Archiwizacja artefaktów z pakietem **archivist**

Przemysław Biecek, Marcin Kosiński
przemyslaw.biecek@gmail.com, m.p.kosinski@gmail.com

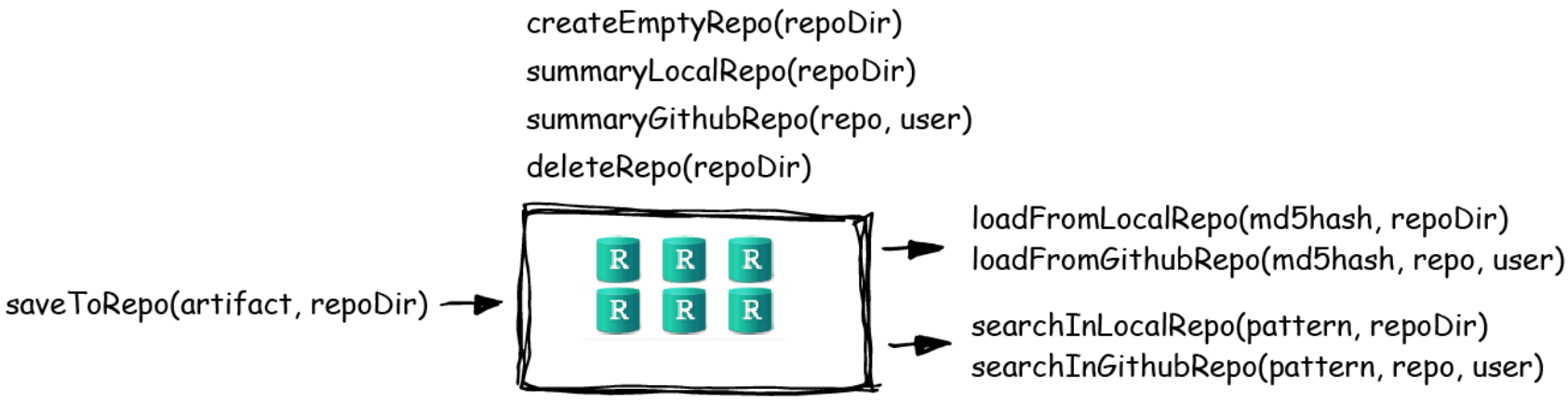
https://github.com/pbiecek/archivist

16 Października, 2014

Do czego służy **archivist**?

- pozwala **magazynować i archiwizować** obiekty w repozytoriach przechowywanych na dysku bądź na Githubie
- udostępnia proste w obsłudze funkcje uprawniające **szukanie** i **odzyskiwanie** obiektów
- wspiera filozofię **powtarzalnych badań** (*reporoducible research*)
- idealnie sprawuje się jako **pamięć podręczna**

Jak działa archivist?



Each repository contains a database with objects metadata.
Objects are stored as binary files.
Each object has a unique key - md5 hash.
Metadata, like object class, name, creation date, relations with other objects are useful when searching for an object in a repository.

Case Study: Pamięć podręczna

Funkcja `getMaxDistribution()` podsumowuje rozkład maksimum z N niezależnych obserwacji z rozkładu D dla R replikacji.

```
getMaxDistribution <- function(D = rnorm, N = 10,
                               R = 1000000) {
  res <- replicate(R, max(D(N)))
  summary(res)
}
system.time( getMaxDistribution(rnorm, 10) )
```

```
   user  system elapsed
  49.11    0.41   50.53
```

```
system.time( getMaxDistribution(rexp, 20) )
```

```
   user  system elapsed
  49.02    0.13   50.06
```

Case Study: Pamięć podręczna

Spróbuj sam. Funkcję można łatwo pobrać z repozytorium przechowywanego na Githubie.

```
install.packages("archivist")
library("archivist")
# library(devtools)
# install_github("pbiecek/archivist")

loadFromGithubRepo("c", user="MarcinKosinski",
                   repo="Museum")
```

Case Study: Pamięć podręczna

```
cache <- function(cacheRepo, FUN, ...) {
  tmpl <- list(...)
  tmpl$.FUN <- FUN
  outputHash <- digest(tmpl)
  isInRepo <- searchInLocalRepo(paste0("cacheId:",
                          outputHash), cacheRepo)
  if (length(isInRepo) > 0)
    return(loadFromLocalRepo(isInRepo[1], repoDir = cacheRe
  output <- do.call(FUN, list(...))
  attr( output, "tags") <- paste0("cacheId:", outputHash)
  attr( output, "call") <- ""
  saveToRepo(output, repoDir = cacheRepo, archiveData = TRU
             archiveMiniature = FALSE, rememberName = FALSE
  output
}
loadFromGithubRepo("7", user="MarcinKosinski",
                   repo="Museum")
```

Case Study: Pamięć podręczna

Używając pakietu **archivist** można przygotować repozytorium przechowujące wywołania funkcji `cache()`, aby uniknąć ich powtórnego wywołania w przyszłości.

```
cacheRepo <- tempdir()
createEmptyRepo( cacheRepo )
library( "digest" )
system.time( cache(cacheRepo, getMaxDistribution,
                   rnorm, 10) )
```

```
   user  system elapsed
  46.85    0.17   47.16
```

```
system.time( cache(cacheRepo, getMaxDistribution,
                   rexp, 10) )
```

```
   user  system elapsed
  44.29    0.08   46.52
```

Case Study: Pamięć podręczna

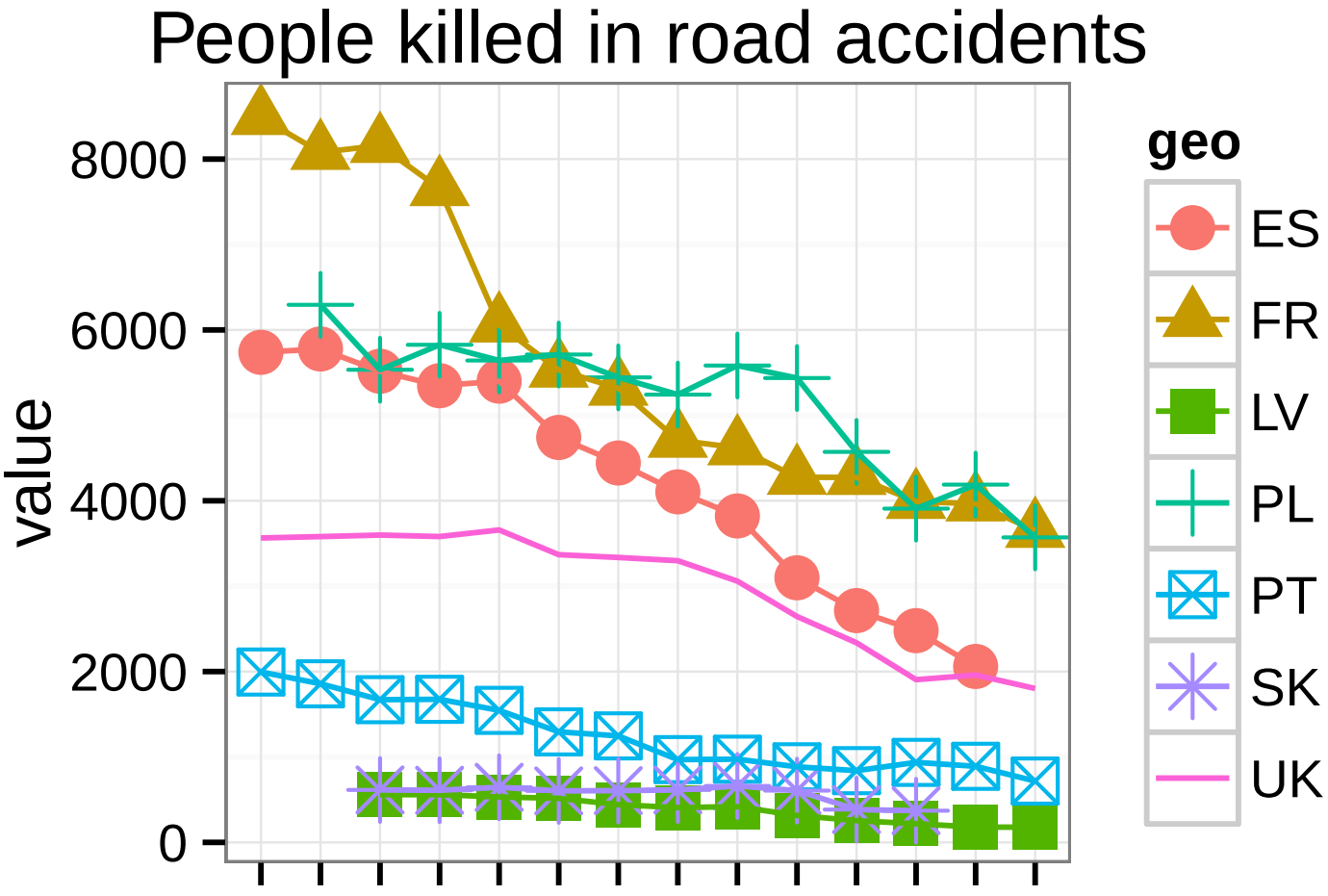
Pierwsze wywołanie nie różniło się prędkością od zwykłego wywołania samej funkcji `getMax..`, jednak drugie wykorzystuje wyniki z pierwszego wywołania i jest o wiele szybsze.

```
system.time( cache( cacheRepo, getMaxDistribution,
                    rnorm, 10) )
```

```
   user  system elapsed
   0.03    0.00    0.03
```

Inne przykłady

```
wykres <-loadFromGithubRepo( md5hash = "fcd7" ,
user = "pbiecek", repo = "graphGallery", value = TRUE)
print( wykres )
```



Inne przykłady

```
hash <- searchInGithubRepo( "name:crime",
user="MarcinKosinski", repo="Museum", fixed = FALSE)
getTagsGithub( hash, user="MarcinKosinski",
                        repo="Museum")
```

```
name:crime.by.state %.%
filter(State == "New York",  Year == 2005) %.%
  arrange(desc(Count))   %.%
 select(Type.of.Crime, Count) %.%
 mutate(Proportion = Count/sum(Count)) %.%
 group_by(Type.of.Crime) %.%
     summarise(num.types = n(), counts = sum(Count))
```

Więcej?

Te i inne zastosowania można zobaczyć pod tymi linkami lub na stronie http://pbiecek.github.io/archivist/

Cache with the archivist package

Retrieving all plots with other github repository (example with flights data from Hadley Wickham useR!2014 tutorial)

Archiving artifacts with their chaining code

Just get the object

Lazy load with **archivist**