archivist: Tools for Storing, Restoring and Searching for R Objects

Przemyslaw.Biecek@gmail.com
M.P.Kosinski@gmail.com

University of Warsaw
Faculty of Mathematics, Informatics, and Mechanics

> sessionInfo()
[2] "Aalborg, Denmark"
Motivation: StatLink (at) OECD

Source: OECD, PISA 2012 Database, Table II.2.1.
StatLink http://dx.doi.org/10.1787/888932964794
Reproducible research

With great tools, like knitr or Sweave, one can prepare excellent and reproducible report/article.

However:

- sometimes raw data are large or with limited access,
- computations take a lot of time or require specialized hardware,
- require specific versions of packages,
- ...

Instead of reproducing all results we may ask for only for scripts that retrieve required results.

How this may be useful?
Let's see some examples.
Use Case 1:

We found an interesting plot/table in an article.

Is there a way to retrieve corresponding data?
Figure 1 | Somatic mutation frequencies observed in exomes from 3,083 tumour-normal pairs. Each dot corresponds to a tumour-normal pair, with vertical position indicating the total frequency of somatic mutations in the exome. Tumour types are ordered by their median somatic mutation frequency, with the lowest frequencies (left) found in haematological and paediatric tumours, and the highest (right) in tumours induced by carcinogens such as tobacco smoke and ultraviolet light. Mutation frequencies vary more than 1,000-fold between lowest and highest across different cancers and also within several tumour types. The bottom panel shows the relative proportions of the six different possible base-pair substitutions, as indicated in the legend on the left. See also Supplementary Table 2.
Hooks to R objects

With archivist, for any data.frame, R plot, R object, one can generate a simple one line instruction that retrieves R object. Include it in figure/table caption, blog post, stackoverflow...

```r
# the full object name is 32 characters long, but first few is enough
# archivist::aread("pbiecek/graphGallery/2166dfbd3a7a68a91a2f8e6df1a44111")
archivist::aread("pbiecek/graphGallery/2166d")
```
Hooks to R objects

With archivist, you can print calling cards for R objects and keep best objects in your wallet.
Use Case 2:

Saving objects should be as easy as possible.
Storing objects should be as easy as possible

Let's create a plot.

```r
library("ggplot2")
pl <- ggplot(iris, aes(y=Petal.Length, x=Sepal.Length, color=Species)) +
     geom_point() + theme_bw()
```

With archivist, saving an object is just a single call of `saveToRepo()`.

```r
library("archivist")
repo <- "archivist_test"
createEmptyRepo(repo)
saveToRepo(pl, repo)
```

```
[1] "fcbbeae563766ce7fb042a57f4d44f28"
attrib("data")
[1] "ff575c261c949d073b2895b05d1097c3"
```
Storing objects should be as easy as possible

Let's create a plot.

```r
library("ggplot2")
pl <- ggplot(iris, aes(y=Petal.Length, x=Sepal.Length, color=Species)) + geom_point() + theme_bw()
```

With archivist, saving an object is just a single call of `saveToRepo()`.

```r
library("archivist")
repo <- "archivist_test"
createEmptyRepo(repo)
saveToRepo(pl, repo)
showLocalRepo(repo, "tags")
```

<table>
<thead>
<tr>
<th>artifact</th>
<th>tag</th>
<th>createdDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcbbeae563766ce7fb042a57f4d44f28</td>
<td>labelx:Sepal.Length</td>
<td>2015-07-01 08:42:28</td>
</tr>
<tr>
<td>fcbbeae563766ce7fb042a57f4d44f28</td>
<td>labely:Petal.Length</td>
<td>2015-07-01 08:42:28</td>
</tr>
<tr>
<td>fcbbeae563766ce7fb042a57f4d44f28</td>
<td>class:gg</td>
<td>2015-07-01 08:42:28</td>
</tr>
<tr>
<td>fcbbeae563766ce7fb042a57f4d44f28</td>
<td>class:ggplot</td>
<td>2015-07-01 08:42:28</td>
</tr>
<tr>
<td>fcbbeae563766ce7fb042a57f4d44f28</td>
<td>name:pl</td>
<td>2015-07-01 08:42:28</td>
</tr>
<tr>
<td>ff575c261c949d073b2895b05d1097c3</td>
<td>date:2015-07-01 08:42:28</td>
<td>2015-07-01 08:42:28</td>
</tr>
</tbody>
</table>
How the repository looks like?

Each repository has following structure:

- SQLite database stored in the file `backpack.db`
- directory named `gallery`, with objects and miniatures (rda, png and txt files).

Tags and artifact's meta data are stored in two tables.
Use Case 3:

Few weeks ago we have created an R object and now we would like to find it.

How we can find it?
Searching in the repository

With archivist, you can search for artefacts by pointing their properties, like class, object’s attributes, variable names and others.

Let’s find all objects of the class gg

```r
plots <- asearch("pbiecek/graphGallery",
                  patterns = "class:gg")
length(plots)

[1] 4
```
Searching in the repository

With archivist, you can search for artefacts by pointing their properties, like class, object’s attributes, variable names and others.

Let’s find all objects of the class gg

```r
plots <- asearch("pbiecek/graphGallery",
                  patterns = "class:gg")
length(plots)
```

After retrieving all plots that fit given pattern, you can plot them all.

```r
library(gridExtra)
do.call(grid.arrange, plots)
```
Retrieved objects might be updated

Objects may be also updated or additionally tagged. Here we add titles with plot’s MD5 hashes for each plot.

```r
plots2 <- lapply(plots,
  function(x)
    x + ggtitle(paste("MD5:", substring(digest::digest(x), 1, 8))))
do.call(grid.arrange, plots2)
```
Use Case 4:

Explore the repository in an interactive fashion
Interactive browser for R objects

With archivist, you can interactively explore artefacts in the repository with the shiny app created on-the-fly.

```r
repo <- "/Users/pbiecek/GitHub/graphGallery/"
shinySearchInLocalRepo(repo)
```
Use Case 5:

We have an R object.

Is there a way to check how the object was created?
Object’s pedigree

We have extended the `%>%` operator from magrittr. The new operator saves all calls and results with additional meta information that allow to recreate a path from which the object was created.

If this operator is used, then for any resulting object we can restore it’s pedigree.

```r
library("dplyr")
setLocalRepo("/Users/pbiecek/GitHub/graphGallery/"

iris %>%
  filter(Sepal.Length < 6) %>%
  lm(Petal.Length ~ Species, data=. %>%
  summary() -> tmp
```
Object’s pedigree

We have extended the `%>%` operator from magrittr. The new operator saves all calls and results with additional meta information that allow to recreate a path from which the object was created.

If this operator is used, then for any resulting object we can restore it’s pedigree.

```
library("dplyr")
setLocalRepo("/Users/pbiecek/GitHub/graphGallery/")

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

Calls and partial results are stored as tags in archivist repository.

```
ahistory(tmp)
```

```
iris
  -> filter(Sepal.Length < 6) [ff575c261c949d073b2895b05d1097c3]
  -> lm(Petal.Length ~ Species, data = .) [d3696e13d15223c7d0bbccbb33cc20a11]
  -> summary() [990861c7c27812ee959f10e5f76fe2c3]
ahistory(md5hash = "050e41ec3bc40b3004bc6bdd356acae7")
```

```
iris
  -> filter(Sepal.Length < 6) [ff575c261c949d073b2895b05d1097c3]
  -> lm(Petal.Length ~ Species, data = .) [d3696e13d15223c7d0bbccbb33cc20a11]
  -> summary() [990861c7c27812ee959f10e5f76fe2c3]
```
Use Case 6:

We have an approved scoring model.

We want to make sure that exactly this model is used.

We need a way to check if we are using the right model.
Verification of identity of an object

In archivist, unique MD5 hashes identify objects. Hashes can be easily verified.

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

```
[1] "2a6e492cb6982f230e48cf46023e2e4f"
```
Verification of identity of an object

In archivist, unique MD5 hashes identify objects. Hashes can be easily verified.

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

[1] "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 t value Pr(>|t|)
(Intercept)        -1.70234    0.23013   -7.397  1.0e-11 ***
Sepal.Length        0.63211    0.04527    13.962  < 2e-16 ***
Speciesversicolor  2.21014    0.07047    31.362  < 2e-16 ***
Speciesvirginica   3.09000    0.09123    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
Use Case 7:

Can we use achivist to cache function results?
Cache

With archivist, you can use cache function to accumulate results from previous calls.

```r
library(lubridate)
cacheRepo <- tempdir()
createEmptyRepo( cacheRepo )
# some toy function
fun <- function(n) {replicate(n, summary(lm(Sepal.Length~Species, iris))$r.squared)}

# first execution
system.time( cache(cacheRepo, fun, 100) )
```

```
user  system elapsed
 0.148   0.002   0.150
```
Cache

With archivist, you can use cache function to accumulate results from previous calls.

```r
library(lubridate)
# a temporary directory as a repo
cacheRepo <- tempdir()
createEmptyRepo( cacheRepo )
# some toy function
fun <- function(n) {replicate(n, summary(lm(Sepal.Length ~ Species, iris))$r.squared)}

# first execution
system.time( cache(cacheRepo, fun, 100) )

user  system elapsed
 0.159   0.005   0.165

# second execution is much faster
system.time( cache(cacheRepo, fun, 100) )

user  system elapsed
 0.003   0.000   0.003

system.time( cache(cacheRepo, fun, 100, notOlderThan = now() - hours(1)) )

user  system elapsed
 0.008   0.001   0.007

deleteRepo( cacheRepo )
rm( cacheRepo )
```
### What other functions are available in archivist?

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>GitHub</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic repo functions</strong></td>
<td><code>createEmptyRepo</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>deleteRepo</code></td>
<td></td>
</tr>
<tr>
<td><strong>Save and load objects</strong></td>
<td><code>saveToRepo</code></td>
<td><code>aread</code>, <code>loadFromGithubRepo</code></td>
</tr>
<tr>
<td></td>
<td><code>loadFromLocalRepo</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>rmFromRepo</code></td>
<td></td>
</tr>
<tr>
<td><strong>Search in repo</strong></td>
<td><code>%a%</code></td>
<td><code>searchInGithubRepo</code>, <code>multiSearchInLocalRepo</code></td>
</tr>
<tr>
<td></td>
<td><code>searchInLocalRepo</code></td>
<td><code>searchInGithubRepo</code>, <code>multiSearchInLocalRepo</code></td>
</tr>
<tr>
<td><strong>Set default repo</strong></td>
<td><code>setGithubRepo</code></td>
<td><code>setLocalRepo</code></td>
</tr>
<tr>
<td><strong>Repo statistics</strong></td>
<td><code>showLocalRepo</code></td>
<td><code>showGithubRepo</code>, <code>summaryGithubRepo</code></td>
</tr>
<tr>
<td></td>
<td><code>summaryLocalRepo</code></td>
<td></td>
</tr>
<tr>
<td><strong>Helper functions</strong></td>
<td><code>copyLocalRepo</code></td>
<td><code>copyGithubRepo</code></td>
</tr>
<tr>
<td></td>
<td><code>saveSetToRepo</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>shinySearchInLocalRepo</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>zipLocalRepo</code></td>
<td><code>zipGithubRepo</code></td>
</tr>
</tbody>
</table>
Where I can find more?

The latest version (1.5) is available on GitHub and CRAN.

More information, examples, use-cases and documentation about this package is available on http://pbiecek.github.io/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.