Język R

Postawowe informacje

Język programowania oraz implementujący go system R jest środowiskiem do statystycznej analizy danych. Umożliwia pracę interaktywną i skryptową.

R jest zainstalowany :

- W pracowni 4.29 i 4.30
- na serwerze jabba.icsr.agh.edu.pl (dostep bezposredni)
- na klastrze vcluster (dostepny przez gandalfa)
- Na terminalach PCoIP (obraz WIN 10 ICSR System Podstawowy), wygodnym narzędziem tam zaistalowanym jest RStudio.

Instalacja na Ubuntu:

apt-get install r-base

- Strona główna projektu R
- Quick R:
- RStudio

Interaktywna praca z systemem

- Uruchamiamy konsolę R bezpośrednio poleceniem R
- Otwarcie pomocy w przeglądarce:

```
help.start()
```

- Pomoc w konsoli:

```
help()
help(plot)
```

- bieżący katalog

```
getwd()
```

- zmiana katalogu bieżącego

```
setwd("/my/directory/")
```

- Proste obliczenia

```
> 2+2
[1] 4
```

- przypisanie wiecej o rodzajach przypisania

```
>  a<-2
>  a=2
```

- dzialanie

```
>  a*2
[1] 4
```

- Listy

```
x = c(1,2,3,4)
y = c(2,4,6,8)
z = x*2
z
[1] 2 4 6 8
x+y
[1]  3  6  9 12
```

- Funkcje

```
seq(1,10)
[1]  1  2  3  4  5  6  7  8  9 10
x = seq (1,10)
sum(x)
[1] 55
sqrt(x)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
[9] 3.000000 3.162278
```

- Wyjście z konsoli

```
q()
Save workspace image? [y/n/c]: y
```

Import i przekształcanie danych

R umożliwia wczytywanie danych z baz danych i plików w różnych formatach.

- Przykład: import z pliku CSV
  Do importu służy funkcja read. Przykładowy plik: data1.txt. Kolumny t1 i t2 zawierają wyniki pomiarów czasu dla 2 algorytmów.

```
results = read.csv("data1.txt")

results
    n   t1   t2
1   1  1.1  2.3
2   2  2.0  4.1
3   3  2.8  6.0
4   4  4.2  8.2
5   5  5.1  9.9
6   6  5.9 12.1
7   7  6.7 14.1
8   8  8.0 15.9
9   9  9.1 18.1
10 10 10.1 19.9
```

  Zaimportowane dane są w postaci tabeli (data frame).
- Pobranie kolumny jako wektor kolumnowy lub lista:

```
results[1]
    n
1   1
2   2
3   3
4   4
5   5
6   6
7   7
8   8
9   9
10 10

results[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10
```

- pobranie kolumny po nazwie:

```
results["t1"]
     t1
1   1.1
2   2.0
3   2.8
4   4.2
5   5.1
6   5.9
7   6.7
8   8.0
9   9.1
10 10.1
```

- Pobranie wiersza lub wierszy:

```
results[2,]
  n t1  t2
2 2  2 4.1

results[c(2,4,6),]
  n  t1   t2
2 2 2.0  4.1
4 4 4.2  8.2
6 6 5.9 12.1
```

- attach pozwala na pojedyncze odwolania do kolumn bez odwolywania sie do calej tabeli:

```
> attach(results)
> t1
 [1]  1.1  2.0  2.8  4.2  5.1  5.9  6.7  8.0  9.1 10.1
> t2
 [1]  2.3  4.1  6.0  8.2  9.9 12.1 14.1 15.9 18.1 19.9
> n
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Rysowanie wykresów

  Uzywamy pakietu ggplot2

```
> install.packages("ggplot2")
> library("ggplot2")
```

Rysowanie wykresu:

```
> ggplot(results,aes(n,t2)) + geom_point()
> last_plot() + geom_line(data=results, aes(n,t1))
```

Przykład analizy danych

- Przetwarzanie danych w tabeli
  Dane w pliku data2.txt mają postać tabeli faktów, w której wiersze są wynikami pomiarów.

```
results = read.csv("data2.txt")

results
     n alg       time         error
1    1  t1  1.0105721  0.0105721477
2    2  t1  1.9816890 -0.0183109931
3    3  t1  3.0662680  0.0662680427
4    4  t1  4.0602206  0.0602205696
5    5  t1  5.0295021  0.0295020510
[...]
```

- Dodanie nowej kolumny:

```
results$speed = results$n / results$time

results

     n alg       time         error     speed
1    1  t1  1.0105721  0.0105721477 0.9895385
2    2  t1  1.9816890 -0.0183109931 1.0092401
3    3  t1  3.0662680  0.0662680427 0.9783880
4    4  t1  4.0602206  0.0602205696 0.9851682
[...]
```

- Grupowanie (agregacja)
  Przyklad: zastosowanie funkcji FUN (tutaj: mean czyli obliczającej średnią) do wszystkich wartosci time dla takich samych n i alg:

```
avg_results = aggregate( time ~ n:alg, data=results, FUN=mean)
> avg_results
    n alg       time
1   1  t1  0.9990494
2   2  t1  2.0152520
3   3  t1  2.9867132
[...]
10 10  t1  9.9948604
11  1  t2  1.9878299
12  2  t2  3.9599609
13  3  t2  5.9861750
[...]
20 10  t2 19.9883028
```

- Dołączenie kolumny z obliczonym odchyleniem standardowym (sd) do tabeli:

```
avg_results$sd = aggregate( time ~ n:alg, data=results, FUN=sd)$time
```

- Agregacja kilku wielkości:

```
avg_results = aggregate( cbind(time,speed) ~ n:alg, data=results, FUN=mean)
```

- Filtrowanie danych:

```
avg_results[avg_results$alg=="t1",]
```

- Rysowanie wykresów.
  Wykres time w funkcji n:

```
ggplot(avg_results, aes(n,time)) + geom_point()
```

- Przykład skryptu rysującego wykres. Skrypt data2_plot.R uruchamiamy poleceniem:
  W wersji okienkowej: Plik -> Otworz skrypt i potem "uruchom linijke lub zaznaczenie"
  albo w konsoli:

```
source("data2_plot.R")
```

w bashu za pomoca polecenia:

```
Rscript data2_plot.R
```

W wyniku powstaje plik: myplot.png

Przyklad aproksymacji wielomianami

- Generujemy ciąg punktów, który będziemy aproksymować

```
x = c(32,64,96,118,126,144,152.5,158)
y = c(99.5,104.8,108.5,100,86,64,35.3,15)
mydata = data.frame(x,y)
```

- Fitujemy zwyklym wielomianem stopnia 3

```
fit = lm(y ~ poly(x, 3, raw=TRUE), data=mydata)
```

- Mozemy wyswietlic wspolczynniki wielomianu:

```
fit
```

- Rysujemy wykres punktow:

```
ggplot(mydata,aes(x,y)) + geom_point()
```

- Generujemy gestą listę punktów dla wyliczenia wartości wielomianu aproksymujacego

```
newdata = data.frame(x = seq(30,160, length.out=250))
```

- Dodajemy do wykresu krzywą wielomianu

```
newdata$y = predict(fit, newdata)
last_plot() + geom_line(data=newdata, aes(x,y))
```

- Multiple Regression in R

Zadanie

Za pomocą języka R przeanalizować dowolne dane z poprzednich laboratoriów. Elementy obowiązkowe:

- W jednej tabeli języka R należy umieścić wyniki dwóch różnych eksperymentów (t.j. czas działania dwóch różnych funkcji) dla różnych parametrów (t.j. rozmiarów wektora(macierzy)).
- Tabela powinna zawierać dane z 10-krotnego uruchomienia tego samego eksperymentu dla tych samych parametrów (dla każdego takiego zestawu)
- Należy przedstawić wykresy średnich czasów obliczenia wybranych funkcji w zależności od rozmiaru wektora (macierzy)
- Wykresy powinny zawierać informację o odchyleniu standardowym dla uzyskanych wyników. Słupki błędów można narysować korzystając z funkcji geom_errorbar pakietu ggplot2 .
  Uwaga, potrzebna może być instalacja biblioteki ggplot2. W RStudio wystarczy dodać odpowiedni pakiet w zakładce "Packages" w prawym dolnym rogu oraz oznaczyć ten pakiet jako "user library".

  Przy korzystaniu z konsoli:

```
install.packages("ggplot2")
library("ggplot2")
```

- Uzyc aproksymacji wielomianowej dostepnej w jezyku R do znalezienia odpowiednich wielomianow, ktore najlepiej pasuja do wynikow kazdego z algorytmow. Dodac wykresy uzyskanych wielomianow do wykresu.