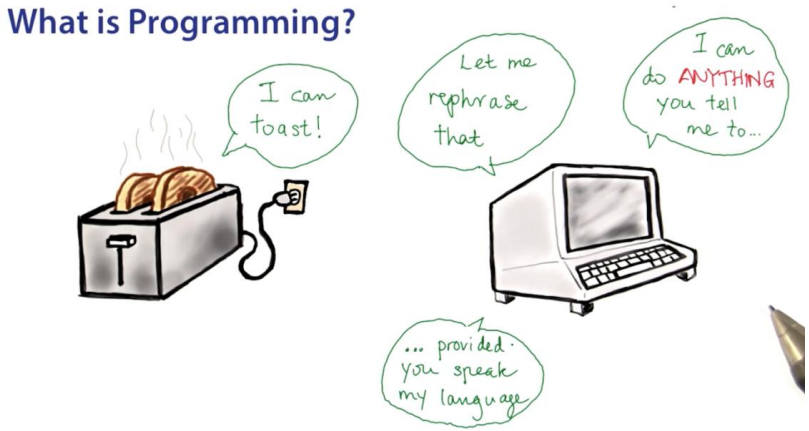
Wprowadzenie do programowania i środowiska R



Biomatematyka - wykład 2

Dr Wioleta Drobik-Czwarno

Dlaczego biolodzy powinni nauczyć się podstaw programowania?

- Pomaga lepiej zrozumieć działanie programów komputerowych z których korzystamy w codziennej pracy
  - Każdy program jest pisany przez ludzi i może zawierać błędy
- Jest całkiem prawdopodobne, że kiedyś będziecie pracowali z programistą/bioinformatykiem, więc dobrze znać język jakim się posługują
- Co jeżeli nikt jeszcze nie stworzył programu komputerowego do waszej nowej, genialnej metody której chcecie/musicie użyć w badaniach?

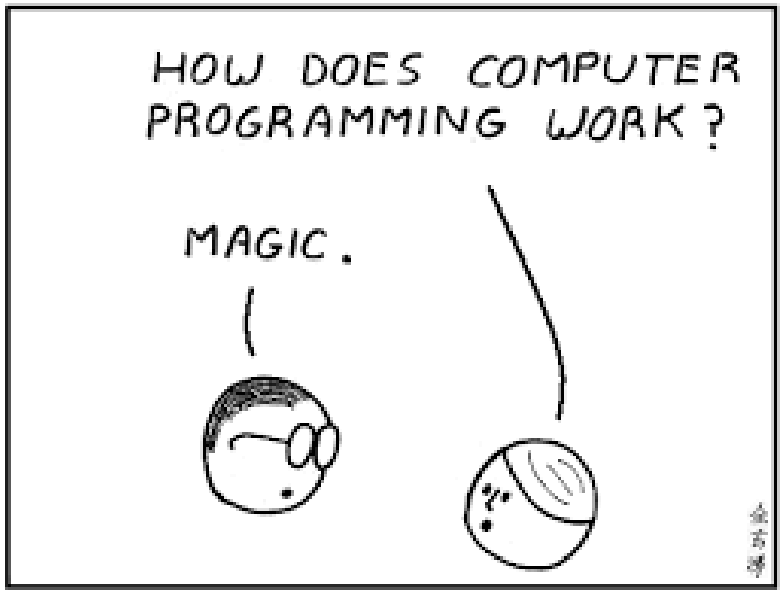
Model Lotka-Volterra w R

```
library(deSolve)

LotVmod <- function (Time, State, Pars) {
    with(as.list(c(State, Pars)), {
        dx = x*(alpha - beta*y)
        dy = -y*(gamma - delta*x)
        return(list(c(dx, dy)))
    })
}

Pars <- c(alpha = 2, beta = .5, gamma = .2, delta = .6)
State <- c(x = 10, y = 10)
Time <- seq(0, 100, by = 1)

out <- as.data.frame(ode(func = LotVmod, y = State, parms = Pars, times = Time))

matplot(out[,-1], type = "l", xlab = "time", ylab = "population")
legend("topright", c("Cute bunnies", "Rabid foxes"), lty = c(1,2), col = c(1,2), box.lwd = 0)
```

Źródło: https://www.r-bloggers.com/

Czym jest programowanie?

- Programowanie **nie jest** zajęciem dla wybrańców posiadających ogromną i niemal tajemną wiedzę



Języki programowania

- **Język programowania** to język stworzony do opisu kolejnych kroków, które mają być podjęte przez komputer.
  - ~ instrukcja dawana komputerowi przez człowieka.
- Język używany przez procesor to **kod maszynowy**
- Kod pisany w języku programowania jest przetwarzany na kod maszynowy tak, by mógł zostać przetworzony przez procesor

Klasyfikacja

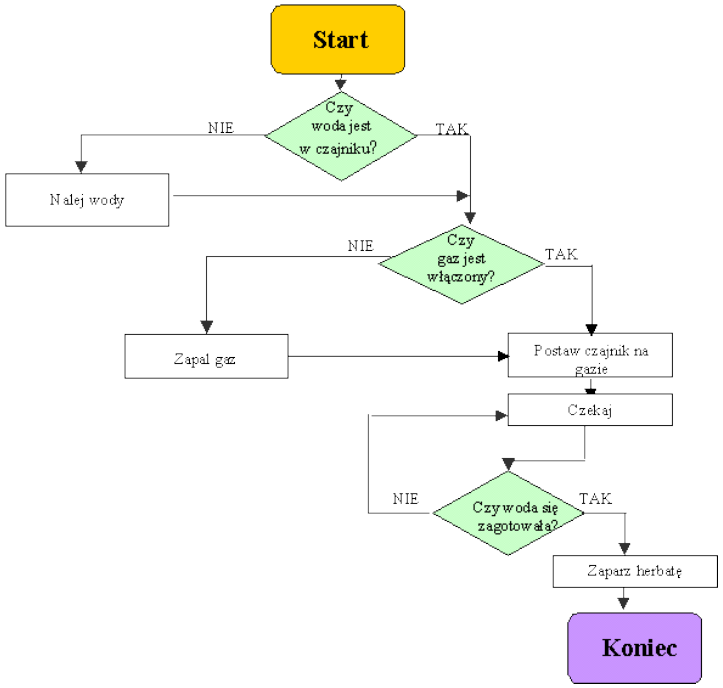
- Interpretowane
  - Na bieżąco tłumaczone na język maszynowy komputera przez program zwany **interpreterem**
- Kompilowane
  - Kod źródłowy jest tłumaczony na kod maszynowy (wykonywalny program) przez program zwany **kompilatorem**
  - Jedna instrukcja w kodzie źródłowym to kilka (języki niskiego poziomu), kilkaset instrukcji (języki wysokiego poziomu) w kodzie maszynowym

Klasyfikacja

- Poziom
  - *Języki niskiego poziomu* - języki maszynowe (zapisane są w postaci liczb binarnych) oraz asemblery (języki symboliczne)
  - *języki wysokiego poziomu –w tych* językach jedna instrukcja jest tłumaczona na kilka, kilkadziesiąt instrukcji procesora

Algorytm

- Czym jest algorytm?
  - Algorytm to jednoznaczny przepis wykonania pewnej czynności w skończonym czasie np. zmiana pewnych danych wejściowych do pewnych danych wynikowych



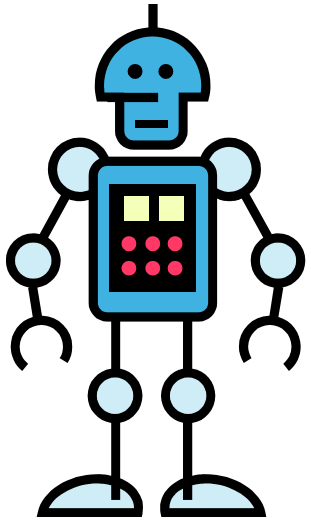
Od pojęć do kodu…

- Typowe formy zapisu algorytmu:
  - Opis słowny
  - Schemat blokowy
  - Pseudokod
  - Lista kroków do wykonania
  - Kod źródłowy programu

- Program to algorytm zapisany w postaci (zależnie od języka programowania) umożliwiającej jego automatyczną realizację za pomocą komputera

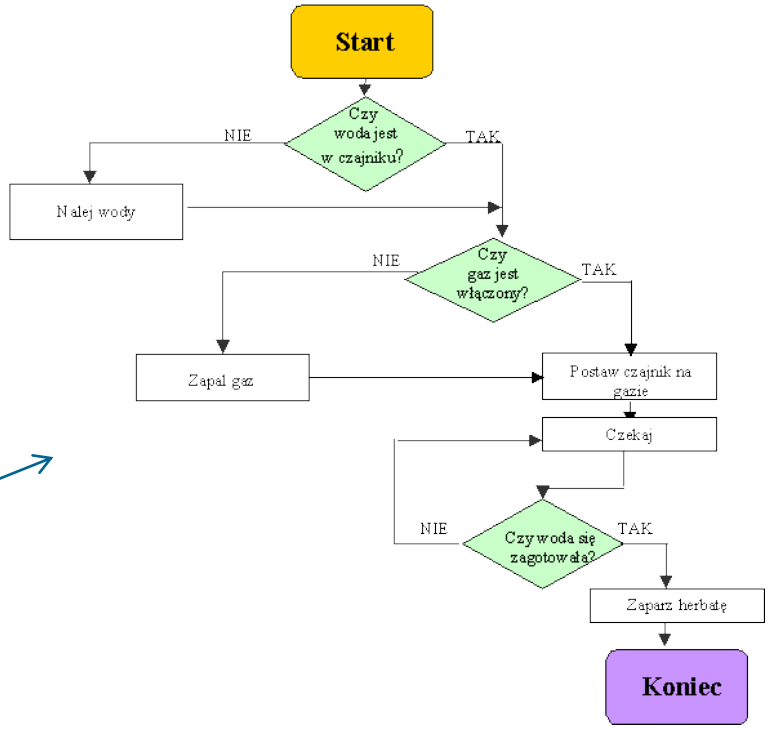
Od problemu do programu

1. Sformułowanie problemu (definicja zadania)
2. Analiza problemu
3. Wybór metody (metod) rozwiązania
4. Opracowanie algorytmu
5. Kodowanie (implementacja) programu
6. Testowanie programu
7. Sporządzenie dokumentacji



Główne składowe każdego programu

- Zmienne
- Funkcje
- Sterowanie przepływem wykonywania programu
  - instrukcje warunkowe
  - pętle



Schemat blokowy algorytmu

Instrukcje warunkowe

- Najczęściej wykorzystywane elementy w języku programowania
- Słownie: Jeżeli spełniony został warunek wykonaj operacje X, jeżeli nie, wykonaj operacje Y.
- np. robienie sałatki warzywnej dla Jasia
  - Zrobimy sos na bazie…. Czy Jaś ma nietolerancje laktozy?

TAK

NIE

Instrukcje warunkowe w R

Przykład:

Skrypt (widok w edytorze tekstu)

```
x<-5

if(x<10){
   x+2
}else{
   x-3
}
```

Widok w konsoli

```
> x<-5
>
> if(x<10){
+    x+2
+ }else{
+    x-3
+ }
[1] 7
```

wynik

Pętle

- Rodzaj instrukcji pozwalających na wielokrotne powtórzenie wykonania zaprogramowanych operacji
- Np. doprawiaj sałatkę dopóki nie będzie dobra
  - Przykłady w R:

```
> wektor<-c(1:10)
> for(i in wektor){print(i)}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

```
> i<-0
> while(i<5){
+ print(i)
+ i<-i+1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
```

Biblioteki

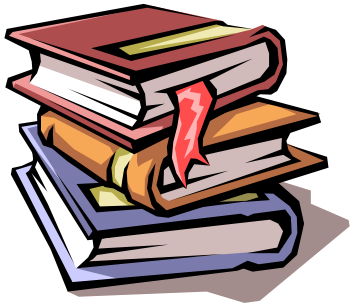
- To zbiór klas, funkcji możliwych do wykorzystania przez programistę / użytkownika
- W R nazywane są **pakietami**
- Biblioteka standardowa:
  - Podstawowy zestaw funkcji pozwalający zrealizować najważniejsze operacje

Jak nauczyć się podstaw programowania?

coursera education for everyone

edX

Kursy online



Książki

PRAKTYKA !!!

ROSALIND

Rozwiązywanie problemów biologicznych przez programowanie

Środowisko R

Narzędzia

- Najpopularniejsze środowiska do obliczeń numerycznych

- MATLAB
- OCTAVE
- R

-obliczenie inżynierskie i naukowe
- symulacje

-obliczenia statystyczne i zaawansowana grafika

Narzędzia

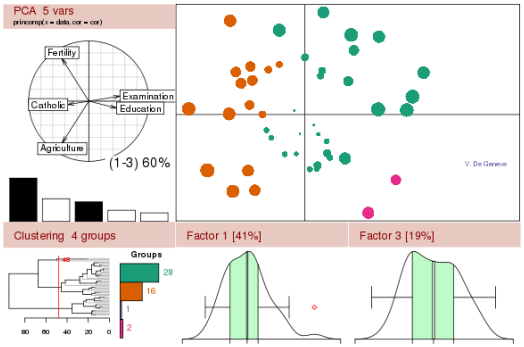
- Najpopularniejsze programy do statystyki i analizy danych
  - Arkusze kalkulacyjne (np. MS Excel)
  - SAS
  - Statistica
  - SPSS
  - S-PLUS
  - **R**

Czym jest **R**?

- GNU R to **język programowania i środowisko** do m.in..:
  - obliczeń statystycznych
  - wizualizacji danych
  - symulacji

- Cieszy się rosnącą popularnością i uznaniem zwłaszcza w Europie Zachodniej i USA, gdzie od lat jest standardem w dziedzinie analiz statystycznych

[R]
THERPROJECT.ORG



Trochę HistoRii

- Pierwsza wersja R została napisana przez Roberta Gentlemana i Ross Ihake (znanych jako R&R) pracujących na Wydziale Statystyki Uniwersytetu w Auckland
  - Pakiet początkowo służył jako pomoc dydaktyczna do uczenia statystyki na tym uniwersytecie

- Od roku 1997 rozwojem R kieruje:
  - zespół ponad dwudziestu osób nazywanych **core team**
  - fundacja *„The R Foundation for Statistical Computing”* z setkami aktywnych uczestników

- Wkład tysięcy osób z całego świata publikujących własne biblioteki z bardzo różnych dziedzin

Polecana literatura:

- Biecek P. 2014. Przewodnik po pakiecie R. Oficyna wydawnicza GIS. – Rozdział 1 wersji ze starszego wydania dostępny za darmo w Internecie: http://cran.r-project.org/doc/contrib/Biecek-R-basics.pdf
- Komsta Ł. 2004. Wprowadzenie do R. Wersja PDF dostępna za darmo w Internecie: http://cran.r-project.org/doc/contrib/Komsta-Wprowadzenie.pdf
- Rybiński M. 2011. Krótkie wprowadzenie do R dla programistów, z elementami statystyki opisowej. WMIM UW. Dostępny za darmo w Internecie: http://www.mimuw.edu.pl/~trybik/edu/0809/rps/r-skrypt.pdf
- Paradis E. 2005. R for Beginners. Dostępny za darmo w Internecie: http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

Ważne miejsca w sieci

- **The Comprehensive R Archive Network**
  - Instalacja, pakiety, materiały
  - **http://cran.r-project.org/**
- Informacje i tutoriale
  - http://www.r-bloggers.com/

- Tutoriale statystyczne i nie tylko
  - http://www.r-tutor.com/

- Try R
  - http://tryr.codeschool.com/

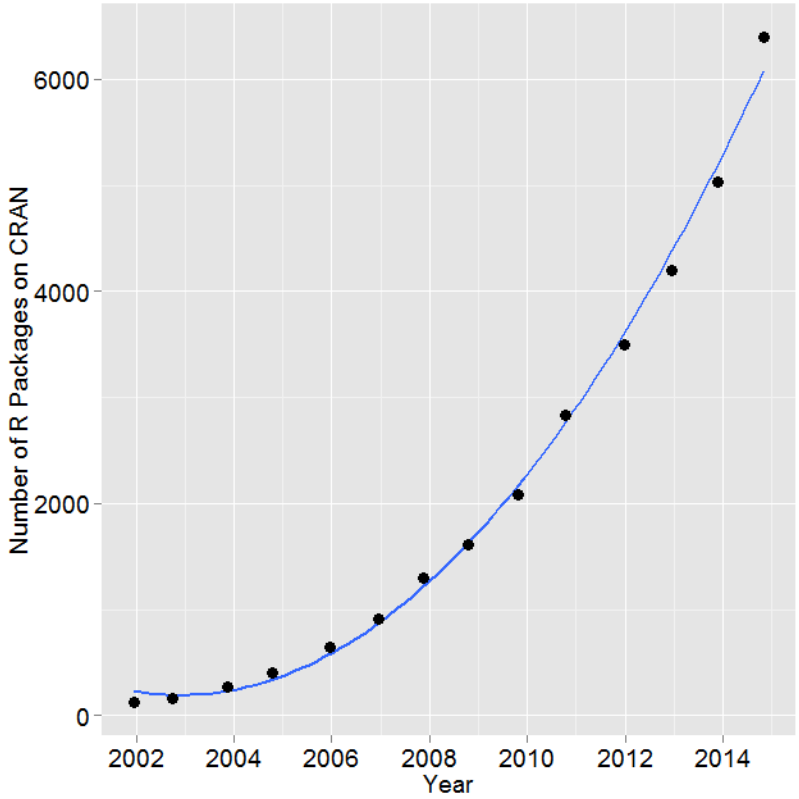
Dlaczego waRto?

- Oprogramowanie na licencji GPL do zastosowań prywatnych oraz komercyjnych
  - Darmowa dystrybucja
  - Dostępność kodu źródłowego
- Dynamiczny rozwój, tysiące dostępnych pakietów
  - Możliwość tworzenia pakietów zawierających nowe funkcjonalności
- Wsparcie naukowe
- Aktywna społeczność użytkowników w Polsce i na świecie
- Wieloplatformowość (Windows, Linux, Mac OS)
- Optymalizacja obliczeń
- Praca z dużymi zbiorami danych

Where in the world is R and Rstudio. http://blog.rstudio.org/2012/10/01/where-in-the-world/

Zalety platformy **R**

- R pozwala na tworzenie i upowszechnianie pakietów zawierających nowe funkcjonalności

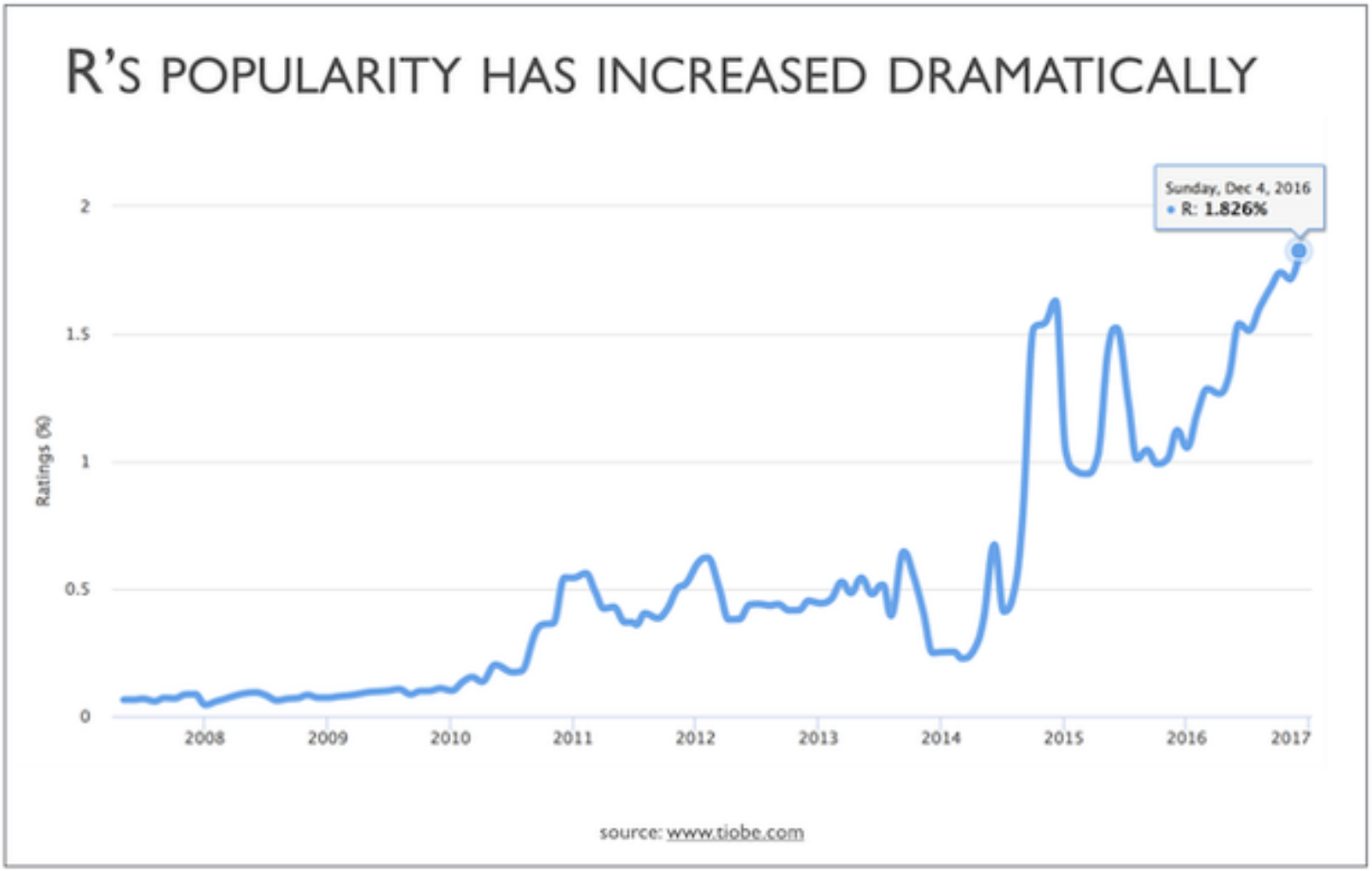


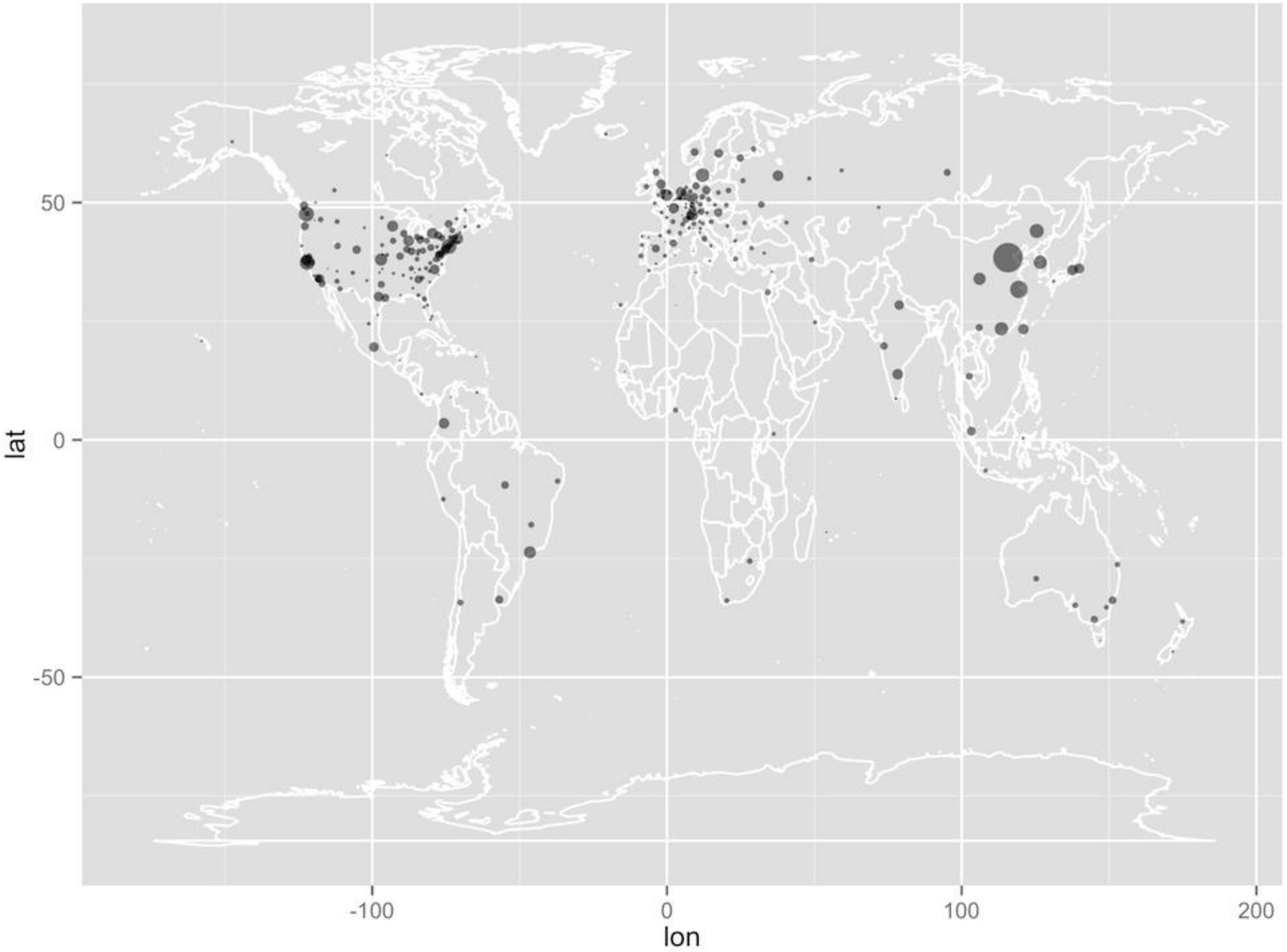
koniec 2014 r.,
ponad 6000 pakietów

Wykres 1. Liczba pakietów dostępnych w kolejnych latach

Źródło: r4stats.com/2016/04/19/rs-growth-continues-to-accelerate/

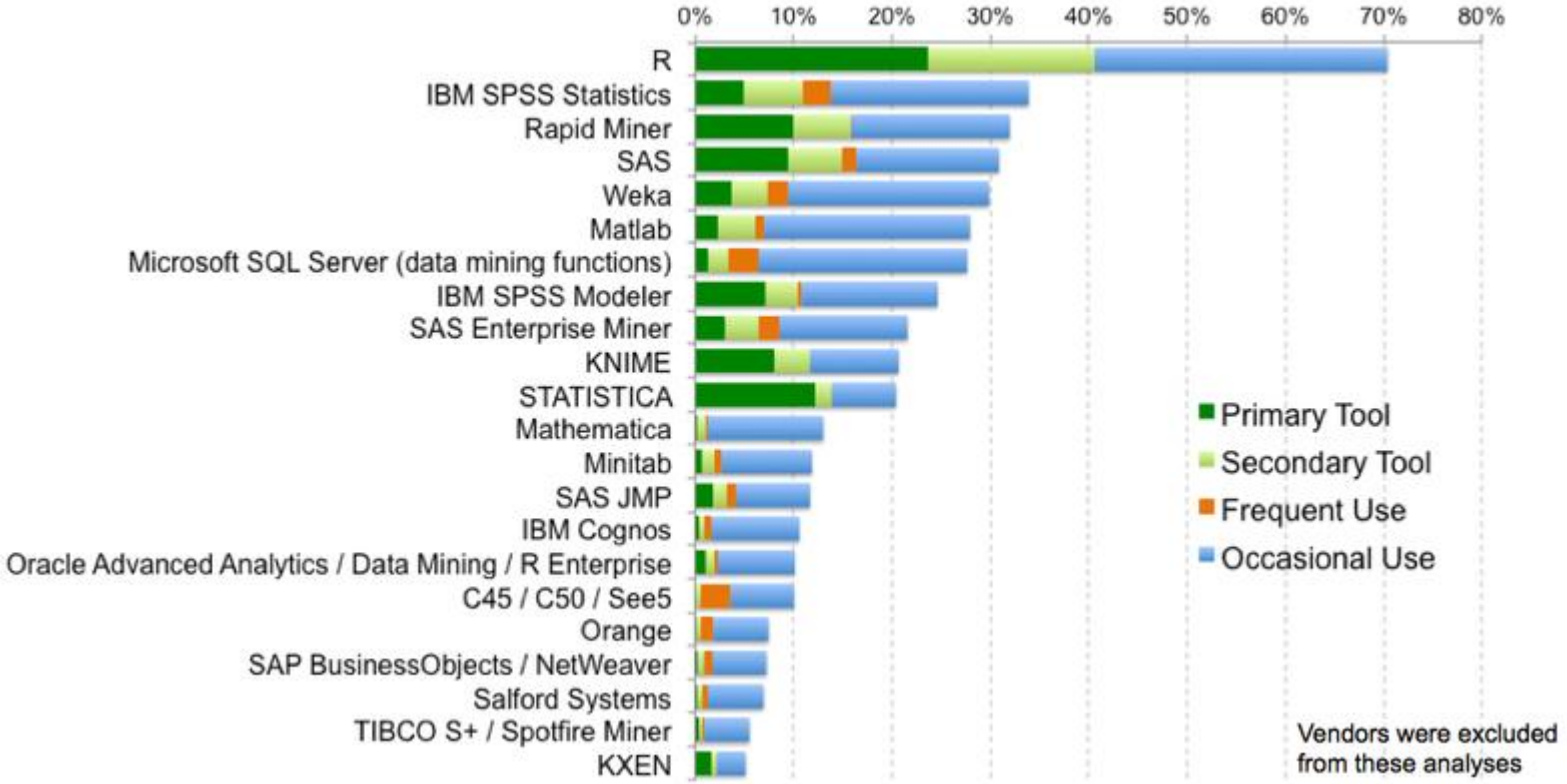
Popularność





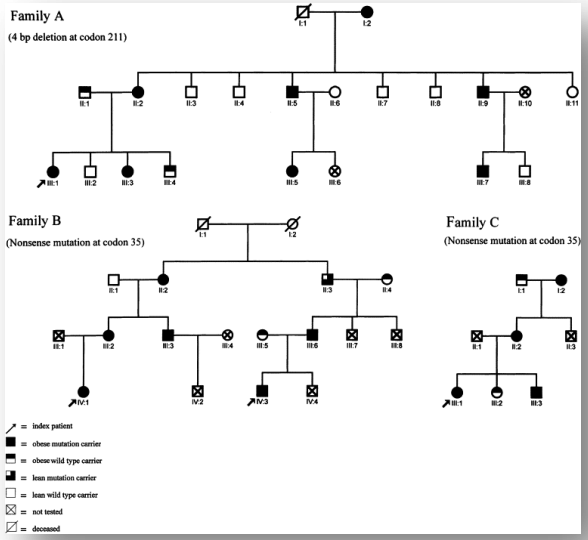
Źródło: http://blog.revolutionanalytics.com/2010/01/r-package-growth.html

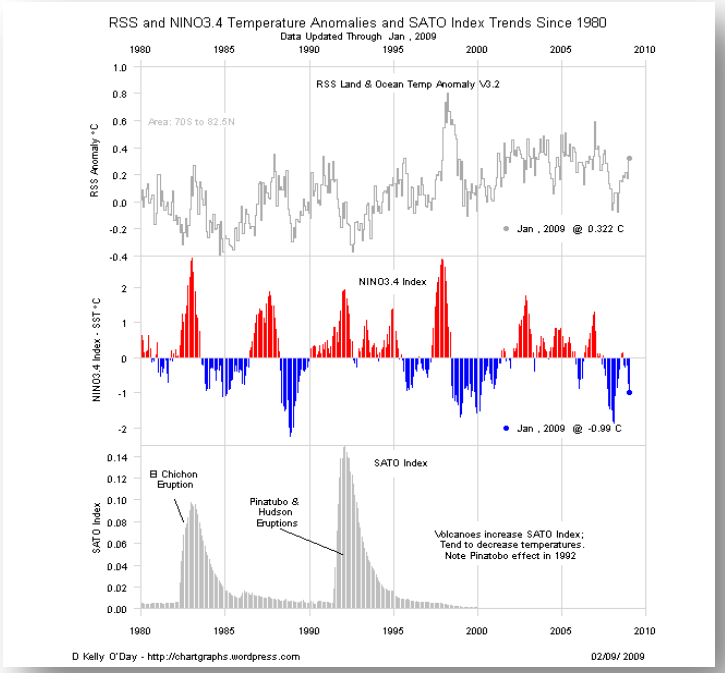
R JEST OBECNIE NAJPOPULARNIEJSZYM NARZĘDZIEM W ANALIZIE DANYCH NA ŚWIECIE

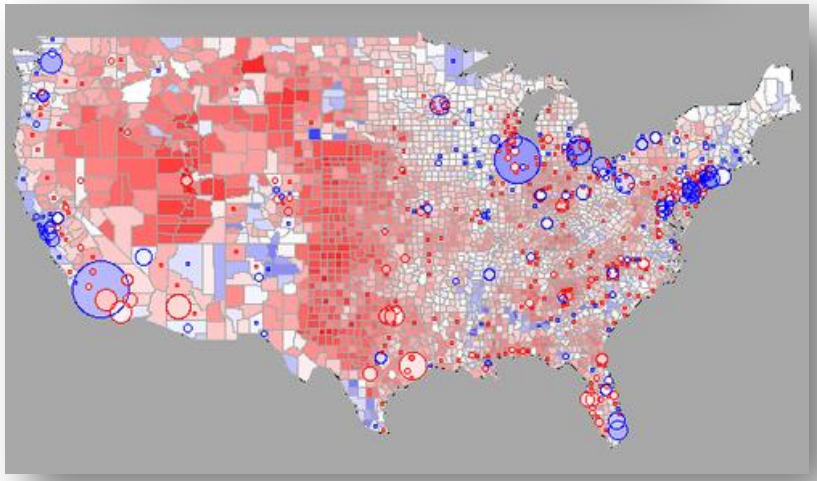


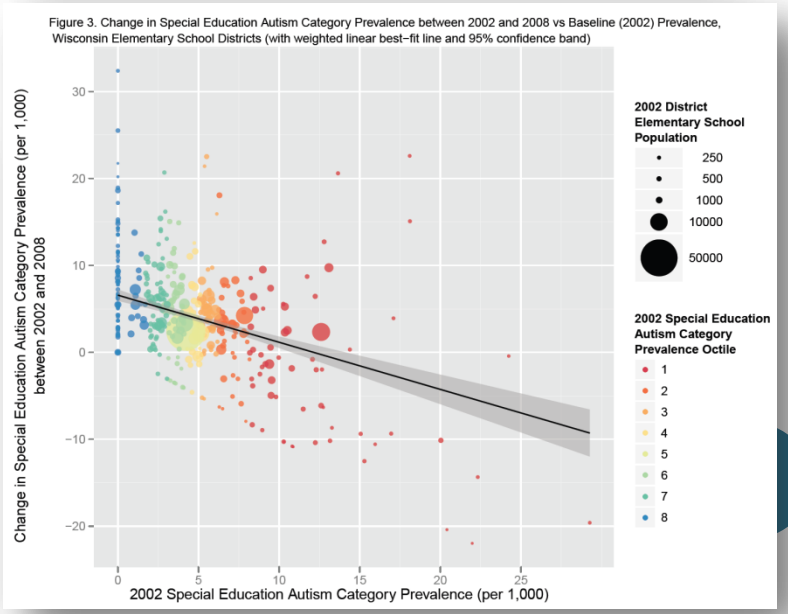
Źródło: http://blog.revolutionanalytics.com/2013/10/r-usage-skyrocketing-rexer-poll.html

MOŻLIWOŚCI GRAFICZNE







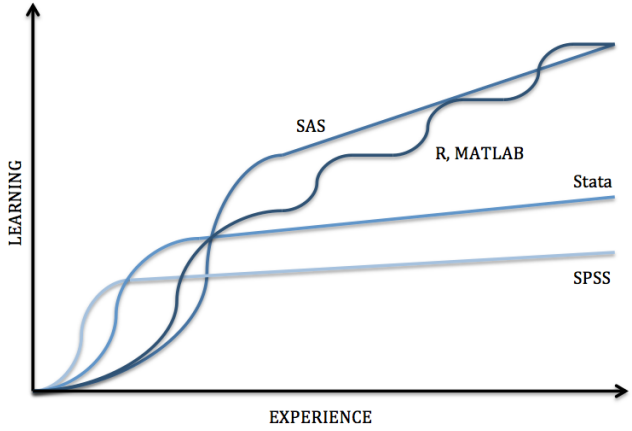


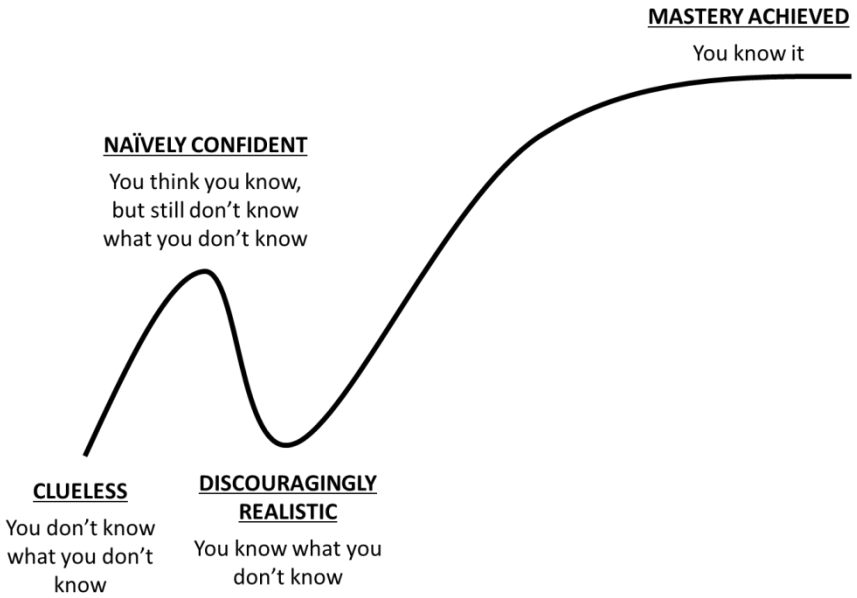
Cechy charakterystyczne:

- Interpretowalność języka R
- Odmienność od arkuszy kalkulacyjnych
  - Praca w linii komend lub nakładki graficzne
- Nakładki i narzędzia
  - R Studio – wygodne środowisko obliczeń zintegrowane z R, udogodnienia dla analityków i programistów
  - R Commander – interfejs okienkowy

Czy **R** ma jakieś wady?

- Stroma krzywa uczenia się

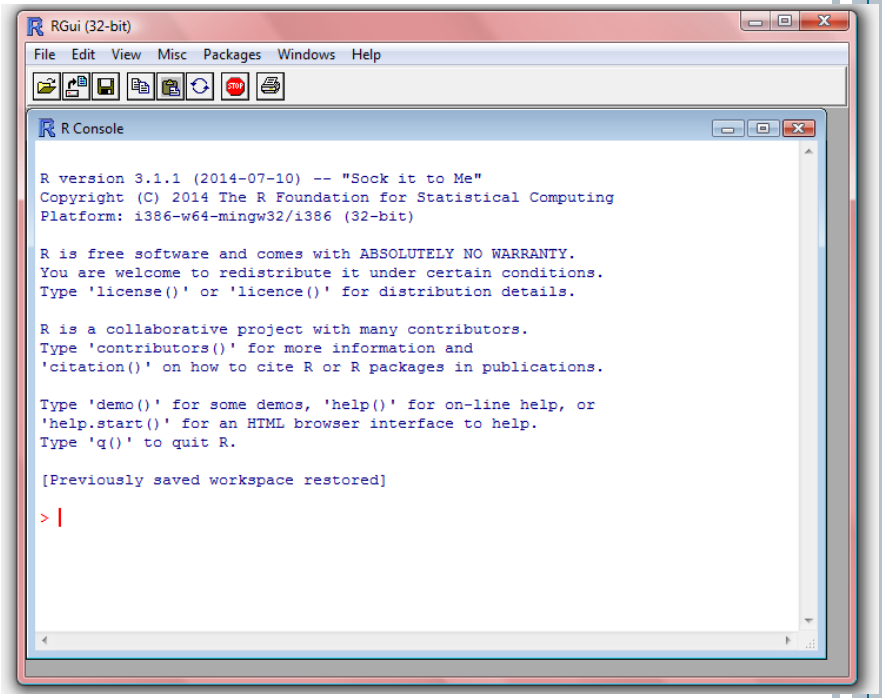




- Brak oszczędności w użyciu pamięci RAM
- Nie zawsze najlepsza jakość dokumentacji oraz pakietów (w zależności od pakietu)

JAK WYGLĄDA PRACA Z **R**

- Język R jest językiem **interpretowanym**, a korzystanie sprowadza się do podania ciągu komend, które mają zostać wykonane
- Znak > jest zachętą do wprowadzania poleceń
- Znak + jest znakiem kontynuacji
- Kolejne komendy mogą być wprowadzane linia po linii z klawiatury lub też mogą być wykonywane jako **skrypt**



Skrypt

- Jest to program wykonywany **wewnątrz** pewnej aplikacji , w odróżnieniu od programów „normalnych” wykonywanych niezależnie, bezpośrednio w systemie operacyjnym
- Składa się z wykonywalnego kodu oraz komentarzy
- **#** jest znakiem komentarza
  - wszystko co znajdzie się za tym znakiem jest pomijane przez program

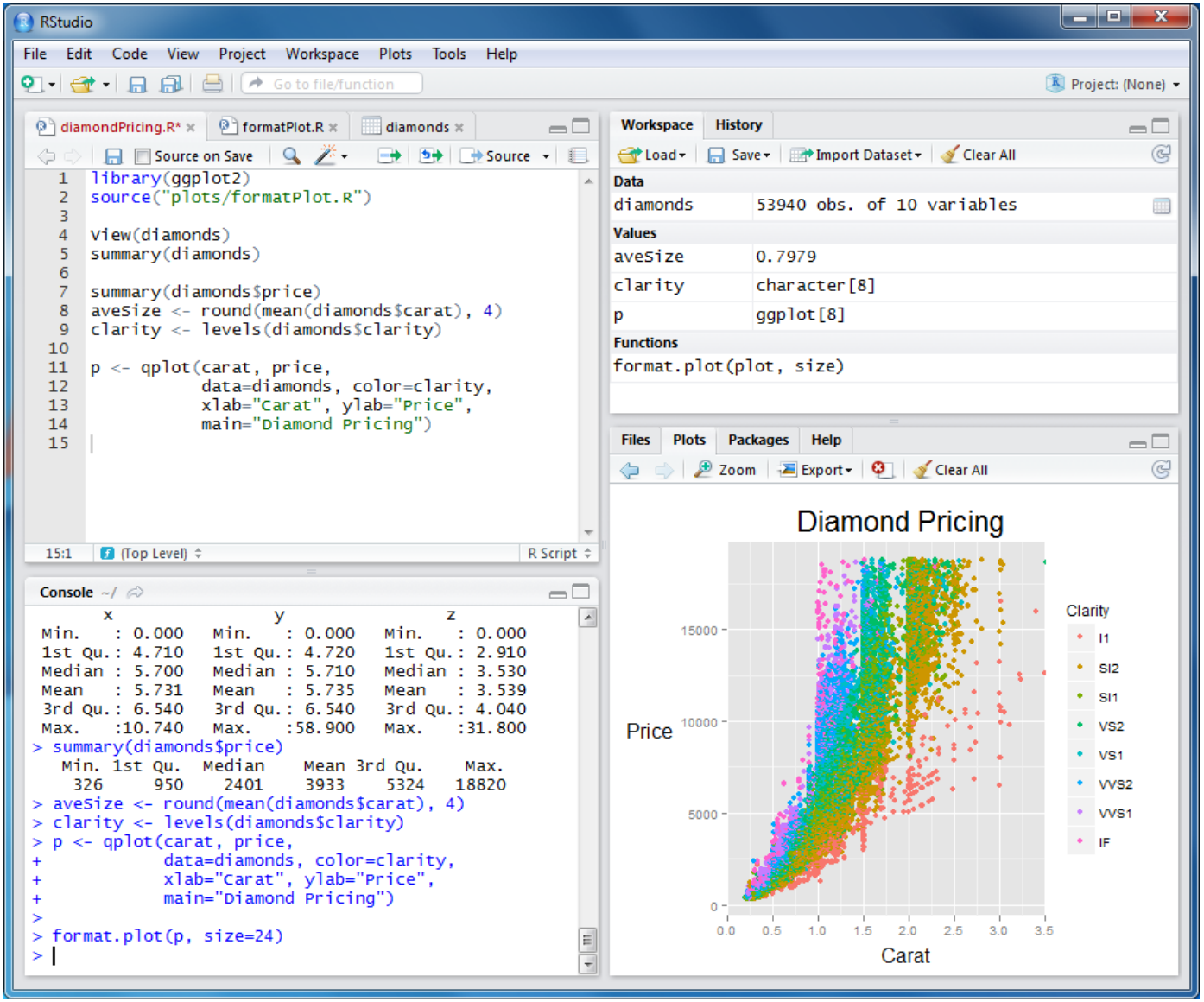
Pojęcia i komendy

- Obszar roboczy (ang. workspace)
  - Wszystkie obiekty znajdujące się aktualnie w pamięci operacyjnej
- ESC
  - Przerwanie aktualnie wykonywanej czynności
- q()
  - Zamknięcie środowiska

Praca z **R** studio

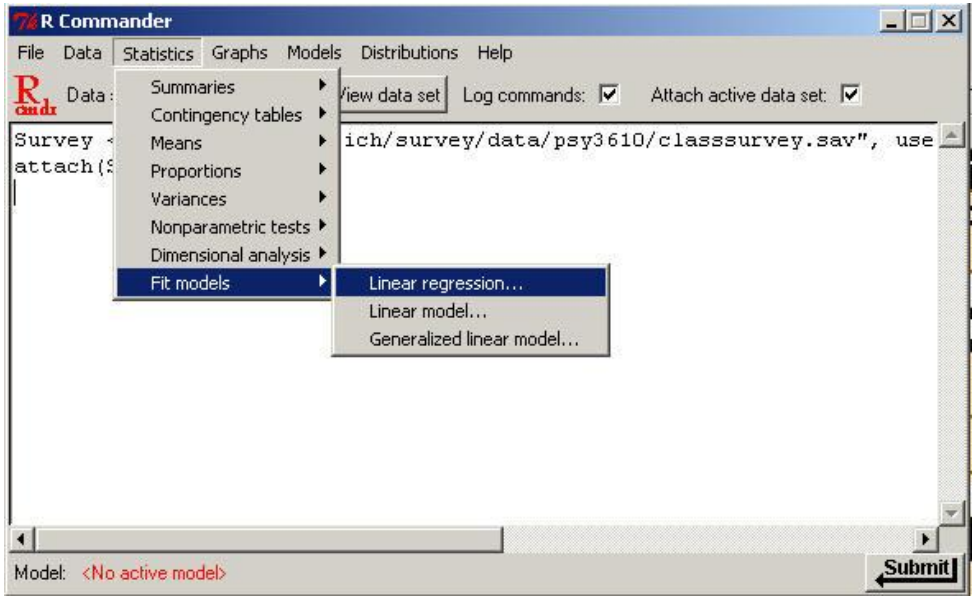
- Dostępne na stronie: *http://www.rstudio.com/*
  - Zarządzenie wieloma plikami/projektami
  - Wyświetlanie obiektów obecnych w przestrzeni nazw
  - Wbudowany edytor – kolorowanie składni
  - Podpowiadanie składni (skrót klawiaturowy: shift+enter)
  - Łatwa instalacja pakietów

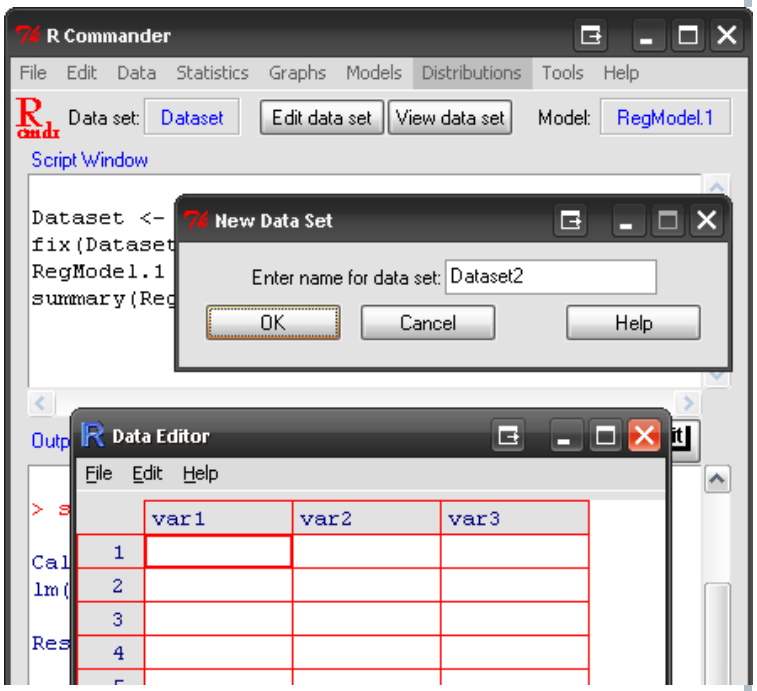
Praca z **R** studio



R Commander

- Interfejs okienkowy dla środowiska R
- Instalacja:
  - *install.packages("Rcmdr",dependencies=TRUE)*





Podstawy składni języka R

Wszystko z czym mamy do czynienia z R jest **obiektem**

- Obiektem mogą być:
  - Zmienne
  - Macierze
  - Funkcje
  - Itd..
- Obiekty posiadają nazwy i inne atrybuty takie jak np. wymiary, nazwy kolumn

Czym jest zmienna w informatyce?

- Zmienne są sposobem przechowywania informacji w komputerze
- Miejsce w pamięci które trzeba odpowiednio nazwać i zaadresować
- Np.
  - tworzymy zmienną o nazwie „wiek” i przechowujemy w niej wiek danej osoby
  - tworzymy zmienną kot….

Zmienne w R

- Etapy procesu tworzenia zmiennej:
  1. Podaj nazwę zmiennej np. *lata*
  2. Dodaj znak przypisania wartości do zmiennej „**<-**” lub znak równości „=”
  3. Dodaj zawartość zmiennej np. 5 i naciśnij enter
  4. Sprawdź zawartość zmiennej przez wpisanie jej nazwy. W konsoli powinien pojawić się wynik

- Przykłady:

```
> lata<-15
> lata
[1] 15
```

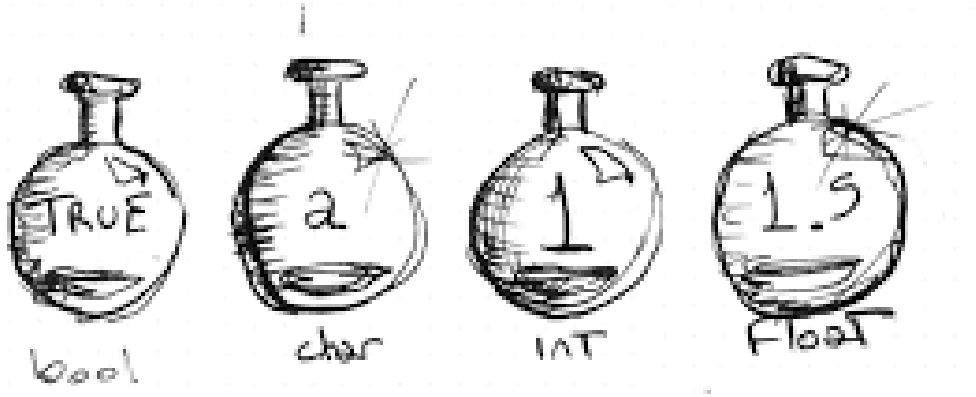
```
> imiona<-c("Ania","Kasia","Basia")
> imiona
[1] "Ania"  "Kasia" "Basia"
```

Zmienne w R

- Nazwy zmiennych:
  - Istotna jest wielkość liter
  - Powinny zaczynać się od litery
  - Mogą składać się z liter, cyfr, kropek, podkreślników
  - Nie może zawierać m.in. spacji

Typy zmiennych

- Typ liczbowy
  - Liczby całkowite, rzeczywiste
  - Kropką dziesiętna jest **kropka**, nie przecinek
  - **NaN** - wartość specjalna z ang. „not a number” czyli „nie liczba”
  - Oznaczenia Inf oraz –Inf – to plus i minus nieskończoność



Typy zmiennych

- Typ czynnikowy (kategoryczny, ang. *factor*)
  - Przydatny do przechowywania wartości występujących na kilku poziomach (w kilku kategoriach) → zmienne o charakterze jakościowym np. płeć, rasa, wykształcenie
  - Przechowuje informacje o powtórzeniach takich samych wartości oraz o zbiorze unikalnych wartości tego ciągu
  - Najczęściej służy do definiowania grup
  - Zmienne takie tworzymy z użyciem funkcji factor()

```
> plec<- factor(c("samiec","samica","samica","samiec","samiec"))
> plec
[1] samiec samica samica samiec samiec
Levels: samica samiec
> summary(plec)
samica samiec
     2      3
```

Typy zmiennych

- Typ znakowy (character)
  - Wartościami obiektów są napisy (łańcuchy znaków)
  - Rozpoczynają się i kończą znakiem ” lub ‘
  - Znaki specjalne:
    - \n – znak nowej linii
    - \t – znak tabulacji
  - Z łańcuchów znaków można wycinać fragmenty, sklejać, wyszukiwać podciągi znaków, itd.

```
> "To jest napis"
[1] "To jest napis"
> 'To również'
[1] "To również"
> 'AAAACCCGTGTGGTTTGGG'
[1] "AAAACCCGTGTGGTTTGGG"
```

Typy zmiennych

- Typ logiczny (logical)
  - Obiekty tego typu przechowują jedną z dwóch wartości:
    - logiczną prawdę (TRUE lub jego skrót T, odpowiednik liczbowy 1)
    - logiczny fałsz (FALSE lub jego skrót F, odpowiednik liczbowy 0)
  - Dzięki nim można wykonywać operacje logiczne, np.

```
> a<-TRUE
> b<-"TRUE"
> a==b
[1] TRUE
```

```
> a<-F
> b<-FALSE
> a==b
[1] TRUE
```

  - Wartości logiczne są szczególnie istotne przy instrukcjach warunkowych

Wektor (vector)

- Jest to ciąg zmiennych tego samego typu, np. liczb, znaków, wartości logicznych
- Do tworzenia wektorów służy funkcja **c()**
- Przykładowe funkcje:
  - *length(wektor)* – długość wektora
  - *sort(wektor)* – sortowanie, rosnąco
  - sample(wektor, *length(wektor) ,FALSE)*

```
> liczby<-c(1,2,3,4,45,7)
> logiczny<-c(T,F,T,F)
> tekst
[1] "Ola"   "Zosia" "Kasia"
> liczby
[1]  1  2  3  4 45  7
> logiczny
[1]  TRUE FALSE  TRUE FALSE
```

Wektor (vector)

- Wektory są strukturami, na których można wykonywać praktycznie wszystkie działania arytmetyczne. Np.:
  - pierwiastek z wektora jest wektorem, którego elementy są pierwiastkami elementów oryginalnego wektora
  - suma dwóch wektorów o tej samej długości jest wektorem zawierającym sumy poszczególnych elementów (dwa pierwsze, dwa drugie etc.)

- Jeśli wektory mają różne długości, to po dojściu do końca pierwszego wektora obliczenia zaczynają się od początku.
  - w konsoli wyświetlane jest ostrzeżenie

Indeksowanie

- Odnosimy się do konkretnych elementów stanowiących podzbiór całości
  - otrzymujemy wektor ale jest on krótszy i zawiera jedynie elementy z wyjściowego wektora
  - polega na dodaniu wyrażenia w nawiasach kwadratowych

```
> tekst
[1] "Ola"   "Zosia" "Kasia"
> tekst[2]
[1] "Zosia"
> tekst[c(1,3)]
[1] "Ola"   "Kasia"
```

Indeksowanie

- Jako indeksu możemy użyć:
  - Wektora zawierającego liczby naturalne (od 1 do długości wektora)
  - Wektora zawierającego ujemne liczby całkowite
    - wektor zawiera wszystkie elementy oprócz wskazanych
  - Wektora logicznego o tej samej długości
    - otrzymany wektor zawiera tylko te liczby dla których indeks zawiera wartość TRUE

Macierz (*matrix*)

- Jest wektorem, dodatkowo zawierającym informacje o uporządkowaniu elementów
- Najczęściej stosowana jest macierz dwuwymiarowa
- Indeksowanie:
  - Macierz[wiersz, kolumna]

```
> A<-matrix(c(2,4,5,6,7,8),ncol=2)
> A
     [,1] [,2]
[1,]    2    6
[2,]    4    7
[3,]    5    8
> A[1,1]
[1] 2
```

Ramka danych (*data frame*)

- Macierz (tablica), której poszczególne kolumny sa wektorami lub/i czynnikami o tej samej długości
- Kolumny mogą różnic się pomiędzy sobą typem elementów, jakie zawierają.
- Konstruktorem jest funkcja data.frame

```
> head(possum)
    case site Pop sex age hdlngth skullw totlngth taill
C3     1    1 Vic   m   8    94.1   60.4     89.0  36.0
C5     2    1 Vic   f   6    92.5   57.6     91.5  36.5
C10    3    1 Vic   f   6    94.0   60.0     95.5  39.0
C15    4    1 Vic   f   6    93.2   57.1     92.0  38.0
C23    5    1 Vic   f   2    91.5   56.3     85.5  36.0
C24    6    1 Vic   f   1    93.1   54.8     90.5  35.5
```

Funkcje

- Wydzielona część programu do wielokrotnego wykorzystania
  - Schemat instrukcji, które mają zostać wykonane na wybranych argumentach
- Jej konstruktorem jest funkcja *function()*

mojafunkcja <- function(*argument1, argument2, ...*)
{
statements
return(*object*)
}

```
funkcja<-function(a,b){
  c<-(a+b)/10
  wynik<-sqrt(c)
  print(wynik)
}
```

```
> funkcja(3,8)
[1] 1.048809
```

Pakiety

- Na przykładzie pakietu Rcmdr
- Instalacja:
  - *install.packages("Rcmdr",dependencies=TRUE)* – instaluje pakiet Rcmdr oraz wszystkie wspomagające jego działanie
  - Wyświetli sie lista serwerów z repozytoriami - wybieramy najbliższy (szybkość)
- Funkcja library(*Rcmdr*):
  - Ładowanie biblioteki do aktywnej przestrzeni roboczej (workspace)
- Aby wyświetlić listę załadowanych bibliotek używamy tej samej funkcji, ale bez argumentu

Dziękuje za uwagę