Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Magdalena Łuniewska
Nr albumu: 277386

Modele nieliniowe a funkcja `nls()`

Praca licencjacka
na kierunku MATEMATYKA W RAMACH MIĘDZYWYDZIAŁOWYCH INDYWIDUALNYCH STUDIÓW MATEMATYCZNO-PRZYRODNICZYCH

Praca wykonana pod kierunkiem
dra inż. Przemysława Biecka
Instytut Matematyki Stosowanej i Mechaniki

wrzesień 2012

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autorki pracy

Świadoma odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autorki pracy

Streszczenie

Równania regresji nieliniowej, umożliwiające przewidywanie wartości jednej zmiennej w oparciu o wartości innych zmiennych, są powszechnie stosowane w naukach społecznych oraz przyrodniczych. W pracy przedstawiono równania regresji nieliniowej wraz z obszernymi przykładami ich zastosowań (zaczerpniętymi z dorobku nauk takich, jak biologia, chemia czy socjologia). Dokładnie opisano także procedurę opracowywania i diagnostyki modeli regresji nieliniowej w programie **R** z wykorzystaniem funkcji `nls()` oraz funkcji pomocniczych. Praca zawiera też przykład zastosowania przedstawianej teorii oraz opisywanych metod w analizie danych rzeczywistych dotyczących odsetka osób palących papierosy w różnych grupach wiekowych populacji polskiej.

Słowa kluczowe

regresja nieliniowa, modele nieliniowe, **R**, funkcja `nls()`, palenie papierosów

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.2 Statystyka

Klasyfikacja tematyczna

62J02 General nonlinear regression

Tytuł pracy w języku angielskim

Nonlinear models and the model fitting function `nls()`

Spis treści

Wprowadzenie

Równania regresji należą do najpowszechniejszych metod statystycznych stosowanych zarówno w instytutach badawczych (np. w naukach społecznych bądź przyrodniczych), jak i w agencjach marketingowych czy urzędach statystycznych. W niniejszej pracy przedstawiono teorię regresji nieliniowej oraz jej wykorzystanie w programie **R** wraz z obszernymi przykładami.

Praca składa się z trzech części. Pierwsza z nich przedstawia regresję nieliniową. Druga część pracy zawiera opis opracowania modelu regresji nieliniowej w programie **R** przy użyciu funkcji `nls()`. Omówione jest działanie samej funkcji `nls()`, trudności mogące pojawić się przy jej stosowaniu (np. problemy związane z nieprawidłowym wyborem wartości początkowych algorytmu) oraz metody umożliwiające sprawniejsze stosowanie funkcji. Ostatni rozdział zawiera przykład zastosowania praktycznego metod opisanych we wcześniejszych częściach pracy. W części tej przedstawione są dwa modele regresji: liniowy i nieliniowy ze względu na parametry. Modele te dotyczą zależności pomiędzy wiekiem a odsetkiem palaczy w populacji polskiej. Przedstawione modele porównane są ze sobą pod względem dopasowania do danych oraz spełnienia założeń dotyczących modeli regresji nieliniowej.

0.1. Oznaczenia

W pracy przytaczane są fragmenty kodu z programu **R**. Każdy przykład zapisywany jest przy pomocy czcionki o stałej szerokości i umiejscawiany jest w ramce, jak poniżej.

```
> przykladowa.komenda(sample)

przykladowa odpowiedz
```

Nazwy funkcji programu **R**, a także nazwy wykorzystywanych obiektów (np. zbiorów danych), zapisywane są w tekście przy użyciu czcionki o stałej szerokości, np. tutaj przestawiona jest `przykladowa.komenda`, która jest bardzo użyteczna.

Przytaczane równania matematyczne zapisywane są pochyloną czcionką i opatrywane są kolejnymi numerami, jak poniżej w równaniu 1:

$$y = f(x). \tag{1}$$

Nazwy zmiennych w równaniach oraz funkcji matematycznych zapisywane są pochyloną czcionką, np. $x_1, \ldots, x_n$.

Rozdział 1

Regresja nieliniowa

1.1. Model regresji

Regresja jest metodą statystyczną, przy użyciu której badać można zależność pomiędzy zmiennymi oraz przewidywać nieznane wartości jednych zmiennych (wyjaśnianych, zależnych) na podstawie znanych wartości innych zmiennych (wyjaśniających, niezależnych). W modelach regresji zmienną jednowymiarową, oznaczaną najczęściej przez y, wiąże się z wektorem zmiennych wyjaśniających $\mathbf{x}$ za pomocą funkcji f, zgodnie z równaniem:

$$y = f(\mathbf{x}, \beta). \tag{1.1}$$

Często wektor zmiennych wyjaśniających jest jednowymiarowy – wtedy mówi się o zmiennej niezależnej x. Budowa modelu regresji polega na wyznaczeniu wartości parametrów β, na podstawie wartości zmiennej wyjaśnianej i zmiennych wyjaśniających oraz ogólnej (znanej z dokładnością do p-wymiarowego wektora parametrów $\beta = (\beta_1, ..., \beta_p)$) postaci funkcji f.

1.2. Model regresji nieliniowej

W dalszej części pracy rozważane są modele oparte o następujące równanie:

$$y_i = E(y_i|\mathbf{x}_i) + \epsilon_i = f(\mathbf{x_i}, \beta) + \epsilon_i. \tag{1.2}$$

Wyjaśnijmy, co oznaczają poszczególne składowe równania 1.2. Zmienne zależna i wektor zmiennych niezależna oznaczanych są odpowiednio przez y i $\mathbf{x}$. Budując model regresji nieliniowej, opieramy się na znanych n wartościach realizacji wektora zmiennych niezależnych $\mathbf{x}$ i zmiennej zależnej y: $(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)$. Zatem y_i i $\mathbf{x_i}$ są wartościami zmiennej wyjaśnianej i wektora zmiennych wyjaśniających w i-tej obserwacji. Zacznijmy od wyjaśnienia prawej strony równania: $f(\mathbf{x_i}, \beta) + \epsilon_i$. Widzimy, że wartość y_i zależy od wartości funkcji f dla określonych wartości $\mathbf{x_i}$ oraz wartości parametrów β oraz od ϵ_i.

Funkcja f opisuje wcześniej wspominaną zależność pomiędzy wartościami zmiennej niezależnej i wartościami zmiennej zależnej. Ta funkcja jest *nieliniowa* względem co najmniej jednego z parametrów β.

ϵ_i oznacza błąd pomiaru zmiennej zależnej w i-tej obserwacji. Często mamy do czynienia z pomiarami, w których zarówno zmienna zależna, jak i zmienna niezależna mierzone są z pewną dokładnością. Zarówno wynik pomiaru długości przy pomocy linijki, jak i wynik testu psychologicznego obarczone są pewnej wielkości błędem, wynikającym np. z niedokładności stosowanych narzędzi. ϵ_i oznacza właśnie tego rodzaju błędy w pomiarze zmiennej zależnej.

Innymi słowy, prawą stronę równania należy rozmieć następująco: wartość zmiennej wyjaśnianej w i-tej obserwacji zależy od wartości funkcji dla i-tej wartości zmiennej wyjaśniającej i danego zestawu parametrów oraz od wartości błędu pomiaru ϵ_i.

Przyjrzyjmy się teraz środkowej części równania: $y_i = E(y_i|\mathbf{x_i}) + \epsilon_i$. Porównując ją z prawą częścią równania, stwierdzamy, że $= E(y_i|\mathbf{x_i}) = f(\mathbf{x_i}, \beta)$. Należy rozumieć to następująco: ponieważ w pomiarze zmiennej niezależnej także występują błędy, równość pomiędzy wartością y_i a wartością $f(\mathbf{x_i}, \beta)$ zachodzi z pewną dokładnością, stąd uprawnione jest zastąpienie symbolu funkcji f warunkową wartością oczekiwaną.

Należy podkreślić, że funkcja f jest nieliniowa względem co najmniej jednego z parametrów $\beta_1, ..., \beta_p$. Budując model regresji nieliniowej znamy wzór funkcji f, wiemy, jakiego rodzaju jest to funkcja (wielomianowa, wykładnicza itp.). Nie znamy natomiast wartości parametrów β. Zadanie budowy modelu regresji polega na wyznaczeniu wartości tych parametrów w oparciu o wartości obserwacji $(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)$ i o ogólną postać funkcji f. Omawiana w rozdziale 2 funkcja `nls()` służy właśnie do wyznaczania wartości parametrów w zadanym równaniu. Więcej na temat modeli regresji nieliniowej można przeczytać w [18] i [3].

W powyższym rozumowaniu niejawnie zakłada się, że osoba przystępująca do analizy danych posiada pewną wiedzę na temat tego, jakiego rodzaju funkcja f powinna być wykorzystana. Z tego powodu metody regresji nieliniowej przeznaczone są do analizy danych, dla których istnieją ugruntowane teoretycznie lub empirycznie zależności pomiędzy zmiennymi niezależnymi a zmienną zależną.

1.3. Przykład modelu: Model Bevertona-Holta

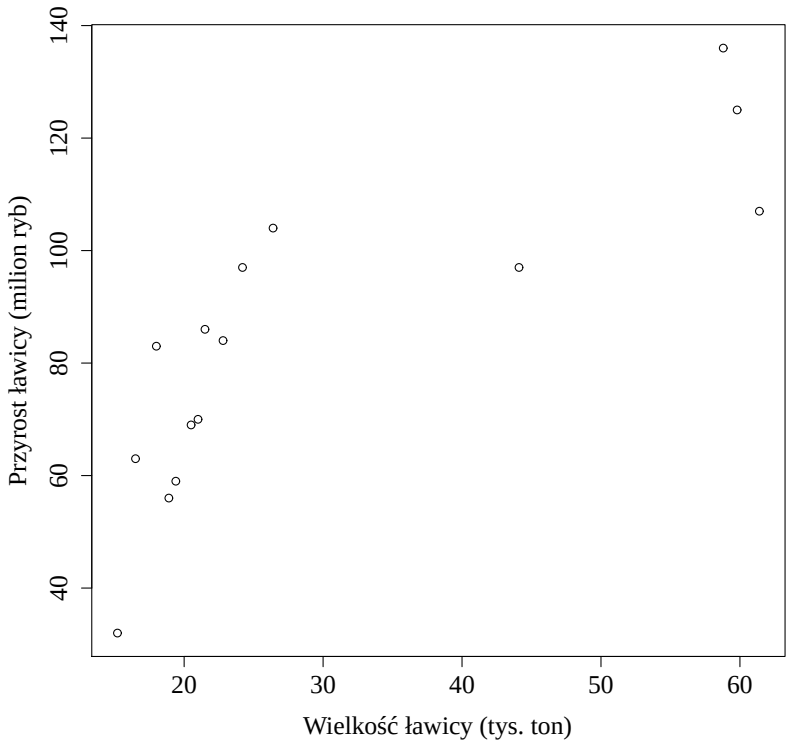
W tej sekcji teoria przedstawiona w sekcji 1.2 jest zilustrowana przykładem pochodzącym z ichtiologii. W ichtiologii istnieje wiele modeli opisujących zależność pomiędzy rozmiarem stada tarłowego (jego biomasy) a rozmiarem ławicy po okresie tarła. Zbiór danych `M.merluccius` z pakietu `nlrwr` ([17]) dla programu **R** zawiera m.in. dwie zmienne: `spawn.biomass` (rozmiar ławicy przed tarłem mierzony w tonach biomasy), `num.fish` (rozmiar ławicy po tarle mierzony w milionach ryb), które w niniejszej pracy służą do zilustrowania idei regresji nieliniowej.

Zależność pomiędzy rozmiarem stada tarłowego a rozmiarem ławicy po okresie tarła przedstawiona jest na rysunku 1.1 na stronie 9. Posłużenie się różnymi miarami na osiach wykresu uzasadniają zbadane empirycznie prawidłowości biologiczne – na liczebność przyrostu ławicy wpływają zarówno liczebność stada tarłowego, jak i masa każdej z ryb, dlatego uprawnione są próby szacowania *liczebności* ławicy przy pomocy danych o *biomasie*.

Jednym z modeli opisujących rozrost ławicy przy użyciu wyżej wymienionych zmiennych jest model Bevertona-Holta [5]:

$$P_{t+1} = f(S_t, (\alpha, k)) = \frac{\alpha \cdot S_t}{1 + S_t/k}. \tag{1.3}$$

W powyższym zapisie P_{t+1} jest liczebnością ławicy w pokoleniu $t+1$, S_t to biomasa stada tarłowego pokolenia t, natomiast α i k są szacowanymi parametrami. Parametr α można interpretować jako współczynnik rozrodu, natomiast dla parametru k zachodzi $(\alpha-1)\cdot k = M$, gdzie M jest pojemnością środowiska, czyli maksymalnym rozmiarem ławicy mogącej przeżyć w środowisku.



Rysunek 1.1: Wielkość ławicy tarłowej a liczebność ławicy po okresie tarła.

1.4. Dopasowywanie parametrów modelu

1.4.1. Reszty i resztowa suma kwadratów

Rozważany jest model określony równaniem 1.2, w którym dane są wartości wektora zmiennych wyjaśniających $\mathbf{x}_1, \ldots, \mathbf{x}_n$[1], wartości zmiennej wyjaśnianej $y_1, \ldots, y_n$ oraz wzór funkcji f zależnej od pewnych nieznanych parametrów $\beta_1, \ldots, \beta_p$. Jak wspominano w sekcji 1.2, proces budowy modelu regresji nieliniowej polega na ustalaniu wartości parametrów $\beta_1, \ldots, \beta_p$. Jako oszacowania parametrów wybiera się takie $\hat{\beta}_1, \ldots, \hat{\beta}_p$, dla których funkcja f jest *średnio najlepiej dopasowana* do danych $y_1, \ldots, y_n$. Ujmując rzecz precyzyjniej, używa się parametrów uzyskanych przy minimalizacji resztowej sumy kwadratów (*residual sums of squares, RSS*) względem β:

$$RSS(\beta_1, \ldots, \beta_p) = RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i, \beta))^2. \tag{1.4}$$

Zatem estymatorem parametru β jest takie $\hat{\beta}$, w którym RSS osiąga globalne minimum.

1.4.2. Metody numeryczne

W odróżnieniu od przypadku regresji liniowej (np. [3], 1-29), minimalizacja RSS (równanie 1.4) w ogólności nie jest liniowa, z uwagi na nieliniowość funkcji f względem parametrów.

[1] $\mathbf{x}_1, \ldots, \mathbf{x}_n$ należy rozumieć jako wartości wektora zmiennych wyjaśniających w n obserwacjach, nie wartości kolejnych n współrzędnych tego wektora.

Dlatego konieczne jest zastosowanie *numerycznych metod optymalizacji*. Numeryczne metody optymalizacji są iteracyjnymi procedurami, które w pewnej liczbie kroków pozwalają na dokładne określenie optymalnych wartości szukanych parametrów. W każdym kolejnym kroku algorytm określa nowy zestaw parametrów oparty na posiadanych danych, wybranym modelu i dotychczasowym zestawie parametrów. Najpopularniejszym algorytmem pozwalającym na oszacowanie parametrów w regresji nieliniowej jest Metoda Gaussa-Newtona, opierająca się na liniowym przybliżaniu funkcji nieliniowej f w kolejnych krokach ([8], 71-81; [3], 40-43). Na tej metodzie opiera się także działanie domyślnego algorytmu funkcji `nls()` (sekcja 2.1.1).

Numeryczne metody optymalizacji parametrów są dalekie od doskonałości. Dwoma najczęstszymi problemami związanymi z ich użyciem są:

1. wybór parametrów początkowych i sposób rozpoczynania działania algorytmu,
2. upewnienie się, że w wyniku działania algorytmu zostały wybrane parametry będące globalnym, a nie lokalnym minimum funkcji.

Problemy te są ze sobą powiązane. Jeśli parametry początkowe są odpowiednio bliskie optymalnym parametrom, w efekcie działania algorytmu w skończonej liczbie kroków optymalne wartości parametrów zostają osiągnięte (w takiej sytuacji mówimy o zbieżności algorytmu). Zły wybór wartości początkowych często prowadzi do błędnych oszacowań parametrów, a co za tym idzie – otrzymania błędnego modelu ([17]).

Rozwiązania problemów regresji nieliniowej (czyli wybrane optymalne zestawy parametrów β) są obliczane numerycznie, dlatego mogą wpływać na nie takie czynniki, jak: wybór algorytmu, wybór implementacji danego algorytmu (np. określenie definicji zbieżności danej procedury albo to, czy pierwsze pochodne funkcji liczone są numerycznie czy odgórnie zadane), wybór parametryzacji oraz wybór wartości początkowych estymowanych parametrów. Jednak przeważnie wartości otrzymywane dla określonego modelu i wybranego zbioru danych nie różnią się między sobą zasadniczo[2].

[2]Jeśli dla obranego modelu i określonego zbioru danych dysproporcje pomiędzy wartościami optymalnych parametrów oszacowanymi przez różne algorytmy albo różne implementacje tego samego algorytmu są duże, możliwe, że należy zmienić model, do którego próbuje się dobrać parametry, na model prostszy.

Rozdział 2

Funkcja `nls()`

2.1. Wprowadzenie

W tej części pracy przedstawiona jest metoda doboru parametrów do modelu regresji nieliniowej, zgodnie z teorią przedstawioną w sekcji 1.2. W rozdziale 1 kilkakrotnie wspomniano, że analiza regresji nieliniowej polega na ustaleniu takich wartości parametrów $\beta_1, \ldots, \beta_p$, dla których RSS (równanie 1.4) osiąga minimum globalne. Pakiet statystyczny **R** ([13]) w podstawowej instalacji (w pakiecie `stats`, [13]) zawiera funkcję `nls()`, która wylicza wartości estymatorów parametrów $\hat{\beta}_1, \ldots, \hat{\beta}_p$ na podstawie zadanego równania regresji oraz danych wartości zmiennych niezależnych i zależnych.

Dodatkowo w tym rozdziale przedstawione są podstawowe sposoby optymalizowania działania funkcji `nls()` poprzez zadawanie odpowiednich wartości początkowych parametrów (sekcja 2.3.1), ręczne zadawanie wartości pochodnych (sekcja 2.3.2) oraz wybór innego niż domyślny algorytmu w przypadku modeli z parametrami warunkowo liniowymi (sekcja 2.3.3). W dalszej kolejności w rozdziale przedstawione są informacje ułatwiające korzystanie z funkcji `nls()`: informacje o zmianach ustawień funkcji (sekcja 2.3.4) oraz wyjaśnienia najpopularniejszych komunikatów o błędach, pojawiających się w trakcie działania funkcji `nls()` (sekcja 2.3.5).

Warto zaznaczyć, że chociaż w niniejszej pracy uwaga skupiona jest na modelach o jednej zmiennej wyjaśniającej, funkcja `nls()` umożliwia także szacowanie wartości parametrów w modelach o dwóch i więcej zmiennych niezależnych (więcej na ten temat znaleźć można w [17]).

2.1.1. Podstawowe informacje o funkcji `nls()`

Chcąc użyć funkcji `nls()`, można określić wartości szeregu argumentów. Przykładowe argumenty funkcji `nls()` zamieszczone są w tabeli 2.1 na stronie 12. Przykładowe użycia niektórych z argumentów wymienionych w tabeli 2.1 (np. `formula`, `data`, `start` oraz `trace`) zawiera sekcja 2.2. Więcej informacji na temat argumentów funkcji `nls()` znaleźć można w pomocy do tej funkcji ([2]).

Funkcja `nls()` umożliwia dodatkowo użycie wielu metod. Część z nich zostanie wykorzystana w przykładach w dalszej części pracy[2]. Do najbardziej użytecznych metod należą metody wymienione w tabeli 2.2 na stronie 13.

[2]Mimo tego, że w pracy omówiona będzie większość wymienionych funkcji, przydatne mogą być także strony z pomocą do **R**.

Tabela 2.1: Przykładowe argumenty funkcji `nls()`.

Argument	Zastosowanie
`algorithm`	Wybór algorytmu używanego do estymacji wartości parametrów. Może przyjmować następujące wartości: 1. `default`: algorytm domyślny, czyli algorytm Gaussa-Newtona, 2. `plinear`: algorytm używany w modelach wykorzystujących *warunkowo liniowe* parametry[1], 3. `port`: algorytm używany w modelach o znanych z góry ograniczeniach parametrów. W przypadku, kiedy wartości początkowe dla parametrów są nieokreślone w algorytmach `default` oraz `port`, tzn. nie określono wartości argumentu `start`, funkja `nls()` wszystkim parametrom pisze na początku wartości równe 1 (algorytm `plinear` w takiej sytuacji nie rozpocznie działania).
`control`	Zmiana ustawień funkcji `nls()` (patrz: sekcja 2.3.4).
`data`	Wybór pakietu danych do opracowania modelu.
`formula`	Określenie funkcji w wykorzystywanym modelu.
`na.action`	Określenie postępowania z brakami danych. Domyślną wartością jest `na.omit()`, innymi możliwymi wartościami są `na.exclude()` oraz `na.fail()`.
`subset`	Wybór podzbioru danych, który ma być wykorzystany do budowy modelu.
`trace`	Umożliwienie śledzenia kolejnych kroków pracy algorytmu.

Tabela 2.2: Metody używane przez funkcję `nls()`.

Metoda	Zastosowanie
`coef`	Podaje estymacje parametrów w modelu.
`confint`	Pozwala na przedziałowe oszacowanie parametrów.
`deviance`	Podaje minimum RSS.
`dr.residual`	Podaje liczbę stopni swobody $(n - p)$.
`fitted`	Wymienia oszacowane wartości zmiennej zależnej dla wartości zmiennych niezależnych (ze zbioru danych, na podstawie których oszacowano model).
`formula`	Podaje wzór, za pomocą którego określona została funkcja w modelu.
`logLik`	Podaje maksimum wartości logarytmu funkcji wiarogodności $(log(L(\hat{\beta}, \hat{\sigma})))$.
`plot`	Rysuje wykres wystandaryzowanych reszt.
`predict`	Podaje wartości zmiennej zależnej dla zadanych przez użytkownika wartości zmiennych niezależnych.
`print`	Krótko podsumowuje dopasowanie modelu.
`profile`	Podaje wartości statystyk t (przydatne w testowaniu hipotez).
`residuals`	Podaje wartości reszt w modelu.
`summary`	Podsumowuje dopasowanie modelu.
`vcov`	Podaje wartości wariancji i kowariancji oszacowań parametrów.

2.2. Przykład zastosowania funkcji `nls()`

2.2.1. Przykładowe dane

W tej sekcji przedstawiony jest najprostszy przykład zastosowania funkcji `nls()`. Wykorzystuje on dane empiryczne z eksperymentu przeprowadzonego przez Cedergreen i Madsena ([6]). Cadergreen i Madsen badali podatności pędów i korzenia rzęsy drobnej (*Lemna minor*) na wchłanianie związków azotu. Rzęsa drobna jest gatunkiem rośliny wodnej swobodnie pływającej po powierzchni wody, powszechnie występującym na całej kuli ziemskiej[3]. Roślina składa się ze spłaszczonych skórzastych pędów (o średnicy do 3 mm), kształtem przypominających liście, oraz pojedynczego korzenia. Podczas eksperymentu izolowano korzeń rzęsy drobnej oraz jej pędy (listki), a następnie wystawiono je na działanie związków azotu NH_4^+ oraz NO_3^- w różnym, kontrolowanym, stężeniu. Na koniec eksperymentu części roślin wysuszono, a poziom podatności na wchłanianie związków azotu obliczono na podstawie suchej masy rośliny oraz zmian stężenia tych związków w czasie. Na dane, których użyjemy w celu zilustrowania działania funkcji `nls()`, składają się dane o początkowym stężeniu związków azotu (`conc`) oraz o stopniu wchłonięcia tychże związków (`rate`); dane te dostępne są w pakiecie `nlrwr` w zbiorze `L.minor` ([17]). Tabela 2.3 ukazuje dane ze zbioru `L.minor`, dane te przedstawione są także na wykresie 2.1.

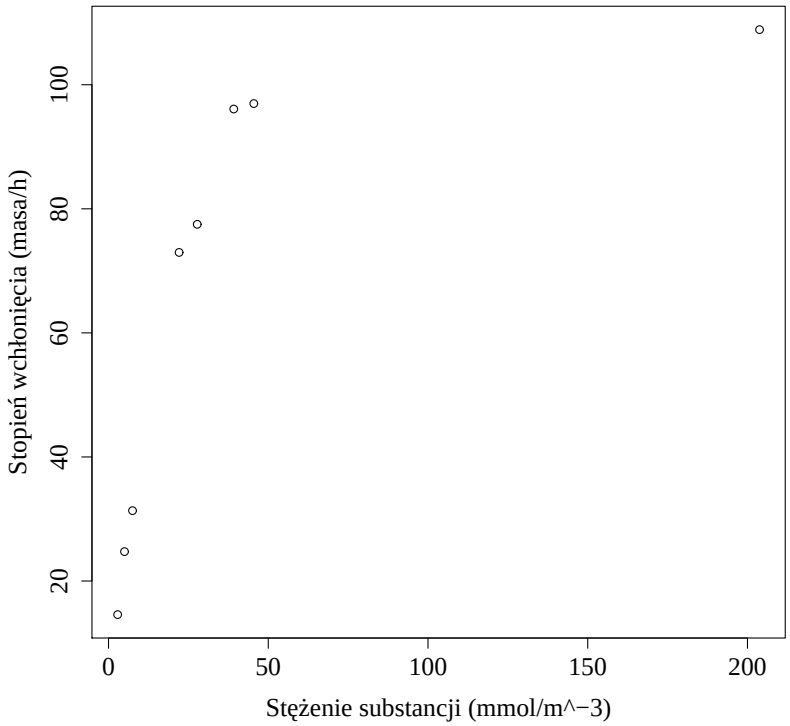
2.2.2. Opracowanie modelu

Dane z badań eksperymentalnych dotyczących przyswajania związków w zależności od ich stężenia, takich, jak badania opisane powyżej, często opisywane są przy pomocy modelu

[3]W Polsce rzęsa drobna jest uznawana za roślinę pospolitą. Powszechnie występuje na powierzchni zbiorników wodnych.

Tabela 2.3: Stężenie początkowe związków azotu i stopień ich wchłonięcia.

	conc	rate
1	2,86	14,58
2	5,01	24,74
3	7,52	31,35
4	22,10	72,97
5	27,77	77,50
6	39,20	96,09
7	45,48	96,97
8	203,78	108,88



Rysunek 2.1: Dane ze zbioru `L.minor`: stężenie substancji a stopień jej wchłonięcia.

Michaelisa-Menten ([9]). Model ten, opracowany w drugiej dekadzie XX wieku, jest jednym z najprostszych i najszerzej wykorzystywanych modeli do opisu zjawisk z zakresu kinetyki enzymatycznej, czyli zjawisk związanych z czasowym przebiegiem reakcji chemicznych katalizowanych przez enzymy. Tradycyjnie model Michealisa-Menten opisuje się równaniem 2.1, w którym v to szybkość reakcji chemicznej, a $[S]$ to stężenie substratu. W modelu wykorzystywane są dwa parametry: K i V_m. V_m oznacza maksymalną osiąganą wartość v, a K jest stałą Michaelisa, czyli takim stężeniem substratu, przy którym szybkość reakcji enzymatycznej (v) jest równa połowie szybkości maksymalnej (V_m).

$$v = f([S], (K, V_m)) = \frac{V_m \cdot [S]}{K + [S]} \tag{2.1}$$

$$y = f(x, (K, V_m)) = \frac{V_m \cdot x}{K + x} \tag{2.2}$$

Dla danych ze zbioru `L.minor` model opisuje szybkość wchłaniania związków azotu (y, `rate`) przez rzęsę drobnolistną w zależności od ich stężenia (x, `conc`) i przyjmuje postać daną równaniem 2.2. Przyjrzymy się temu równaniu. Dla $x = 0$ (czyli dla zerowego stężenia związków azotu), wartość funkcji $f(x, (K, V_m))$ wynosi 0 (więc związki azotu nie są wchłaniane), wraz ze wzrostem wartości x wartość y rośnie zbiegając do wartości asymptotycznej V_m. Zatem wraz ze wzrostem stężenia związków azotu w środowisku, wzrasta szybkość ich wchłaniania przez roślinę. Istnieje jednak pewna nieprzekraczalna granica szybkości wchłaniania. Dobór wartości parametrów K i V_m dla danych ze zbiory `L.minor` można wygodnie przeprowadzić korzystając z funkcji `nls()`, jak w poniższym przykładzie.

```
> L.minor.m1 <- nls(rate ~ Vm * conc/(K + conc),
+ data = L.minor, start = list(K = 20, Vm = 120),
+ trace = TRUE)

624.3282 :       20      120
244.5460 :       15.92382 124.57148
234.5198 :       17.25299 126.43877
234.3595 :       17.04442 125.96181
234.3533 :       17.08574 126.04671
234.3531 :       17.07774 126.03016
234.3531 :       17.07930 126.03340
234.3531 :       17.07899 126.03276
```

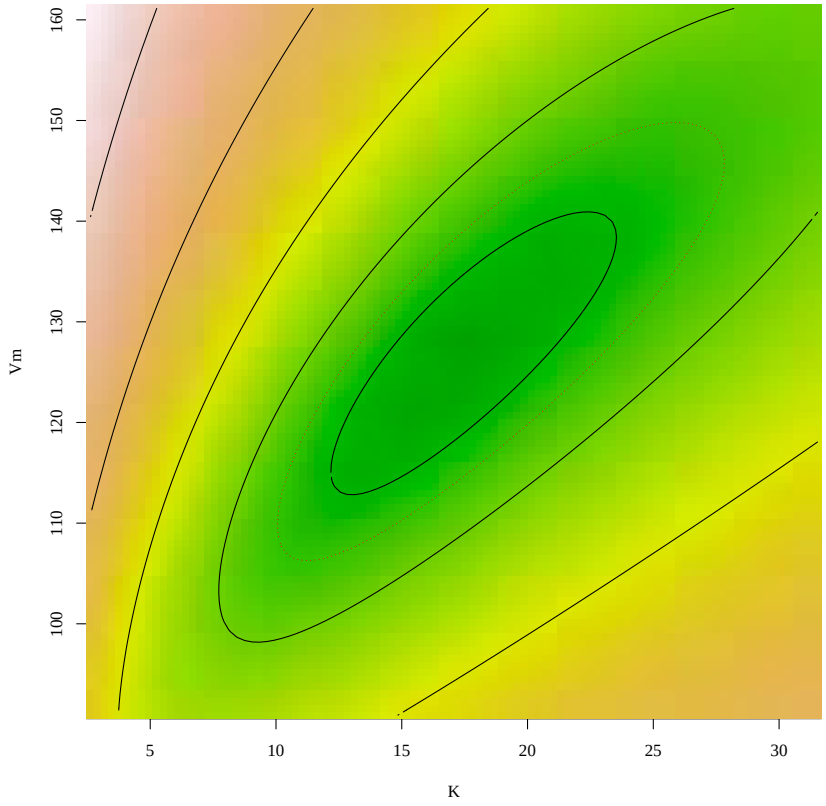
Wyjaśnijmy znaczenie poszczególnych części tej formuły. Funkcja `nls()` w tym przypadku korzysta z czterech argumentów: `formula`, `data`, `start` oraz `trace`, zgodnie z ich przeznaczeniem opisanym w tabeli 2.1 na stronie 12. Pierwszy z nich – `formula` – przyjmuje wartość: `rate ~ Vm * conc/(K + conc)`. Jest to zapis równania 2.2. W równanie funkcji włączone są zarówno parametry K i V_m (o zadanych z góry nazwach), jak i zmienna wyjaśniająca (`conc`). Innym sposobem zadania parametrów funkcji `nls()` może być wpisanie wcześniej zdefiniowanej w **R** funkcji w miejsce napisu `rate ~ Vm * conc/(K + conc)`. Warto zauważyć, że parametry w podanym wyżej równaniu funkcji nie odnoszą się do nazw zmiennych w wykorzystywanym zbiorze danych, nie są też powszechnie stosowanymi symbolami działań matematycznych ani funkcji (np. nie występuje wśród nich ani symbol `e`, ani `pi`).

Drugi argument – `data` – określa zbiór danych, z którego pochodzą wartości zmiennych wyjaśnianej (`rate`) i wyjaśniającej (`conc`). Aby rozpocząć działanie algorytmu szacującego wartości parametrów, konieczne jest określenie wartości początkowych. W tym celu używany jest argument `start`. Pamiętając o interpretacji parametrów w wykorzystywanym modelu, z wykresu 2.1 można odczytać przybliżone wartości parametrów. Widać z niego, że asymptota górna wykresu (V_m) wynosi około 120, natomiast mniej więcej połowę tej wartości (60) widać dla wartości $K = 20$.

Argument `trace` określa, czy wartości parametrów oraz resztowych sum kwadratów (RSS) mają być wyświetlane po każdym kroku procedury iteracyjnej (domyślna wartość: `FALSE`). Podawanie wartości RSS (albo wartości resztowego błędu standardowego lub wariancji) może być użyteczne jako sumaryczna miara dopasowania modelu. Taka miara jest przydatna do

porównań dopasowań różnych modeli (różnych funkcji zmiennej wyjaśniającej) do wybranego zbioru danych.

Przyjrzyjmy się danym zwróconym przez argument `trace` oraz ich ilustracjom na wykresach 2.2 i 2.3. Wykres 2.2 przedstawia zmiany w wartościach RSS w zależności od wartości parametrów K i V_m. Obszary oznaczone na zielono odpowiadają najniższym wartościom RSS. Obszary pomiędzy dwiema liniami (poziomicami) odpowiadają jednakowym wartościom RSS. Z rysunku widać, że RSS przyjmuje najniższe wartości dla $K \approx 17$ i $V_m \approx 125$.



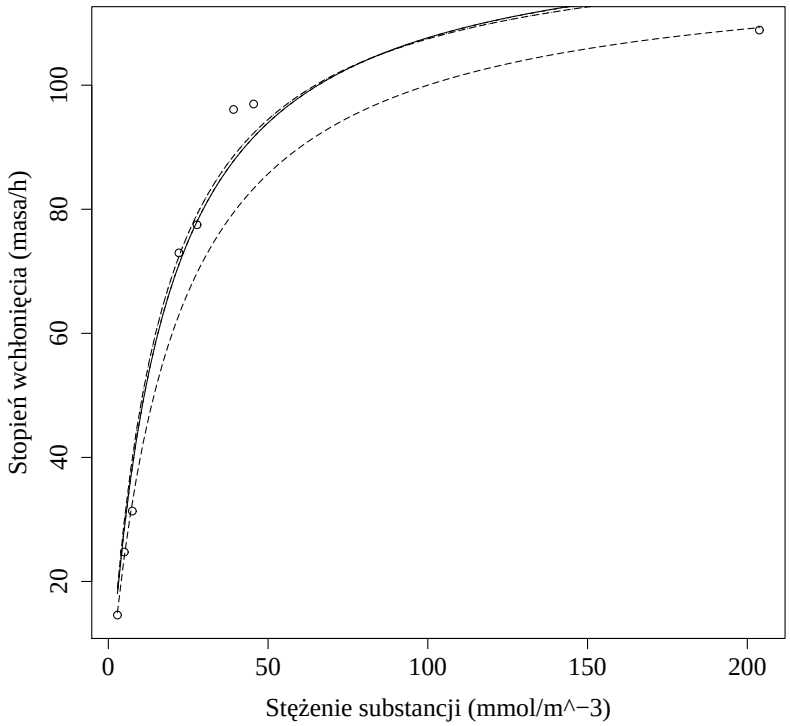
Rysunek 2.2: RSS w zależności od wartości parametrów K i V_m.

Wykres 2.3 przedstawia krzywe regresji zadane równaniem 2.2 dla kolejnych estymowanych wartości parametrów. Na wykresie przedstawiono krzywe oparte na parametrach otrzymanych jedynie w części iteracji, z uwagi na wysokie podobieństwo krzywych otrzymywanych w ostatnich iteracjach. Na rysunku widać, że jedyną krzywą, która wyraźnie odbiega od pozostałych, jest krzywa złożona z kresek – oparta na wybranych wartościach początkowych parametrów. Pozostałe krzywe – złożona z kropek i złożona z naprzemiennych kropek i kresek – niemal pokrywają się z krzywą narysowaną linią ciągłą – ostatecznym rozwiązaniem.

Istnieje szereg metod wymienionych w tabeli 2.2, umożliwiających poznanie wartości dotyczących budowanego modelu. Aby znaleźć oszacowania parametrów możemy użyć metody `coef`.

```
> coef(L.minor.1)
        K         Vm
 17.07899 126.03276
```

Aby poznać minimalną wartość RSS w modelu można się posłużyć metodą `deviance`.



Rysunek 2.3: Krzywe regresji dla wartości parametrów V_m i K otrzymanych w kolejnych iteracjach. Kreski – wartości początkowe, kropki – wartości po pierwszej iteracji, naprzemienne kropki i kreski – wartości po trzeciej iteracji, linia ciągła – wartości po ostatniej iteracji.

```
> deviance(L.minor.m1)
[1] 234.3531
```

Dokładniejsze informacje na temat opracowanego modelu można uzyskać, stosując metodę `summary`.

```
> summary(L.minor.1)

Formula: rate ~ Vm * conc/(K + conc)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
K    17.079      2.953   5.784  0.00117 **
Vm  126.033      7.173  17.570 2.18e-06 ***
---
Signif. codes:  0 ‘’*** 0.001 ‘’** 0.01 ‘’* 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 6.25 on 6 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 8.144e-06
```

Metoda ta pozwala na zebranie kilku informacji. Pierwsza z nich to zadane równanie z modelu (tutaj: $rate\ V_m * conc/(K + conc)$). Następnie podawane są estymowane wartości parametrów i powiązane z nimi wielkości. Oszacowania parametrów podane są w pierwszej kolumnie. Asymptota pozioma przyswajalności związków azotu występuje dla wartości 126, a wartość stężenia substancji, skutkująca wzrostem przyswajalności od 0 do V_m to $17,08$. Druga kolumna zawiera oszacowania standardowych błędów estymacji parametrów. Kolejne dwie kolumny zawierają wartości testu t (więcej na ten temat można przeczytać np. w [17]). W ostatniej części wymienione są niektóre szczegóły procedury estymacji, takie jak: liczba iteracji przeprowadzonych przez algorytm szacujący (w tym wypadku wykonano siedem iteracji), oraz różnica między wartościami parametrów oszacowanymi w ostatnich dwóch krokach.

Podsumowując powyższą część, można stwierdzić, że dla danych pochodzących ze zbioru `L.minor` dopasowanym modelem Michealisa-Menten będzie krzywa dana równaniem:

$$rate = \frac{126,03 \cdot conc}{17,08 + conc}. \tag{2.3}$$

2.3. Dodatkowe informacje o funkcji `nls()`

Niekiedy, na przykład w sytuacji wielokrotnego oszacowywania wartości parametrów w tym samym modelu albo w sytuacji pracy z dużymi zbiorami danych, przydatna może być wiedza o tym, co zrobić, aby funkcja `nls()` działała szybciej.

Działanie funkcji `nls()` można usprawnić na kilka sposobów. Po pierwsze, warto zatroszczyć się o dobry wybór wartości początkowych (sekcja 2.3.1). Może to nie tyle usprawnić działanie funkcji `nls()`, co raczej umożliwić jej właściwą funkcjonalność. Po drugie, działanie funkcji `nls()` może odbywać się szybciej (szczególnie w przypadku pracy z dużymi zbiorami danych), jeśli określone zostaną wartości pochodnych cząstkowych (sekcja 2.3.2). Po trzecie, niekiedy mamy do czynienia z funkcjami określającymi model, które są liniowe względem części parametrów. W takiej sytuacji może być wykorzystany inny algorytm, usprawniający działanie funkcji `nls()` (sekcja 2.3.3). Ponadto działanie funkcji `nls()` może być monitorowane przy użyciu argumentu `control` (sekcja 2.3.4). Podrozdział zakończony jest informacją o najczęstszych błędach komunikowanych przez funkcję `nls()` (sekcja 2.3.5).

2.3.1. Wybór wartości początkowych

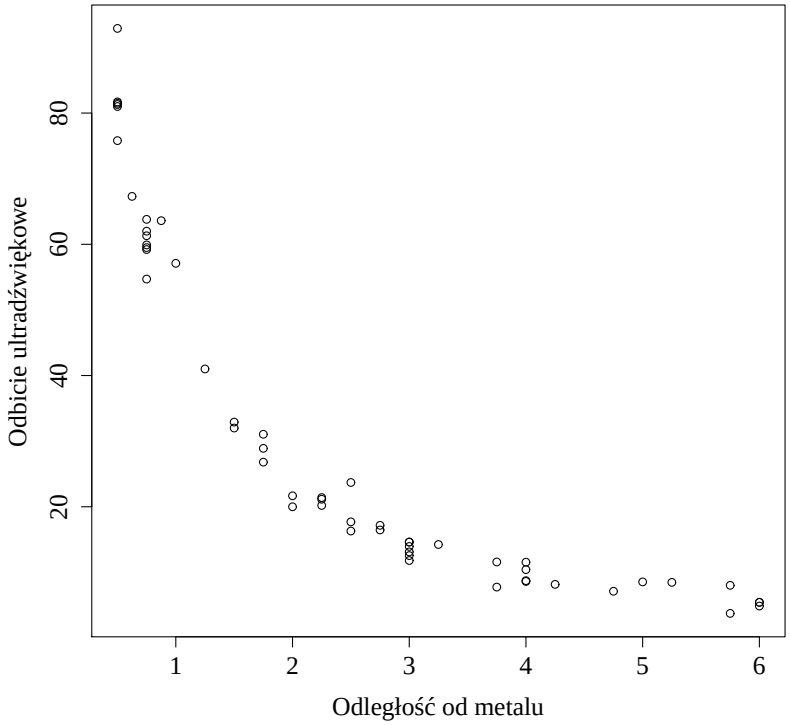
W tej części pracy opisane są sposoby wybierania wartości początkowych dla algorytmu, którym posługuje się funkcja `nls()`. Przedstawionych jest kilka metod znajdywania startowych wartości parametrów. Omówione jest także użycie funkcji samostartujących (*self-starters*).

W poprzedniej części pracy skuteczną metodą określania początkowych wartości parametrów była metoda „na oko”. Jest to najpopularniejszy sposób szacowania wartości początkowych, chociaż wymaga pewnego rodzaju doświadczenia w prowadzeniu analiz statystycznych. Metoda określania początkowych wartości parametrów przez zgadywanie może być szczególnie skuteczna w sytuacjach, kiedy łatwo jest interpretować parametry. W przeciwnym przypadku użyteczne może być przyjrzenie się wykresowi funkcji z modelu nieliniowego bądź bardziej skomplikowane poszukiwanie dobrych wartości startowych.

2.3.1.1. Analiza graficzna

Zazwyczaj (choć nie zawsze) przynajmniej część parametrów daje się interpretować (np. przy modelowaniu zjawisk przyrodniczych bądź fizycznych). Zbiór danych `Chwirut2` z pakietu `NISTnls` ([17]) zawiera dane z eksperymentu przeprowadzonego przez Narodowy Instytut

Standardów i Technologii (*National Institute of Standards and Technology*, `http://www.nist.gov.`) na temat odbicia ultradźwiękowego (*ultrasonic response*) w zależności od odległości metalu ([10]). Dane te przedstawione są na wykresie 2.4.



Rysunek 2.4: Dane ze zbioru `Chirwut2`: odległość metalu a promieniowanie ultradźwiękowe.

Dla tego zbioru danych NIST sugeruje zastosowanie modelu rozkładu wykładniczego określonego równaniem 2.4, w którym zmienną wyjaśnianą jest y, oznaczające natężenie promieniowania ultradźwiękowego, a zmienną wyjaśniającą jest x – odległość od metalu.

$$y = f(x, (\beta_1, \beta_2, \beta_3)) = \frac{exp(-\beta_1 x)}{\beta_2 + \beta_3} \tag{2.4}$$

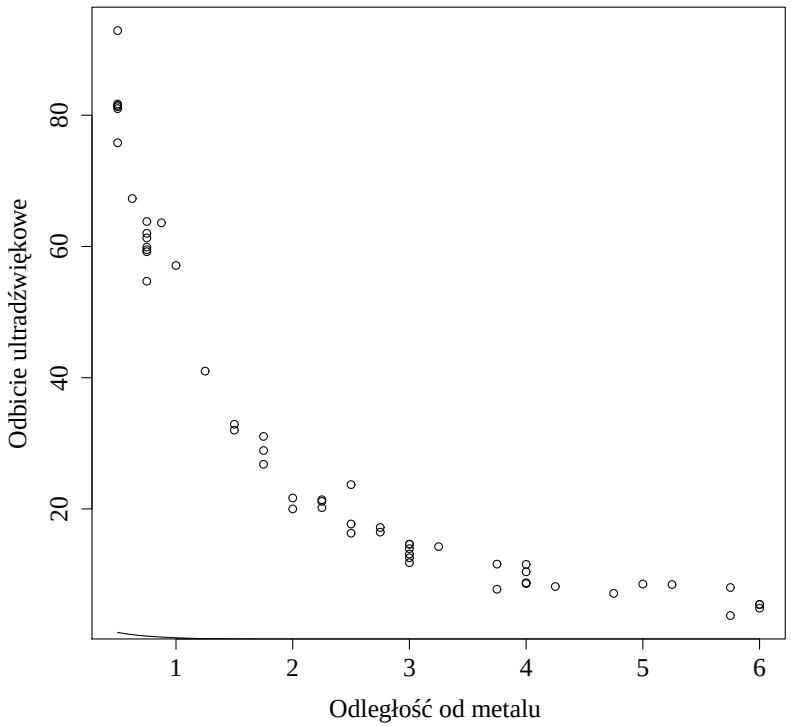
Dla $x = 0$ otrzymujemy wartość y równą $1/\beta_2$, dlatego za β_2 możemy wziąć wartość y_i dla najmniejszego zaobserwowanego x_i. Wartość ta w przybliżeniu wynosi $0,01$. Pozostałe dwa parametry – β_1 i β_3 nie mają równie prostych interpretacji. Dlatego spróbujmy prześledzić, jak wartość funkcji f zmienia się w zależności od wartości parametrów β_1 i β_3.

W pierwszym kroku zdefiniujmy funkcję z równania 2.4 w programie **R**.

```
> expFct <- function(x, beta1, beta2, beta3) {
+ exp(-beta1*x)/(beta2 + beta3 * x)}
```

Następnie narysujmy wykresy funkcji `expFct` dla różnych wartości parametrów. Wykorzystamy do tego funkcję `curve()`. Zacznijmy od przyjrzenia się wykresowi funkcji f dla wartości parametrów: $\beta_1 = 1$, $\beta_2 = 0,01$ (zgodnie z powyższym oszacowaniem) i $\beta_3 = 1$. Dla tego zestawu parametrów wykres funkcji zadanej równaniem 2.4 jest *bardzo bliski* osi x wszędzie poza wartościami x bliskimi 0 (rysunek 2.5).

```
> expFct <- function(x, beta1, beta2, beta3){
+ exp(-beta1*x)/(beta2+beta3*x)}
> plot(y~x, data=Chwirut2, xlab="Odległość od metalu",
+ ylab="Odbicie ultradźwiękowe",       ylim= c(0,100))
> curve(expFct(x, beta1=1, beta2=0.01, beta3=1), add=TRUE)
```

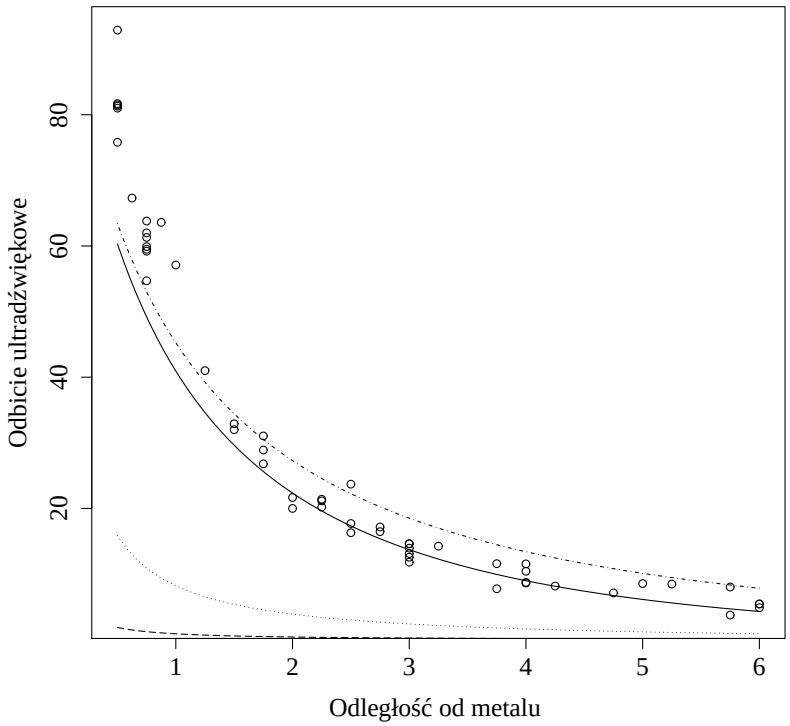


Rysunek 2.5: Wykres funkcji zadanej równaniem 2.4 dla $\beta_1 = 1$, $\beta_2 = 0,01$ i $\beta_3 = 1$.

Ponieważ wybór wartości początkowych $\beta_1 = 1$ i $\beta_3 = 1$ był chybiony, wypróbujmy inne konfiguracje wartości parametrów β_1, β_2 i β_3: $(0,1; 0,01; 1)$, $(0,1; 0,01; 0,1)$, $(0,1; 0,01; 0,01)$ i $(0,2; 0,01; 0,01)$. Przechodząc od jednego układu wartości do drugiego zmieniamy wartość tylko jednego parametru (β_1 lub β_3) – np. przechodząc z konfiguracji drugiej do trzeciej zmieniamy jedynie wartość β_3 z $0,1$ na $0,01$. Wykresy funkcji f dla czterech kolejnych zestawów parametrów przedstawione są na rysunku 2.6.

Na rysunku 2.6 widać jak zmienia się wykres funkcji f w zależności od wartości parametrów β_1, β_2 i β_3. Z rysunku wynika, że wraz ze zmianami wartości parametrów wykres funkcji jest coraz bliższy danym. W praktyce zazwyczaj potrzeba więcej niż tylko kilku prób, by dostać wykres funkcji bliski danym. Ostatni zestaw parametrów *mniej więcej* pasuje do danych, dlatego można ten układ przyjąć jako układ wartości początkowych dla funkcji `nls()`. Aby poznać szczegółowe informacje o opracowanym modelu, posłużymy się metodą `summary`, podobnie jak w sekcji 2.2.2.

```
Chwirut2.m1 <- nls(y ~ expFct(x, beta1, beta2, beta3),
+data=Chwirut2, start=list(beta1=0.2, beta2=0.01, beta3=0.01))
```



Rysunek 2.6: Wykresy funkcji f dla zestawów parametrów: $(0,1; 0,01; 1)$ – kreski, $(0,1; 0,01; 0,1)$ – kropki, $(0,1; 0,01; 0,01)$ – naprzemienne kropki i kreski, $(0,2; 0,01; 0,01)$ – linia ciągła.

```
>summary(Chwirut2.m1)

Formula: y ~ expFct(x, beta1, beta2, beta3)

Parameters:
       Estimate Std. Error t value Pr(>|t|)
beta1 0.1665753  0.0383032   4.349 6.56e-05 ***
beta2 0.0051653  0.0006662   7.753 3.54e-10 ***
beta3 0.0121501  0.0015304   7.939 1.81e-10 ***
---
Signif. codes:  0 ‘’*** 0.001 ‘’** 0.01 ‘’* 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 3.172 on 51 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 4.67e-06
```

Algorytm zatrzymał się po pięciu iteracjach, a w efekcie jego działania otrzymano wartości, które *wyglądają rozsądnie* (są bliskie naszym, znalezionym *metodą prób i błędów* oszacowaniom parametrów, które uznaliśmy za akceptowalne na podstawie rysunku 2.6).

Sprawdźmy jeszcze, co wydarzy się, jeśli za wartości początkowe wybierzemy taki zestaw parametrów, dla którego wykres funkcji f jest daleki od danych, na przykład wartości obrane

przez nas na samym początku: $\beta_1 = 1$, $\beta_2 = 0,01$ i $\beta_3 = 1$.

```
Chwirut2.m1 <- nls(y ~ expFct(x, beta1, beta2, beta3),
+data=Chwirut2, start=list(beta1=1, beta2=0.01, beta3=1))

Error in numericDeriv(form[[3L]], names(ind), env) :
 Missing value or an infinity produced when evaluating the model
```

Dla niepoprawnie wybranych wartości początkowych funkcja `nls()` zamiast oszacowań parametrów zwraca komunikat o błędzie (patrz: sekcja 2.3.5 na stronie 28).

2.3.1.2. Szukanie po siatce

W sytuacji braku jakichkolwiek przesłanek do twierdzenia, że wartości parametrów powinny mieścić się w jakimś przedziale, przydać może się metoda *szukania po siatce* (*grid searching*): analiza resztowej sumy kwadratów (równanie 1.4 na stronie 9) w celu znalezienia takiego zakresu parametrów, dla którego RSS jest minimalne.

Wróćmy do zbioru danych `Chwirut2` i funkcji zadanej równaniem 2.4. Przypomnijmy: oszacowaliśmy, że wartość parametru β_2 wynosi około $0,01$, ale *a priori* niczego nie wiemy o wartościach parametrów β_1 i β_3, poza tym, że są to liczby dodatnie. Możemy poszukiwać ich wartości w dość wąskim przedziale od $0,1$ do 1, zmieniając z kroku na krok wartości parametrów o $0,1$. Innymi słowy, możemy przeszukiwać *siatkę* spełniającą warunki:

1. $0,1 \leq \beta_1 \leq 1$
2. $\beta_2 = 0,1$
3. $0,1 \leq \beta_3 \leq 1$.

Opisane powyżej przedziały dla β_1 i β_3 możemy otrzymać, generując dwa wektory zawierające wartości parametrów za pomocą funkcji `seq()`. Funkcja `seq()` umożliwia generowanie sekwencji [4] liczb zapisywanej jako wektor. Funkcja `expand.grid` natomiast pozwala na utworzenie iloczynu kartezjańskiego wektorów, które wskaże się jako jej argumenty. Aby otrzymać siatkę zawierającą liczby od $0,1$ do $1,0$ z rozstępem $0,1$ dla wymiarów odpowiadających parametrom β_1 i β_3 postąpimy następująco:

```
> grid.Chwirut2 <- expand.grid(list(beta1= seq(0.1, 1, by=0.1),
+beta2=c(0.01), beta3= seq(0.1, 1, by=0.1)))
```

Otrzymana siatka `grid.Chwirut2` zawiera trzy kolumny o nazwach odpowiadających nazwom parametrów. Każdemu możliwemu układowi parametrów przypisany jest jeden wiersz w `grid.Chwirut2`, więc wszystkich wierszy jest 100 (10 możliwych wartości parametru β_1 dla każdej z 10 możliwych wartości parametru β_3).

Do przeanalizowania wartości RSS dla zestawów parametrów z `grid.Chwirut2` można wykorzystać funkcję `nls2()` z pakietu o tej samej nazwie ([7]). Specyfikacja funkcji `nls2()` jest bardzo zbliżona do specyfikacji funkcji `nls()`. Podstawową różnicą są argumenty `start` i `method`. Dla funkcji `nls2()` argument `start` może przyjmować wartość zarówno *ramki danych* (`data frame`), jak i nawet modelu opracowanego wcześniej przez funkcję `nls2()`. Wartość `"brute-force"` argumentu `method` sprawia, że RSS analizowane jest dla wartości parametrów określonych przez argument `start`.

[4] Za sekwencję uznaje się tu ciąg arytmetyczny.

```
> Chwirut2.m2a <- nls2(y ~ expFct(x, beta1, beta2, beta3),
+ data=Chwirut2, start=grid.Chwirut2, algorithm="brute-force")
```

Wyniki działania funkcji `nls2()` zawiera minimalną wartość RSS oraz zestaw parametrów, dla którego to minimum zostało osiągnięte. W przypadku naszych danych minimum RSS dla kombinacji parametrów określonych w `grid.Chwirut2` wynosi 60696. Osiągane jest dla układu parametrów: $(0,1; 0,01; 0,1)$.

```
> Chwirut2.m2a

Nonlinear regression model
  model:  y ~ expFct(x, beta1, beta2, beta3)
   data:  NULL
beta1 beta2 beta3
 0.10  0.01  0.10
 residual sum-of-squares: 60696

Number of iterations to convergence: 100
Achieved convergence tolerance: NA
```

Wartości parametrów β_1, β_2 i β_3 otrzymane w rezultacie działania funkcji `nls2()` mogą być wykorzystane jako wartości początkowe dla zwyczajnej funkcji `nls()`. Otrzymany w efekcie ich użycia model niewiele odbiega od modelu wygenerowanego w poprzedniej części.

```
> Chwirut2.m2 <- nls(y ~ expFct(x, beta1, beta2, beta3),
+ data=Chwirut2, start=list(beta1=0.1, beta2=0.01, beta3=0.1))

> Chwirut2.m2

Nonlinear regression model
  model:  y ~ expFct(x, beta1, beta2, beta3)
   data:  Chwirut2
   beta1    beta2    beta3
0.166578 0.005165 0.012150
 residual sum-of-squares: 513

Number of iterations to convergence: 6
Achieved convergence tolerance: 4.471e-06
```

2.3.1.3. Użycie funkcji samostartujących

W wypadku wielokrotnego stosowania tego samego modelu regresji nieliniowej nieoceniona jest możliwość automatycznego wybierania wartości początkowych. Taką możliwość dają *funkcje samostartujące* (*self-starter functions*). Funkcje te są specyficzne dla każdego modelu, a przy ich pomocy można określić zestaw wartości początkowych dla zadanego zbioru danych. Dla wybranego modelu regresji nieliniowej można skonstruować bardzo wiele rozmaitych funkcji samostartujących. Wykorzystanie wyliczonych przez nie wartości początkowych nie musi skutkować dobrym dopasowaniem modelu, ale odpowiednio przygotowane funkcje samostartujące zazwyczaj zwracają wartości wystarczająco bliskie oszacowaniom parametrów, by algorytm szukania dokładnych wartości parametrów był przy ich wykorzystaniu zbieżny.

Istnieje kilka sposobów znajdowania funkcji samostartujących w **R**. Przy standardowo

zainstalowanym programie **R** instalowana jest także biblioteka podstawowych funkcji samostartujących. Innym miejscem, w którym znaleźć można zestaw funkcji samostartujących jest pakiet `HydroMe` ([20], funkcje z tego pakietu związane są przede wszystkim z modelami regresji nieliniowej wykorzystywanymi do estymacji pewnych hydraulicznych parametrów w modelach używanych przez gleboznawców). Funkcje tego typu mogą być także częściami innych pakietów. Na przykład pakiet `NRAIA` ([1]) zawiera funkcję samostartującą dla modelu Richardsa[5]. W dalszej części pracy uwaga jest skupiona na funkcjach samostartujących zawartych w standardowej wersji **R**. Przedstawiony jest także sposób, w jaki można samodzielnie skonstruować funkcję samostartującą.

Funkcja `nls()` związana jest z kilkoma funkcjami samostartującymi. Aby przedstawić ich działanie wróćmy do przykładu związanego z pakietem danych `L.minor` oraz modelem Michaelisa-Menten. W tym przykładzie parametry w modelu Michaelisa-Menten zostały wyznaczone przy użyciu funkcji `nls()` i ręcznie zadanych parametrów. Spróbujmy teraz ponownie dopasować model do danych, tym razem jednak, posługując się funkcją samostartującą `SSmicmen()`. Jest to funkcja służąca do znajdowania wartości parametrów w modelu Michaelisa-Menten bez konieczności wyboru wartości początkowych.

```
L.minor.m2 <- nls(rate ~ SSmicmen(conc, Vm, K), data=L.minor)
```

Pierwszym argumentem ponownie jest wzór modelu. Tym razem tylda służy do oddzielenia zmiennej wyjaśnianej `rate` od wyjaśniającej `conc`, która jest jednym z argumentów funkcji samostartującej `SSmicmen()`. Pozostałymi argumentami funkcji `SSmicmen()` są parametry równania Michaelisa-Menten V_m i K. Przypomnijmy, że model Michealisa-Menten opisuje równanie 2.2 zamieszczone na stronie 15. Dla działania funkcji oczywiście istotna jest kolejność zadawania parametrów. Drugim (po równaniu zadanym funkcją samostartującą) argumentem funkcji `nls()` jest zbiór danych, zawierający zmienne wyjaśnianą i wyjaśniającą. Ponieważ użyta została funkcja samostartująca, nie było konieczności zadawania początkowych wartości parametrów przy pomocy argumentu `start`. Model otrzymany przy wykorzystaniu funkcji samostartującej jest równie dobrze dopasowany do danych, co model oszacowany przy wyborze wartości początkowych (`L.minor.m1`).

```
> summary(L.minor.m2)

Formula: rate ~ SSmicmen(conc, Vm, K)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
Vm  126.033      7.173  17.570 2.18e-06 ***
K    17.079      2.953   5.784  0.00117 **
---
Signif. codes:  0 ‘’*** 0.001 ‘’** 0.01 ‘’* 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 6.25 on 6 degrees of freedom

Number of iterations to convergence: 0
Achieved convergence tolerance: 2.478e-06
```

W **R** możliwe jest także samodzielne przygotowanie funkcji samostartujących. W tym celu można posłużyć się funkcją `selfStart()`. Sama procedura opracowania funkcji samostartujących jest jednak dość praco- i czasochłonna, co więcej nie będzie wykorzystywana

[5]Jest to model wzrostu zadany równaniem $y = f(x, (b, d, e, f)) = d \cdot \left[1 + exp(\frac{e-x}{b})\right]^{-exp(-f)}$.

w przykładzie zastosowań funkcji `nls()` w ostatnim rozdziale pracy. Dlatego zdecydowano nie przytaczać jej tutaj. Przykłady ręcznego opracowania funkcji samostartujących są opisane szczegółowo np. w [17].

2.3.2. Określanie wartości pochodnych

Minimalizacja RSS przy użyciu algorytmu Gaussa-Newtona polega na liniowej aproksymacji funkcji zadanej w modelu przy użyciu pierwszej pochodnej tej funkcji. Jeśli nie posiadamy informacji o pochodnej funkcji, jest ona obliczana numerycznie przy pomocy funkcji `numericDeriv()`. Niekiedy podanie jawnego wzoru na pierwszą pochodną może przynieść korzyści w postaci szybszej zbieżności algorytmu. Taka sytuacja jest szczególnie korzystna, kiedy ten sam model dopasowuje się do wielu zbiorów danych, ponieważ znacząco oszczędza czas potrzebny do przeprowadzenia obliczeń. W dalszej części pracy przedstawione są dwie możliwe drogi zadawania wzoru na pochodną w omówionym już modelu Michaelisa-Menten.

2.3.2.1. Ręczne zadawanie wartości pochodnych

Po zróżniczkowaniu funkcji w modelu Michaelisa-Menten zadanej równaniem 2.1 względem obu parametrów, otrzymujemy następujące wzory:

$$\frac{\partial f}{\partial K} = \frac{-V_m \cdot x}{(K+x)^2}, \tag{2.5}$$

$$\frac{\partial f}{\partial V_m} = \frac{x}{K+x}. \tag{2.6}$$

Znając wzory 2.5 i 2.6, możemy określić nową funkcję opisującą model Michaelisa-Mentena, tym razem wzbogaconą o wzory na pochodne cząstkowe. Funkcję tę nazwiemy `MMfunction1` i wykorzystamy dalej jako argument funkcji `nls()`.

```
> MMfunction1<-function(conc, K, Vm){
+ numer <- Vm*conc
+ denom <- K+conc
+ mean <- numer/denom
+ partialK<- -numer/(denom)^2
+ partialVm<- mean/Vm
+ attr(mean, "gradient") <- cbind(partialK,partialVm)
+ return(mean)}
```

Pierwsze trzy linijki pomiędzy nawiasami klamrowymi określają funkcję z modelu Michaelisa-Mentena (kolejno określono wartość licznika, mianownika oraz ich ilorazu). Następne linijki definiują pochodne zgodnie z równaniami 2.5 i 2.6 (`partialK` oraz `pratialVm`). W szóstej linii tworzony jest wektor złożony z wartości funkcji z modelu Michaelisa-Menten oraz z obliczonych wartości pochodnych cząstkowych. W ten sposób do funkcji zadającej równanie modelu, wykorzystywanej jako argument w funkcji `nls()`, można dołączyć informacje o wartościach pochodnych cząstkowych.

Sprawdźmy teraz, jak dobrze dopasowany do danych będzie model uzyskany w funkcji `nls()` przy wykorzystaniu informacji o wartościach pochodnych. Jak widać poniżej, podając wartości pochodnych cząstkowych, otrzymano model równie dobrze dopasowany do danych, jak model określony wcześniej – `L.minor.m1` (strona 15). Otrzymano także identyczne jak w modelu `L.minor.m1` wartości oszacowań parametrów.

```
> L.minor.mdv1 <- nls(rate ~MMfunction1(conc, K, Vm),
+ data=L.minor, start=list(K=20, Vm=120))
> summary(L.minor.mdv1)

Formula: rate ~ MMfunction1(conc, K, Vm)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
K    17.079      2.953   5.784  0.00117 **
Vm  126.033      7.173  17.570 2.18e-06 ***
---
Signif. codes:  0 ‘’*** 0.001 ‘’** 0.01 ‘’* 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 6.25 on 6 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 8.148e-06
```

2.3.2.2. Automatyczne zadawanie wartości pochodnych

Innym sposobem zadawania wartości pochodnych cząstkowych jest użycie funkcji `deriv()`, służącej do różniczkowania wyrażeń. Funkcja ta wykorzystuje trzy argumenty: funkcję, która ma być zróżniczkowana, nazwy zmiennych oraz pustą funkcję, która w toku działania funkcji `deriv()` zostanie uzupełniona o wyrażenia określające pochodne cząstkowe. Dla modelu Michealisa-Menten wywołanie funkcji `deriv()` wygląda następująco:

```
> MMfunction2 <- deriv(~Vm*conc/(K+conc),
+ c("K","Vm"), function(conc, K, Vm){})
```

Wykorzystując funkcję `MMfunction2()` jako argument funkcji `nls()`, identycznie, jak wcześniej wykorzystano funkcję `MMfunction1()`, otrzymać można takie samo oszacowanie parametrów, jak powyżej.

Z przedstawionego tu toku rozumowania nie wynikają żadne bezpośrednie korzyści z zastosowania wzorów na pochodne cząstkowe. Przynajmniej częściowo jest to spowodowane rozmiarem danych – pakiet `L.minor` jest stosunkowo niewielki. Jednak przy pracy z większymi pakietami jawne podanie wzoru na pochodne cząstkowe może istotnie zmniejszyć liczbę iteracji.

2.3.3. Parametry warunkowo liniowe

Często funkcja opisująca model jest nieliniowa względem tylko części używanych parametrów. Innymi słowy, funkcja zadająca model może być liniowa względem części swoich parametrów (oczywiście, nie wszystkich!). Parametry, względem których funkcja określająca model jest liniowa, nazywamy *warunkowo liniowymi*. Przypomnijmy model Michaelisa-Menten (równanie 2.2, strona 15). W tym modelu funkcja f jest liniowa względem parametru V_m, a nieliniowa tylko względem parametru K. Prawa strona równania 2.2 może być zapisana tak, by podkreślić liniowość funkcji względem parametru V_m:

$$\frac{x}{K+x} \cdot V_m. \tag{2.7}$$

Estymacja parametrów w sytuacji, kiedy część z nich jest warunkowo liniowa, może przebiegać sprawniej niż w przypadku, w których funkcja zdająca model jest nieliniowa względem wszystkich parametrów. Dzieje się tak, ponieważ chcąc oszacować wartości parametrów warunkowo liniowych, przy ustalonych wartościach parametrów, które nie są warunkowo liniowe, można posłużyć się zwyczajną regresją liniową. Używając funkcji `nls()` do oszacowania wartości takich parametrów należy posłużyć się algorytmem `plinear`, który wspominano już w tabeli 2.1 na stronie 12.

Algorytm `plinear` wymusza użycie nieco innej niż we wcześniejszych przykładach formuły modelu jako argumentu funkcji `nls()`. Mianowicie dla każdego z warunkowo liniowych parametrów należy określić funkcję, usuwając z jej formy część odpowiadającą temu parametrowi. Na przykład dla funkcji zadanej równaniem 2.2 definicja modelu będzie zawierała jedynie formułę $\frac{x}{K+x}$, czyli to, co zostaje z funkcji po usunięciu części odpowiadającej jedynemu parametrowi warunkowo liniowego – V_m. Wywołanie funkcji `nls()` wykorzystującej algorytm `plinear` wygląda następująco:

```
> L.minor.m3<-nls(rate ~ conc/(K+conc),
+ data=L.minor, algorithm="plinear",
+ start=list(K=20))
```

```
> summary(L.minor.m3)

Formula: rate ~ conc/(K + conc)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
K      17.079      2.953   5.784  0.00117 **
.lin  126.033      7.173  17.570 2.18e-06 ***
---
Signif. codes:  0 ‘’*** 0.001 ‘’** 0.01 ‘’* 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 6.25 on 6 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 2.143e-06
```

Model otrzymany tą drogą jest równie dobrze dopasowany do danych, co model `L.minor.m1` otrzymany w wyniku zastosowania funkcji `nls()` z algorytmem domyślnym, nie zakładającym obecności parametrów warunkowo liniowych. Jedynie podsumowanie modelu różni się nieznacznie, ponieważ podczas tworzenia modelu `L.minor.m3` nie została arbitralnie nadana żadna nazwa parametrowi warunkowo liniowemu, został on w podsumowaniu opisany jako `.lin`.

Więcej przykładów dotyczących użycia algorytmu `plinear`, także w modelach o więcej niż jednym parametrze warunkowo liniowym (w takiej sytuacji nieco inaczej zadaje się formułę funkcji określającej model do argumentów funkcji `nls()`) przedstawiono w [21].

2.3.4. Monitorowanie funkcji `nls()`

Jak wspomniano w sekcji 2.1.1, funkcja `nls()` posiada argument `control` służący do monitorowania przebiegu działania algorytmu w ramach funkcji `nls()`. Funkcja `nls.control()` umożliwia wygodne przyjrzenie się wartościom argumentów kontrolnych – aby sprawdzić ich wartości wystarczy wpisać `nls.control()` do linii poleceń.

```
> nls.control()
$maxiter
[1] 50

$tol
[1] 1e-05

$minFactor
[1] 0.0009765625

$printEval
[1] FALSE

$warnOnly
[1] FALSE
```

Podstawowe informacje dotyczące działania funkcji `nls()`, które można otrzymać w wyniku działania funkcji `nls.control()`, wymienione są w tabeli poniżej wraz z krótkimi opisami. Więcej szczegółów na ten temat można znaleźć na stronie pomocy, wpisując do linii poleceń programu **R** `?nls.control`, czyli otwarcie pomocy do funkcji `nls.control`.

Tabela 2.4: Informacje o ustawieniach funkcji `nls()`.

`maxiter`	Maksymalna liczba iteracji, domyślnie jest to 50.
`data`	Poziom tolerancji zbieżności algorytmu. Im niższa jest ta wartość (domyślnie $0,00001$), tym ostrzejsze kryteria muszą spełniać wartości w kolejnych iteracjach, by można było zakończyć pracę algorytmu.
`tol`	Określenie funkcji w wykorzystywanym modelu.
`minFactor`	Minimalna wartość kroku w iteracjach (więcej informacji w sekcji 2.3.5). Domyślnie jest to $\frac{1}{1024}$, czyli $0,0009765625$.
`printEval`	Informacja o tym, czy wyliczone wartości pochodnych powinny być raportowane w każdej iteracji czy nie.
`warnOnly`	Informacja o tym, czy funkcja `nls()` powinna zgłaszać błędy podczas działania czy nie. Jeśli ustawiona jest wartość `FALSE` (wartość domyślna), funkcja `nls()`zatrzyma się natychmiast po wykryciu błędu, zgłaszając komunikat o błędzie. Jeśli ustawiona jest wartość `TRUE`, funkcja `nls()` zgłosi informację o błędzie i zwróci niepełny model (zamiast zatrzymania się). Modelu zwróconego w tym przypadku nie można użyć do opisu danych albo prognozowania wartości zmiennej zależnej, jednak może być on przydatny do stwierdzenia, co spowodowało wystąpienie błędu.

2.3.5. Komunikaty błędów

Korzystając z funkcji `nls()`, niekiedy można natknąć się na komunikaty o błędach. Chcąc wyeliminować zaistniałe błędy, należy dobrze rozumieć wiadomości wysyłane przez funkcję `nls()`. Poniżej opisane są najczęstsze komunikaty błędów zwracane przez funkcję `nls()`. Dotyczą one błędów w zbieżności algorytmu, wynikających bądź z nieprawidłowo określonych ustawień funkcji `nls()` (patrz: sekcja 2.3.4), bądź nieprawidłowo dobranych wartości

początkowych (patrz: sekcja 2.3.1).

```
Error in nls(... :
 number of iterations exceeded maximum of 50
```

```
Error in nls(... :
 step factor 0.000488281 reduced below 'minFactor' of 0.0009765625
```

Dwa powyższe komunikaty odnoszą się do przekroczenia przez algorytm pewnych aktualnie ustalonych limitów. W pierwszym wypadku liczba iteracji przewyższa ustalone maksimum, którego domyślną wartością jest 50 (patrz: sekcja 2.3.4). Takie ustalenie wartości oznacza, że niezależnie od tego, czy algorytm osiągnął zbieżność czy nie, po 50 iteracjach jego praca zostanie przerwana. Niekiedy zwiększenie tej wartości może pomóc w osiąganiu zbieżności i estymowaniu wartości parametrów w modelu. Aby sprawdzić, czy w danej sytuacji mamy do czynienia z takim przypadkiem, należy użyć argumentu `trace=TRUE`, zadając funkcję `nls()` (podobnie jak zrobione było to w sekcji 2.2.2) i ponownie ją uruchomić. Następnie należy na podstawie wyników działania `trace` ocenić, czy oszacowania parametrów zmieniają się z kroku na krok dążąc do jakichś stabilnych wartości. Jeśli tak – przydatne może być zmienienie maksimum iteracji.

Drugi komunikat błędu odnosi się do kroku w iteracjach, czyli wartości jaką należy dodać do wcześniej otrzymanej wartości parametru podczas kolejnej iteracji algorytmu (wiąże się to z użyciem algorytmu Gaussa-Netwona). Wartością domyślną, jak wspomniano w 2.3.4 jest $\frac{1}{1024}$, ale niekiedy opłaca się wartość tę zmniejszyć, aby uzyskać zbieżność algorytmu.

```
Error in nls(... :
 singular gradient
```

```
Error in numericDeruv)form[[3]], names(ind), env) :
 Missing value or an infinity produced when evaluating the model
```

```
Error in nlsModel(formula, mf, start, wts):
 singular gradient matrix at initial parameter estimates
```

Powyższe trzy błędy wiążą się z nieprawidłowo dobranymi wartościami początkowymi. Aby wyeliminować błędy w takiej sytuacji można wybrać inne wartości początkowe (więcej w sekcji 2.3.1), wypróbować działanie innego algorytmu przez zmianę wartości argumentu funkcji `nls()` (np. algorytm `port` może w danym przypadku być skuteczniejszy niż algorytm Gaussa-Newtona) albo zmienić funkcję opisującą model.

Rozdział 3

Analiza rzeczywistych danych

3.1. Wprowadzenie

Rozdział 3 poświęcony jest analizie danych rzeczywistych dotyczących palenia papierosów przez dorosłych Polaków w zależności od ich wieku. Na świecie palenie papierosów powoduje śmierć około $5,5$ miliona osób rocznie ([23]). Używanie wyrobów tytoniowych może skutkować chorobami takim, jak: nowotwory płuc, pęcherza moczowego i żołądka, udary mózgu, wrzody żołądka i dwunastnicy, miażdżyca, ślepota, bezpłodność czy zaburzenia rytmu serca ([23]). Coraz częściej organizowane są poruszające kampanie antynikotynowe ([11]), a w ostatnim czasie w Unii Europejskiej dyskutowany jest pomysł wprowadzenia jednolitych opakowań papierosów ([19]), mających na celu zniechęcenie palaczy do kupowania wyrobów tytoniowych.

Jednak planując kampanię społeczną zniechęcającą do używania papierosów, należy zastanowić się nad grupą, do której kampania ma być skierowana. Aby wybrać grupę docelową kampanii trzeba poznać prawidłowości dotyczące odsetka osób palących w różnych grupach wiekowych. Dalsza część pracy poświęcona jest opisaniu tej zależności. W sekcji 3.2 przedstawione są dane dotyczące udziału osób palących w populacji Polaków w różnych grupach wiekowych. Sekcje 3.3 oraz 3.4 przedstawiają odpowiednio liniowy i nieliniowy ze względu na parametry model regresji opisujący zależność pomiędzy wiekiem członków danej grupy a udziałem wśród nich osób palących.

3.2. Wykorzystane dane

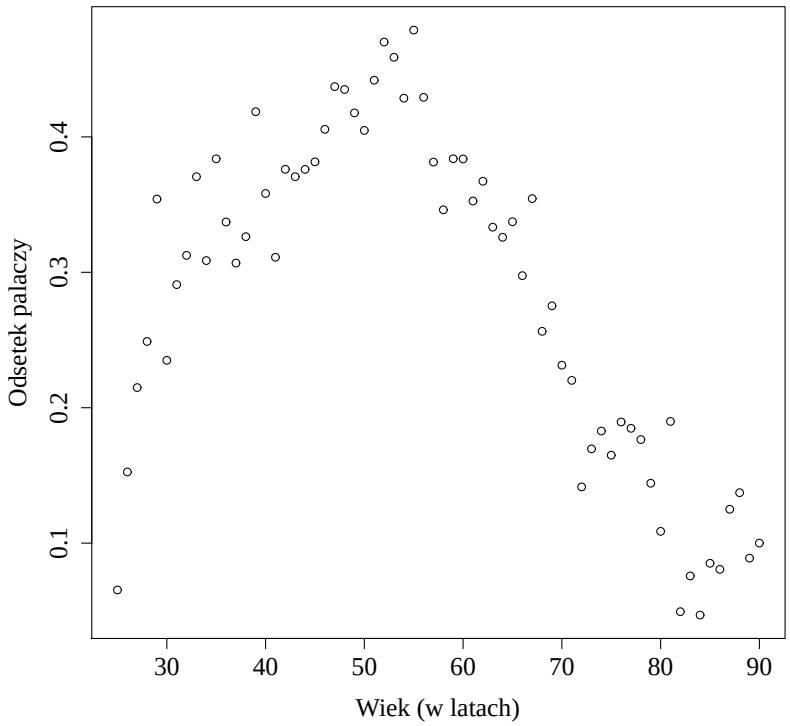
W pracy wykorzystano fragment danych opublikowanych w ramach Diagnozy Społecznej 2011 ([14]). Diagnoza Społeczna jest badaniem kwestionariuszowym przeprowadzanym na terenie Polski, mającym na celu określenie warunków życia Polaków. Badanie przeprowadzane jest od roku 2000 co dwa lub trzy lata na reprezentatywnej próbie kilkudziesięciu tysięcy Polaków. W każdej edycji badania biorą udział te same gospodarstwa i osoby, co kilka lat wcześniej. Celem przeprowadzania Diagnozy Społecznej jest opisanie funkcjonowania ekonomicznego, emocjonalnego i zdrowotnego Polaków.

W dalszej części pracy wykorzystana zostanie część danych pochodzących z Diagnozy Społecznej 2011. Będą to dane dotyczące zależności pomiędzy wiekiem a odsetkiem osób palących papierosy. Pierwsza z analizowanych zmiennych pochodzi bezpośrednio z bazy danych części Diagnozy Społecznej przeznaczonej dla indywidualnych respondentów. Druga zmienna – odsetek osób palących papierosy w danej grupie wiekowej – została opracowana na podstawie odpowiedzi uczestników badania na pytanie o palenie papierosów i informacji o ich wieku. Dane przedstawione są w tabeli 3.1 na stronie 32 oraz na wykresie 3.1 na stronie 33.

Tabela 3.1: Dane dotyczące udziału palaczy w populacji w różnych grupach wiekowych.

wiek	udział palaczy	wiek	udział palaczy
25	0,07	58	0,35
26	0,15	59	0,38
27	0,21	60	0,38
28	0,25	61	0,35
29	0,35	62	0,37
30	0,23	63	0,33
31	0,29	64	0,33
32	0,31	65	0,34
33	0,37	66	0,30
34	0,31	67	0,35
35	0,38	68	0,26
36	0,34	69	0,28
37	0,31	70	0,23
38	0,33	71	0,22
39	0,42	72	0,14
40	0,36	73	0,17
41	0,31	74	0,18
42	0,38	75	0,16
43	0,37	76	0,19
44	0,38	77	0,18
45	0,38	78	0,18
46	0,41	79	0,14
47	0,44	80	0,11
48	0,44	81	0,19
49	0,42	82	0,05
50	0,40	83	0,08
51	0,44	84	0,05
52	0,47	85	0,09
53	0,46	86	0,08
54	0,43	87	0,12
55	0,48	88	0,14
56	0,43	89	0,09
57	0,38	90	0,10

Jak widać na rysunku 3.1 odsetek osób palących w populacji zależy od rozpatrywanej grupy wiekowej, a zależność ta jest nieliniowa. W najmłodszej rozpatrywanej grupie do palenia papierosów przyznaje się 7% respondentów. Odsetek ten wzrasta aż do około 40% u osób w wieku około 50 lat, by potem zmaleć do około 10% w grupie najstarszej – u osób mających około 90 lat.



Rysunek 3.1: Dane dotyczące udziału palaczy w populacji w różnych grupach wiekowych.

3.3. Model liniowy ze względu na parametry

W tej części rozdziału do danych przedstawione w sekcji 3.2 dopasowywany jest model liniowy ze względu na parametry. Parametry w tym modelu są dopasowane przy użyciu wcześniej opisanej metody – funkcji `nls()`. W dalszej części rozdziału przedstawiany tutaj model jest porównywany z modelem nieliniowym ze względu na parametry.

3.3.1. Wybór funkcji

Aby zastosować analizę regresji na danych przedstawionych w sekcji 3.2, należy na początku zdecydować, do jakiego modelu planujemy znaleźć wartości parametrów. Jak wspomniano w rozdziale 1 do opracowania modelu regresji nieliniowej potrzebny jest znany wzór funkcji, dla której dobierać się będzie parametry. Często wzór funkcji wynika z teorii, którą ilustruje się eksperymentem. Jednak w przypadku danych przedstawionych w sekcji 3.2 nieznana jest żadna teoria. Z tego powodu funkcja określająca zależność pomiędzy wiekiem a odsetkiem palaczy zostanie wybrana na podstawie danych pochodzących z badania.

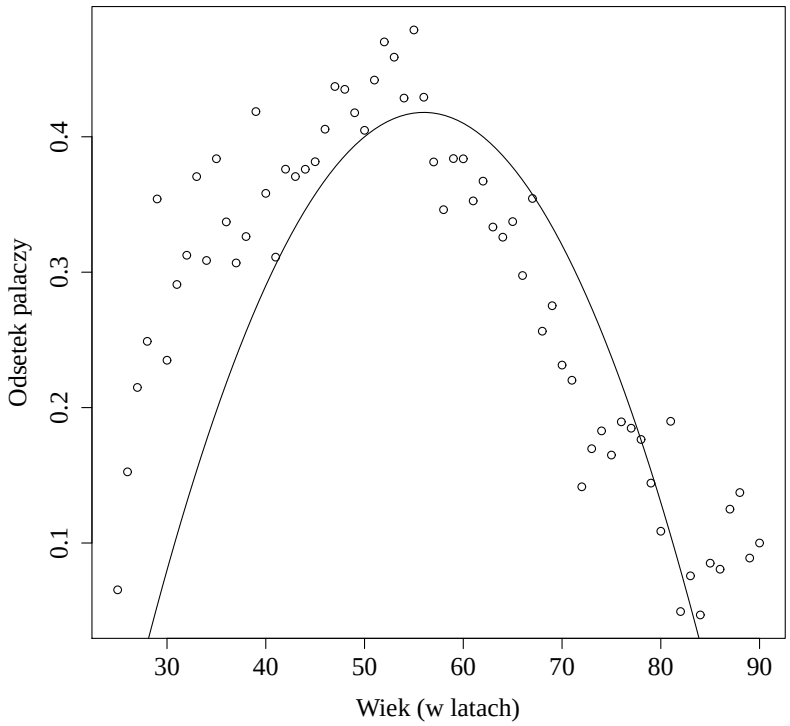
Spróbujmy odczytać informacje o zależności pomiędzy wiekiem a odsetkiem palaczy z rysunku 3.1 na stronie 33. Widać na nim, że mamy do czynienia z funkcją kawałkami monotoniczną – dla niższych wartości zmiennej niezależnej rosnącą, dla wyższych – malejącą, ponadto wykres jest niemal symetryczny. Z tych powodów dobrym pomysłem może być rozważenie funkcji kwadratowej zadanej wzorem:

$$O = a \cdot W^2 + b \cdot W + c, \tag{3.1}$$

gdzie W oznacza wiek wyrażony w latach, a O to odsetek palaczy w grupie wiekowej.

Rysunek 3.2 przestawia przykładową funkcję kwadratową naniesioną na dane z rysunku 3.1. Wykorzystana funkcja jest zadana wzorem:

$$y = (-0,0005) \cdot x^2 + 0,056 \cdot x - 1,15. \tag{3.2}$$



Rysunek 3.2: Funkcja kwadratowa a wykorzystywane dane.

Z rysunku 3.2 widać, że wybrana funkcja dość dobrze oddaje charakter zależności. Oczywiście wstępnie dobrane parametry sprawiają, że wykres funkcji nie pokrywa się z wykresem danych, ale ma odpowiedni kształt. Aby znaleźć odpowiednie wartości parametrów posłużymy się opisaną metodą w rozdziale 2.

3.3.2. Wybór wartości początkowych

Chcąc wykorzystać metodą opisywaną w rozdziale 2, musimy poza wzorem funkcji wybrać także wartości początkowe estymowanych parametrów. Posłużymy się w tym celu metodą analizy graficznej, opisaną w sekcji 2.3.1.1. Przeanalizujemy współrzędne wierzchołka szukanej funkcji.

Wiadomo, że wierzchołek funkcji kwadratowej ma współrzędne $(p; q)$ określone wzorami:

$$\begin{cases} p &= \frac{-b}{2a} \\ q &= \frac{4ac-b^2}{4a} \end{cases}$$

gdzie a,b i c określone są jak powyżej w równaniu 3.1.

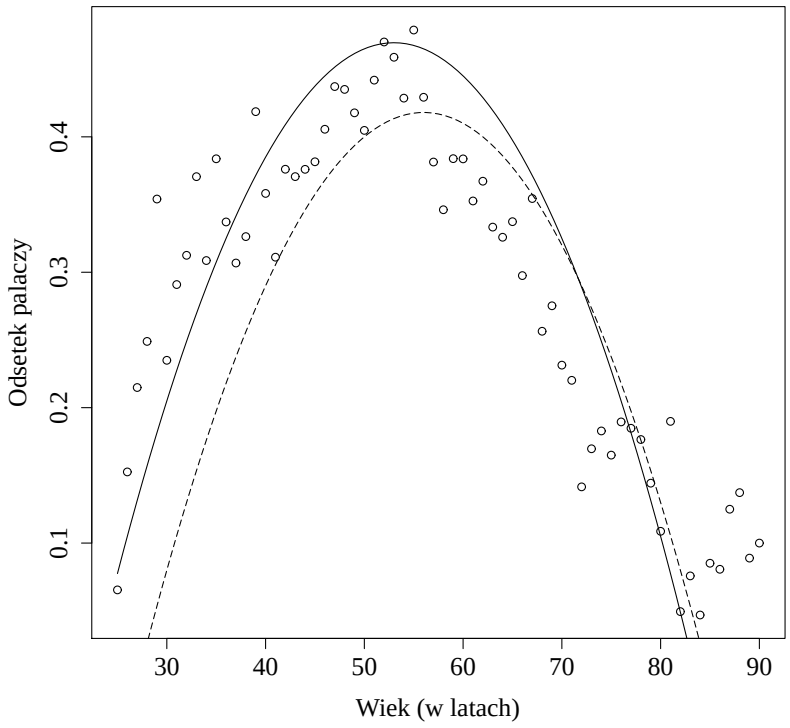
Przyglądając się ponownie rysunkowi 3.2, możemy stwierdzić, że wierzchołek funkcji powinien mieć współrzędne w przybliżeniu równe $(53; 0,47)$, stąd:

$$\begin{cases} \frac{-b}{2a} & = & 53 \\ \frac{4ac-b^2}{4a} & = & 0,47 \end{cases}$$

Zatem parametry b i c można w następujący sposób zapisać jako zależne od parametru a i współrzędnych wierzchołka funkcji kwadratowej:

Przyjmując za a wartość $0,0005$, jak w równaniu 3.2 oraz wartości współrzędnych $(53; 0,47)$, otrzymujemy wartości parametrów $b = 0,053$ i $c = -0,935$. Zatem funkcja opisująca model, przy wybranych wartościach parametrów, jest określona wzorem:

$$O = (-0,0005) \cdot W^2 + 0,053 \cdot W - 0,935. \tag{3.3}$$



Rysunek 3.3: Funkcje kwadratowe a wykorzystywane dane. Linia przerywana: funkcja zadana wzorem 3.2, linia ciągła: funkcja zadana wzorem 3.3.

Rysunek 3.3 przedstawia wykorzystywane dane oraz wykresy dwóch funkcji określonych równaniami 3.2 (linia przerywana) i 3.3 (linia ciągła). Z tego rysunku widać, że nowa funkcja jest bliższa danym niż funkcja wybrana wcześniej. Ponieważ otrzymany wykres funkcji jest dość bliski wykresowi danych, wykorzystane wartości parametrów potraktujemy jako wartości początkowe dla funkcji `nls()`.

3.3.3. Wykorzystanie funkcji `nls()`

Mając wybrane wartości początkowe parametrów, możemy rozpocząć pracę z funkcją `nls()`. Wykorzystamy trzy argumenty funkcji `nls()`: `formula`, `start` oraz `trace`, tak samo jak

w przykładzie w sekcji 2.2.2 na stronie 13. Poniższy fragment kodu ukazuje użycie funkcji `nls()` wraz z odpowiedzią programu (pokazywaną dzięki zastosowaniu wartości `TRUE` argumentu `trace`), zawierająca informacje o wartościach parametrów i wartościach RSS w kolejnych krokach. Widać, że została przeprowadzona tylko jedna iteracja algorytmu Gaussa-Newtona. Kolejne znajdywane wartości były już wystarczająco bliskie znalezionym w pierwszym kroku, żeby zakończyć działanie algorytmu.

```
> palenie.m1 <- nls (odsetek.palaczy ~
+  a * wiek^2 + b * wiek + c,
+ start = list( a = -0.0005, b = 0.053, c = -.935),
+ trace = TRUE)

0.6057796 :  -0.0005  0.0530 -0.9350
0.19711 :  -0.0002636424  0.0267205279 -0.2873351203
```

```
> palenie.m1

Nonlinear regression model
  model:  odsetek.palaczy ~ a * wiek^2 + b * wiek + c
   data:  parent.frame()
         a          b          c
-0.0002636  0.0267205 -0.2873351
 residual sum-of-squares: 0.1971

Number of iterations to convergence: 1
Achieved convergence tolerance: 1.872e-07
```

Na podstawie wyżej przytoczonych fragmentów kodu i odpowiedzi programu **R** określić można zestaw estymatorów parametrów, dla których wartość RSS była najmniejsza. Ten zestaw to:

$$\begin{cases} a &= -0,00026 \\ b &= 0,027 \\ c &= -0,29 \end{cases}$$

A zatem zależność pomiędzy wiekiem a odsetkiem osób palących papierosy opisuje równanie:

$$O \approx -0,00026 \cdot W^2 + 0,027 \cdot W - 0,29. \tag{3.4}$$

Wykres 3.4 na stronie 37 przedstawia wykorzystane dane wraz z krzywą opisaną równaniem 3.4.

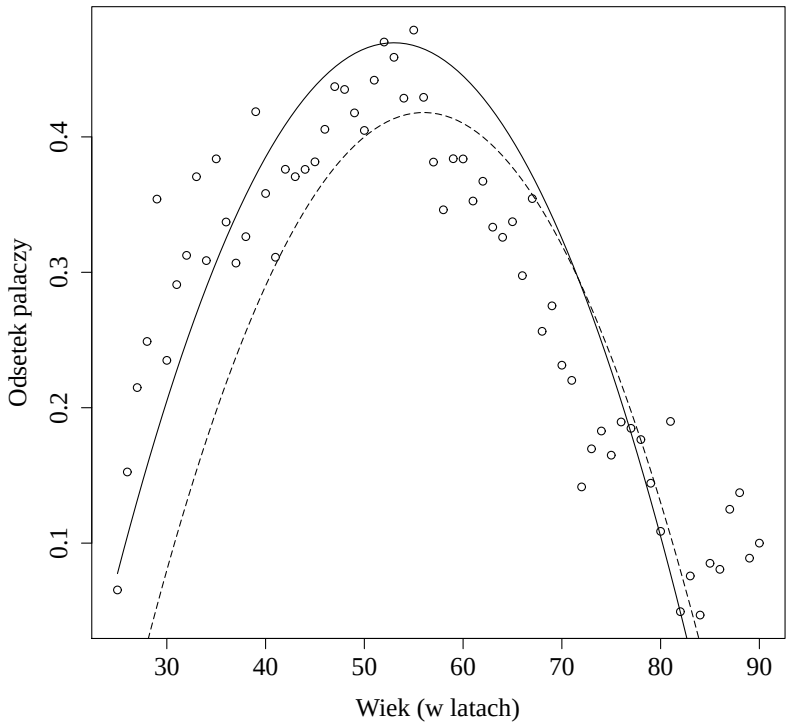
3.4. Model nieliniowy ze względu na parametry

3.4.1. Proponowany model

Rozważmy krzywą zadaną równaniem:

$$\Phi_{a,\mu,\sigma}(x) = \frac{a}{\sigma\sqrt{2\pi}} exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{3.5}$$

Wzór 3.5 jest wzorem dystrybuanty rozkładu normalnego, przemnożonej przez współczynnik a. Funkcja $\Phi_{a,\mu,\sigma}$ zadana równaniem 3.5 zachowuje część właściwości dystrybuanty



Rysunek 3.4: Wykorzystane dane i dopasowana do nich krzywa.

rozkładu normalnego – np. jest symetryczna względem prostej wyznaczonej przez wartość μ. Z uwagi na wprowadzenie parametru a zmienia się jednak pole pod krzywą.

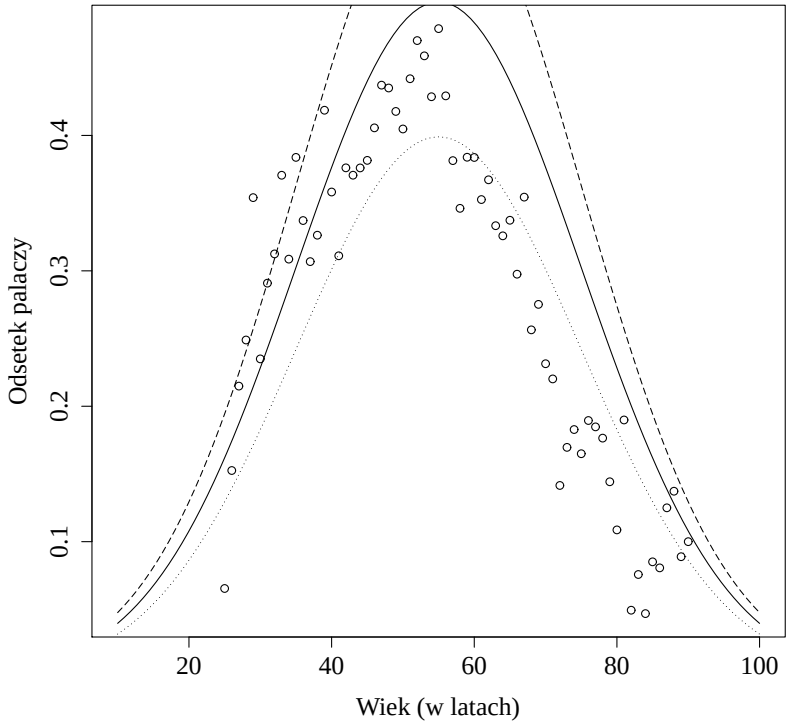
Spróbujmy zastosować tę krzywą do oceny udziału osób palących w populacji w różnych grupach wiekowych. Równanie 3.5 przybiera wówczas postać:

$$O = \frac{a}{\sigma\sqrt{2\pi}} exp\left(\frac{-(W-\mu)^2}{2\sigma^2}\right). \tag{3.6}$$

3.4.2. Wybór wartości początkowych

Wybór wartości początkowych, podobnie jak w sekcji 3.3.2 przeprowadzamy metodą graficzną, znając niektóre właściwości funkcji określonej równaniem 3.5. Wiadomo, że parametr μ określa oś symetrii wykresu funkcji. Wykres funkcji Gaussa jest symetryczny względem punktu, w którym funkcja osiąga maksimum. Stąd za wartość początkową parametru μ możemy wybrać taką wartość zmiennej `wiek`, dla której zmienna `odsetek.palaczy` osiąga maksimum, czyli około 55. Podobnie wiadomo, że wartość parametru σ można określić na podstawie kształtu wykresu – dla wartości zmiennej `wiek` równych $\mu \pm \sigma$ wykres powinien mieć punkty przegięcia. Wypukłość funkcji zmienia się około prostych odpowiadających wartościom zmiennej `wiek` równym 35 i 75, stąd proponowana wartość początkowa zmiennej σ wynosi 20. Na podstawie tych dwóch wartości można oszacować wartość początkową parametru a, przez szukanie wykresu wizualnie najbardziej zbliżonego do danych. Wykresy przedstawiające krzywe w zależności od wartości parametru a przedstawione są na rysunku 3.5. Widać na nim, że najbliższy danym jest wykres otrzymany dla parametru $a = 25$. Dlatego tę wartość

przyjmiemy za wartość początkową.



Rysunek 3.5: Funkcje Gaussa dla różnych wartości parametru a: 30 – kreski, 25 – linia ciągła, 20 – kropki.

3.4.3. Wykorzystanie funkcji `nls()`

Określmy najpierw funkcję zadaną równaniem 3.5 w programie **R**.

```
> nor = function (x, mean, sd) {
(1/(sd*sqrt(2*pi)))*exp(-(x-mean)^2/(2*sd*sd))}
```

Funkcję tę wykorzystajmy dalej jako argument funkcji `nls()`. Za wartości początkowe przyjmiemy zestaw wybrany na podstawie graficznej analizy danych:

$$\begin{cases} a &= 25 \\ \mu &= 55 \\ \sigma &= 20 \end{cases}$$

```
> palenie.m2 <-nls(odsetek.palaczy ~ a*nor(wiek, m, sd),
start=list(a=25, m=55, sd=20), trace="TRUE")

0.5411278 :  25 55 20
0.1193678 :  21.08568 50.58806 20.36142
0.1100274 :  21.00828 49.77668 19.62678
0.1100087 :  20.98971 49.82955 19.59106
0.1100087 :  20.98978 49.82893 19.59104
```

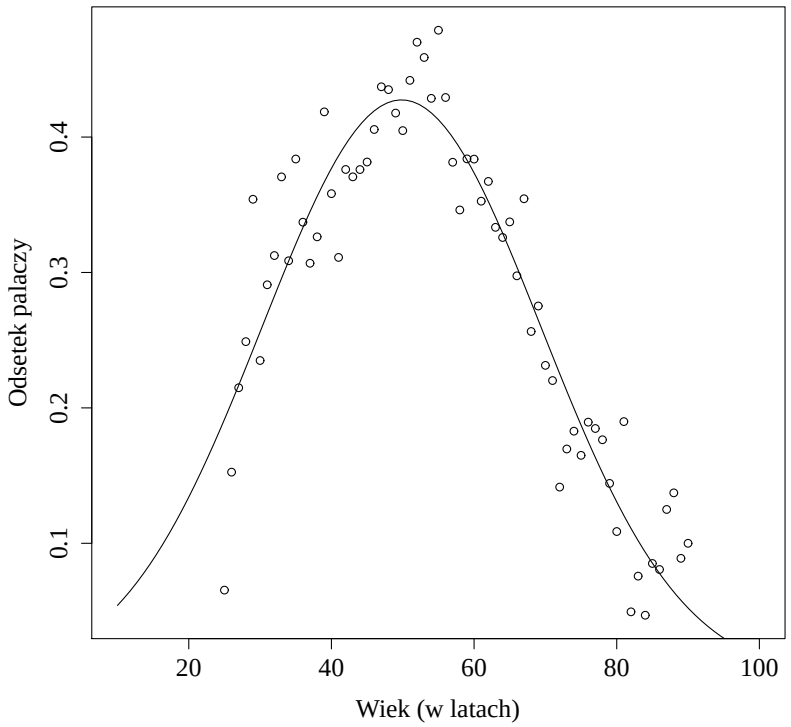
```
>palenie.m2
Nonlinear regression model
  model:  odsetek.palaczy ~ a * nor(wiek, m, sd)
   data:  parent.frame()
    a     m    sd
20.99 49.83 19.59
 residual sum-of-squares: 0.11

Number of iterations to convergence: 4
Achieved convergence tolerance: 2.63e-06
```

W efekcie działania funkcji `nls()` otrzymano następujący zestaw parametrów:

$$\begin{cases} a &= 20,99 \\ \mu &= 49,83 \\ \sigma &= 19,59 \end{cases}$$

Wykres krzywej regresji dla otrzymanych wartości parametrów przedstawia rysunek 3.6.



Rysunek 3.6: Krzywa regresji w modelu wykorzystującym krzywą Gaussa.

3.5. Porównanie modeli

W sekcjach 3.3 i 3.4 przedstawiono dwa modele regresji, dla których wartości parametrów znaleziono wykorzystując funkcję `nls()`. Pierwszy z przedstawionych modelu był liniowy

względem parametrów, drugi – nieliniowy. W tej sekcji przeprowadzone jest porównanie modeli.

3.5.1. RSS

Pierwszym z analizowanych kryteriów porównawczych jest RSS. Zgodnie z teorią przedstawioną w rozdziale 1, szukając wartości parametrów, dąży się do minimalizacji RSS. Dla otrzymanych oszacowań parametrów w modelu wykorzystującym krzywą Gaussa uzyskuje się niższą wartość RSS niż w modelu opisywanym w sekcji 3.3 ($0,11$ w porównaniu do $0,35$).

3.5.2. Diagnostyka modeli

Aby przeprowadzić pełną diagnostykę przedstawionych modeli `palenie.m1` i `palenie.m2` wykorzystujemy funkcje z pakietu `nlstools`([4]): `nlsResiduals` i `test.nlsResiduals`. Pierwsza z nich przygotowuje zestaw danych do wykresów diagnostycznych, druga testuje, czy reszty mają rozkład normalny i czy są losowe (więcej o testowanych tu założeniach można przeczytać w [3]).

Wykorzystajmy najpierw funkcję `test.nlsResiduals`.

```
> test.nlsResiduals(nlsResiduals(palenie.m1))

------
        Shapiro-Wilk normality test

data:  stdres
W = 0.989, p-value = 0.8283


------
        Runs Test

data:  as.factor(run)
Standard Normal = -2.481, p-value = 0.0131
alternative hypothesis: two.sided
```

```
> test.nlsResiduals(nlsResiduals(palenie.m2))

------
        Shapiro-Wilk normality test

data:  stdres
W = 0.9826, p-value = 0.4834


------
        Runs Test

data:  as.factor(run)
Standard Normal = -1.464, p-value = 0.1432
alternative hypothesis: two.sided
```

Na podstawie przeprowadzonych testów Shapiro-Wilka nie można odrzucić hipotezy zerowej, mówiącej o normalności rozkładów reszt w żadnym z modeli. Jednak na podstawie

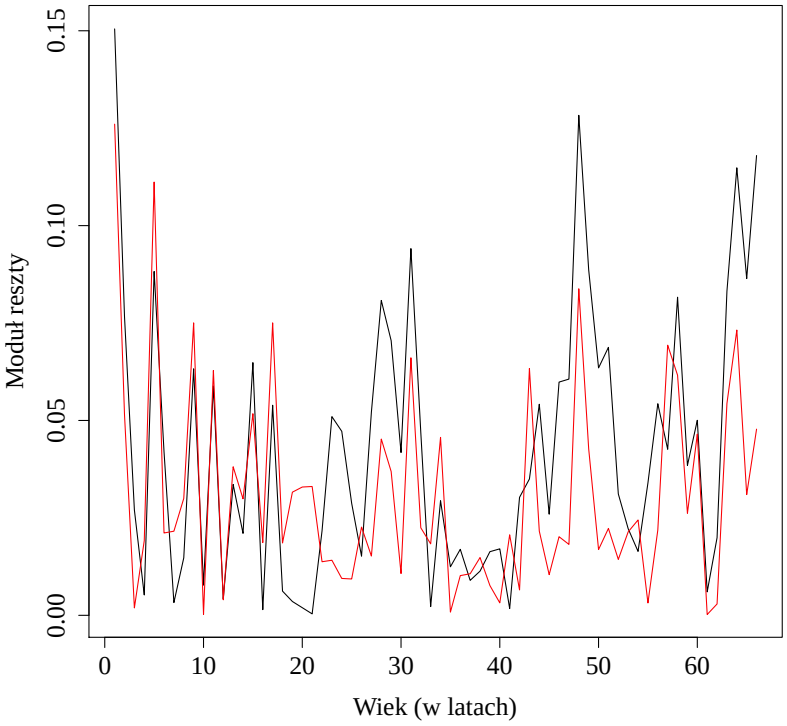
testu serii w pierwszym z modeli można odrzucić hipotezę zerową o losowości rozkładu reszt ($p < 0,05$). Nie można tego jednak zrobić w przypadku drugiego modelu.

W efekcie działania funkcji `nlsResiduals` otrzymano rysunki 3.8 i 3.9 (strony 43 i 44). Rysunki te składają się z czterech wykresów: wykresu reszt (*residuals*), wykresu wystandaryzowanych reszt (*standarized residuals*), wykresu autokorelacji (*autocorrelation*) i wykresu kwantylowo-normalnego (*normalized Q-Q plot*) wystandaryzowanych reszt. Przyjrzyjmy się pierwszemu z wykresów na obu rysunkach. Na rysunku 3.8 widać, że wartość reszt zależy nieliniowo od wartości zmiennej `odsetek.palaczy`. Wykres ma kształt litery „U" – dla „małych" i „dużych" wartości zmiennej zależnej, wartości reszt są większe niż dla „średnich" wartości tej zmiennej. Widoczna zależność i zmiana znaku reszt (dla wartości zmiennej zależnej mniejszych niż około $0,08$ i większych niż około $0,35$ większość reszt jest dodatnia, dla wartości zmiennej wyjaśnianej z przedziału $(0,08; 0,35)$ reszty są ujemne) sugerują, że wybrana krzywa nie pasuje kształtem do danych. Podobnego wrażenia nie można odnieść po analizie rysunku 3.9, na podstawie którego ciężko wskazać jakąkolwiek zależność pomiędzy wartościami zmiennej zależnej i wartościami reszt. Zauważmy dodatkowo, że reszty przedstawione na rysunku 3.8 są co do modułu większe niż reszty przedstawione na rysunku 3.9. Różnice te szczególnie dobrze widać na rysunku 3.7 na stronie 42.

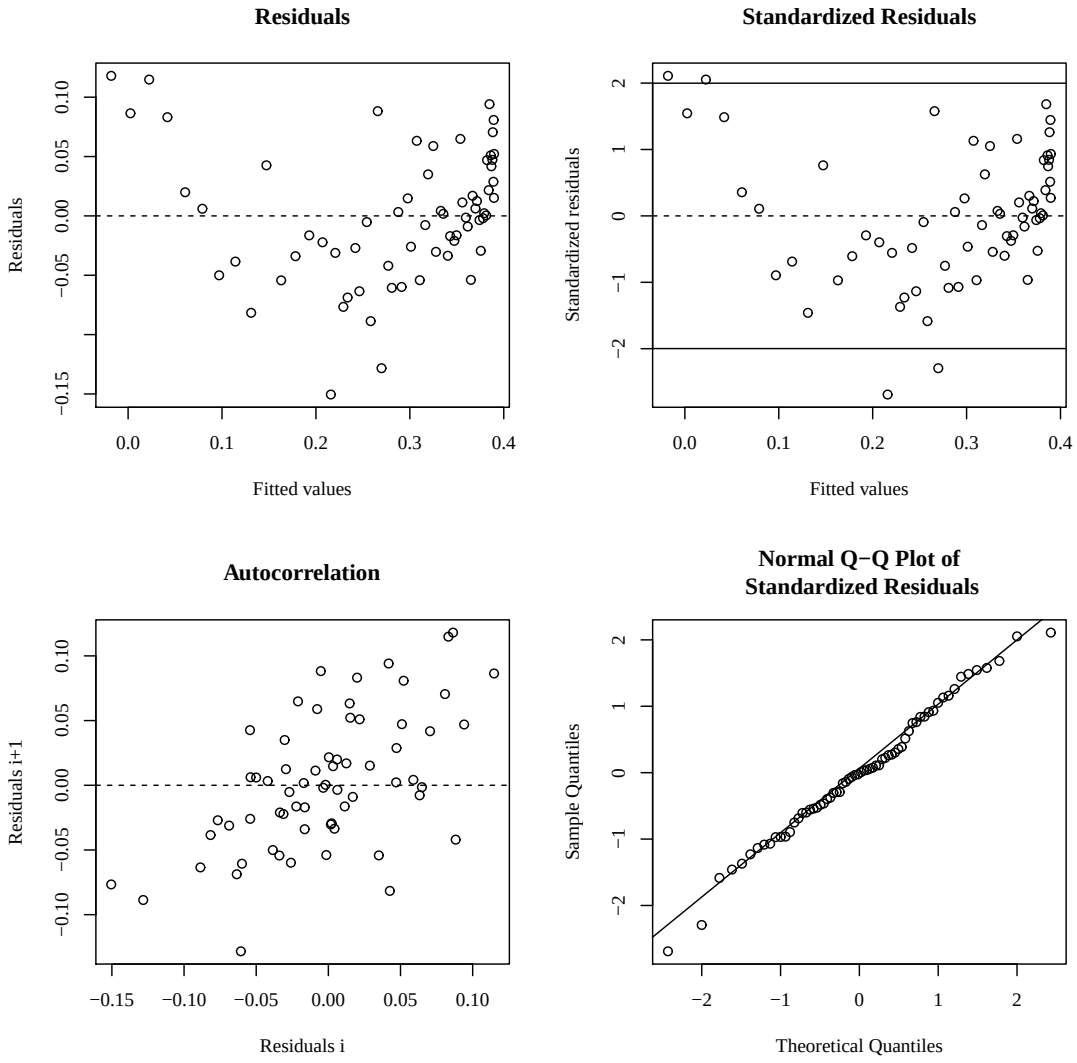
Na podstawie kolejnych wykresów – wykresów wystandaryzowanych reszt – możemy stwierdzić, że wśród reszt z modelu `palenie.m2` mniej jest odbiegających od średniej o ponad 2 odchylenia standardowe niż wśród reszt w modelu `palenie.m1`. Analizując rysunki autokorelacji, można dojść do wniosku, że błędy w modelu `palenie.m1` są skorelowane – dane przedstawione na wykresie układają się w linię. Inaczej jest w przypadku danych dotyczących reszt w modelu `palenie.m2`. Dla tych danych niemożliwe jest wskazanie zależności pomiędzy resztami w kolejnych krokach. Ostatnie wykresy (wykresy kwantylowo-normalne) mogą być pomocne przy analizie normalności rozkładów reszt, ale tę kwestię wyjaśniono już wcześniej przy pomocy testu Shapiro-Wilka.

3.5.3. Podsumowanie

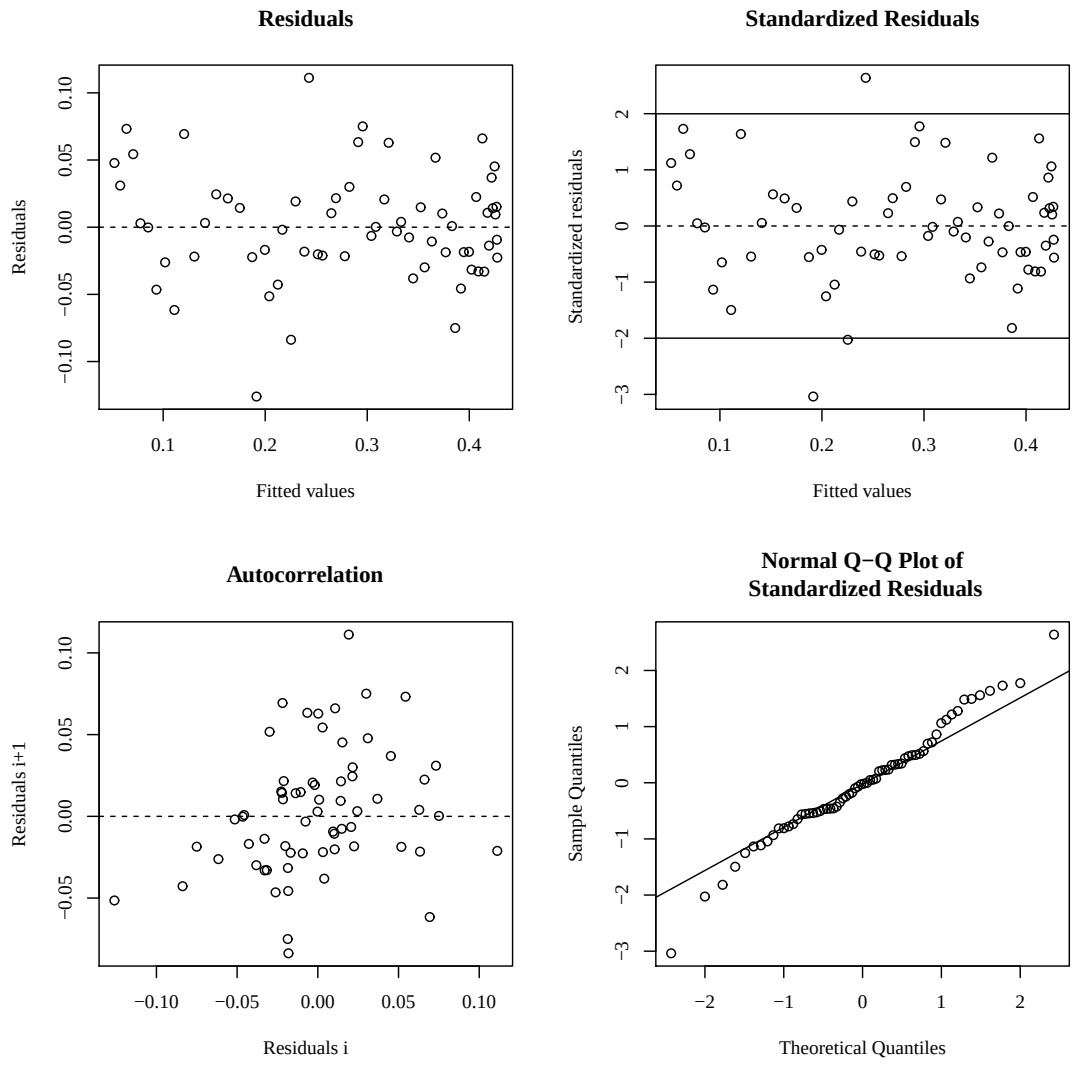
W niniejszym rozdziale przedstawione są dwa modele pozwalające na przewidywanie odsetka osób palących papierosy w danej grupie wiekowej. Pierwszy z zaproponowanych modeli, określony równaniem 3.1, jest modelem, w którym wykorzystuje się funkcję liniową ze względu na parametry. Drugim zaproponowanym modelem (równanie 3.5) jest model nieliniowy ze względu na parametry. Analiza RSS (sekcja 3.5.1) oraz analiza reszt (sekcja 3.5.2) pozwalają na stwierdzenie, że proponowany model nieliniowy jest lepszy niż model liniowy. Po pierwsze, RSS dla modelu nieliniowego jest niższe. Po drugie, reszty w modelu liniowym, w przeciwieństwie do reszt w modelu nieliniowym, nie są losowe. Po trzecie, reszty w modelu liniowym są skorelowane z resztami z wcześniejszych kroków działania algorytmu. Argumenty te uzasadniają w przypadku analizowanych danych wyższość modelu nieliniowego nad modelem liniowym.



Rysunek 3.7: Moduły różnic w modelach `palenie.m1` (linia czarna) i `palenie.m2` (linia czerwona).



Rysunek 3.8: Wykresy diagnostyczne dla reszt w modelu `palenie.m1`.



Rysunek 3.9: Wykresy diagnostyczne dla reszt w modelu `palenie.m2`.

Rozdział 4

Zakończenie

Praca zawiera podstawowe informacje dotyczące modeli regresji nieliniowej. Modele te służą do opisu zależności pomiędzy zmiennymi. Powszechnie stosuje się je w badaniach empirycznych w rozmaitych dziedzinach, na przykład w naukach społecznych i przyrodniczych (np.[3]). W niniejszej pracy model regresji nieliniowej został zastosowany do opisu danych pochodzących z badań społecznych, opisanych wcześniej przy pomocy modelu liniowego.

Rozdział teoretyczny zawierał opis wykorzystywanych modeli regresji, dla których charakterystyczna jest nieliniowa zależność funkcji określającej model od co najmniej jednego z wykorzystywanych parametrów. Kolejne rozdziały poświęcone były zastosowaniom regresji nieliniowej. W pierwszym z nich przedstawiono metodę umożliwiającą estymację wartości parametrów w modelu za pomocą programu **R**. Rozdział ten zawierał wiele przykładów opartych na dostępnych danych z zakresu nauk przyrodniczych. W ostatnim rozdziale przedstawiono dwa modele regresji wyjaśniające zależność pomiędzy wiekiem a odsetkiem palaczy w danej grupie. Przedstawione analizy reszt pozwoliły na stwierdzenie, że proponowany model nieliniowy lepiej opisuje szukaną zależność niż model liniowy.

Spis rysunków

Spis tabel

Bibliografia

[1] Bates, D.M. (2011). *NRAIA: Data sets from "Nonlinear Regression Analysis and Its Applications". R package version 0.9-8.* `http://CRAN.R-project.org/package=NRAIA` [22.08.2012]

[2] Bates, D.M., DebRoy, S., Gay, D.M.(2012.) *nls() function.* W: R Core Team. (2012). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Wiedeń.

[3] Bates, D.M., Watts, D.G. (1988). *Nonlinear Regression Analysis and Its Applications.* Wiley, Canada. 32-66, 67-134.

[4] Baty, F., Delignette-Muller, M.L. (2011). *nlstools: tools for nonlinear regression diagnostics.*

[5] Beverton R.J.H., Holt S.J. (1993). *On the dynamics of exploited fish populations.* Chapman and Hall, Londyn. 26-41.

[6] Cadergeen N., Madsen T.V. (2002). *Nitrogen uptake by the floating macrophyte Lemna minor.* New Phytol. **155**, 285-292.

[7] Grothendieck, G. (2010). *nls2: Non-linear regression with brute force. R package version 0.1-3.* `http://CRAN.R-project.org/package=nls2` [22.08.2012]

[8] Kincaid, D., Cheney, W. (2002). *Numerical Analysis. Mathematics of Scientific Computing.* Tłumaczenie na polski: Paszkowski, S. (2006). *Analiza Numeryczna.* Wydawnictwo Naukowo-Techniczne, Warszawa. 71-81.

[9] Michaelis, L., Menten, M.M.L. (1913). *Die Kinetik der Invertinwirkung.* Biochem, **49**. 333-369. Tłumaczenie na angielski: Goody, S.R., Johnson, K.A. `http://pubs.acs.org/doi/suppl/10.1021/bi201284u/suppl_file/bi201284u_si_001.pdf` [22.08.2012]

[10] NIST (National Institute of Standards and Technology) (1979). *Ultrasonic Reference Study Block (D. Chwirut). Technical report.* `http://www.itl.nist.gov/div898/strd/nls/data/chwirut2.shtml`. [01.05.2012]

[11] Prus, M. (2010). Sugestywne kampanie antynikotynowe. *Ciekawnik.pl. Warto wiedzieć.* `http://ciekawnik.pl/inspiracje/563-kampanie-antynikotynowe` [22.08.2012]

[12] Quarteroni, A., Sacco, R., Saleri, F. (2000). *Numerical Mathematics.* Springer, Nowy Jork. 83-287.

[13] R Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Wiedeń. `http://stat.ethz.ch/R-manual/R-patched/library/stats/html/nls.html`. [01.05.2012]

[14] Rada Monitoringu Społecznego (2011). *Diagnoza społeczna: zintegrowana baza danych.* www.diagnoza.com [21.05.2012]

[15] Shapiro, S.S., Wilk, M.B. (1965). *An analysis of variance test for normality (complete samples).* Biometrika, 52. 591–611.

[16] Radhakrishna Rao, C. (1973). *Linear Statistical Inference and its Applications.* Tłumaczenie na polski: Zieliński, R. (1982). *Modele liniowe statystki matematycznej.* PWN, Warszawa. 277-300.

[17] Ritz, C., Streibig, J.C. (2008). *Nonlinear Regression with R.* Springer, Nowy Jork. 1-54.

[18] Seber, G.A.F., Wild, C.J. (1989). *Nonlinear Regression.* Wiley, Canada. 1-20, 619-623.

[19] Szymańska-Borginon, K. (2012). Opakowania na papierosy będą jednolite w UE? *RMF FM.* http://www.rmf24.pl/fakty/swiat/news-opakowania-na-papierosy-beda-jednolite-w-ue,nId,628748 [22.08.2012]

[20] Thine Omuto, C. (2012). *HydroMe: Estimation of Soil Hydraulic Parameters from Experimental Data. R package version 1.0.* http://CRAN.R-project.org/package=HydroMe [22.08.2012]

[21] Venables, W.N., Ripley, B.D. (2002). *Modern Applied Statistics with S.* Springer, New York.

[22] Wald, A., Wolfowitz, J. (1940). *On a test whether two samples are from the same population.* Annals of Mathematical Statistics. 11. 147-162. http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.aoms/1177731909 [17.09.2012]

[23] World Health Organization. (2012). *Confronting the tobacco epidemic in a new era of trade and investment liberalization.* http://whqlibdoc.who.int/publications/2012/9789241503723_eng.pdf. [22.08.2012] .