Self-Study Exercises

Nishant Gopalakrishnan

Fred Hutchinson Cancer Research Center

17-18 February, 2011

1 Introduction

The genotype information for the GWAS study is provided to you in a comma separated file snpData.csv. This file is available in the extdata folder of the *AdvancedR2011* package. This data has been converted to NetCDF based storage (with byte precision) and stored as a file snpData.nc in the extdata folder.

The exercises that are included in this document are intended to

- Introduce the snp data and explore some of the options in *R* to read the snpData.csv file.
- Get familiar with the application programming interface provided by the *ncdf* package for accessing NetCDF files and create a script to retrieve horizontal slices of the data.
- Create a more useful function `getGWAScols` that access vertical slices of data (snps for all samples) from the NetCDF file. This function will be added to the *StudentGWAS* package that is being developed as this course progresses. The function should be general enough to work with NetCDF files of different sizes.
- Create a unit test to verify that the `getGWAScols` function produces the correct output.

Exercise 1
The goal of this exercise is to explore some of the functionality available in R to read large text files in an efficient manner by reading in smaller chunks at a time.

- *The snp data is available in the file snpData.csv and is located in the extdata folder of the AdvancedR2011 package. Make use of the* `system.file` *to get a path to the snpData.csv file.*
- *Make use of the functions* `scan` *to read in only the 601th row from the snpData.csv file.*

Solution:

```
> library(AdvancedR2011)
> pth <- system.file("extdata", "snpData.csv", package = "AdvancedR2011Data")
> origDat <- scan(pth, what = character(0), sep =",",
+                                 skip = 605, nlines = 1, quiet = TRUE)
```

Exercise 2
The experimental data has been provided to you in the form of a NetCDF (version3) file in the extdata folder of your AdvancedR2011 package. The NetCDF library provides useful command line tools that can be used to explore the contents of a NetCDF file. Make use of the ncdump command line utility to explore the contents of the snpData.nc file provided. (Do not forget to use the -h option). Observe the names of the variables, dimensions and their length.

Solution: ncdump -h snpData.nc

Exercise 3
The goal of this exercise is to create a script to read in all the snips for sample 601 from the NetCDF file snpData.nc and verify that the values read from the NetCDF file match those in the file snpData.csv. Both files are provided in the extdata folder of the AdvancedR2011 package and can be located using the `system.file` function. From the information obtained from the NetCDF command line utility, we know that the file contains 113735 snps (columns) and 1000 samples (rows).

- *Use the function `open.ncdf` to open the NetCDF file snpData.nc.*
- *Use the function `get.var.ncdf` to read in the data for all snips for the sample 601 in the file snpData.nc*
- *Use the `scan` function to read in the snp data for sample 601 from the snpData.csv file and compare its performance with that using the `get.var.ncdf` function. The `system.time` function can be used to compare the performance of the two functions.*
- *Check if the results are identical using the `identical` function in R.*
- *Finally close the NetCDF file using `close.ncdf`*

Solution:

```
> library(ncdf)
> st <- system.time
> ncFile <- system.file("extdata", "snpData.nc", package = "AdvancedR2011Data")
> nc <- open.ncdf(ncFile, write = FALSE)
> st(dat <- get.var.ncdf(nc, varid = "snpData", start= c(601,1), count = c(1, 113735)))
```

```
   user  system elapsed
  0.036   0.076   0.111

> csvFile <- system.file("extdata", "snpData.csv", package = "AdvancedR2011Data")
> st(origDat <- scan(csvFile, what = integer(0), sep =",", skip = 600, nlines = 1,
+             quiet = TRUE))

   user  system elapsed
  4.471   0.105   4.577

> identical(as.vector(dat), origDat)

[1] TRUE

> invisible(close.ncdf(nc))
```

Exercise 4
For most of the software development in the remainder of the course, we will need to access snp data for all the samples(i.e. along the columns of the NetCDF file) in smaller chunks(couple of columns at a time) for block processing of data. To make it convenient to develop code as we go along, a smaller subset of the data that we have looked at so far has been provided as a NetCDF file in the extdata folder of the StudentGWAS package. The file small_snpData.nc contains genotype information of 25 snips for 50 samples.

Our goal in this exercise is to develop a `getGWAScols` *function that retrieves the genotype information for all the samples for the range of snips specified by the user.*

The function should take three arguments

- *nc: An ncdf file pointer obtained by a call to the* `open.ncdf`
- *first: A single integer, the start index for the snpData (column) to be returned*
- *last: A single integer, the end index for the snpData to be returned.*

The function should return a `matrix` *of data type* `raw` *having dimensions with number of rows equal to the number of samples in the small_snpData.nc file and columns corresponding to the range of input specified by the user. The function* `as.raw` *can be used to coerce the integer data type returned by* `get.var.ncdf` *to the raw data type.*

Since the `getGWAScols` *is going to be included in a package and will likely encounter NetCDF files of varying sizes, the function has to be written with general usability in mind. (rather than hard coding number of rows/columns in the NetCDF file etc.) Additionally, suitable checks need to be put in place for the input arguments to make sure they are of the correct type and are within bounds with regard to the dimensions of the NetCDF file.*

To help with the function development, a convenience function `getNcdfVarSummary` has been provided to you in the StudentGWAS package that takes the NetCDF file pointer returned by `open.ncdf` and returns a list. Elements dim1 and dim2 of this list correspond to the number of rows and number of columns in the NetCDF file. The `getNcdfVarSummary` function can be found in the file utils.R in the R folder of the StudentGWAS package.

Solution:

```
> getGWAScols <- function(nc, first, last = first)
+ {
+     if (!is.numeric(first) || length(first) != 1 || is.na(first))
+         stop("'first' must be a single integer")
+     if (!is.numeric(last) || length(last) != 1 || is.na(last))
+         stop("'last' must be a single integer")
+     ncInfo <- getNcdfVarSummary(nc, "snpData")
+     nrows <- ncInfo$dim1
+     ncols <- ncInfo$dim2
+     if (first < 1 || last < first || last > ncols)
+         stop("we need to have 1 <= 'first' <= 'last' <= nb col in NetCDF file")
+     dat <- get.var.ncdf(nc, varid = "snpData", start= c(1,first),
+             count = c(nrows, last - first + 1))
+     structure(as.raw(dat), dim = dim(dat))
+ }
> pth <- system.file("extdata", "small_snpData.nc", package = "StudentGWAS")
> nc <- open.ncdf(pth)
> dat <- getGWAScols(nc, 1, 4)
> invisible(close.ncdf(nc))
```

Exercise 5

An important part of the software development process is to include unit tests that verify that the results returned by our functions are correct. This becomes extremely useful as we try to optimize code as the development of the package progresses.

The goal of this exercise is to develop a unit test function `testGWAScols` that checks if the values returned by the `getGWAScols` function is correct. The result obtained from a call to the function `getGWAScols` for the file small_snpData.nc for all the columns as been provided to you as a file small_snpData.rda in the data folder of the StudentGWAS package. This data can be loaded into an R session using the `load` function.

- *Create a function `testGWAScols` that takes no inputs*
- *Retrieve the genotype information for all the snips and samples in the file small_snpData.nc using the `getGWAScols` function.*

- *Load the target data from the extdata folder of the StudentGWAS package using the `load` function.*

- *Check if the results match using the `identical` function.*

Solution:

```
> library(ncdf)
> testGWAScols <- function()
+ {
+     pth <- system.file("extdata", "small_snpData.nc", package = "StudentGWAS")
+     nc <-  open.ncdf(pth, write = FALSE)
+     ncols <- getNcdfVarSummary(nc, "snpData")$dim2
+     current <- getGWAScols(nc, 1, ncols)
+     load(system.file("extdata", "small_snpData.rda", package = "StudentGWAS"))
+     identical(current, small_snpData)
+ }
> testGWAScols()
```