0.0_processing_phenotypes

Andrea Estandia

16/11/2020

```r
# Knitr settings:
knitr::opts_knit$set(root.dir = rprojroot::find_rstudio_root_file())
knitr::opts_chunk$set(echo = TRUE, eval = FALSE)
options(scipen = 999)
```

```r
source("./src/0.0_island_rule_source.R")
```

Read data

```r
pheno <-
  read_csv(file.path(data_path, "phenotypes/phenotypes+SMC2+AE.csv")) %>%
  mutate(wing_weight=Wing/Weight) %>%
  mutate(wing_weight=ifelse(wing_weight=="Inf", NA, wing_weight)) %>%
  mutate(tarsus_weight=Tarsus/Weight) %>%
  mutate(tarsus_weight=ifelse(tarsus_weight=="Inf", NA, tarsus_weight)) #%>%
  #mutate(id=firstup(id))

pheno[pheno=="<NA>"]=NA
```

PCA prep for GWAS with a) head, wing, tarsus, tail b) bill measurements

```r
body_pca <-
  princomp(~Wing+
             Tarsus+
             Tail+
             HeadLength,
           data=pheno,
           scores=TRUE,
           cor=TRUE,
           na.action=na.exclude)

colnames(body_pca$scores) <-
  c("body_PC1", "body_PC2", "body_PC3", "body_PC4")

bill_pca <-
  princomp(~Bill_length_posterior+
             Bill_depth_anterior+
             Bill_width_anterior,
           data=pheno,
           scores=TRUE,
           cor=TRUE,
           na.action=na.exclude)

colnames(bill_pca$scores) <-
  c("bill_PC1", "bill_PC2", "bill_PC3")

pheno <-
  cbind(pheno,body_pca$scores) %>%
  cbind(bill_pca$scores)

write.csv(pheno, file.path(data_path, "phenotypes/pheno_pca.csv"), row.names = F)
```

Plot trait data per population

```r
plot_phenotypes <- function(dataset, trait, xaxis_name, color){
  plot = dataset %>%
    ggplot(aes(y = .data[[trait]],
               x = reorder(pop, .data[[trait]], mean, na.rm=TRUE),
               color=color)) +
    stat_summary(
      aes(group = pop),
      geom = "pointrange",
      fun.data = mean_cl_boot,
      size = 1
    ) +
    geom_jitter(
      width = 0.1,
      height = 0.1,
      alpha = 0.25
    ) +
    scale_color_manual(values=color)+
    theme_minimal() +
    theme(
      panel.grid.minor = element_blank(),
      panel.grid.major.x = element_blank(),
      axis.text.x = element_text(angle = 45, hjust=1, size=text_size),
      axis.text.y = element_text(size = text_size),
      axis.title = element_text(size = text_size),
      axis.title.x = element_blank(),
      legend.position = "none") +
     labs(
       y = paste0(xaxis_name, "\n"))
    return(plot)
}

trait_list = colnames(pheno)[c(14:17, 24, 29:30)]
output_plot <- list()
for (trait in trait_list){
  output_plot[[trait]] <- plot_phenotypes(pheno, trait, as.character(trait), color="#3d3d3d")
}

trait_plot <- wrap_plots(output_plot)
```

```
ggsave(
  "trait_plot.pdf",
  trait_plot,
  path = figures_path,
  device = "pdf",
  width = 30,
  height = 10,
  dpi = 400
)

pheno$pop <- as.character(pheno$pop)
pheno$pop <- factor(pheno$pop, levels=c("Mainland",
                           "Tasmania",
                           "French_Polynesia",
                           "Norfolk_Island",
                           "Chatham",
                           "New_Zealand",
                           "Heron_Island",
                           "Lord_Howe_Island",
                           "Grand_Terre",
                           "Lifou",
                           "Ouvea",
                           "Mare",
                           "Tanna",
                           "Efate",
                           "Ambrym",
                           "Ambae",
                           "Pentecost",
                           "Malekula",
                           "Santo",
                           "Gaua",
                           "Vanua_Lava"))

plot_phenotypes2 <- function(dataset, trait, xaxis_name, color){
  plot = dataset %>%
    ggplot(aes(y = .data[[trait]],
               x = pop,
               color=color)) +
    stat_summary(
      aes(group = pop),
      geom = "pointrange",
      fun.data = mean_cl_boot,
      size = 1
    ) +
    geom_jitter(
      width = 0.1,
      height = 0.1,
      alpha = 0.25
    ) +
    scale_color_manual(values=color)+
    theme_minimal() +
    theme(
      panel.grid.minor = element_blank(),
```

```r
      panel.grid.major.x = element_blank(),
      axis.text.x = element_text(angle = 45, hjust=1, size=text_size),
      axis.text.y = element_text(size = text_size),
      axis.title = element_text(size = text_size),
      axis.title.x = element_blank(),
      legend.position = "none") +
     labs(
       y = paste0(xaxis_name, "\n"))
    return(plot)
}

trait_list = colnames(pheno)[c(14:17, 24, 29:30)]
output_plot <- list()
for (trait in trait_list){
  output_plot[[trait]] <- plot_phenotypes2(pheno, trait, as.character(trait), color="#3d3d3d")
}

trait_plot2 <- wrap_plots(output_plot)

ggsave(
  "trait_plot2.pdf",
  trait_plot2,
  path = figures_path,
  device = "pdf",
  width = 20,
  height = 10,
  dpi = 400
)

pc_list = colnames(pheno)[c(31:32, 35:36)]
output_plot_pc <- list()
for (trait in pc_list){
  output_plot_pc[[trait]] <- plot_phenotypes(pheno, trait, as.character(trait), color="#3d3d3d")
}

pc_plot <- wrap_plots(output_plot_pc)

ggsave(
  "pc_plot.pdf",
  pc_plot,
  path = figures_path,
  device = "pdf",
  width = 12,
  height = 6,
  dpi = 400
)

trait_list = colnames(pheno)[c(31:32, 35:36)]
output_plot <- list()
for (trait in trait_list){
  output_plot[[trait]] <- plot_phenotypes2(pheno, trait, as.character(trait), color="#3d3d3d")
}
```

```r
pc_plot2 <- wrap_plots(output_plot)

ggsave(
  "pc_plot2.pdf",
  pc_plot2,
  path = figures_path,
  device = "pdf",
  width = 12,
  height = 5,
  dpi = 400
)

for (i in 1:length(output_plot)){
ggsave(
  paste0("plot_", names(output_plot)[i], ".pdf"),
  output_plot[[i]],
  path = figures_path,
  device = "pdf",
  width = 6,
  height = 3,
  dpi = 400
)
}
```

```r
pheno %>% ggplot(aes(y=Tarsus,x=Wing, col=pop))+geom_point()+
  stat_ellipse()+
    theme_minimal() +
    theme(
      panel.grid.minor = element_blank(),
      panel.grid.major.x = element_blank(),
      axis.text.x = element_text(angle = 45, hjust=1, size=text_size),
      axis.text.y = element_text(size = text_size),
      axis.title = element_text(size = text_size),
      axis.title.x = element_blank())
```

Summary stats phenotypes

```r
summary_stats <- list()
total_sample_size <- list()
for (trait in colnames(pheno[c(14:24)])) {
  summary_stats[[trait]] <-
    group_by(pheno, pop) %>%
    drop_na(trait) %>%
    summarise(
      count = n(),
      mean = mean(get(trait), na.rm = TRUE),
      max = max(get(trait), na.rm = TRUE),
      min = min(get(trait), na.rm = TRUE),
      sd = sd(get(trait), na.rm = TRUE),
    )
  total_sample_size[[trait]] <-
    pheno %>%
    dplyr::select(trait) %>%
    drop_na() %>%
    count()
}

#Save list in an Excel workbook with each list on a sheet
for (sublist in 1:length(summary_stats)){
  write.xlsx(summary_stats[sublist],
             file=file.path(figures_path, "summary_stats_pheno.xlsx"),
             sheetName=paste(sublist),
             row.names=FALSE,
             append=T)
}
```

Check normality in phenotypes

```
trait_list <- c(trait_list, "body_PC1", "body_PC2", "bill_PC1", "bill_PC2",
                "Bill_length_posterior","Bill_depth_anterior", "Bill_width_anterior"
                )
plot_normality <- function(dataset, traits, xaxis_name) {
  plot = dataset %>%
    ggplot(aes(x = .data[[trait]])) +
    geom_density() +
    theme_minimal() +
    labs(x = paste0(xaxis_name, "\n"))
  return(plot)
}

plot_qqnormality <- function(dataset, traits, yaxis_name) {
  plot = dataset %>%
    ggplot(aes(sample = .data[[trait]])) +
    stat_qq() + stat_qq_line()+
    theme_classic() +
    labs(y = paste0(yaxis_name, "\n"))
  return(plot)
}

trait_list <- c(trait_list, "body_PC2")
output_norm_plot <- list()
for (trait in trait_list) {
  output_norm_plot[[trait]] <- plot_normality(pheno, trait, as.character(trait))
}

output_qqnorm_plot <- list()
for (trait in trait_list) {
  output_qqnorm_plot[[trait]] <- plot_qqnormality(pheno, trait, as.character(trait))
}

norm_plots <- wrap_plots(output_norm_plot)
qqnorm_plots <- wrap_plots(output_qqnorm_plot)

ggsave(
  "norm_plots.pdf",
  norm_plots,
  path = figures_path,
  device = "pdf",
  width = 8,
  height = 6,
  dpi = 400
)

ggsave(
  "qqnorm_plots.pdf",
  qqnorm_plots,
  path = figures_path,
  device = "pdf",
  width = 8,
  height = 6,
  dpi = 400
)
```

Generate subsets for sequence removal in BEAGLE file

```
samples2keep <- list()
for (trait in trait_list){
  samples2keep[[trait]] <-
    pheno %>%
      dplyr::select(sample_name, id, pop, trait) %>%
      drop_na()
}
##SAMPLES TO KEEP##
#Save list in an Excel workbook with each list on a sheet.
for (sublist in 1:length(samples2keep)){
  write.xlsx(samples2keep[sublist],
             file=file.path(figures_path,"samples2keep.xlsx"),
             sheetName=paste(sublist),
             row.names=FALSE,
             append=T)
}

##SAMPLES TO REMOVE##
#Generate sequence from Ind0 to Ind187
numbers <- seq(0,376)
m <- c()
for (number in numbers){
  x <- paste0("Ind",number,collapse="")
  m <- c(m,x)
}
#Check those samples that are in the main subset but not in the just generated sequence
samples2remove <- list()
for (sublist in samples2keep){
  samples2remove[[colnames(sublist[4])]] <-
    as.data.frame(m) %>%
    filter(m %!in% sublist$id)
}

for (sublist in 1:length(samples2remove)){
  write.xlsx(samples2remove[sublist],
             file=file.path(figures_path,"samples2remove.xlsx"),
             sheetName=paste(sublist),
             row.names=FALSE,
             append=T)
}


for (trait in trait_list){
  tmp <-
    pheno %>%
      dplyr::select(id, trait) %>%
      drop_na() %>%
      dplyr::select(id, trait)
    write_tsv(tmp, file.path(subset_ind_path, paste0(as.character(trait),".tsv")), col_names=F)
}

for (trait in trait_list){
  tmp <-
    pheno %>%
      dplyr::select(trait) %>%
      drop_na()
    write_tsv(tmp, file.path(subset_pheno_path, paste0(as.character(trait),".tsv")), col_names=F)
}
```

Test whether island-colonising silvereyes are larger

```
summary(lm(body_PC1~colonisation, data=pheno))
summary(lm(bill_PC1~colonisation, data=pheno))

prior <-
  set_prior(class="b",
            "normal(0,10)")

# Create a dummy variable for each population
model <-
  brm(body_PC1 ~ colonisation_label_brms +
                     (1|pop),
            prior = prior,
            chains = 4,
            iter = 4000,
            warmup = 1000,
            data = pheno)

model

plot(model)

pheno %>%
  data_grid(colonisation_label_brms) %>%
  add_fitted_draws(model) %>%
  ggplot(aes(x = .value, y = population)) +
  #add_epred_draws(mod) %>%
```

```r
  #ggplot(aes(x = .epred, y = population)) +
  stat_dotsinterval(quantiles = 100)+
  theme_minimal() +
  theme(
    legend.position = "none",
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_blank(),
    axis.text.x = element_text(hjust=1, size = text_size,family="ubuntu"),
    axis.text.y = element_text(hjust=1, size = text_size,family="ubuntu"),
    axis.title.y = element_blank(),
    axis.title = element_text(size = text_size,family="ubuntu"),
    plot.title = element_text(face = "bold",family="ubuntu"),
    plot.subtitle = element_text(color = "#424242",family="ubuntu"))+
  labs(
    title = "Posterior estimates of the group means",
    x = "\nMean CLOCK length (bp)"
  )

# Print summary of model
summary(model)

brms::conditional_effects(model)

pheno %>%
  ggplot(aes(x=colonisation, y=body_PC1, col=colonisation))+
  geom_jitter(width = 0.05)+
  geom_violin(alpha=0.2)+
  scale_color_manual(values = c(
    "#6a994e",
    "#219ebc",
    "#fcba03",
    "#d62828",
    "#c78c16"))+
  theme(
    panel.border = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.text.x = element_text(size=text_size),
    axis.text.y = element_text(size = text_size),
    axis.title = element_text(size = text_size),
    legend.position = "right",
    legend.text = element_text(size=11),
    legend.title = element_text())+
  labs(x = "\nColonisation time",
       y = "Body PC1\n")
```