7. Storing and retrieving large data

Thomas Lumley
Ken Rice

UW Biostatistics

Seattle, June 2010
Large data

“R is well known to be unable to handle large data sets.”

Solutions:

• Get a bigger computer: Linux computer with 16Gb memory for < $2500

• Don’t load all the data at once (methods from the mainframe days).
Storage formats

R has two convenient data formats for large data sets

- For ordinary large data sets, the RSQLite package provides storage using the SQLite relational database.

- For very large ‘array-structured’ data sets such as whole-genome SNP chips, the ncdf package provides storage using the netCDF data format.
Large data

netCDF was designed by the NSF-funded UCAR consortium, who also manage the National Center for Atmospheric Research.

Atmospheric data are often array-oriented: eg temperature, humidity, wind speed on a regular grid of \((x, y, z, t)\).

Need to be able to select ‘rectangles’ of data – eg range of \((x, y, z)\) on a particular day \(t\).

Because the data are on a regular grid, the software can work out where to look on disk without reading the whole file: efficient data access.
Array oriented data (position on genome, sample number) for genotypes, probe intensities.

Potentially very large data sets:

2,000 people \times 300,000 = \text{tens of Gb}

16,000 people \times 1,000,000 \text{SNPs} = \text{hundreds of Gb.}

Even worse after imputation to 2,500,000 SNPs.

R can’t handle a matrix with more than \(2^{31} - 1 \approx 2\) billion entries even if your computer has memory for it. Even data for one chromosome may be too big.
Using netCDF data

With the ncdf package:

open.ncdf() opens a netCDF file and returns a connection to the file (rather than loading the data)

get.var.ncdf() retrieves all or part of a variable.

close.ncdf() closes the connection to the file.
Variables can use one or more array dimensions of a file.

Dimensions

SNP

Sample

Genotypes

Chromosome
library("ncdf")
nc <- open.ncdf("hapmap.nc")

## read all of chromosome variable
chromosome <- get.var.ncdf(nc, "chr", start=1, count=-1)
## set up list for results
runs<-vector("list", nsamples)

for(i in 1:nsamples){
  ## read all genotypes for one person
  genotypes <- get.var.ncdf(nc, "geno", start=c(1,i),count=c(-1,1))
  ## zero for htzygous, chrm number for hmzygous
  hmzygous <- genotypes != 1
  hmzygous <- as.vector(hmzygous*chromosome)
Example

```r
## consecutive runs of same value
r <- rle(hmzygous)
begin <- cumsum(r$lengths)
end <- cumsum(c(1, r$lengths))
long <- which ( r$lengths > 250 & r$values !=0)
runs[[i]] <- cbind(begin[long], end[long], r$lengths[long])
```

close.ncdf(nc)

Notes

- chr uses only the 'SNP' dimension, so start and count are single numbers
- geno uses both SNP and sample dimensions, so start and count have two entries.
- rle compresses runs of the same value to a single entry.
Creating netCDF files

Creating files is more complicated

- Define dimensions

- Define variables and specify which dimensions they use

- Create an empty file

- Write data to the file.
**Dimensions**

Specify the name of the dimension, the units, and the allowed values in the `dim.def.ncdf` function.

One dimension can be ‘unlimited’, allowing expansion of the file in the future. An unlimited dimension is important, otherwise the maximum variable size is 2Gb.

```r
snpdim<-dim.def.ncdf("position","bases", positions)
sampledim<-dim.def.ncdf("seqnum","count",1:10, unlim=TRUE)
```
Variables are defined by name, units, and dimensions

```
varChrm <- var.def.ncdf("chr","count",dim=snpdim,
               missval=-1, prec="byte")
varSNP <- var.def.ncdf("SNP","rs",dim=snpdim,
               missval=-1, prec="integer")
vargeno <- var.def.ncdf("geno","base",dim=list(snpdim, sampledim),
               missval=-1, prec="byte")
vartheta <- var.def.ncdf("theta","deg",dim=list(snpdim, sampledim),
               missval=-1, prec="double")
varr <- var.def.ncdf("r","copies",dim=list(snpdim, sampledim),
               missval=-1, prec="double")
```
Creating the file

The file is created by specifying the file name and a list of variables.

genofile<-create.ncdf("hapmap.nc", list(varChrm, varSNP, vargeno, vartheta, varr))

The file is empty when it is created. Data can be written using `put.var.ncdf()`. Because the whole data set is too large to read, we might read raw data and save to netCDF for one person at a time.

for(i in 1:4000){
    geno<-readRawData(i)  ## somehow
    put.var.ncdf(genofile, "geno", genc,
    start=c(1,i), count=c(-1,1))
}

7.12
Efficient use of netCDF

Read all SNPs, one sample
Efficient use of netCDF

Read all samples, one SNP
### Efficient use of netCDF

Read some samples, some SNPs.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Genotypes</th>
<th>Chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Efficient use of netCDF

Random access is not efficient: eg read probe intensities for all missing genotype calls.
Efficient use of netCDF

- Association testing: read all data for one SNP at a time

- Computing linkage disequilibrium near a SNP: read all data for a contiguous range of SNPs

- QC for aneuploidy: read all data for one individual at a time (and parents or offspring if relevant)

- Population structure and relatedness: read all SNPs for two individuals at a time.