Odtwarzalne analizy i obliczenia

Przemysław Biecek

5 XII 2015

Odtwarzalność

Za Wikipedią: [1]

Reproducibility is the ability of an entire experiment or study to be duplicated, either by the same researcher or by someone else working independently. Reproducing an experiment is called replicating it. Reproducibility is one of the main principles of the scientific method.

Ostatnie zdanie, sugeruje, że jest to bardzo ważne pojęcie. Więc rozpoczniemy od ćwiczenia.

Znajdź trzy powody, dla których powtarzalność analiz jest ważna (a może uważasz, że nie jest taka ważna?).

-
-
-

[1] `https://en.wikipedia.org/wiki/Reproducibility`

Knitr

Jednym z najbardziej użytecznych narzędzi do tworzenia powtarzalnych analiz jest pakiet `knitr`.[2] Jest on często postrzegany jako następca funkcji `Sweave` [3] i koncepcji `literate programming`.

Wszystkie materiały do tego szkolenia zostały zbudowane z użyciem pakietu `knitr`. Pozwala on na łączenie kodu R z opisem w języku naturalnym. Jako wynik otrzymuje się raport, który łatwo odtworzyć, łatwo też zrozumieć. Pakiet `knitr` umożliwia wygenerowanie raportu do formatu `doc`, `pdf`, `md`, `html` a dzięku użyciu programu do konwersji `pandoc` możemy też korzystać z dziesiątek innych formatów.

Program RStudio wspiera tworzenie dokumentów opartych o pakiet `knitr`. Poleceniem `Plik / Nowy plik... / R Markdown...` można zbudowac błyskawicznie szablon raportu lub prezentacji.

Kod źródłowy dla pakietu `knitr` to dokument, najczęściej w formacie Markdown, z dodanymi wstawkami. Wstawka to blok kodu, rozpoczynający się od znaków `'''{r}` a kończący się znakami `'''`. Na poniższym zdjęciu przedstawiamy fragment takiego raportu.

Wybrane użyteczne argumenty wstawek:

- `cache=`, czy wynik danego fragmentu ma być cache'owany czy nie,
- `eval=`, czy dana wstawka ma być wywołana czy nie,

[2] `http://yihui.name/knitr/`

[3] Friedrich Leisch (2002) Dynamic generation of statistical reports using literate data analysis.

- `echo=`, czy kod R ma być umieszczony w wyniku czy nie,
- `fig.height=, fig.width=`, parametry kontrolujące grafikę,
- `results=`, czy wyniki mają być wklejone do raportu czy nie,
- `warning=, message=`, czy komentarze mają być wklejone do raportu czy nie.

````
## Przykłady

Pakiet `archivist` pozwala na udostępnianie i pobieranie obiektów R z lokalnego lub zdalnego repozytorium.

```{r, message=FALSE}
library(archivist)

archivist::aread("pbiecek/graphGallery/2166dfbd3a7a68a91a2f8e6df1a44111")
```
```{r, message=FALSE, eval=FALSE}
archivist::aread("pbiecek/graphGallery/2166d")
```

Repozytorium może być udostępnione jako katalog na współdzielonym dysku, GitHubie lub może znajdować się w pakiecie R.

```{r, eval=FALSE}
setLocalRepo(repoDir = system.file("graphGallery", package = "archivist"))
archivist::aread("2166dfbd3a7a68a91a2f8e6df1a44111")
```

W repozytorium, można przechowywać dowolne obiekty, też modele statystyczne.
````

Figure 1: Fragment dokumentu w formacje Rmd. Opis w języku naturalnym przeplata się z wstawkami kodu R

Zadanie

Jakie widzisz zastosowania dla takich raportów? Spróbujmy rozszerzyć poniższą listę.

- Studenckie projekty
- Prace domowe
- Analizy naukowe danych
- Dokumentacja pakietów lub innych funkcjonalności

Zbuduj raport korzystając z szablonu Tuftego.

- Zbuduj nowy raport (File -> New Rmarkdown) lub prezentację,
- W raporcie przedstaw wykres i model liniowy (używając funkcji lm) prezentujący zależność pomiędzy ceną auta a wiekiem auta dla zbioru `auta2012`.

- Skompiluj je przyciskiem `knitr`,
- Dodaj kilka nowych wstawek tworzących wykres, tabelę lub wynik liczbowy.

Zbuduj prezentację używając pakietu `slidify`. Skopiuj do niej zawartość raportu wykonanego powyżej.

Reprodukowalność / Odtwarzalność

Wszystko płynie i nic nie pozostaje takie samo. Heraklit z Efezu

Program R oraz większość z dostępnych tysięcy pakietów bardzo szybko ewoluuje. Z roku na rok należy spodziewać się nowych wersji pakietów być może różniących się funkcjonalnością. Może się zdarzyć, że nawet na podstawie tego samego kodu źródłowego po kilku latach w nowszej wersji otrzymamy inne wyniki.

Co można zrobić by umożliwić odtworzenie uprzednio uzyskanych wyników?

Aby określić przy jakich ustawieniach wykonano określone instrukcje, wygodnie jest wykorzystać funkcję `session_info`.

```
library(devtools)
session_info()
```

```
Session info ------------
 setting  value
 version  R version 3.2.2 (2015-08-14)
 system   x86_64, darwin13.4.0
 ui       RStudio (0.99.441)
 language (EN)
 collate  en_US.UTF-8
 tz       Europe/Warsaw
 date     2015-12-04

Packages -----------------
 package  * version date       source
 devtools * 1.9.1   2015-09-11 CRAN (R 3.2.0)
 digest     0.6.8   2014-12-31 CRAN (R 3.2.0)
 memoise    0.2.1   2014-04-22 CRAN (R 3.1.0)
```

Archivist

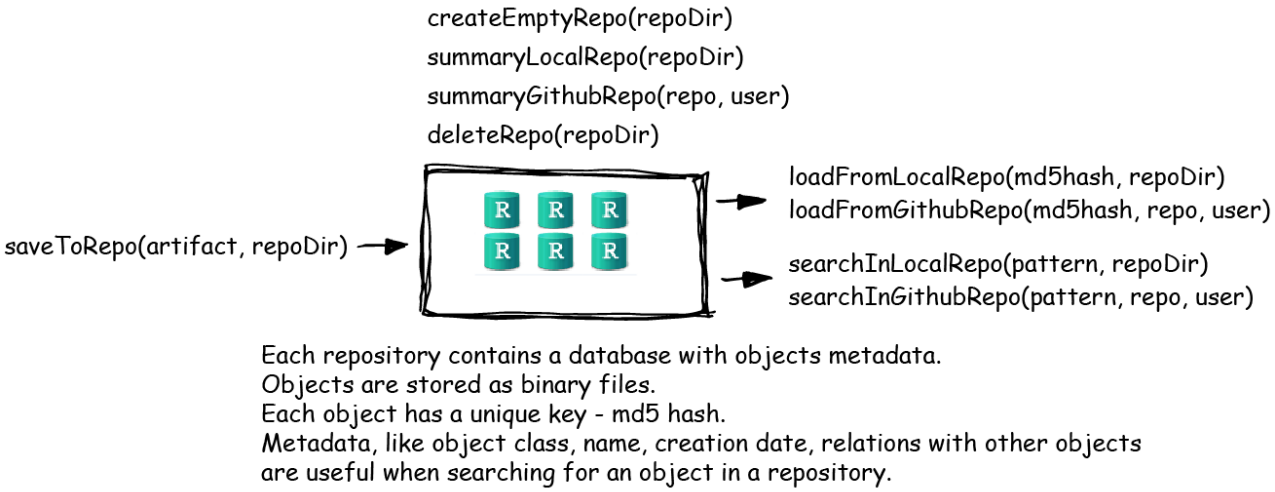
Pakiet archivist jest wzorowany na możliwościach, które daje system `StatLinks`[4] wykorzystywany przez OECD[5] w raportach statystycznych.

Kluczowe funkcjonalności tego pakietu opisuje poniższy rysunek. Są to:

[4] `http://statlinks.oecdcode.org`

[5] `https://pl.wikipedia.org/wiki/Organizacja_Wsp%C3%B3%C5%82pracy_Gospodarczej_i_Rozwoju`

- Tworzenie/modyfikowanie/usuwanie repozytoriów,
- Dodawanie/usuwanie obiektów oraz ich meta danych z repozytoriów,
- Odczytywanie obiektów z repozytoriów,
- Przeszukiwanie repozytoriów.



Strona z przykładami pakietu `archivist` jest dostępna na serwisie GitHub [6].

[6] `http://pbiecek.github.io/archivist/`

Przykłady

Pakiet `archivist` pozwala na udostępnianie i pobieranie obiektów R z lokalnego lub zdalnego repozytorium.

Kod 2166dfbd3a7a68a91a2f8e6df1a44111 oznacza identyfikator obiektu do odczytania, jest to tzw. hash md5

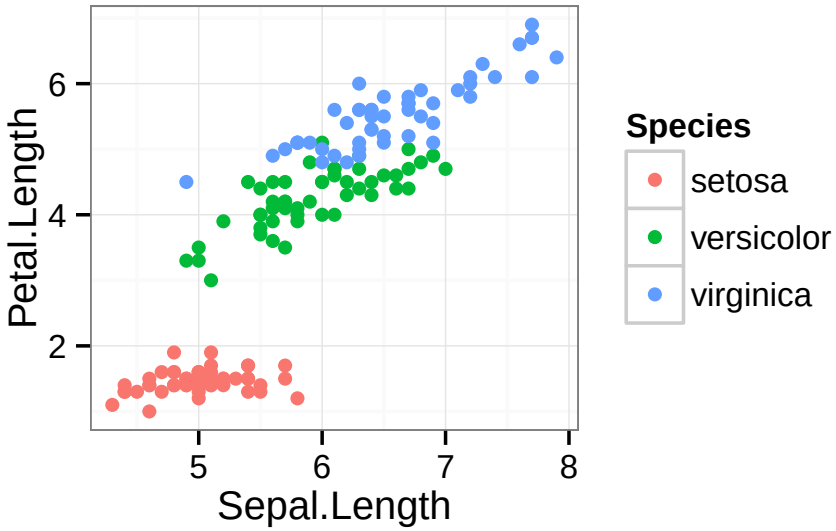
```
library(archivist)
archivist::aread("pbiecek/graphGallery/2166dfbd3a7a68a91a2f8e6df1a44111")
```

```
archivist::aread("pbiecek/graphGallery/2166d")
```

Repozytorium może być udostępnione jako katalog na współdzielonym dysku, GitHubie lub może znajdować się w pakiecie R (w tym przypadku pakiet jest nośnikiem repozytorium z obiektami).

```
setLocalRepo(repoDir = system.file("graphGallery",
    package = "archivist"))
archivist::aread("2166dfbd3a7a68a91a2f8e6df1a44111")
```

W repozytorium, można przechowywać dowolne obiekty, też modele statystyczne.




```
model <- archivist::aread("pbiecek/graphGallery/2a6e492cb6982f230e48cf46023e2e4f")
summary(model)
```

```
##
## Call:
## lm(formula = Petal.Length ~ Sepal.Length + Species, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.76390 -0.17875  0.00716  0.17461  0.79954
##
## Coefficients:
##                   Estimate Std. Error
## (Intercept)       -1.70234    0.23013
## Sepal.Length       0.63211    0.04527
## Speciesversicolor  2.21014    0.07047
## Speciesvirginica   3.09000    0.09123
##                   t value Pr(>|t|)
## (Intercept)        -7.397 1.01e-11 ***
## Sepal.Length       13.962  < 2e-16 ***
## Speciesversicolor  31.362  < 2e-16 ***
## Speciesvirginica   33.870  < 2e-16 ***
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  ' ' 1
##
## Residual standard error: 0.2826 on 146 degrees of freedom
## Multiple R-squared:  0.9749, Adjusted R-squared:  0.9744
## F-statistic:  1890 on 3 and 146 DF,  p-value: < 2.2e-16
```

Wyszukiwanie

Zaletą przechowywania obiektów w repozytoriach jest to, że repozytoria można przeszukiwać wykorzystując meta informacje o obiektach. Listę wszystkich dostępnych tagów, po których można wyszukiwać, można przejrzeć używając instrukcji `?Tags`.

Przykładowo, znajdźmy wszystkie modele liniowe ze zmienną `Sepal.Length`.

```
models <- asearch("pbiecek/graphGallery", patterns = c("class:lm",
    "coefname:Sepal.Length"))

lapply(models, coef)
```

```
## [[1]]
##  (Intercept) Sepal.Length
##    -7.101443     1.858433
##
## [[2]]
##       (Intercept)      Sepal.Length
##         -1.7023422         0.6321099
## Speciesversicolor  Speciesvirginica
##         2.2101378         3.0900021
```

Można również przeszukiwać zdalne repozytoria. Pobierzmy wszystkie wykresy klasy `gg` i wyświetlmy je na jednym wykresie.

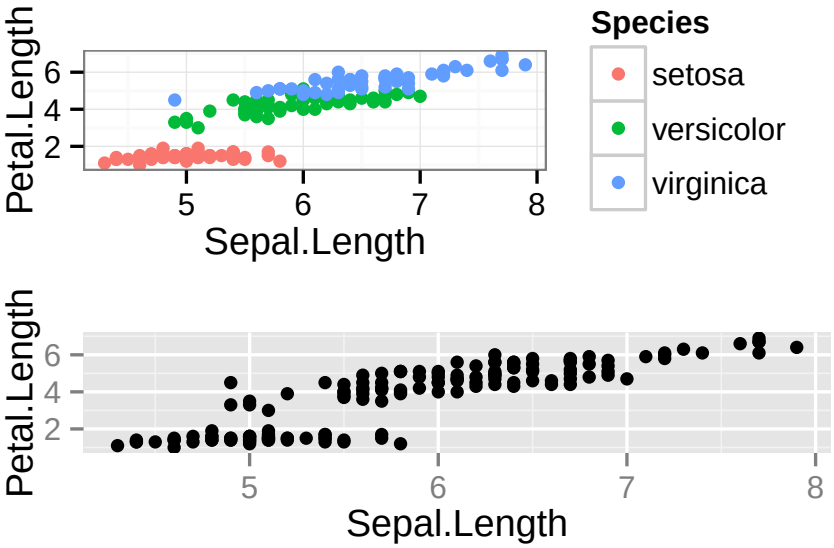
```
plots <- asearch("pbiecek/graphGallery", patterns = c("class:gg",
    "labelx:Sepal.Length"))

library("gridExtra")
do.call(grid.arrange, plots)
```

Historia obiektu

Zapamiętując obiekty, razem z meta danymi, mamy możliwość pamiętania również obiektów, wykorzystanych do stworzenia obecnego obiektu. Umożliwia to odczytanie historii obiektu, czyli listy instrukcji, która doprowadziła do jego powstania.

```
library("dplyr")
library("ggplot2")
library("archivist")
createEmptyRepo("tmp_archivist")
setLocalRepo(repoDir = "tmp_archivist")
```




```
tmp <- iris %a% filter(Sepal.Length < 6) %a% lm(Petal.Length ~
    Species, data = .) %a% summary()

ahistory(tmp)

##    iris                                    [ff575c261c949d073b2895b05d1097c3]
## -> filter(Sepal.Length < 6)                [d3696e13d15223c7d0bbccb33cc20a11]
## -> lm(Petal.Length ~ Species, data = .)    [990861c7c27812ee959f10e5f76fe2c3]
## -> summary()                               [050e41ec3bc40b3004bc6bdd356acae7]

ahistory(md5hash = "050e41ec3bc40b3004bc6bdd356acae7")
```

Statystyki

Mając meta dane o obiektach, możemy wyświetlać statystyki dotyczące zawartości repozytoriów.

Przygotujmy nowe repozytorium, skopiujmy do niego jeden obiekt z repozytorium zdalnego i wyświetlmy statystyki tego repozytorium.

```
repo <- "new_repo"
createEmptyRepo(repoDir = repo)
copyGithubRepo(repoTo = repo, md5hashes = "2166dfbd3a7a68a91a2f8e6df1a44111",
    user = "pbiecek", repo = "graphGallery")
```

Dla repozytorium można przedstawiać statystyki dotyczące zawartości obiektów.

```
showLocalRepo(repoDir = repo, method = "tags")
summaryLocalRepo(repoDir = system.file("graphGallery",
    package = "archivist"))
```

```
## Number of archived artifacts in the
## Repository: 27 Number of archived datasets
## in the Repository: 3 Number of various
## classes archived in the Repository: Number
## lm 9 data.frame 2 summary.lm 1 Saves per day
## in the Repository: Saves 2014-09-03 10
## 2015-06-22 2 2015-06-24 10 2015-06-25 12
```

Zapisywanie

Kluczową operacją w pracy z repozytorium jest zapisywanie obiektów.

Za zapisywanie odpowiedzialna jest instrukcja `saveToRepo`. Wykonuje ona następujące kroki:

- wyznacza md5 hash dla obiektu,
- zapisuje obiekt jako plik binarny `rda`,
- wyciąga i zapisuje w bazie danych SQLite meta dane takie jak klasa obiektu, data powstania, tagi.

```
repo <- "new_repo"
# pierwszy obiekt
setLocalRepo(repoDir = repo)
saveToRepo(iris)
```

```
## [1] "ff575c261c949d073b2895b05d1097c3"
```

```
# teraz zapisujemy wykres
pl <- qplot(Sepal.Length, Petal.Length, data = iris)
saveToRepo(pl, repoDir = repo)
```

```
## [1] "de1f79010d5d69f7e0fefab9fccabac1"
## attr(,"data")
## [1] "ff575c261c949d073b2895b05d1097c3"
```

Test na tożsamość

Zaletą wykorzystywania haszy md5 jako nazwy obiektu jest możliwość sprawdzenia tożsamości odtworzonego obiektu. Dzięki temu możemy zweryfikować, czy obiekt udostępniony na przykład w raporcie lub publikacji jest dokładnie tym samym, który pobieramy z repozytorium.

Wystarczy na nowo wyznaczyć jego hash md5 używając funkcji `digest` z pakietu `digest`.

```
model <- aread("pbiecek/graphGallery/2a6e492cb6982f230e48cf46023e2e4f")
digest::digest(model)
```

```
## [1] "2a6e492cb6982f230e48cf46023e2e4f"
```

Przeszukiwanie repozytorium

Często, pracując dłuższy czas z danymi pamiętamy, że pewne analizy wykonaliśmy, ale nie pamiętamy gdzie one się znajdują. W tym przypadku bardzo przydatne może być przeszukiwanie obiektów w repozytorium.

Repozytorium można przeszukiwać na różne sposoby, podając tagi, daty, klasę poszukiwanego obiektu.

```
searchInGithubRepo(pattern = "class:gg", user = "pbiecek",
    repo = "graphGallery") %>% head(2)
```

```
## [1] "2166dfbd3a7a68a91a2f8e6df1a44111"
## [2] "d74472d5b4eee352ba17c5a6f2472c07"
```

```
searchInGithubRepo(pattern = list(dateFrom = "2014-09-01",
    dateTo = "2014-09-30"), user = "pbiecek",
    repo = "graphGallery") %>% head(2)
```

```
## [1] "f9ebb370fa8fed2057be1bb11005f5fb"
## [2] "6f6623ab33ae7f98bf8f5d7457c112eb"
```

```
multiSearchInGithubRepo(pattern = c("class:gg",
    "labelx:Sepal.Length"), user = "pbiecek",
    repo = "graphGallery") %>% head(2)
```

```
## [1] "2166dfbd3a7a68a91a2f8e6df1a44111"
## [2] "92ada1e052d4d963e5787bfc9c4b506c"
```

Zadanie

Wróćmy do danych o samochodach `auta2012`.

- W pętli, dla różnych marek samochodów zbuduj model zależności pomiędzy ceną a rocznikiem (dla każdej marki osobno).
- Następnie dodaj te wszystkie modele do nowo utworzonego repozytorium `archivist`.
- Następnie (np. w innej sesji R) wyszukaj obiekty klasy `lm` w tym repozytorium.
- Możesz udostępnić repozytorium opracowanych obiektów innym osobom lub używać tego repozytorium jako magazyn modeli w aplikacji SaaS.