Obesity susceptibility genes in a Spanish population using sequencing data

Isaac De la Hoz

9 de abril de 2019

Introduction

Obesity

Obesity is defined as an increase in fat mass that is sufficient to adversely affect health.According World Health Organization, people with a body mass index (BMI; weight in kg/height in m^2) higher than 30 $\frac{kg}{m^2}$ are considered obese. Nowadays, obesity is considered as a worldwide epidemic associated with increased morbidity and mortality that imposes an enormous burden on individual and public health (Xu and Tong 2011). In the Europe population, for instance, 10%-20% of people are classified as obese (Klaauw and Farooqi 2015).

Obesity and genetics

Around 40-70% of inter-individual variability in BMI, commonly used to assess obesity, has been attributed to genetic factors(Xu and Tong 2011). The evidence for genetic contributions to body weight comes from family, twin, and adoption studies. Other studies cumulatively demonstrate that the heritability (fraction of the total phenotypic variance of a quantitative trait attributable to genes in a specified environment) of BMI is between 0.71 and 0.86 (Klaauw and Farooqi 2015).

Objectives

In order to expand the catalog of BMI susceptibility SNPs we perform an study on whole exome sequence data from 16 different obese individuals.

Methodology

From already aligned data, a pipeline which include variant calling, variant annotation and statistical analysis was performed.

Variant Calling

In order to find the best way to obtain variant from the alignment files, two different variant callers were proved and compared. One of them was selected for being used in this analysis.

R package: VariantTools

VariantTools is R package which allows to perform the variant calling using R. The following code was the used to perform the variant calling:

First of all, the libraries needed were loaded

```
library(GenomicAlignments)
library(VariantAnnotation)
library(Rsamtools)
library(VariantTools)
library(GenomicRanges)
```

```
library(BiocParallel)
library(BSgenome.Hsapiens.UCSC.hg38)
library(gmapR)
```

Once libraries were loaded, the Gmap genome of human Hg38 version (the human genome version used to perform the alignment) was created:

```
#Gmap genome object creation. The human genome is indexed
## IMPORTANT: the following lines have to be executed 1 time,
## the needed to create the dependency.
setwd("/scratch")
chrs <- standardChromosomes(Hsapiens)
hs <- getSeq(Hsapiens, chrs)
##genome seqlevels style correction so that it is equal to bam files' seqlevels
seqlevels(hs)[25]<-"chrMT"
names(hs)[25]<-"chrMT"
##Gmap genome creation
gmapGenomePath <- file.path(getwd(), "HSG")
gmapGenomeDirectory <- GmapGenomeDirectory(gmapGenomePath, create = TRUE)
gmapGenome <- GmapGenome(genome=hs, directory=gmapGenomeDirectory,
                         name="hg38", create=TRUE, k = 14L)
```

This last code have to be run only one time because once run, a gmap object is created and stored in the directory selected. The following code take charge of the variant calling and genotyping. It was executed once per Bam file you have, changing each time the object `FILE` by the correspondent bam file name.

```
#data loading
##filename
FILE <- "<file name>"
bamFile <- sprintf("~/data/WES_obesity/Data/%s.bam", FILE)

#Tallies creation (VRanges object with variant information)

chrs <- standardChromosomes(Hsapiens)
hs <- getSeq(Hsapiens, chrs)
seqlevels(hs)[25]<-"chrMT"
names(hs)[25]<-"chrMT"
gmapGenomePath <- file.path("/scratch/HSG")
gmapGenomeDirectory <- GmapGenomeDirectory(gmapGenomePath)
HGmapGenome <- GmapGenome(genome=hs, directory=gmapGenomeDirectory, name="hg38")
tiles <- tileGenome(seqinfo(HGmapGenome), ntile = 100)
param <- TallyVariantsParam(HGmapGenome, which = unlist(tiles), indels = TRUE)
bpparam <- MulticoreParam(workers = 5)
tallies <- tallyVariants(bamFile, param, BPPARAM = bpparam)
mcols(tallies) <- NULL
sampleNames(tallies) <- FILE

#Calling and filtering
##Call genotypes
cov <- coverage(bamFile)
params <- CallGenotypesParam(HGmapGenome, p.error = 1/1000, which = tiles)
genotypes <- callGenotypes(tallies, cov, params, BPPARAM = bpparam)

##The default variant calling filters:
##VariantCallingFilters(read.count = 2L, p.lower = 0.2, p.error = 1/1000)
```

```
calling.filters <- VariantCallingFilters()
post.filters <- VariantPostFilters()
variants <- callVariants(genotypes, calling.filters, post.filters)

### Saving as a R data
vcf <- asVCF(sort(variants))
save(vcf, file=sprintf("~/data/WES_obesity/genotypedVariants/%s.rda", FILE))
```

The resulting files were saved as R data to be easily read by R in the later analysis. This last procedure can be run iteratively through the following code:

```
Files <- c("<vector with the name of all bam files>")

chrs <- standardChromosomes(Hsapiens)
hs <- getSeq(Hsapiens, chrs)
seqlevels(hs)[25]<-"chrMT"
names(hs)[25]<-"chrMT"
gmapGenomePath <- file.path("/scratch/HSG")
gmapGenomeDirectory <- GmapGenomeDirectory(gmapGenomePath)
HGmapGenome <- GmapGenome(genome=hs, directory=gmapGenomeDirectory, name="hg38")
tiles <- tileGenome(seqinfo(HGmapGenome), ntile = 100)
param <- TallyVariantsParam(HGmapGenome, which = unlist(tiles), indels = TRUE)
bpparam <- MulticoreParam(workers = 4)
params <- CallGenotypesParam(HGmapGenome, p.error = 1/1000, which = tiles)
calling.filters <- VariantCallingFilters()
post.filters <- VariantPostFilters()

for (i in Files){
  bamFile <- sprintf("~/data/WES_obesity/Data/%s.bam", i)
  ##Tallies creation (VRanges object with variant information)
  tallies <- tallyVariants(bamFile, param, BPPARAM = bpparam)
  print("1")
  mcols(tallies) <- NULL
  sampleNames(tallies) <- i
  cov <- coverage(bamFile)
  ##Call genotypes
  genotypes <- callGenotypes(tallies, cov, params, BPPARAM = bpparam)
  ##Call variants and filtering
  variants <- callVariants(genotypes, calling.filters, post.filters)
  ##saving vcf file
  vcf <- asVCF(sort(variants))
  save(vcf, file=sprintf("~/data/WES_obesity/genotypedVariants/%s.rda", i))
}
```

Once we have all variant from all bam files, we need to obtain the minor allele frequency of the variants in order to perform the statistical analysis. For doing that, we need to merge all VCF files in only one multi-sample VCF file. But, there is a problem here. The VCFs do not have a line for every single locus. Samples that match consensus at a positions do not have an entry for that position, so if there is a SNP in a sample at a given position, other samples could have no entry for that position, and we do not know if the other samples really math consensus there, or if they have low coverage there. Therefore, the variants can not be safely called.

GATK haplotype caller

Considering the problem with the VariantTool R packag, we proved the tool HaplotypeCaller from java-based tool named GATK. This tool allow us to call variants individually on each sample using it in -ERC GVCF mode, leveraging the previously introduced reference model to produce a comprehensive record of genotype likelihoods and annotations for each site in the exome, in the form of a gVCF file. By this way, we can safely call all variants.

The following bash code was used to obtain the gVCFs.

```
# Directories
DWD="$(pwd)"
DIRBAM=$DWD/Data
DIRVCF=$DWD/VCF_HapCaller
GREF=$DWD/hg38.fa

# 1. HaplotypeCaller

##Changing from chrM to chrMT
sed 's/chrM/chrMT/g' hg38.fa
##Reference dictionary creation
gatk CreateSequenceDictionary -R hg38.fa -O hg38.dict
##Creating index file
samtools faidx hg38.fa

##HaplotypeCaller
for BAM in `ls $DIRBAM| grep "bam$"|grep -v "2F759.bam"`
do
    gatk --java-options "-Xmx4g" HaplotypeCaller -R $GREF \
    -I $BAM -O $DIRVCF/${BAM%".bam"}.raw.snps.indels.g.vcf \
    -ERC GVCF &
done
wait
```

Once we got all gVCFs, we used the gatk's tool name `ValidateVarints` in order to validate the correctness of the formatting of VCF files. In addition to standard adherence to the VCF specification, this tool performs extra strict validations to ensure that the information contained within the file is correctly encoded. These include:

- REF. correctness of the reference base(s)
- CHR_COUNTS. accuracy of AC and AN values
- IDS. tests against rsIDs when a dbSNP file is provided
- ALLELES. that all alternate alleles are present in at least one sample

```
# 2. Variant validation
for FILE in `find $DIRVCF -name "*.raw.snps.indels.g.vcf"`
do
    gatk ValidateVariants -R $GREF -V $FILE &
done
wait
```

Once validated, we combined all gVCFs in only one VCF file. Through the GATK's tool named `CombineGVCFs`.

```
#3. Combine GVCFs
```

```
find $DIRVCF -name "*.raw.snps.indels.g.vcf" > $DWD/input.list
gatk CombineGVCFs -R $GREF -V $DWD/input.list -O $DIRVCF/RawVariants.vcf
```

Once we got the multi-sample VCF, we used the tool `GenotypeGVCFs` in order to perform joint genotyping

```
# 4. GVCF Genotyping

mkdir $DIRVCF/finalVCF
gatk --java-options "-Xmx4g" GenotypeGVCFs -R $GREF -V $DIRVCF/RawVariants.vcf -O $DIRVCF/finalVCF/varia
```

From this last code, a jointly genotyped VCF file was obtained. The next step consisted in filtering all low quality variants from the VCF file.

In order to filter in the best way, first of all, the SNPs and INDELs were separated because each type of variant has different filtering parameters. For selecting SNPs and INDELs the tool `SelectVariants` was used.

```
gatk SelectVariants -V $DIRVCF/finalVCF/variants.vcf -select-type SNP -O $DIRVCF/finalVCF/variants.snps
gatk SelectVariants -V $DIRVCF/finalVCF/variants.vcf -select-type INDEL -O $DIRVCF/finalVCF/variants.in
```

Once selected, the following filtering parameters were applied:

For SNPs

- QD< 2.0
- MQ < 40.0
- FS > 60.0
- SOR > 3.0
- MQRankSum < -12.5
- ReadPosRankSum < -8.0

For INDELs

- QD < 2.0
- ReadPosRankSum < -20.0
- InbreedingCoeff < -0.8
- FS > 200.0
- SOR > 10.0

```
#SNPs filtration
gatk VariantFiltration -V $DIRVCF/finalVCF/variants.snps.vcf \
-filter "QD < 2.0" --filter-name "QD2" \
-filter "QUAL < 30.0" --filter-name "QUAL30" \
-filter "SOR > 3.0" --filter-name "SOR3" \
-filter "FS > 60.0" --filter-name "FS60" \
-filter "MQ < 40.0" --filter-name "MQ40" \
-filter "MQRankSum < -12.5" --filter-name "MQRankSum-12.5" \
-filter "ReadPosRankSum < -8.0" --filter-name "ReadPosRankSum-8" \
-O $DIRVCF/finalVCF/variants.snps_filtered.vcf

##Indels filtration

gatk VariantFiltration -V $DIRVCF/finalVCF/variants.indels.vcf -filter "QD < 2.0" \
--filter-name "QD2" \
```

```
-filter "QUAL < 30.0" --filter-name "QUAL30" \
-filter "FS > 200.0" --filter-name "FS200" \
-filter "ReadPosRankSum < -20.0" --filter-name "ReadPosRankSum-20" \
-O $DIRVCF/finalVCF/variants.indels_filtered.vcf
```

Once filtered, SNPs and INDELs filtered were merged in a file and unfiltered variants were selected.

```
gatk MergeVcfs \
          -I $DIRVCF/finalVCF/variants.snps_filtered.vcf \
          -I $DIRVCF/finalVCF/variants.indels_filtered.vcf \
          -O $DIRVCF/finalVCF/variants_filtered.vcf

# Selection of unfiltered variants
gatk SelectVariants \
              -V $DIRVCF/finalVCF/variants_filtered.vcf \
              -exclude-filtered true -O $DIRVCF/finalVCF/variant_filtered.vcf

bgzip -c $DIRVCF/finalVCF/variants_filtered.vcf > $DIRVCF/finalVCF/variants_filtered.vcf.gz
tabix -f -p vcf $DIRVCF/finalVCF/variants_filtered.vcf.gz
```

Variant annotation

Statistical analysis

References

Klaauw, Agatha A Van Der, and I Sadaf Farooqi. 2015. "Review The Hunger Genes : Pathways to Obesity." *Cell* 161 (1). Elsevier Inc.: 119–32. doi:10.1016/j.cell.2015.03.008.

Xu, Yuanzhong, and Qingchun Tong. 2011. "Expanding neurotransmitters in the hypothalamic neurocircuitry for energy balance regulation." *Protein & Cell* 2 (10): 800–813. doi:10.1007/s13238-011-1112-4.