OrthologStatus_DGEsComparedtoHuman

Purpose

Generating bar graphs that show for the genes differentially expressed between the response of human to poly(I:C) versus that of each NHP species, how many of those genes also have a one-to-one human ortholog in each of the NHP species. This is to help determine if we are seeing certain genes as differentially expressed in their response in human versus the NHP species to poly(I:C) simply because those genes only have a one-to-one human ortholog in a given species.

Loading packages

```
library(dplyr)
library(stringr)
library(ggplot2)
library(reshape2)
library(purrr)
library(plyr)
library(AMR)
library(biomaRt)
```

Uploading homolog files and databases needed

```
all_homologs <- get(load("2019-07-04homology_feature_NHP.Rdata"))
all_homologs_simple <- llply(all_homologs, function(x) dplyr::select(x, c(2, 3))) %>%
    llply(., function(x) distinct(x, hsapiens_homolog_ensembl_gene, .keep_all = TRUE))


##Looking at currently available databases in biomaRt
#listMarts()

##Specifying that we want to work with the ENSEMBL database
ensembl <- useMart("ENSEMBL_MART_ENSEMBL",
          host = "http://apr2019.archive.ensembl.org")

human_ensembl <- useDataset("hsapiens_gene_ensembl", mart = ensembl)

human_ENSEMBL_IDs <- getBM(attributes = c('ensembl_gene_id', 'description', 'external_gene_name'), mart
human_ENSEMBL_IDs <- dplyr::rename(human_ENSEMBL_IDs, hsapiens_homolog_ensembl_gene = ensembl_gene_id)

joined <- llply(all_homologs_simple, function(x) left_join(human_ENSEMBL_IDs, x, by = "hsapiens_homolog
  llply(., function(x) distinct(x, hsapiens_homolog_ensembl_gene, .keep_all = TRUE)) %>%
  llply(., function(x) dplyr::select(x, c(hsapiens_homolog_ensembl_gene, hsapiens_homolog_orthology_typ
  reduce(full_join, by = "hsapiens_homolog_ensembl_gene")
joined_clean <- filter_at(joined, 2:9, any_vars(!is.na(.)))
colnames(joined_clean) <- c("hsapiens_ENSEMBL_ID", names(all_homologs[1:5]), "Pt_mac", "Rh_mac", "Sq_mo
joined_clean[is.na(joined_clean)] <- 0

joined_clean.m <- melt(joined_clean, id = "hsapiens_ENSEMBL_ID") %>%
  freq(variable, value) %>%
    dplyr::select(., item, count)
joined_clean.m$species <- str_extract(joined_clean.m$item, "^\\w+(_?)\\w+")
joined_clean.m$type <- str_extract(joined_clean.m$item, "ortholog.*$")
joined_clean.m[is.na(joined_clean.m)] <- "none"
```

```
joined_clean.m$species <- factor(joined_clean.m$species, levels = c("Chimp", "Bonobo", "Gorilla", "Orang
```

```
##Bring in the DGE profiles between each NHP species and human when you mapped the RNASeq reads to the s
##genomes
species_sourcer <- file.path("Expanded_Design_Factor_SpeciesHomologs_Outputs_SpeciesMapped")
species_mapped <- Sys.glob(file.path(species_sourcer, "*DifferFromHuman*"))

species_mapped_read <- llply(species_mapped, function(x) read.csv(x))

names(species_mapped_read) <- species_mapped %>%
  str_replace(., "Expanded_Design_Factor_SpeciesHomologs_Outputs_SpeciesMapped/", "")

##Make the csv read in for the genes that are differentially expressed relative to human for each specie
##into a list of data frames (each data frame a different species).
unique_each_species <- split(species_mapped_read$`2019-08-24 DifferFromHuman_SpeciesMapped_EachSpeciesUn
                             f = species_mapped_read$`2019-08-24 DifferFromHuman_SpeciesMapped_EachSpeci
  llply(., function(x) dplyr::select(x, X, SYMBOL, variable))

##Exclude the list elements that contain "human_related" in their name since we just want the unique
##SYMBOLS for each species, regardless of whether the value was mapped to the human genome or the specie
##specific genome.
unique_each_species<-unique_each_species[-grep("human_related", names(unique_each_species))]
unique_joined <- llply(unique_each_species, function(x) joined_clean[joined_clean$hsapiens_ENSEMBL_ID %
                         melt(., id = "hsapiens_ENSEMBL_ID") %>%
                         freq(variable, value))  ##Make a table of the frequency of the different orthol
                                                   ##different species.
unique_joined.m <- melt(unique_joined) %>%
  .[grep("one2one", .$item),] %>%
  filter(., variable == "count")
```

```
## Using item as id variables
## Using item as id variables
## Using item as id variables
## Using item as id variables
## Using item as id variables
## Using item as id variables
## Using item as id variables
## Using item as id variables
```

```
unique_joined.m$extra <- paste(unique_joined.m$item, unique_joined.m$L1)
unique_joined.mm <- unique_joined.m %>% arrange(L1, desc(.$value))
unique_joined.mm$extra <- factor(unique_joined.mm$extra,
        levels = unique_joined.mm$extra[order(unique_joined.mm$L1)])

##setting the color scheme for the different species
colcoloring <- function(item) {
  ifelse(grepl("Pt_mac", item), "darkolivegreen3",
      ifelse(grepl("bab", item), "darkgreen",
        ifelse(grepl("Sq_monkey", item), "purple",
          ifelse(grepl("orang", item), "#bee1f4",
             ifelse(grepl("gorilla", item), "#7dc3e8",
               ifelse(grepl("bonobo", item), "#0088ce",
                 ifelse(grepl("chimp", item), "#00659c",
                   ifelse(grepl("Rh_mac", item), "forestgreen", "grey")))))))
```

```
}
colcolors <- unlist(lapply(unique_joined.m$item, colcoloring))
unique_joined.mm$color  <- colcolors

plotting <- ggplot(unique_joined.mm, aes(x = extra, y = value, fill = extra)) + geom_col() +
  facet_wrap(~L1, ncol = 4, scales = "free") + xlab("") + ylab ("Number of one-to-one Orthologs") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major = element_line(colour = "black")) +
  theme(axis.text.y = element_text(size = rel(1)), axis.title.y = element_text(size = rel(1.2)),
        legend.position = "none", axis.ticks.x = element_blank(), axis.text.x = element_blank()) +
  scale_fill_manual(values = unique_joined.mm$color)+
  theme(strip.background = element_rect(colour="black", fill="white", linetype="solid"),
                                        strip.text = element_text(size = 10))
ggsave(filename = paste(Sys.Date(), "One2OneOrthoCounts.pdf"),
       plot = plotting, dpi = 300, width = 12, height = 6, units = "in", device = "pdf")



##The remaining genes we are examining are part of groups (i.e. all Old World Monkeys, all monkeys, etc
##so we just need to extract the unique genes from those groups and then query their ortholog "status"
##across all species.

##Get the unique ENSEMBL IDs from each of the data frames in the list to match up with our
##data frame "joined_clean" to identify types of orthologs we have.
species_unique_orthos <- llply(species_mapped_read, function(x) unique(x[x$X,]))

##Really only care about the Old World Monkey and All Monkey data frames in this
species_other <- llply(species_unique_orthos, function(x) joined_clean[joined_clean$hsapiens_ENSEMBL_ID
names(species_other)
```

```
## [1] "2019-08-24 DifferFromHuman_SpeciesMapped_AllMonkeys.csv"
## [2] "2019-08-24 DifferFromHuman_SpeciesMapped_CommonAllSpecies.csv"
## [3] "2019-08-24 DifferFromHuman_SpeciesMapped_EachSpeciesUnique.csv"
## [4] "2019-08-24 DifferFromHuman_SpeciesMapped_OWMsOnly.csv"
```

```
AllMonkeys <- species_other[[1]] %>%
  melt(., id = "hsapiens_ENSEMBL_ID") %>%
                          freq(variable, value)  ##Make a table of the frequency of the different orthol
AllMonkeys$type <- "AllMonkeys"

AllOWMs <- species_other[[4]] %>%
  melt(., id = "hsapiens_ENSEMBL_ID") %>%
                          freq(variable, value)  ##Make a table of the frequency of the different orthol
AllOWMs$type <- "AllOWMs"

Monkeys_OWMs <- rbind(AllMonkeys, AllOWMs)

Monkeys_OWMs.m <- melt(Monkeys_OWMs) %>%
  .[grep("one2one", .$item),] %>%
  filter(., variable == "count")
```

```
## Using item, type as id variables
```

```
Monkeys_OWMs.m$extra <- paste(Monkeys_OWMs.m$item, Monkeys_OWMs.m$type)
Monkeys_OWMs.mm <- Monkeys_OWMs.m %>% arrange(type, desc(.$value))
```

```
Monkeys_OWMs.mm$extra <- factor(Monkeys_OWMs.mm$extra,
        levels = Monkeys_OWMs.mm$extra[order(Monkeys_OWMs.mm$type)])

colcoloring <- function(item) {
  ifelse(grepl("Pt_mac", item), "darkolivegreen3",
      ifelse(grepl("bab", item), "darkgreen",
        ifelse(grepl("Sq_monkey", item), "purple",
          ifelse(grepl("orang", item), "#bee1f4",
             ifelse(grepl("gorilla", item), "#7dc3e8",
               ifelse(grepl("bonobo", item), "#0088ce",
                 ifelse(grepl("chimp", item), "#00659c",
                   ifelse(grepl("Rh_mac", item), "forestgreen", "grey")))))))
}
colcolors_MonkeysOWMs <- unlist(lapply(Monkeys_OWMs.mm$item, colcoloring))
Monkeys_OWMs.mm$color <- colcolors_MonkeysOWMs

plotting2 <- ggplot(Monkeys_OWMs.mm, aes(x = extra, y = value, fill = extra)) + geom_col() +
  facet_wrap(~type, ncol = 2, scales = "free") + xlab("") + ylab ("Number of one-to-one Orthologs") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major = element_line(colour = "black")) +
  theme(axis.text.y = element_text(size = rel(1)), axis.title.y = element_text(size = rel(1.2)),
        legend.position = "none", axis.ticks.x = element_blank(), axis.text.x = element_blank()) +
  scale_fill_manual(values = Monkeys_OWMs.mm$color)+
  theme(strip.background = element_rect(colour="black", fill="white", linetype="solid"),
                                        strip.text = element_text(size = 10))
ggsave(filename = paste(Sys.Date(), "One2OneOrthoCounts_MonkeysOWMs.pdf"),
       plot = plotting2, dpi = 300, width = 12, height = 6, units = "in", device = "pdf")
```

Session Info

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] bindrcpp_0.2.2 biomaRt_2.38.0 AMR_0.7.0      plyr_1.8.4
## [5] purrr_0.2.5    reshape2_1.4.3 ggplot2_3.1.0  stringr_1.3.1
## [9] dplyr_0.7.8
##
## loaded via a namespace (and not attached):
##  [1] progress_1.2.0       tidyselect_0.2.5     xfun_0.4
##  [4] colorspace_1.4-0     htmltools_0.3.6      stats4_3.5.2
```

```
##  [7] yaml_2.2.0           blob_1.1.1           XML_3.98-1.16
## [10] rlang_0.3.1          pillar_1.3.1         glue_1.3.0
## [13] withr_2.1.2          DBI_1.0.0            BiocGenerics_0.28.0
## [16] bit64_0.9-7          bindr_0.1.1          munsell_0.5.0
## [19] gtable_0.2.0         evaluate_0.12        memoise_1.1.0
## [22] labeling_0.3         Biobase_2.42.0       knitr_1.21
## [25] IRanges_2.16.0       curl_3.3             parallel_3.5.2
## [28] AnnotationDbi_1.44.0 Rcpp_1.0.0           scales_1.0.0
## [31] backports_1.1.3      S4Vectors_0.20.1     bit_1.1-14
## [34] microbenchmark_1.4-6 hms_0.4.2            digest_0.6.18
## [37] stringi_1.2.4        grid_3.5.2           tools_3.5.2
## [40] bitops_1.0-6         magrittr_1.5         lazyeval_0.2.1
## [43] RCurl_1.95-4.11      tibble_2.0.1         RSQLite_2.1.1
## [46] crayon_1.3.4         pkgconfig_2.0.2      prettyunits_1.0.2
## [49] data.table_1.12.0    httr_1.4.0           assertthat_0.2.0
## [52] rmarkdown_1.11       R6_2.3.0             compiler_3.5.2
```