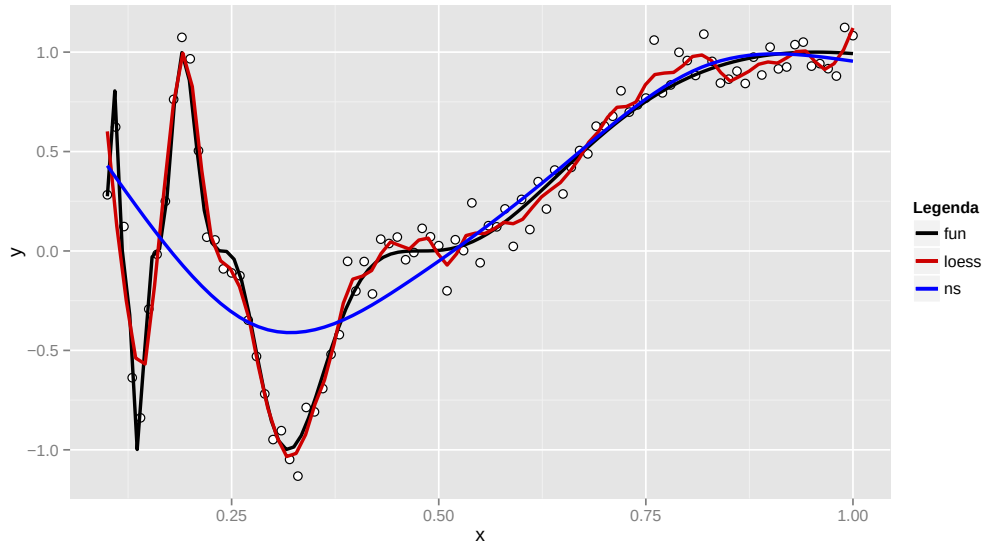
```
# algorytm Levenberga-Marquardta:
fit4=nlsLM(V~SSmicmen(S,Vmax,Km))
```

Dużo większa rodzina funkcji wzrostu lub sigmoidalnych jest dostępna w pakiecie `drc`. W artykułach: *Bioassay Analysis using R* oraz *Dose response curves and other non-linear curves in Weed Science and Ecotoxicology with the add-on package drc in R* można znaleźć wiele przykładów wykorzystania tego pakietu w praktyce.

W tej części opracowania przedstawimy kilka funkcji do wygładzania danych. W pierwszej kolejności wykorzystamy dwie funkcje: `loess(stats)` oraz `ns(splines)`.

```
library(splines)
set.seed(193)
x=seq(0.1,1,by=.01)
y=sin(1.5/x)^3+ rnorm(x, s=0.1)
f=function(x) sin(1.5/x)^3
d=data.frame(x,y)
# wykres:
```

```
ggplot(d,aes(x,y))+
geom_point(pch=21,col="black",bg="white",size=2.5)+
stat_function(fun=f,aes(colour="a"),size=1)+
stat_smooth(span=0.1,se=F,aes(colour="b"),size=1)+
stat_smooth(method= "lm",formula=y~ns(x,5),se=F,aes(colour="c"),
size=1)+
scale_colour_manual(name= "Legenda",
values= c("black","red3","blue"),
labels=c("fun","loess","ns"))
```



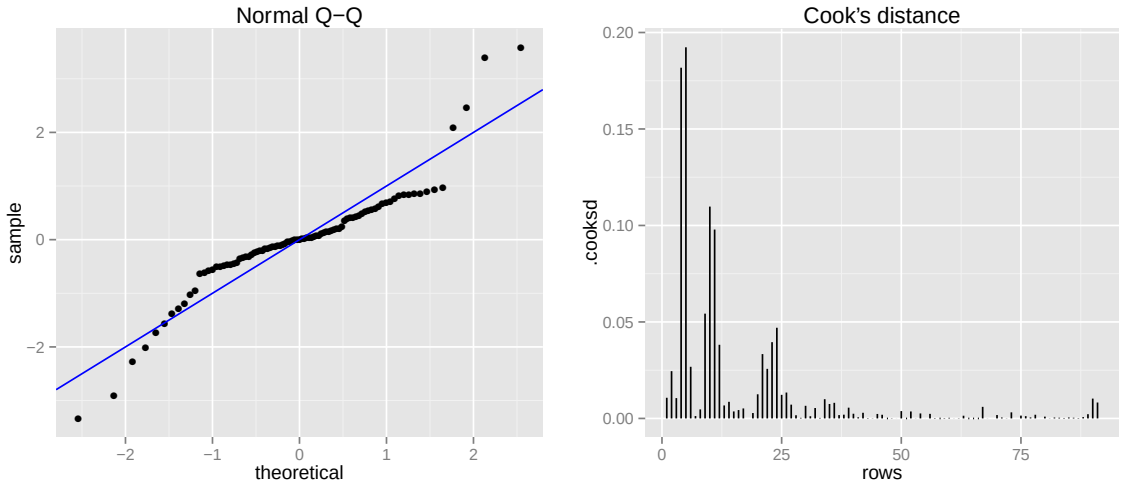
Rys. 3.4. Wygładzanie.

W pakiecie `ggplot2` możemy wykonać szereg wykresów diagnostycznych. Wszystkie niezbędne dane są dostępne w tabeli `fortify(ns)` do której możemy dopisywać kolejne zmienne.

```
# zmienne: .hat, .sigma, .cooksd, .fitted, .resid, .stdresid:
diag1= fortify(ns)
# dodanie nowej zmiennej rows:
diag2= cbind(diag1, rows=1:nrow(diag1))
```

Poniżej wykonamy dwa wykresy diagnostyczne dla modelu `ns`. Pierwszy z wykorzystaniem danych z funkcji `fortify(ggplot2)` natomiast drugi z wykorzystaniem dopisanej zmiennej `rows` z tabeli `diag2`.

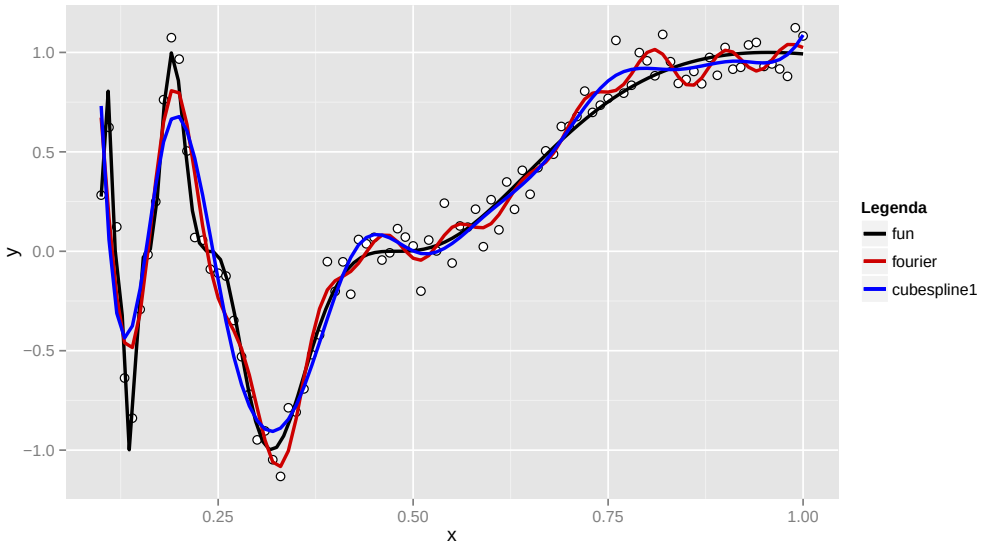
```
# wykresy:
plot1=
ggplot(ns, aes(sample= .stdresid)) +stat_qq()+
geom_abline(colour="blue")+labs(title= "QQ normal")
plot2=
ggplot(diag2, aes(rows, .cooksd, ymin=0, ymax=.cooksd))+
geom_linerange()+labs(title= "Cook's distance")
# rozmieszczenie wykresów:
library(grid)
pushViewport(viewport(layout= grid.layout(1, 2)))
print(plot1, vp= viewport(layout.pos.row= 1, layout.pos.col= 1))
print(plot2, vp= viewport(layout.pos.row= 1, layout.pos.col= 2))
```



Rys. 3.5. Wykresy diagnostyczne modelu ns.

Bardzo ciekawe rozwiązania są dostępne również w pakiecie `aratio`. Na rysunku (3.6) rezultat wygładzania danych metodą `fourier1` oraz `cubespline1`.

```
library(aratio)
fitf= fourier1(y~x,minq=1,maxq=10)
fitc= cubespline1(y~x,mink=1,maxk=10)
# wykres:
ggplot(d,aes(x,y))+
geom_point(pch=21,col="black",bg="white",size=2.5)+
stat_function(fun=f,aes(colour="a"),size=1)+
geom_line(aes(x,fitf$yhat,colour="b"),size=1)+
geom_line(aes(x,fitc$yhat,colour="c"),size=1)+
scale_colour_manual(name="Legenda",
values= c("black","red3","blue"),
labels=c("fun","fourier","cubespline1"))
```



Rys. 3.6. Wygładzanie.

Warto także wspomnieć o pakiecie `signal` który oferuje szeroki zestaw funkcji do przetwarzania sygnałów. W tej bibliotece znajdziemy filtr Savitzkyego-Golaya, Czybyszewa oraz wiele innych technik przeznaczonych do usuwania szumów z sygnałów skażonych czyli do ich wygładzania i filtrowania.

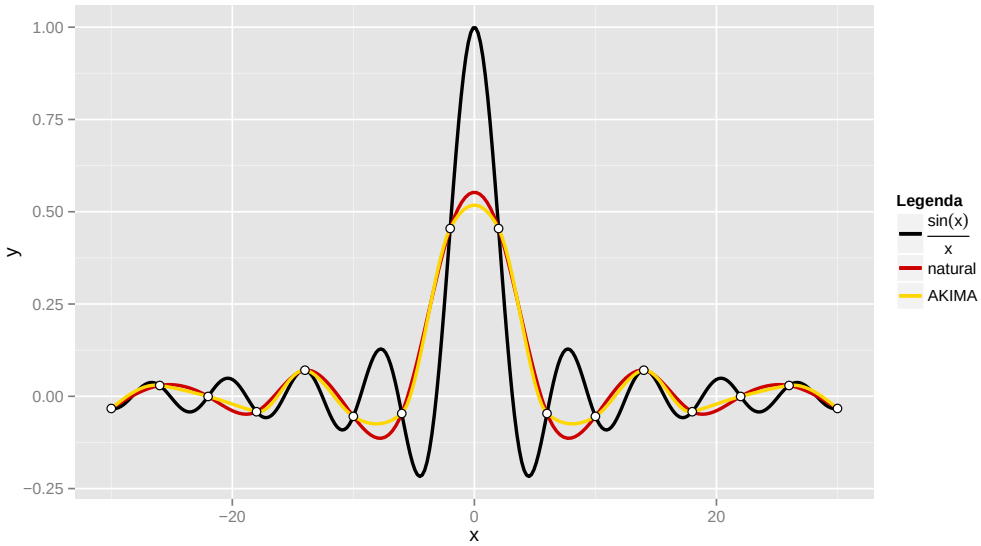
4. Interpolacja

W tym rozdziale przedstawimy dwa algorytmy interpolacyjne: Cubic splines – `spline(stats)` oraz Akima splines – `aspline(akima)`.

Pierwszy przykład to obliczanie wartości funkcji $f(x) = \frac{\sin(x)}{x}$ która jest zadana w postaci 16 równoodległych punktów na przedziale $x \in \langle -30; 30 \rangle$.

```
x=seq(-30,30,by=4); y=sin(x)/x
t=data.frame(x,y)
```

```
# wykresy:
library(akima)
ggplot(t,aes(x,y))+
stat_function(
fun= function(x) sin(x)/x,n=300,aes(colour= "a"), size=1)+
geom_line(data=data.frame(
spline(t, n=200,method="natural")),aes(colour="b"),size=1)+
geom_line(data=data.frame(
aspline(t, n=200)),aes(colour="c"),size=1)+
geom_point(size=2.5,pch=21,col="black",bg="white")+
scale_colour_manual(name= "Legenda",
values= c("black","red3","gold"),
labels=c(bquote(frac(sin(x), x)),"natural","AKIMA"))
```

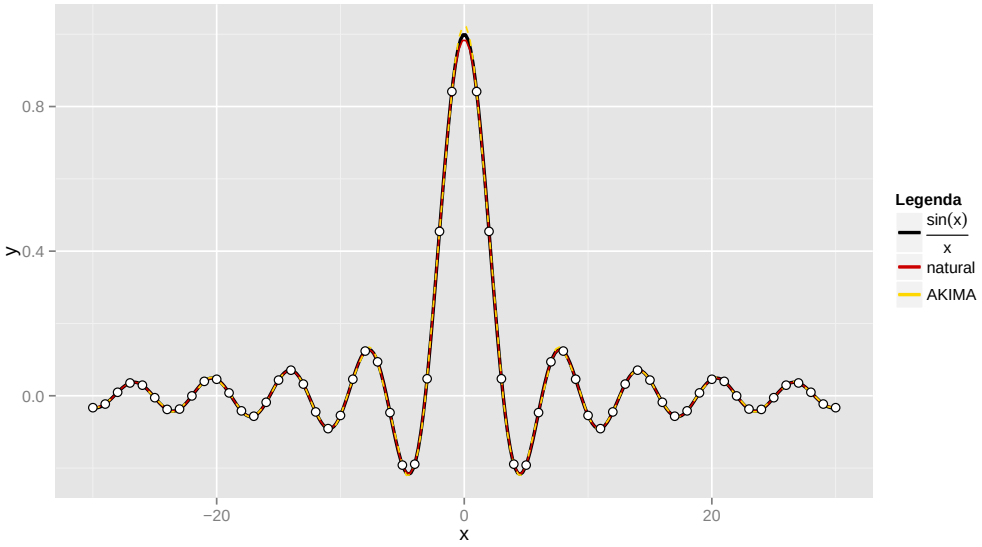


Rys. 4.1. Interpolacja.

Przybliżenie funkcji będzie lepsze gdy weźmiemy pod uwagę więcej punktów np. 61 równoodległych punktów.

```
x=seq(-30,30,by=1);y=sin(x)/x
t=data.frame(x,y)
```

```
# wykresy:
ggplot(t,aes(x,y))+
stat_function(
fun= function(x) sin(x)/x,n=300,aes(colour= "a"), size=1)+
geom_line(data=data.frame(
spline(t, n=200,method="natural")),aes(colour="b"),size=0.5)+
geom_line(data=data.frame(
aspline(t, n=200)),aes(colour="c"),size=0.5,lty=2)+
geom_point(size=2.5,pch=21,col="black",bg="white")+
scale_colour_manual(name= "Legenda",
values= c("black","red3","gold"),
labels=c(bquote(frac(sin(x), x)),"natural","AKIMA"))
```



Rys. 4.2. Interpolacja.

Kolejny przykład dotyczy tzw. zagęszczania tablic. Na podstawie danych w tabeli 4.1 obliczymy jaka była temperatura o godzinie 14:30.

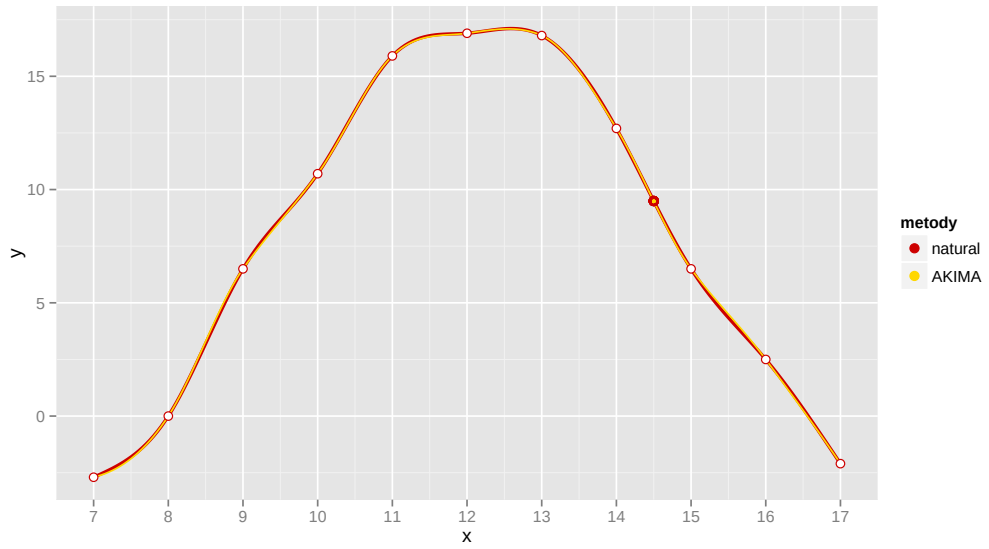
Tabela 4.1. Temperatura.

godzina [h]	7	8	9	10	11	12	13	14	15	16	17
temperatura [°C]	-2,7	0,0	6,5	10,7	15,9	16,9	16,8	12,7	6,5	2,5	-2,1

```
x=7:17
y=c(-2.7,0,6.5,10.7,15.9,16.9,16.8,12.7,6.5,2.5,-2.1)
t= data.frame(x,y)
```

```
# natural:
Yn= spline(x,y,method="natural",xout=c(14.5))$y; Yn
[1] 9.495417
# AKIMA:
Ya= aspline(x,y,xout=c(14.5))$y; Ya
[1] 9.479256
```

```
# wykresy:
ggplot(t,aes(x,y))+
geom_line(data=data.frame(
spline(t, n=200,method="natural")),col="red3",size=1)+
geom_line(data=data.frame(
aspline(t, n=200)),col="gold",size=0.5)+
geom_point(col="red3",size=2.5,pch=21,bg="white")+
geom_point(aes(x=14.5,y=Yn,colour="a"),pch=20,size=4)+
geom_point(aes(x=14.5,y=Ya,colour="b"),pch=20,size=1)+
scale_colour_manual(name= "metody",
values= c("red3","gold"),
labels=c("natural","AKIMA"))+
scale_x_continuous(limits=c(7,17), breaks=7:17)
```



Rys. 4.3. Interpolacja.