cgmisc

Package containing various functions useful in computational genetics, especially in genome-wide association studies

Marcin Kierczak

August 12, 2014

Introduction

Synopsis

Package cgmisc contains miscellaneous functions, hopefully useful for extending genome-wide association study (GWAS) analyses.

Getting help

Like every other R function, the functions provided in this package are documented in the standard R-help (Rd) format and can be easily accessed by issuing **help**() or its shorter version, **?** function. For instance, if you want to get more information on how to use the clump.markers() function, type either help(clumpmarkers) or ?clump.markers and press return/enter. To see this document from within R you type vignette(‘cgmisc‘).

Purpose of this document

This document aims at presenting how to use functions provided in this package in a typical GWAS data analyses workflow. It is, however, not pretending to be a GWAS tutorial as such.

Conventions

- All R commands are written in terminal type: myfun(foo=T, bar=54)
- In the above example: *myfun* is a function and both *foo* and *bar* are its arguments

Working with `cgmisc`

Installation

In order to install `cgmisc`, you either use one of the R GUIs (native R GUI, RStudio etc.) or type the following command:

```
install.packages("cgmisc", repos="")
```

Functions in the `cgmisc` package often complement or use `GenABEL` package functions and data structures.`GenABEL` is an excellent and widely-used R package for performing genome-wide association studies and much more...

Therefore GenABEL will be loaded automagically when loading cgmisc. You can load cgmisc package by as follows:

```
require("cgmisc")

## Loading required package:  cgmisc
## Warning:  there is no package called 'cgmisc'
```

After having loaded the package it is time to load some data:

```
  load('data/data.rda')
```

Association Analysis

Some of cgmisc functions use data which are the result of GWAS analyses. Let's perform GWAS on our data to obtain GenABEL scan.gwaa-class object :

```
an0 <- qtscore(response ~ sex, data = data)

## Warning:  1 observations deleted due to missingness
```

And have a look at top 5 markers

```
summary(an0, top = 5)

## Summary for top 5 results, sorted by P1df
##                   Chromosome Position Strand A1 A2   N    effB se_effB
## BICF2P1063345             34 40399702      u  T  G 206 -0.2834 0.06954
## BICF2P682714               1 15399848      u  A  G 189  0.5887 0.14714
## BICF2G630569243            6 80458945      u  C  A 206  0.3071 0.07797
## BICF2S2366791              6 70667322      u  C  T 206  0.2628 0.06682
## BICF2G630450144           34 40416964      u  A  G 206 -0.2705 0.06933
##                   chi2.1df      P1df   effAB   effBB chi2.2df      P2df
## BICF2P1063345        16.60 4.609e-05 -0.3644 -0.5354    17.51 1.579e-04
## BICF2P682714         16.01 6.309e-05  0.5912  1.1515    16.01 3.339e-04
## BICF2G630569243      15.51 8.216e-05  0.2904  0.6358    15.57 4.167e-04
## BICF2S2366791        15.46 8.410e-05  0.4558  0.4732    20.15 4.202e-05
## BICF2G630450144      15.22 9.573e-05 -0.3339 -0.5161    15.77 3.756e-04
##                       Pc1df
## BICF2P1063345     7.357e-05
## BICF2P682714      9.911e-05
## BICF2G630569243 1.274e-04
## BICF2S2366791     1.302e-04
## BICF2G630450144 1.472e-04
```

Once this is done, we can proceed with cgmisc functions.

Functions

Plot.Manhattan.LD

The `plot.Manhattan.LD` function allows you to visualize the LD pattern in a genome fragment on an enchanced Manhattan plot. You select one marker, typically the one with the strongest association to the analysed trait and all other markers in the region are coloured according to the degree of linkage disequilibrium with this index marker.

```
plot.manhattan.LD(data, an0, chr=34, region=c(39e6,42e6),
                  index.snp = 'BICF2P1063345', bonferroni = F)

## Error:  could not find function "plot.manhattan.LD"
```

Clump.markers

`clump.markers` function implements clumping procedure described in PLINK documentation. Clumping is based on linkage disequilibrium. The function returns list of clumps which can be used for further analyses or plotted using `plot.clumps` function included in our package.

```
clumps <- clump.markers(data, gwas.result = an0, chr = 6,
                        bp.dist = 250e3, p1 = 0.0001, p2 = 0.01, r2 = 0.5, image=T)

## Error:  could not find function "clump.markers"
```

plot.clumps

`plot.clumps` function plots clumps resulting from running the `clump.markers` function on Manhattan Plot.

```
plot.clumps(gwas.result = an0, clumps = clumps, chr = 1, region = c(3330897,
    3497239))

## Error:  could not find function "plot.clumps"
```

Compute.Fstats

Fixation index Fst is a measure of population differentiation due to genetic structure. Given a set of genotypes in two populations, the function computes fixation index (Fst) and corresponding indices: Fit and Fis. Pops is a vector of two values indicating to which population an individual belongs. Typically, one uses a vector of zeroes and ones where 0 marks an individual belonging to population 1 and 1 marks an individual belonging to population 2. Often, the vector is a result of clustering in MDS-scaled genomic kinship space.

```
fstats <- compute.Fstats(data, pops)

## Error:  could not find function "compute.Fstats"
```

Plot.fstats

Plot results of `compute.Fstats` function.

```
plot.fstats(data, fstats)

## Error:  could not find function "plot.fstats"
```

create.Haploview.info

The program PHASE implements a Bayesian statistical method for reconstructing haplotypes from population genotype data. `Create.Haploview.info` function prepares PHASE and fastPHASE input files from GWAS data object. It can be used on a single entire chromosome or on a specified chromosomal region.

```
create.Haploview.info(data=data, chr=2, coords=c(3030587,5030587),
                      outFile="~user/test.info")
```

get.adjacent.markers

This function returns the number of markers which are adjacent to the given within a distance set as parameter.

```
adj <- get.adjacent.markers(data, "BICF2P647127")

## Error:  could not find function "get.adjacent.markers"
```

```r
print(adj[5:10])

## Error:  object 'adj' not found
```

get.chr.midpoints

`Get.chr.midpoints` function might be very useful for plotting chromsome labels on x-axis. It returns a list of chromosome midpoints coordinates which are independent of coordinates used.

```r
mids <- get.chr.midpoints(data)

## Error:  could not find function "get.chr.midpoints"
```

get.overlapping.windows

```r
olw <- get.overlapping.windows(data, chr = 3, size = 1e+06, overlap = 2500)

## Error:  could not find function "get.overlapping.windows"
```

het.for.overlap.wind

```r
het <- het.for.overlap.wind(olw)

## Error:  could not find function "het.for.overlap.wind"
```

get.window.means

open.region.UCSC

```r
open.region.UCSC(chr = 3, coords = c(40455299, 64160299), assembly = "canFam3")
```

pop.allele.counts

```
pac <- pop.allele.counts(data, pops, progress = TRUE)

## Error:  could not find function "pop.allele.counts"
```

plot.pac

```
plot.pac(data, pac, plot.LD = FALSE)

## Error:  could not find function "plot.pac"
```