R Scripts to Model Varaince-heterogeneity

2019-05-17

2019-05-17

- **Principal component analysis (PCA)**
- **Double Generalized Linear Model (DGLM)**
  - **Step 1**: Create a function for DGLM analysis
  - **Step 2**: Use the for-loop to run function
- **Hierarchical Generalized Linear Model (HGLM)**
  - **Step 1**: Construct the Z matrix for HGLM analysis
    - First we construct G-matrix (VanRanden 2008)
  - **Step 1**: Second get Cholesky decomposition of G matrix
  - **Step 3**: Create a function for HGLM analysis
  - **Step 4**: Now build function to process the HGLM output and determine p-values
- **Circular Manhattan Plot**
  - Load the GWAS, DGLM and HGLM outputs
  - Draw the circular Manhattan plot
- **Violin plots**
- **Heat map of epistasis and interaction effective plot**

Load the required libraries

```
Packages <- c("dglm", "hglm", "gdata", "readR", "tidyr", "ggplot", "qqman",
    "CMplot", "dplyr", "ggplot2", "SNPRelate", "ggcorrplot", "statmod")
lapply(Packages, library, character.only = TRUE)
```

Note: Before showing the codes for DGLM and HGLM analysis, I will first begin with the principal component (PC) analysis of marker data and extract the top fours PCs which is required later for DGLM analysis to account for population structure

Principal component analysis (PCA)

- In this section we will extract the PC in package SNPRelate using SNP marker data which latter will be used in DGLM analysis to account for population structure.

```r
# read the vcf file from the folder
    vcf.hww <- "~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/Data_final/genotype.vcf"
# Reformat the data
    genofile<-snpgdsVCF2GDS(vcf.hww, "hww.gds", method="biallelic.only")
#Run PCA
    geno_hww<- snpgdsOpen(genofile)
    pca_hww1<- snpgdsPCA(geno_hww, num.thread=2)
# variance proportion (%)
    pc.percent <- pca_hww1$varprop*100
    head(round(pc.percent, 2))
## make a data.frame
    pca_data<- data.frame(sample.id = pca_hww1$sample.id,
                          EV1 = pca_hww1$eigenvect[,1],    # the first eigenvector
                          EV2 = pca_hww1$eigenvect[,2],    # the second eigenvector
                          EV3 = pca_hww1$eigenvect[,3],
                          EV4 = pca_hww1$eigenvect[,4],
                          EV5 = pca_hww1$eigenvect[,5],
                          stringsAsFactors = FALSE)
    head(pca_data)
#save the pca data as r objective for future use
saveRDS(pca_data, file="~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/results/outputs/
            PCA/pca_hww_299.rds")
# Draw the plot
    plot(pca_data$EV2, pca_data$EV1, xlab="PC1",  col="blue",ylab="PC2",
      yaxt="n",font.lab = 2, col.axis="black", cex.axis=1, tck=-0.02)
```

Double Generalized Linear Model (DGLM)

- Here we will be presenting the codes to show how to model variance-heterogeneity using DGLM in the context of GWAS.
- As the analysis is computationally demanding and time consuming we performed this analysis on Holland Computing Center(HCC) server at University of Nebraska, Lincoln.
- Here we are just providing the snippet of codes and not showing any outputs. ## Read the marker and phenotypic, and principal components

```r
# First we will be reading the marker data that is in rds formate and assign
# it to object geno
geno <- readRDS(file = "geno.rds")
# Read the cadmium data
pheno_data <- readRDS(file = "pheno_ok2_final.rds")
# Read the principal components data and assign it to object covar
covar <- readRDS("pca_hww_299.rds")
```

Step 1: Create a function for DGLM analysis

Description of the code:

- First we created a function so that we can use this function to run the DGLM analysis for all the markers.
- Five arguments are passed to the function including CT (representing column for particular trait in phenotypic data file, here named as phenos), i (representing markers), geno(representing marker data file), and covar (representing PCs of marker data).
- Here we are fitting top four PCs as covariates to account for population structure.
- The basic syntax of DGLM model we are fitting is as follows:

```
phenotype=marker_effect+ covariates (modelling mean), marker_effect(modeling dispersion/vari
```

- Finally we are extracting the variables including coefficients (beta), standard error (s.e), P.mean (p-value for mean), P.disp (p-value for dispersion) and assigning them to data frame ‘out’

```
my.pdglm <- function(cT, i, Phenos, geno, covar) {
    y <- Phenos[, cT]
    model <- dglm(y ~ geno[, i] + covar[, 2] + covar[, 3] + covar[, 4] + covar[,
        5], ~geno[, i], family = gaussian(link = identity))
    P.mean <- summary(model)$coef[2, 4]  # Extarct p values for mean part
    P.disp <- anova(model)$Adj.P[2]  # Extract P values for dispersion part
    s.model <- summary(model$dispersion.fit)
    beta <- s.model$coef[2, 1]  # Extarct cofficients
    se <- s.model$coef[2, 2]  # Extract standard errors
    out <- data.frame(Beta = beta, SE = se, P.mean = P.mean, P.disp = P.disp,
        stringsAsFactors = FALSE)  # Save all the extracted variables in data frame out
    return(out)
    # print(i)
}
```

Step 2: Use the for-loop to run function

Description of the code:

- First we are creating a data frame “TF” with rows equal to number of markers and columns equal to number of variables extracted from the DGLM analysis above.
- Then we are using for-loop to run the above function *my.pdglm()* for all the markers one by one and save the output in TF file.
- We are using function *try()* to continue the loop if error is encountered as some of the markers do not converge.
- Finally we are saving the output as csv file.

```r
TF <- matrix(NA, nrow = dim(geno)[2], ncol = 4)
for (i in 1:dim(geno)[2]) {
    try({
        outm <- my.pdglm(cT = 1, i = i, Phenos = pheno_data, geno, covar)
        TF[i, ] <- as.numeric(outm)
        print(i)
    }, silent = TRUE)
}
# save the output in folder
write.csv(TF, file = paste("TF_", 1, "CD_OKLOHOMA.csv", sep = ""), row.names = FALSE)
```

Hierarchical Generalized Linear Model (HGLM)

Description:

- Here we will be modeling variance-heterogeneity using HGLM in hglm package.
- More details on HGLM modelling for genetics data can be found here The hglm Package (Version 2.0)
- Again HGLM analysis was performed on Holland Computing Center (HCC) server at University of Nebraska, Lincoln.
- Here we are just providing the snippet of codes and not showing any outputs.

Step 1: Construct the Z matrix for HGLM analysis

First we construct G-matrix (VanRanden 2008)

```r
# First upload the marker data
geno <- readRDS(file = "geno.rds")
# Scale the marker data
Xs <- scale(geno, center = TRUE, scale = TRUE)
# Construct G matrix
G <- Xs %*% t(Xs)/ncol(Xs)
dim(G)
```

Step 1: Second get Cholesky decomposition of G matrix

```r
chol.G <- chol(G)
Z0 <- diag(1, nrow = nrow(G), ncol = ncol(G))
Z <- Z0 %*% chol.G
# Z0 is the identity matrix
```

Step 3: Create a function for HGLM analysis

Description of the code:

- First we created a function so that we can use this function to run the hglm analysis for all the markers.
- Six arguments are passed to the function including CT (representing column for particular trait in phenotypic data file, here named as phenos), i (representing markers), Phenos (representing phenotypic data file), X (representing marker matrix), Z(representing Z matrix), and X.disp (representing marker matrix for dispersion part).
- Here we are modeling correlated random effects using Z matrix as random effect which is not possible in DGLM.
- The basic syntax of HGLM model we are fitting is as follows:

```
phenotype=marker_effect (fixed)+ Z (random) (modeling mean), marker_effect(modeling dispersi
```

- Finally we are extracting the variables including coefficients (beta), standard error (s.e), P.mean (p value for mean), P.disp (p value for dispersion) and assigning them to data frame 'out'

```
# Run hglm model for all SNPs using for loop
my.hglm <- function(cT, i, Phenos, Z, X, X.disp) {
    y <- Phenos[, cT]
    y2 <<- Phenos[, cT]
    outm <- hglm(y = y, X = as.matrix(geno[, i]), Z = chol.G2, X.disp = as.matrix(geno[,
        i]), family = gaussian(link = log))
    estimates_fix <- outm$fixef
    SE_Mean <- outm$SeFe
    DF <- outm$dfReFe
    DP_Mm <- outm$varFix
    DP_RM <- outm$varRanef
    estimates_rand <- outm$SummVC1[1]
    S.E_rand <- outm$SummVC1[2]
    out <- data.frame(estimates_fix = estimates_fix, SE_Mean = SE_Mean, DF = DF,
        DP_Mm = DP_Mm, DP_RM = DP_RM, estimates_rand = estimates_rand, S.E_rand = S.E_rand,
        stringsAsFactors = FALSE)
    return(out)
}

# Aanalysis for cadmium phenotype
TF <- matrix(NA, nrow = dim(geno)[2], ncol = 7)

for (i in 1:dim(geno)[2]) {
    try({
        outm <- my.hglm(cT = 5, i = i, Phenos = pheno_data, Z = chol.G2, X = geno,
            X.disp = geno)
```

```r
        TF[i, ] <- as.numeric(outm)
        print(i)
    }, silent = TRUE)
}
# save the output
write.csv(TF, file = paste("TF_", 5, "CD_OKLOHOMA.csv", sep = ""), row.names = FALSE)
```

Step 4: Now build function to process the HGLM output and determine p-values

Description of the code:

```r
# read the raw data file
my.hglm1 <- function(pheno, geno1, map1) {
    # add the column names
    colnames(pheno) <- c("estimates_fix", "SE_Mean", "DF", "DP_Mm", "DP_RM",
        "estimates_rand", "S.E_rand")
    # add the marker name and position
    markernames <- data.frame(colnames(geno1))
    # Now combine the output file
    pheno <- cbind(markernames, pheno)
    # Now estimate the p-values for mean and dispersion part using library dplyr
    library(dplyr)
    pheno <- mutate(pheno, p.mean = 2 * pt(-abs(estimates_fix/SE_Mean), df = 1),
        p.disp = 2 * pt(-abs(estimates_rand/S.E_rand), df = 1))
    # now remove the markers with NA values in the file
    pheno <- pheno %>% filter(!is.na(p.mean))
    # match the markers between map file and outfile
    colnames(pheno) <- c("marker", "estimates_fix", "SE_Mean", "DF", "DP_Mm",
        "DP_RM", "estimates_rand", "S.E_rand", "p.mean", "p.disp")
    map <- map
    # check the marker names is same in both files now combine the mapfile and
    # outputfile
    pheno <- cbind(map, pheno)
    # now select the appropriate columns for Manhattan plot
    pheno <- select(pheno, marker, chrom, pos, p.mean, p.disp)
}
```

Circular Manhattan Plot

- Here we will draw the circular Manhattan plot using p-values including P.mean (equivalent to Traditional GWAS), P.dispersion (obtained in DGLM analysis), and P.dispersion (obtained through HGLM analysis): Figure 1 in the manuscript.

Load the GWAS, DGLM and HGLM outputs

Description of the code:

- Here we will load the data containing p.values for all the three including traditional GWAS, dglm and hglm analysis.
- We will create function to upload all the CSV files
- Combined the p-values from all the three files into one file and rename the columns

```
# First we will create a function to upload all the 3 csv files function
setwd("~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/data_manhattan_all")
import.all <- function(mypath, mypattern, ...) {
    tmp.list.1 <- list.files(mypath, pattern = mypattern)
    tmp.list.2 <- list(length = length(tmp.list.1))
    for (i in 1:length(tmp.list.1)) {
        tmp.list.2[[i]] <- read.csv(tmp.list.1[i], ...)
    }
    names(tmp.list.2) <- tmp.list.1
    tmp.list.2
}

# Now upload the all the files
csv.import <- import.all("~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/data_manhattan_all",
    "csv$", sep = ",")
# here we define the separator of entries in the csv files to be comma.
gwas_cd <- csv.import$gwas_cd.csv
dglm_cd <- csv.import$dglm_cd.csv
hglm_cd <- csv.import$hglm_cd.csv
# subset the data
colnames(gwas_cd) <- c("marker", "chrom", "pos", "gwas_p")
dglm_cd <- dglm_cd[, c(1, 5)]
colnames(dglm_cd) <- c("marker", "dglm_p.disp")
hglm_cd <- hglm_cd[, c(1, 4)]
colnames(hglm_cd) <- c("marker", "hglm_p.disp")
# Now merge all the three data files
all_p <- Reduce(merge, list(gwas_cd, dglm_cd, hglm_cd))
```

Draw the circular Manhattan plot

```
CMplot(all_p, plot.type = "c", r = 0.4, col = matrix(c("grey30", "slategray",
    NA, "lightskyblue1", "lightseagreen", NA, "thistle", "pink", NA), 3, 3,
    byrow = T), chr.labels = paste("", c("1A", "1B", "1D", "2A", "2B", "2D",
    "3A", "3B", "3D", "4A", "4B", "4D", "5A", "5B", "5D", "6A", "6B", "6D",
    "7A", "7B", "7D", "UN"), sep = ""), cir.legend = TRUE, cir.legend.cex = 0.8,
    threshold = 1e-05, outward = FALSE, amplify = TRUE, threshold.lty = c(1,
      2), signal.line = 1, signal.col = "orangered", threshold.col = "blue",
    threshold.lwd = 0.5, signal.cex = 1.5, signal.pch = 19, cir.chr.h = 1.5,
    chr.den.col = "black", file = "jpg", memo = "", dpi = 300)
```

Violin plots

- Here we are providing sample codes used ti draw the effective plots (Figure 2 in the manuscript)

```
ggplot(data_2a_final, aes(x=marker, y=avg1))+
  geom_violin(aes(fill = Genotype), trim = FALSE, position = position_dodge(0.9))+
  geom_hline(yintercept = 0.01309, color = "darkred", size =0.6,show.legend = TRUE,linetype
  geom_boxplot(aes(fill = Genotype), width = 0.2, position = position_dodge(0.9))+
  scale_fill_manual(values = c("#00AFBB", "#E7B800"))+
  #stat_summary(fun.y=mean, shape=8, geom="point",size=2, aes(group=genotype),
  #col="black") + #display mean as an asterisk
  #geom_text(data=chr_b_means, aes(label=cd, x=marker,y=cd),
  #colour="black", size=3)
  labs(title = "", y="Cadmium conc. (mg/kg)", x="Markers")+
  theme_few() + #change background of the plot
  theme(plot.title = element_text(color="black", face="bold",size=12, hjust=0.5))+
  theme(axis.text.x=element_text(colour='black', size=12)) + #asthetics of x-axis text
  theme(axis.text.y=element_text(colour='black', size=12)) + #asthetics of y-axis text
  theme(axis.title.x = element_text(colour='black', size=12, vjust=0.0, face="bold")) + #as
  theme(axis.title.y = element_text(colour='black', size=12, face="bold"))+
  theme(legend.title = element_text(colour="darkred", size=14, face="bold"),
        legend.text = element_text(colour="grey0", size=11, face="bold"))+
  guides(fill=guide_legend(title="Genotype"))
```

Heat map of epistasis and interaction effective plot

- Here we are providing the sample codes used to draw the heat map of epistasis between markers associated with varaince-heterogeneity (Figure 3 of the manuscript) and interaction effective plot (Figure 4 of the manuscript)

```
# Read the data file
epis<-read.csv(file="~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/Epistasis/
               Epistasis_output/epistasis_effectiveplot_DATA.csv", header=TRUE)
# Filter the two markers for interaction
epis_filter<-epis%>%filter(marker==c("IAAV3067", "IWA7579"))
# Create function for data summary
data_summary <- function(data, varname, groupnames){
  require(plyr)
  summary_func <- function(x, col){
    c(mean = mean(x[[col]], na.rm=TRUE),
      sd = sd(x[[col]], na.rm=TRUE))
  }
  data_sum<-ddply(data, groupnames, .fun=summary_func,
                  varname)
  data_sum <- rename(data_sum, c("mean" = varname))
  return(data_sum)
}
epis_sum <- data_summary(epis_filter, varname="phenotype",
                    groupnames=c("marker", "allele"))
epis_sum$phenotype<-round(epis_sum$phenotype, digits=3)
epis_sum$sd<-round(epis_sum$sd, digits=3)
# Drop Na
epis_sum<-epis_sum%>%na.omit()
# Draw the effetive plot
ggplot(epis_sum, aes(x=allele, y=phenotype, group=marker, color=marker)) +
  geom_errorbar(aes(ymin=phenotype-sd, ymax=phenotype+sd), width=.1) +
  geom_line() + geom_point()+
  scale_color_manual(values=c("darkred", "blue"))+
  labs(title="",x="", y = "Cadmium conc.(mg)")+
  theme_classic()+
  theme(axis.text= element_text(color = "black", size = 12))+
  theme (plot.title = element_text(),
         axis.title.y = element_text(color="black", size=12, face="bold")) +
  theme(legend.title = element_text(colour="black", size=14),
        legend.text = element_text(colour="black", size=12))+ # add and modify the legends
  guides(color=guide_legend(title=""))+
  theme(legend.position = c(0.15, 0.9))

# Now draw the HEAT MAP

epis_pvalues<-read.csv(file="~/Box/Postdoc/Dataanalysis/Wheat_HWW_panel/ Epistasis/Epistasis
                       header=TRUE, row.names = 1)
# Convert NA into zeros all the values into -log10
epis_pvalues[is.na(epis_pvalues)]<-1
epis_pvalues<-as.matrix(epis_pvalues)
epis_pvalues<- -log10(epis_pvalues)
# draw the corrplot
heatmap_epis<-ggcorrplot(epis_pvalues, method="square", show.diag = TRUE,
                         tl.cex = 6, show.legend = FALSE)
heatmap_epis
# draw the corrplot
heatmap_epis<-ggcorrplot(epis_pvalues, method="square", show.diag = TRUE,
                         tl.cex = 6, show.legend = FALSE)+
  theme(plot.margin = unit(c(0.5,0.5,2,2), "cm"))
  heatmap_epis
```

====================**END**====================