Using cupcake with UK BioBank

Olly Burren

13/10/2017

Introduction

Here we introduce cupcake a set of functions for scaling GWAS summary statistics so that they are amenable to Principal Component Analysis (PCA). This vignette illustrates the neccessary steps rather than dealing with the underlying technical details.

Note we assume that define to variables - gwas.dir - This is the location of GWAS summary statistic files that should be incorporated into the basis. - support.dir - This is the location of a dir containing support files (in the code below these have specific names but these are arbitrary).

Install cupcake

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.3.2
```

```
install_github('ollyburren/cupcake')
```

```
## Downloading GitHub repo ollyburren/cupcake@master
## from URL https://api.github.com/repos/ollyburren/cupcake/zipball/master
```

```
## Installing cuPCAke
```

```
## '/Library/Frameworks/R.framework/Resources/bin/R' --no-site-file  \
##   --no-environ --no-save --no-restore --quiet CMD INSTALL  \
##   '/private/var/folders/zw/s4pwsm154rs7fjpf1v8jcvym0000gp/T/RtmpAqm3Bs/devtools12fc0731d9da/ollyburr
##   --library='/Library/Frameworks/R.framework/Versions/3.3/Resources/library'  \
##   --install-tests
```

```
##
```

```
library(cuPCAke)
```

```
## Loading required package: data.table
```

```
## Warning: package 'data.table' was built under R version 3.3.2
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colnames,
##     do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, lengths, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff,
##     sort, table, tapply, union, unique, unsplit

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:data.table':
##
##     first, second

## The following objects are masked from 'package:base':
##
##     colMeans, colSums, expand.grid, rowMeans, rowSums

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:data.table':
##
##     shift

## Loading required package: GenomeInfoDb

## Loading required package: magrittr
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.3.2
```

Input Files

Due to the mutable nature of variants, cupcake uses a concatenation of chromosome and position in order to create a unique index for each variant. The software therefore assumes the following.

- The set of variants is constant across all traits to be examined.
- The position of these variants is uniquely defined by chromosome and position.

In order to compute underlying statistics the following input files need to be supplied.

ref_af_file

This file contains a list of reference allele frequencies for each variant relevant to the population being studied. These might be generated for example using data from the the 1000 genome project.

```
ref_af_file<-file.path(support.dir,'as_basis_snps.tab')
fread(ref_af_file,nrows=3L)
```

```
##    chr position   maf
## 1:   1  2033256 0.354
## 2:   1  2040936 0.112
## 3:   1  2058023 0.108
```

ld_file

This file contains a list of genomic regions specificying how to split the genome into rough recombination blocks. cupcake makes the simplifying assumption that there is essentially no LD between these blocks. This file might be generated for example using recombination frequencies available from the International HapMap project.

```
ld_file<-file.path(support.dir,'all.1cM.tab')
fread(ld_file,nrows=3L)
```

```
##    chr   start     end
## 1:   1       1  888659
## 2:   1  888660 1891263
## 3:   1 1891264 2299651
```

m_file

This is the manifest file and specifies metadata about the GWAS summary statistics you wish to include in your basis. This example includes data generously provided by the Neale Lab

1. trait - A user friendly label for trait can be anything but should be unique.
2. file - Filename for input GWAS data (see later for description)
3. cases - Number of cases for study/trait
4. controls - Number of controls for study/trait
5. pmid - Pubmed ID or some other unique reference (for Neale Lab bb summary stats we use bb Field Code)
6. basis_trait - An integer where 1 = TRUE and 0 FALSE. If set to 1 then trait will be used to compute the basis (see technical details for more information).

```
m_file<-file.path(support.dir,'as_basis_manifest.tab')
fread(m_file,nrows=3L)
```

```
##             trait                                          file cases
## 1: bb_hypertension bb:20002_1065:self_reported_hypertension.tab 87690
## 2:       bb_angina       bb:20002_1074:self_reported_angina.tab 10612
## 3:       bb_asthma       bb:20002_1111:self_reported_asthma.tab 39049
##    controls       pmid basis_trait
## 1:   249469 20002_1065           0
## 2:   326547 20002_1074           0
## 3:   298110 20002_1111           0
```

gwas

This file contains the summary statistics for a given trait. One non trivial endeavour in generating these files is making sure that alleles and therefore odds ratio's are correctly aligned. In the future we hope to create a package to semi automate this proceedure but for the time being it's an exercise for the user.

1. id - unique identifier for variant (note cupcake does not use this to integrate data)
2. chr - chromosome
3. position
4. p.val - Univariate association p.value
5. or - Odds Ratio (Note: If you use Neale Lab bb summary stats you will need to convert linear regression β to logistic regression odds ratio)

```
eg.gwas<-fread(m_file,nrows=1L)[]$file
fread(file.path(gwas.dir,eg.gwas),nrows=3L)
```

```
##           id chr position      p.val       or
## 1: 1:2033256   1  2033256 0.18904155 0.992000
## 2: 1:2040936   1  2040936 0.01687641 1.020363
## 3: 1:2058023   1  2058023 0.01736402 1.020306
```

Create a PCA Basis using PCA

Assuming all the files are arranged as specified above then computing the basis is achieved by running the following commands.

Load in and integrate GWAS summary statistics and support files to create an integrated data.table object.

```
basis.DT<-get_gwas_data(m_file,ref_af_file,ld_file,gwas.dir)
```

```
## Processing CD
## Processing CEL
## Processing MS
## Processing PSC
## Processing PBC
## Processing RA
## Processing SLE
## Processing UC
## Processing T1D
## Adding reference minor allele freq.
## Assigning LD Blocks
```

Next we compute the shrinkage parameters for each variant in the basis.

```
shrink.DT<-compute_shrinkage_metrics(basis.DT)
```

```
## Computing maf_se_empirical
## Computing maf_se_estimated
## Computing Bayesian shrinkage
```

Combine to create a matrix of shrunk $\log(OR)$ suitable for basis generation using prcomp. Note that we supply a parameter as to which maf standard error to use. This can be emp or est.

```
basis.mat.emp <- create_ds_matrix(basis.DT,shrink.DT,'emp')
```

```
## Using emp
```

We need to add a control trait where $\log(OR) = 0$

```
basis.mat.emp<-rbind(basis.mat.emp,control=rep(0,ncol(basis.mat.emp)))
```

Finally compute the basis using prcomp

```
pc.emp <- prcomp(basis.mat.emp,center=TRUE,scale=FALSE)
```

Projection of other GWAS summary statistics onto the basis

Here we show one proof of principle application: We use our basis above to see how traits observed in a separate cohort (UK BioBank) cluster with what we have 'learned' in the basis.

```
bb_traits<-fread(m_file)[grep('bb_',trait),]$trait
bb.DT<-get_gwas_data(m_file,ref_af_file,ld_file,gwas.dir,bb_traits)
```

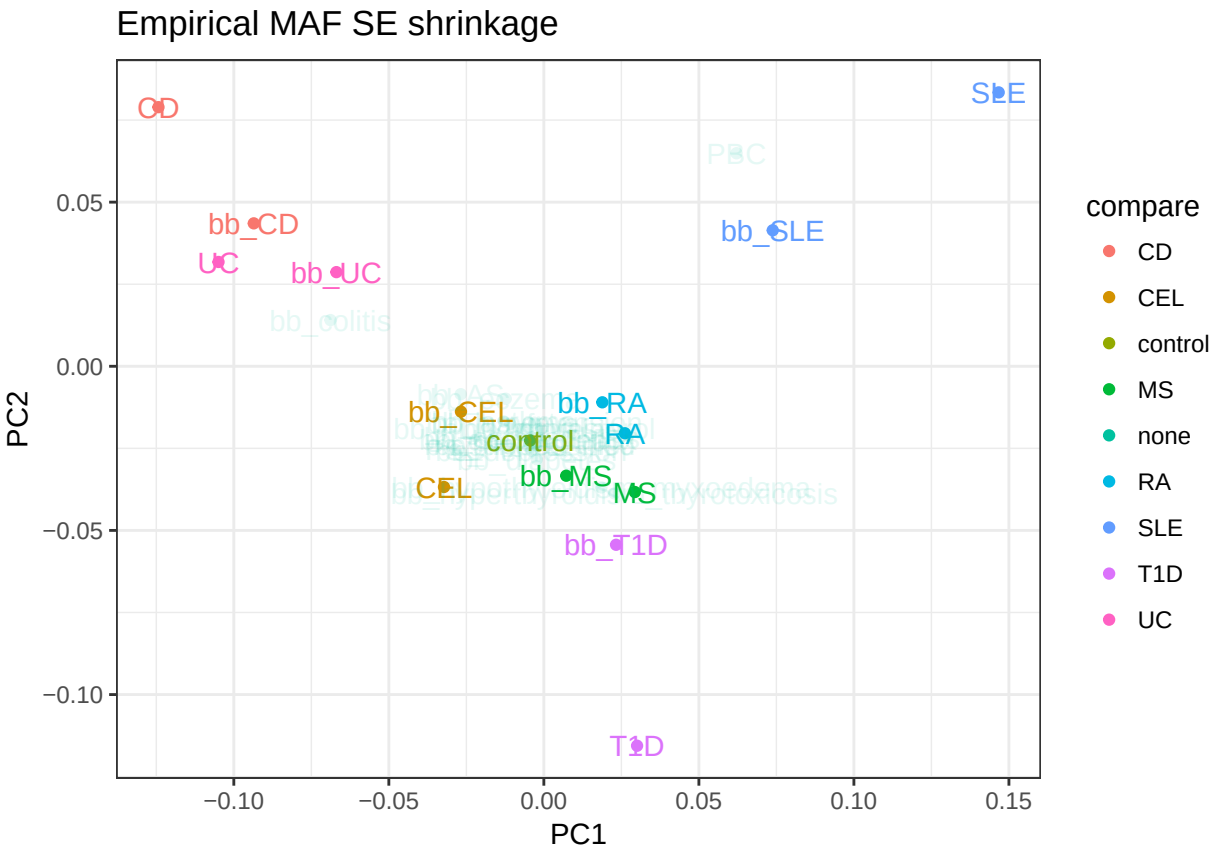
```
## Processing bb_hypertension
## Processing bb_angina
## Processing bb_asthma
## Processing bb_gastro_reflux
## Processing bb_diabetes
## Processing bb_T1D
## Processing bb_hyperthyroidism_thyrotoxicosis
## Processing bb_hypothyroidism_myxoedema
## Processing bb_MS
## Processing bb_migraine
## Processing bb_depression
## Processing bb_slipped_disc
## Processing bb_AS
## Processing bb_SLE
## Processing bb_hayfever
## Processing bb_eczema
## Processing bb_CEL
## Processing bb_colitis
## Processing bb_CD
## Processing bb_UC
## Processing bb_RA
## Processing bb_osteoarthtitis
```

```
## Processing bb_high_cholesterol
## Processing bb_unclassified
## Adding reference minor allele freq.
## Assigning LD Blocks
bb.mat.emp<-create_ds_matrix(bb.DT,shrink.DT,'emp')
## Using emp
pred.emp <- predict(pc.emp,newdata=bb.mat.emp)
emp<-rbind(pc.emp$x,pred.emp)

ml<-list(
    CD = 'bb_CD',
    CEL = 'bb_CEL',
    MS = 'bb_MS',
    RA = 'bb_RA',
    SLE = 'bb_SLE',
    T1D = 'bb_T1D',
    UC = 'bb_UC'
)

g <- function(M){
    M <- cbind(as.data.table(M),trait=rownames(M))
    M$compare<-"none"
    for(i in seq_along(ml)) {
        M[trait %in% c(names(ml)[i], ml[i]), compare:=names(ml)[i]]
    }
    M[trait=="control",compare:="control"]
    M
}

emp<-g(emp)
ggplot(emp,aes(x=PC1,y=PC2,color=compare,label=trait,alpha=compare!='none')) +
    geom_point() + geom_text(show.legend=FALSE) + theme_bw() +
    ggtitle('Empirical MAF SE shrinkage') + scale_alpha_discrete(guide=FALSE)
```



As expected we seem to see clustering where we expect it showing the method has promise.