Wprowadzenie do R dla programistów innych języków

Artur Suchwałko, quantup.pl

2014-02-23

Wstęp

Dokument to możliwie krótkie wprowadzenie do systemu R dla osób znających inne języki programowania. Celem jest jak najszybsze opanowanie podstaw R i nauczenie się operacji znanych z innych narzędzi. Nie jest celem przedstawianie metod statystycznych czy metod analizy danych.

W dokumencie znajdują się krótkie wskazówki: jakie komendy poznać, czego się nauczyć oraz informacje, co jest ważne w pracy z R.

Przeczytanie tego dokumentu powinno zająć najwyżej godzinę. Poczytanie na temat wymienianych funkcji, przećwiczenie ich używania i nauczenie się posługiwania się wymienionymi pakietami na pewno zajmie dużo więcej czasu.

Oczywiście, jeśli to nie wystarcza, to trzeba sięgnąć po inne źródła (łatwo znaleźc w Sieci) albo skorzystać ze szkoleń firmy QuantUp.

Zakładam, że czytelnik:

- Wie, co potrafi R i że R mu się przyda – jeśli nie, to zachęcam do przeczytania krótkiej informacji zachęcającej.
- Zna przynajmniej podstawy języka angielskiego – do samodzielnego czytania dokumentacji i poszukiwania informacji w Internecie.
- Umie używać jakiegoś języka programowania. Jakiego i jak dobrze – w zasadzie bez znaczenia.
- Chce poświęcić czas na naukę R – inaczej żadna książka czy materiały nie pomogą.

Powodzenia w nauce R!

1 Wprowadzenie

1.1 Podstawy

Instalacja i początek

- Ściągnij R (dla Windows: http://cran.at.r-project.org/bin/windows/base/)
- Żeby w razie otrzymania niezrozumiałych komunikatów o błędach móc korzystać z pomocy w Sieci, zmień język na angielski:
  - odznacz tłumaczenie treści podczas instalacji R
  - a jeśli R jest już zainstalowany, to zmień na `LANGUAGE=en` w pliku `Rconsole` (linia około 70) w katalogu, w którym zainstalowany jest R
- Zainstaluj RStudio http://www.rstudio.com/ide/ lub Notepad++ http://notepad-plus-plus.org/ + NppToR http://sourceforge.net/projects/npptor/.
- RStudio jest bardzo wygodnym narzędziem integrującym wiele funkcjonalności, np. do pracy z `knitr`. Podstawowa komenda: `Ctrl + Enter` przesyła fragment skryptu z edytora do R.
- Jak czegoś nie wiesz: szukaj w Sieci.
- Gdy czegoś nie rozumiesz: eksperymentuj i szukaj dalej.

Jak działa R?

- R jest językiem programowania.
- Działa w trybie interaktywnym: piszemy polecenie i otrzymujemy (lub nie) odpowiedź od R.
- To od nas zależy, czy otrzymamy odpowiedź.
- Wynik działania polecenia możemy przypisać do zmiennej: `x <-2 + 2`. Wtedy odpowiedzi nie uzyskamy.
- Odpowiedź uzyskamy np. w sytuacji: `summary(x)`.
- Wyświetlenie obiektu: `x` (i `Enter`) albo `print(x)`.
- Na takim obiekcie możemy wykonać kolejne polecenia: `y <-x + 3`.
- Wszystkie obiekty znajdują się w przestrzeni roboczej. O tym później.

Praca z R w praktyce

- Powtarzanie ostatniego polecenia: strzałka w górę.

- Czyszczenie konsoli: `Ctrl + L`
- `+` na początku linii oznacza niedokończone polecenie. Dokończ lub naciśnij `Esc`.
- Ustaw katalog roboczy tam, skąd chcesz odczytywać i gdzie chcesz zapisywać pliki: `getwd`, `setwd`.
- Instalacja pakietów: `install.packages(...)`. Wczytywanie zainstalowanego pakietu: `library(nazwa.pakietu)`.
- Możesz zapisać historię poleceń: polecenie `Save History...` z menu `File`.
- Wyjście z R: `q()`.
- Wczytywanie kodu z pliku: `source`.
- Zapoznaj się z menu R.

Pomoc i materiały

- Podstawa: `?` lub `help`. Uwaga: spróbuj `?if` oraz `?"if"`
- Przykład: `example(plot)`
- Pomoc w html: `help.start()`
- StackOverflow, pytania oznaczone przez `r`: http://stackoverflow.com/questions/tagged/r. Część programistów udzielających się w tym serwisie jest bardzo aktywna – łatwo i szybko można otrzymać pomoc.
- Jak zadawać pytania, żeby otrzymać odpowiedź: http://rtfm.killfile.pl/.
- Google: dodaj do zapytania twitterowy `\#Rstats`
- Dedykowana wyszukiwarka: http://rseek.org
- Agregator blogów: http://www.r-bloggers.com

1.2 Proste, techniczne

Przypisanie

- `x <-2`, możliwe też `2 -> x`
- Unikamy `=`, chociaż jest możliwe.

Nazwy zmiennych

- Jak w innych językach. Rozróżnia się małe i wielkie litery.
- Separatorem w nazwach jest kropka, ale może być też `_`, np. `macierz.reszt`. Zamiast `.` z innych języków, np. do dostępu do pól, używa się `$`.
- Uwaga na jednoliterowe zmienne lub funkcje używane przez R. Można je zepsuć przez przypisanie: `c`, `q`, `s`, `t`, `C`, `D`, `F`, `I`, `T`.

Komentarze

- Rozpoczynają się znakiem `\#` i obejmują całą linię.
- Nie ma komentarzy blokowych.
- Ale za to w RStudio po zaznaczeniu bloku naciskając `Ctrl + Shift + C` wstawiamy automatycznie komentarz w każdej linii z tego bloku.

Obiekty

- Obiekty w przestrzeni roboczej: `ls()`
- Usuwanie obiektów: `rm()`

2 Rodzaje obiektów

2.1 Wektory

Wektory – podstawy

- Wektor to podstawowy typ w R.
- Wektor zawiera elementy jednego typu (`logical`, `integer`, `double`, `character`).
- Oczywiście, ten typ może być różny dla różnych wektorów.
- Liczba jest wektorem jednoelementowym.
- Indeksujemy od `1`.
- Wektory tworzy się korzystając z funkcji `c`, np. `c(2, 3, 17)`.

Wektory – operacje

- Operacje na wektorach wykonywane są element po elemencie, np. `x *2` wygeneruje wektor, którego elementami są pomnożone przez `2` elementy `x`
- Uwaga na dodawanie wektorów różnej długości! Spróbuj `rep(2, 3)+ rep(4, 6)` oraz `rep(2, 3)+ rep(4, 5)` i na koniec `rep(2, 3)+ 4`. To jest tzw. recycling rule.
- Dostęp do wektora `x <-c(1, 2); x[2]`. Można używać zmiennych logicznych `x[TRUE, FALSE]`.
- Ciągi elementów: `1:17`, funkcja `seq`, np. `seq(from = 1, to = 10, by = 2)`.
- Sortowanie: `sort`, `order`.
- Przetestuj działanie funkcji `which`: `which(1:17 > 3)`

2.2 Typy obiektów

Typy obiektów

- Wektor, macierz, ramka danych, ...
- Konwersje między typami: funkcje `as.*()`, popatrz np. na `as.data.frame()`.
- Sprawdzenie typów obiektów: funkcje `is.*()`, popatrz np. na `is.numeric()`.
- Właściwości obiektów
  - `class()` – klasa obiektu, można modyfikować
  - `typeof()` – typ obiektu,
  - `mode()` – sposób pamiętania obiektu,
  - `length()` – długość obiektu,
  - `attributes()` – atrybuty obiektu; można je zmieniać z pomocą tej samej funkcji,
  - `attr()` – dostęp do wybranych atrybutów obiektu
- Uwaga: Na typy zmiennych patrzymy tutaj głównie z punktu widzenia analizy danych.

2.3 Operacje logiczne

Operacje logiczne

- Stałe: `T` lub `TRUE`, `F` lub `FALSE`.
- Można tworzyć wektory wartości logicznych.
- Operatory `&` i `|` działają element po elemencie.
- Operatory `&&` i `||` zwracają wektor jednoelementowy.
- Zapoznaj się z `any()` i `all()`.

2.4 Macierze

Macierze

- Tworzenie macierzy z wektora `matrix(1:12, 3, 4)`. Wiersz po wierszu: `x <- matrix(1:12, 3, 4, byrow = T)`.
- Możliwe są wszystkie standardowe operacje macierzowe, np. `t(x)`.
- Operacje arytmetyczne (np. `+`, `log`) wykonywane są element po elemencie.
- Do elementów dostajemy się tak: `x[2, 3]`. Sprawdź różnicę między `x[2, 3]` a `x[2, 3, drop = F]`.

- Dostęp do kolumn: `x[, 1]` i do wierszy: `x[1:2, ]`.
- Uwaga na zasadę recyclingu: sprawdź, jak działa `matrix(1:6, 2, 3)+ c(1, 4)`
- Nazwy wierszy i kolumn: `rownames`, `colnames`
- Wektoryzacja operacji (ważne!): `apply`, `sapply`
- Łączenie macierzy: `rbind` (wierszami), `cbind` (kolumnami)

2.5 Listy

Listy

- Tak, jak wektory, ale elementy mogą być różnych typów.
- Tworzenie listy: `x <-list(nazwisko = "Iksinski", wiek = 38, dzieci = c(12, 3))`
- Informacja o zawartości: `attributes(x)`
- Do elementów dostajemy się tak: `x[[1]]`, `x$nazwisko`
- Uwaga: `x[1]` to lista jednoelementowa złożona z pierwszego elementu.
- Dostęp `x[[4]]` nie zadziała, ale przypisanie `x[[4]] <-2` spowoduje powiększenie listy.
- Operacja element po elemencie na liście: `lapply`

2.6 Tekst

Tekst

- Tworzenie napisów: `x <-"to jest napis"`.
- Wypisywanie tekstu: `cat`, `print`.
- Łączenie napisów: `paste`, `paste0`.
- Do operacji na tekście jak najszybciej zacznij korzystać z pakietu `stringr`, jest łatwiejszy w użyciu od standardowych funkcji R (z `base`).
- Możliwe jest wykorzystywanie wyrażeń regularnych.

3 Dane i ich analiza

3.1 Ramki danych (data frames)

Ramki danych – I

- Odpowiedni typ służący do przechowywania danych nazywa się `data.frame`. Można myśleć o nim jak o prostokątnej tabeli lub tabeli w bazie danych.
- W poszczególnych kolumnach zmienne mogą być różnych typów. Poczytaj o typie `factor` (zmienna czynnikowa).
- Obejrzyj przykładową ramkę danych: `library(MASS); data(Cars93)`.
- Ramka danych jest zaimplementowana w R jako lista kolumn.
- Do kolumn dostajemy się np. tak `dane$nazwa.kolumny`.
- Do elementów dostajemy się jak do elementów macierzy.

Ramki danych – II

- Poczytaj o funkcjach:
  - `names`, `rownames` (nazwy),
  - `dim`, `nrow`, `ncol` (wymiary),
  - `apply`, `aggregate`, `subset` (operacje).
- Wczytywanie i zapisywanie danych z plików tekstowych: `read.table`, `write.table` .
- Zapisywanie i odczyt danych binarnych: `save`, `load`.
- Najważniejsze parametry tych funkcji: `sep`, `header`, `dec`.

3.2 Statystyki opisowe

Statystyki opisowe

- Wypróbuj działanie funkcji dla kolumn i całych danych: `mean`, `min`, `summary`, `range` oraz `table`.
- Podstawowe wykresy: `plot`, `pie`, `barplot`, `hist`, `dotchart`.

3.3 `Not a Number` (`NaN`) i brakujące wartości (`NA`, `Not Available`)

`NaN` i `NA`

- Operacje na liczbach mogą zwrócić `NaN`, np. `0/0`.
- Sprawdzamy ich obecność funkcją `is.nan`.
- Brakujące wartości są kodowane przez `NA`.
- Odpowiednie kodowanie jest bardzo ważne w analizie danych.

- Sprawdzamy ich obecność funkcją `is.na`.
- Obsługa: przyjrzyj się dokumentacji funkcji `na.omit`.

4 Programowanie

4.1 Funkcje

Funkcje – wywoływanie

- Wywoływanie funkcji – przećwicz na przykładzie: `seq()`, `seq(1)`, `seq(1, 2)`, `seq(to = 2, from = 1)`, `seq(1, 2, length.out = 3)`
- Pamiętaj o nawiasach wywołując funkcję bezargumentową, np. tak `seq()`.
- Parametry przekazywane są przez wartość.
- Można użyć zmiennych globalnych aby przekazać przez referencję. Oczywiście, to jest brzydkie rozwiązanie.

Funkcje – tworzenie

- Do stworzenia funkcji używamy funkcji R `function` w taki sposób:

```
fun <- function(x, delta = 2)
{
  y <- x + delta + 1
  y
}
```

- Domyślne wartości – pokazane powyżej.
- Możliwe jest użycie zmiennej liczby argumentów. Patrz: ....

Funkcje – jak zwrócić więcej wartości?

- Żeby zwrócić więcej niż jedną wartość, umieszczamy wyniki na liście. **Uwaga**: To jest bardzo użyteczne rozwiązanie.

```
policz.2.3 <- function(x)
{
  return(list(x.2 = x^2, x.3 = x^3))
}
```

4.2 Błędy i wyjątki

Błędy i wyjątki

- Funkcja `message("Licze...")` generuje komunikat diagnostyczny.
- Funkcja `warning("Jest problem...")` generuje ostrzeżenie.
- Funkcja `stop` zatrzymuje bezwarunkowo wykonanie programu generując błąd.
- `stopifnot` zatrzymuje program warunkowo.
- `try` używamy do wyliczenia wyrażenia, które może powodować problem, np. `try(log("a"))`. Dzięki temu nie przerywamy wykonania programu. Można też wyłączyć komunikaty: `options(show.error.messages = FALSE)`
- `tryCatch` umożliwia wykorzystanie własnych komunikatów o błędach (handlerów).
- Poczytaj więcej: `?conditions`

4.3 Sterowanie przepływem kodu

Instrukcje warunkowe: `if` i `ifelse`

```
if (liczba %% 2) {
  cat("nieparzysta")
} else {
  cat("parzysta")
}

ifelse(1:5 > 2, "wariant 1", "wariant 2")
```

Pętla `for`

```
for (i in 1:5) cat(i)
for (i in 5:1) cat(i)
for (i in c("a", "b")) cat(i)
```

Pętla `while`

```
liczba <- 7
while (liczba > 0) {
  cat(liczba)
  liczba <- liczba - 1
}
```

5 Podstawy grafiki

Funkcje niskiego i wysokiego poziomu

- Funkcja wysokiego poziomu otwiera okno graficzne oraz rysuje wykres, funkcja niskiego poziomu dorysowuje obiekty.
- Prześledź przykład: `plot(1:2, 2:3)`, `abline(h = 2.5)`
- Najważniejsza funkcja wysokiego poziomu: `plot()`
- Zapoznaj się z podstawowymi funkcjami niskiego poziomu:
  - `abline()`, `lines()`, `points()`, `text()`,
  - `title()`, `axis()`, `legend()`.

Parametry graficzne i pozostałe sprawy

- Zapoznaj się z podstawowymi parametrami funkcji `plot`: `type` (typ wykresu; szczególnie ważny jest `type = "n"`), `col` (kolor), `xlim` i `ylim` (zakresy osi – wektory), `xlab` i `ylab` (etykiety osi), `main` (tytuł), `cex` (wielkość), `lty` (typ linii), `lwd` (grubość linii)
- Wiele wykresów na jednym: `par(mfrow=c(w,k))`
- Poczytaj o funkcji `par()`.
- Dowiedz się, jak działają urządzenia graficzne, np. `pdf()`. Używając urządzeń graficznych zawsze pamiętaj o `dev.off()`.

6 Efektywna praca w R

Nauka R

- Nieustannie ucz się R.
- Ucz się programować w R oraz używać narzędzi pomocniczych.
- Notuj sobie nazwy często używanych funkcji w słowniczku. Błyskawicznie się je zapomina.
- Subskrybuj podsumowania z list dysksyjnych o R.
- Odwiedzaj R-Bloggers: http://www.r-bloggers.com/

Organizacja pracy

- Rób wszystko porządniej, niż się wydaje, że trzeba.

- Miej porządek w plikach i materiałach do pracy.
- Miej każdą analizę w osobnym katalogu oraz porządek w każdym z tych katalogów.
- Notuj więcej, niż się wydaje, że trzeba.
- Pamiętaj o komentarzach. Wklejaj do kodu linki i informacje o źródłach. Pamiętaj o `getwd` i `setwd` w kodzie.
- Pisz kod starannie, zgodnie z zasadami kodowania: odpowiednie nazwy, wcięcia, odstępy itp.

Organizacja pracy – automatyzacja i powtarzalność

- Automatyzuj wszystko, co się da. Przydaje się w najmniej spodziewanym momencie. Uważaj, żeby nie wkładać w to więcej pracy niż warto.
- Zawsze pisz skrypt (z rozszerzeniem `.R`) wykonujący analizę. Staraj się, żeby analizy były powtarzalne (szukaj: ''reproducible analysis / research").
- Można zapisywać workspace i pliki binarne z krokami analizy, ale lepiej zapisywać kod, który ją odtworzy.
- Staraj się przechowywać jak najwięcej wyników w postaci kodu źródłowego, który je wygeneruje i jak najmniej w postaci binarnej (reproducible research!).
- Czasami warto zrobić pakiet. Zrobienie pierwszego może być trudniejsze, ale z każdym następnym będzie łatwiej...
- Koduj pliki w UTF-8. To ułatwia używanie różnych pomocniczych narzędzi.
- Używaj VCS (systemu kontroli wersji).

Środowisko pracy

- Używaj dobrego edytora i naucz się jego skrótów klawiaturowych (np. Notepad++ i NppToR, TINN-R, RStudio).
- Spróbuj korzystać ze zintegrowanych środowisk typu Eclipse + StatET lub ESS (Emacs Speaks Statistics).
- Znajdź odpowiednie dla siebie środowisko wspierające pracę z R (http://www.sciviews.org/_rgui/ dosyć stare zestawienie). Uwaga: niektóre nie działają dobrze z wieloma monitorami.
- Dowiedz się, jakiego GUI używają inni: http://www.kdnuggets.com/polls/2011/r-gui-used.html

- Dostosuj środowisko pracy, np. zmiana kolorów w edytorze na ciemne tło i jasne litery.

7 Inne sprawy

7.1 R jako język programowania

- Jest językiem interpretowanym. Oznacza to między innymi, że pętle w R wykonują się bardzo wolno i należy unikać ich stosowania.
- Możliwe (ale nie niezbędne) jest programowanie obiektowe.
- Można łączyć kod w R z kodem w C/C++, Javą, C# i innymi językami. Wystarczy poszukać.
- Jest dosyć podobny do Matlaba.

7.2 R w trybie wsadowym

R w trybie wsadowym

- To wygodny sposób automatycznego wykonywania programów R, np. o ustalonych godzinach.
- Użyj polecenia typu: `\"C:\\Program Files\\R\\R-3.0.1\\bin\\R.exe\"CMD BATCH --vanilla --slave \"moj_skrypt.R\"`
- Wyniki (tekst, obrazki) trafiają do plików o standardowej nazwie, dopóki nie obsłużysz tego w specjalny sposób w kodzie.

7.3 Inżynieria oprogramowania

Inżynieria oprogramowania

- Edytory: Notepad++
- Środowiska IDE: RStudio, Eclipse + StatET
- Testy jednostkowe: `testthat`
- Dokumentacja: `roxygen2`
- Raportowanie: pakiet `knitr`

7.4 Dodatkowe pożyteczne funkcje

Dodatkowe pożyteczne funkcje

- `sessionInfo()` – wykorzystywane pakiety
- `R.Version()` – wersja R

7.5 Naucz się później

Naucz się później

- Operacje na danych, np. grupowanie: pakiet `reshape2`
- Grafika: np. http://www.statmethods.net/graphs/index.html, pomyśl o nauce `ggplot2` http://docs.ggplot2.org/current/
- Operacje na tekście: pakiet `stringr`
- Aplikacje webowe: `Shiny`
- Wektoryzacja operacji.
- Jeśli potrzebujesz szybkich operacji na danych, np. grupowania, to poczytaj o pakiecie `dplyr`.