Illustrate ***knitr*** chunk option "cache = TRUE"

Sarah Gerster (sarah.gerster@isb-sib.ch)
Frédéric Schütz (frederic.schutz@isb-sib.ch)
Bioinformatics Core Facility
SIB Swiss Institute of Bioinformatics
Lausanne, Switzerland

March 19, 2014

1 Caching a chunk that takes too long to be executed each time

We want the result of a chunk which takes quite some time to execute. So ideally, we only want to compute it once, and then just recall the saved output when re-compiling the report. We define a code chunk called "simple_cache" to illustrate this:

```
## some silly code which takes long to execute
for (i in 1:10) {
  cat("Processing loop", i, "of 10\n")
  Sys.sleep(2)
}

## Processing loop 1 of 10
## Processing loop 2 of 10
## Processing loop 3 of 10
## Processing loop 4 of 10
## Processing loop 5 of 10
## Processing loop 6 of 10
## Processing loop 7 of 10
## Processing loop 8 of 10
## Processing loop 9 of 10
## Processing loop 10 of 10

cat("Done processing long code\n")

## Done processing long code
```

This was simple enough, because the chunk does not depend on any other chunk nor on external data. As long as we do not modify anything in the chunk called "simple_cache", it will not be reprocessed when the document is compiled. The stored objects created in this chunk are (lazy-)loaded from previously saved files.

2 Caching with dependence on other code chunks

Let us first set the seed for the random number generator in a chunk called "set_seed":

```
## set a random seed
set.seed(17)
```

Now we define a chunk called "rng_cache_1" to simulate some data from a standard normal distribution and display the first few lines of the generated matrix:

```
## generate a matrix with random numbers
x <- matrix(rnorm(500), ncol = 5)
## time consuming part
Sys.sleep(10)
## what we want to display as output
print(xtable(head(x)), floating = FALSE)
```

	1	2	3	4	5
1	0.06	0.09	0.01	0.24	0.11
2	-0.18	0.52	-0.69	0.90	-0.50
3	1.60	1.34	1.93	-0.35	1.55
4	0.96	-0.44	0.00	1.09	1.25
5	1.51	1.42	-0.13	-1.39	1.55
6	-0.44	-1.15	-1.33	-0.22	-1.30

We now define a chunk called "dep_cache_mod" that modifies the matrix `x`:

```
x <- x - 3
```

We compute the mean of `x` in a cached chunk "dep_cache_1":

```
mean(x)

## [1] 7.019
```

And then again in a cached chunk that keeps an eye on modifications to the R chunk "dep_cache_mod":

```
mean(x)

## [1] -2.981
```

Why can the values returned by the two chunks be different?

3 Caching and random number generator seeds

Be careful when caching code while using random number generators. The way random seeds are handled (and what you need to take care of "by hand") is not straight forward when caching is involved. Check `http://yihui.name/knitr/demo/cache/` for more information.

We prepare another R chunk called "rng_cache_2" basically doing the same as "rng_cache_1". However, this time we include a chunk option to specify that the caching depends on the chunk "set_seed".

```
## generate a matrix with random numbers
x <- matrix(rnorm(500), ncol = 5)
## time consuming part
Sys.sleep(10)
## what we want to display as output
print(xtable(head(x)), floating = FALSE)
```

	1	2	3	4	5
1	-1.46	0.40	-0.82	1.36	1.80
2	0.99	-0.22	0.64	-0.99	-0.49
3	0.83	0.84	1.01	-1.28	-1.72
4	1.61	0.76	-0.54	0.16	-1.91
5	0.05	-0.06	1.56	-0.85	0.47
6	-0.58	0.96	1.42	-0.83	1.32

Why do the values in `x` not (always) change when the argument in the `set.seed()` statement is altered in the "set_seed" chunk?

4 Reprocess cached chunks when the R version or external data changes

You can use additional options in the chunk header to track additional changes. For example, you could add an option `cache.rversion` in your global settings for the chunks to track the current version of R :

```
opts_chunk$set(cache.rversion = R.version.string)
```

In the current setting, `R.version.string` = R version 3.0.2 (2013-09-25). If the cached chunks were processed with a different version of R , the MD5 digests will be different, and the chunks get re-evaluated.

Working on data loaded from external files is a related issue. The chunk below relies on data stored in a file called `my_ext_data.csv` which I received from a partner.

```
## load data from external file
my_ext_data <- read.table(file = "my_ext_data.csv", header = TRUE, sep = ",")

## compute average of each column in loaded matrix
colMeans(my_ext_data)

##        V1        V2        V3        V4        V5
## -0.031434 -0.050982 -0.045279 -0.010599 -0.001778
```

Assume your partner sends you a new version of the data file. How could you use and additional chunk option `cache.extra` to ensure that your cached chunk is re-evaluated when this external data file changes?

```
## load data from external file
my_ext_data <- read.table(file = "my_ext_data.csv", header = TRUE, sep = ",")

## compute average of each column in loaded matrix
colMeans(my_ext_data)

##       V1       V2       V3       V4       V5
## -0.11433  0.05347 -0.14666  0.10848  0.04683
```

5 R session information

Always include session information in your reports:

```
## R version 3.0.2 (2013-09-25)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] xtable_1.7-1    ggplot2_0.9.3.1 knitr_1.5
##
## loaded via a namespace (and not attached):
##  [1] MASS_7.3-29        RColorBrewer_1.0-5 codetools_0.2-8
##  [4] colorspace_1.2-4   dichromat_2.0-0    digest_0.6.3
##  [7] evaluate_0.5.1     formatR_0.10       grid_3.0.2
## [10] gtable_0.1.2       highr_0.3          labeling_0.2
## [13] munsell_0.4.2      plyr_1.8           proto_0.3-10
## [16] reshape2_1.2.2     scales_0.2.3       stringr_0.6.2
## [19] tools_3.0.2
```