Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanM

Load required libraries

```r
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tibble)
library(stringr)
library(biomaRt)
library(genefilter)
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## 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:dplyr':
##
##     combine, intersect, setdiff, union

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

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, 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, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:plyr':
##
##     rename

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:plyr':
##
##     desc

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## The following objects are masked from 'package:genefilter':
##
##     rowSds, rowVars
## The following object is masked from 'package:dplyr':
##
##     count
```

```
## The following object is masked from 'package:plyr':
##
##     count
```

```
## Loading required package: BiocParallel
```

```
##
## Attaching package: 'DelayedArray'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

```
## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```r
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(ggplot2)
library(RColorBrewer)
library(stringr)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
library(devtools)
library(ggrepel)
library(reshape2)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
```

```
## The following object is masked from 'package:SummarizedExperiment':
##
##     shift
## The following object is masked from 'package:GenomicRanges':
##
##     shift

## The following object is masked from 'package:IRanges':
##
##     shift

## The following objects are masked from 'package:S4Vectors':
##
##     first, second

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'

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

## The following object is masked from 'package:DelayedArray':
##
##     simplify

## The following object is masked from 'package:GenomicRanges':
##
##     reduce

## The following object is masked from 'package:IRanges':
##
##     reduce

## The following object is masked from 'package:plyr':
##
##     compact
```

```r
library(viridis)
library(devtools)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:Biobase':
##
##     combine

## The following object is masked from 'package:BiocGenerics':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(UpSetR)
```

Purpose

To compare the DGE profiles created in "Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped".

```r
##counts from reads that were aligned to the species-specific genome
DGE_source <- file.path("Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped")
DGE_source_names <- Sys.glob(file.path(DGE_source, "*vs_human_dds1 HumanMapped_DGE_results.txt"))

DGE_readin <- lapply(DGE_source_names, function(x) read.delim(x))
names(DGE_readin) <- DGE_source_names %>%
  str_replace("Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped/", "") %>%
  sub('\\d+-\\d+-\\d+\\s', '', .) %>%
  sub('_treated_v_mock.*|.treatmenttreated.*', '', .) %>%
  sub('species', '', .) %>%
  str_replace("DGE_results.txt", "")

filtering <- function(x) {
  y <- dplyr::filter(x, padj <= 0.05) %>%
    dplyr::filter(abs(log2FoldChange) >= 3)
  y
}

DGEs_filtered <- llply(DGE_readin, filtering)
llply(DGEs_filtered, function(x) nrow(x))
```

```
## $bonobo
## [1] 182
##
## $chimpanzee
## [1] 169
##
## $gorilla
## [1] 150
##
## $olive_baboon
## [1] 346
##
## $orangutan
## [1] 233
##
## $pigtailed_macaque
## [1] 282
##
## $rhesus_macaque
## [1] 311
##
## $squirrel_monkey
## [1] 452
```

```r
##Are there genes in common across all of these?
common_DGEs_all <- map(DGEs_filtered, ~.$X) %>% purrr::reduce(intersect)
common_DGEs_all
```

```
## [1] "ENSG00000084628" "ENSG00000118113" "ENSG00000141668" "ENSG00000155629"
## [5] "ENSG00000167105" "ENSG00000168334" "ENSG00000175946"
```

```
##What are the expression patterns of these common_DGEs_all across species?
common_DGE_values <- llply(DGEs_filtered, function(x) x[x$X %in% common_DGEs_all,])

sapply(common_DGE_values, '[[', 'SYMBOL')
```

```
##      bonobo    chimpanzee gorilla   olive_baboon orangutan
## [1,] "NKAIN1"  "NKAIN1"   "NKAIN1"  "NKAIN1"     "NKAIN1" 
## [2,] "MMP8"    "MMP8"     "MMP8"    "MMP8"       "MMP8"   
## [3,] "CBLN2"   "CBLN2"    "CBLN2"   "CBLN2"      "CBLN2"  
## [4,] "PIK3AP1" "PIK3AP1"  "PIK3AP1" "PIK3AP1"    "PIK3AP1"
## [5,] "TMEM92"  "TMEM92"   "TMEM92"  "TMEM92"     "TMEM92" 
## [6,] "XIRP1"   "XIRP1"    "XIRP1"   "XIRP1"      "XIRP1"  
## [7,] "KLHL38"  "KLHL38"   "KLHL38"  "KLHL38"     "KLHL38" 
##      pigtailed_macaque rhesus_macaque squirrel_monkey
## [1,] "NKAIN1"          "NKAIN1"       "NKAIN1"       
## [2,] "MMP8"            "MMP8"         "MMP8"         
## [3,] "CBLN2"           "CBLN2"        "CBLN2"        
## [4,] "PIK3AP1"         "PIK3AP1"      "PIK3AP1"      
## [5,] "TMEM92"          "TMEM92"       "TMEM92"       
## [6,] "XIRP1"           "XIRP1"        "XIRP1"        
## [7,] "KLHL38"          "KLHL38"       "KLHL38"
```

```
sapply(common_DGE_values, '[[', 'log2FoldChange')
```

```
##         bonobo chimpanzee   gorilla olive_baboon orangutan
## [1,] -4.400864  -5.140453 -3.562226    -7.528281 -7.765408
## [2,] -3.269553  -3.209302 -4.488019    -4.614959 -5.164880
## [3,] -4.517991  -5.429966 -4.667004    -3.489783 -4.997133
## [4,] -4.742451  -7.619573 -3.073232    -9.407388 -8.446704
## [5,] -3.417300  -3.745676 -3.249307    -3.294926 -5.894951
## [6,] -6.531261  -5.613583 -4.844036    -6.597235 -5.000498
## [7,] -5.686699  -5.468382 -4.257567    -3.881083 -6.210851
##      pigtailed_macaque rhesus_macaque squirrel_monkey
## [1,]         -6.958418      -6.635930       -6.156844
## [2,]         -5.814896      -4.434914       -3.986549
## [3,]         -4.538994      -3.339902       -4.720114
## [4,]         -8.809943     -10.499732       -3.748193
## [5,]         -4.547135      -3.962032       -4.939452
## [6,]         -4.535333      -6.422391       -5.086611
## [7,]         -4.618930      -4.977527       -5.047068
```

```
##All the values are negative, indicating these genes' expression is less than what is observed in
##human

zoo <- llply(DGEs_filtered, function(x) dplyr::select(x, X))
names(zoo) <- c("bonobo", "chimp", "gorilla", "baboon", "orangutan", "pigtailed", "rhesus",
                "squirrel_monkey")

zoo_log2FC <- llply(DGEs_filtered, function(x) dplyr::select(x, X, log2FoldChange))
names(zoo_log2FC) <- c("bonobo", "chimp", "gorilla", "baboon", "orangutan", "pigtailed", "rhesus",
                       "squirrel_monkey")
```

```r
zoo_log2FC_df <- zoo_log2FC %>%
  purrr::reduce(full_join, by = "X")
```

```
## Warning: Column `X` joining factors with different levels, coercing to
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector
```

```r
colnames(zoo_log2FC_df) <- c("X", names(zoo))
zoo_log2FC_df$`Average Expression` <- rowMeans(zoo_log2FC_df[,2:9], na.rm=TRUE)
zoo_log2FC_df[is.na(zoo_log2FC_df)] <- 0


counter <- function(x) {
  y <- x %>% add_count(X)
  y
}

zoo_counter <- llply(zoo, counter)
for (i in seq_along(zoo_counter)){
  colnames(zoo_counter[[i]]) <- c("X", names(zoo_counter[i]))
}

zoo_df <- zoo_counter %>%
  purrr::reduce(full_join, by = "X")
```

```
## Warning: Column `X` joining factors with different levels, coercing to
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector

## Warning: Column `X` joining character vector and factor, coercing into
## character vector
```

```
zoo_df[is.na(zoo_df)] <- 0
zoo_df <- zoo_df[c("X", "chimp", "bonobo", "gorilla", "orangutan", "baboon", "rhesus", "pigtailed",
                   "squirrel_monkey")]

zoo_avg_df <- full_join(zoo_log2FC_df[c(1, 10)], zoo_df, by = "X")
write.csv(zoo_avg_df, paste(Sys.Date(), "HumanMapped_SummaryCrossSpeciesExpression.csv"))
```

```
##Now need to load in the DGEs for each species treated vs mock (not relative to human)
##and then the corresponding human DGEs limited to the one-to-one orthologs for that species.

HumanMapped_DGE <- "Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped"
output_dir <-"Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped"
sampleFiles_HumanMapped <- list.files(basename(Sys.glob(file.path(HumanMapped_DGE))),
                  pattern = "treated dds1 HumanMapped_DGE_results.txt|*related.*.txt")

sampleNames_HumanMapped <- sub('_treated_v_mock|.treatmenttreated.*', '', sampleFiles_HumanMapped) %>%
  sub('\\d+-\\d+-\\d+\\s', '', .) %>%
  sub("HumanMapped.*", "", .) %>%
  sub('species', '', .)
length(sampleFiles_HumanMapped)
```

```
## [1] 16
```

```
feature_human_innate <- read.csv("2019-10-20 InnateDBGeneFeatures.csv")

reader <- function(files) {
  d <- read.delim(files)
  d
}

Human_DGEs <- llply(file.path(HumanMapped_DGE, sampleFiles_HumanMapped), reader)

names(Human_DGEs)  <- sampleNames_HumanMapped

##Sorting the DGEs by species so that a given NHP species DGEs are with the corresponding
##human DGEs for that species.
bab_cts <- Human_DGEs[grep("baboon", names(Human_DGEs))]

bonobo_cts <- Human_DGEs[grep("bonobo", names(Human_DGEs))]

orangutan_cts <- Human_DGEs[grep("orangutan", names(Human_DGEs))]

chimp_cts <- Human_DGEs[grep("chimpanzee", names(Human_DGEs))]

sqm_cts <- Human_DGEs[grep("squirrel", names(Human_DGEs))]

rhmac_cts <- Human_DGEs[grep("rhesus", names(Human_DGEs))]
```

```
ptmac_cts <- Human_DGEs[grep("pigtailed", names(Human_DGEs))]

gorilla_cts <- Human_DGEs[grep("gorilla", names(Human_DGEs))]

##Pulling the genes only differentially expressed compared to human for each NHP species
names(zoo_df)
```

```
## [1] "X"                "chimp"            "bonobo"           "gorilla"
## [5] "orangutan"        "baboon"           "rhesus"           "pigtailed"
## [9] "squirrel_monkey"
```

```
chimp <- zoo_df %>% filter_at(., 3:9, all_vars(. == 0))
bonobo <- zoo_df %>% filter_at(., c(2, 4:9), all_vars(. == 0))
gorilla <- zoo_df %>% filter_at(., c(2:3, 5:9), all_vars(. == 0))
orang <- zoo_df %>% filter_at(., c(2:4, 6:9), all_vars(. == 0))
bab <- zoo_df %>% filter_at(., c(2:5, 7:9), all_vars(. == 0))
rhesus <- zoo_df %>% filter_at(., c(2:6, 8:9), all_vars(. == 0))
ptmac <- zoo_df %>% filter_at(., c(2:7, 9), all_vars(. == 0))
sqm <- zoo_df %>% filter_at(., c(2:8), all_vars(. == 0))
zoo_list <- list(bonobo, orang, chimp, gorilla, sqm, rhesus, ptmac, bab)
##Our different groups we want to examine
OWMs <- zoo_df %>% filter_at(., c(6:8), all_vars(. == 1 )) %>%
  filter_at(., c(2:5, 9), all_vars(. == 0))

monkeys <- zoo_df %>% filter_at(., c(6:9), all_vars(. == 1 )) %>%
  filter_at(., c(2:5), all_vars(. == 0))

all_species <- zoo_df %>% filter_at(., c(2:9), all_vars(. == 1))

##############################
##Genes that are differentially expressed compared to human in ALL species
all_cts <- list(bonobo_cts, orangutan_cts, chimp_cts, gorilla_cts, sqm_cts, rhmac_cts, ptmac_cts,
                bab_cts)
names(all_cts) <- c("bonobo", "orangutan", "chimpanzee", "gorilla", "squirrel_monkey",
    "rhesus_macaque", "pigtailed_macaque", "olive_baboon")

all_DGE <- llply(all_cts, llply, function(x) (x[x$X %in% all_species$X, ])) %>%
  llply(., llply, function(x) dplyr::select(x, X, SYMBOL, padj, log2FoldChange))
all_DGE_names <- all_DGE

for (i in 1:length(all_DGE)) {
 all_DGE[[i]] <- all_DGE[[i]] %>% purrr::reduce(full_join, by = "X")
}

all_DGE_limited <- all_DGE %>% llply(., function(x) x[c(1:4, 6:7)]) ##then select columns we care
                                                                   ##about (corresponds to X, SYMBOL,
                                        ##padj and log2FoldChange for the NHP species and the related
                                        ##human file) for each data frame within the list

##Now rename the columns within the data frames in our list so that they make sense once we melt it
##down for plotting.
colnames <- lapply(seq_along(all_DGE_limited), function(i)
  colnames(all_DGE_limited[[i]]) <-
    c("X", "SYMBOL", paste("padj_species"), names(all_DGE_names[[i]][1]),
      paste("padj_human"), names(all_DGE_names[[i]][2])))
for (i in 1:length(all_DGE_limited)) {
  names(all_DGE_limited[[i]]) <- colnames[[i]]
}

##Order the symbols as desired, with the human DGEs related to a given species in
## order from highest to lowest log2FoldChange.
significance <- function(x) {
  y <- dplyr::arrange(x, desc(x[,4]))
  y$padj_species <- ifelse((y$padj_species > 0.05), "ns", "sig")
  y$padj_human <- ifelse((y$padj_human > 0.05), "ns", "sig")
  y
}

all_DGE_ordered <- llply(all_DGE_limited, significance)
names(all_DGE_ordered) <- names(all_cts)

##Melt it all down.
all_species.m <- melt(all_DGE_ordered)
```

```
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
```

```
##Break apart all_species.m into the constituent melted species so we can then re-set the symbol order
##to what we want based on what we did for all_DGE_ordered above.
leveler_species <- function(DGE_set, X) {
species.m <- filter(all_species.m, L1 == !!X)
species.m$SYMBOL <- factor(species.m$SYMBOL, levels = DGE_set$SYMBOL[order(DGE_set[,4])])
species.m
}

common_species <- mapply(leveler_species, all_DGE_ordered, names(all_cts), SIMPLIFY = FALSE) %>%
  do.call("rbind", .)
rownames(common_species) <- c()
common_species$padj_species <- ifelse(grepl("related", common_species$variable), "",
                                      common_species$padj_species)
common_species$padj_human <- ifelse(!grepl("related", common_species$variable), "",
                                    common_species$padj_human)

write.csv(common_species, file.path(output_dir, paste(Sys.Date(),
            "DifferFromHuman_HumanMapped_CommonAllSpecies.csv")))

##Need a "copy" of 'all.m_species' to create a geom_text plot along with our heatmap later. This idea
##came from https://github.com/tidyverse/ggplot2/issues/2656 and the rest of it was modified
##and implemented below in making the faceted heat maps.
common_species_text <- common_species

##For every gene that is in our feature_human_innate data frame derived from InnateDB, we want to
```

```r
##highlight them in our heatmap by making them in bold, italic face.
common_species_text$faces <- ifelse((common_species_text$X %in%
    feature_human_innate$hsapiens_homolog_ensembl_gene), 'bold.italic', 'plain')

##Since we are making faceted plots, we only need one set of SYMBOLs shown per facet, so in our
##geom_text data source we will discard any of the rows for variables that have "related" in its name,
##i.e. getting rid of the second copy of symbols for a given species.
common_species_text <- dplyr::filter(common_species_text, !grepl(".*related",variable))

##########################
##All monkeys

##Using the list above to limit our DGE profiles for each species
monkeys_cts <- list(sqm_cts, rhmac_cts, ptmac_cts, bab_cts)
names(monkeys_cts) <- c("squirrel_monkey", "rhesus_macaque", "pigtailed_macaque", "olive_baboon")

monkeys_DGE <- llply(monkeys_cts, llply, function(x) (x[x$X %in% monkeys$X, ])) %>%
  llply(., llply, function(x) dplyr::select(x, X, SYMBOL, padj, log2FoldChange))

##Could not for the life of me find an easier way to do this, just joining each list of data frames
##for each species so that you now have one data frame (instead of the original two) per species,
##containing the DGE for that species AND the DGE for the related human file.
monkeys_joined <- list((monkeys_DGE[[1]] %>%
                          purrr::reduce(full_join, by = "X")), (monkeys_DGE[[2]] %>%
                            purrr::reduce(full_join, by = "X")), monkeys_DGE[[3]] %>%
                         purrr::reduce(full_join, by = "X"), monkeys_DGE[[4]] %>%
                         purrr::reduce(full_join, by = "X")) %>%
  llply(., function(x) x[c(1:4, 6:7)]) ##then select columns we care about (corresponds to X, SYMBOL,
                                        ##padj and log2FoldChange for the NHP species and the related
                                        ##human file) for each data frame within the list

##Now rename the columns within the data frames in our list so that they make sense once we melt it
##down for plotting.
colnames <- lapply(seq_along(monkeys_joined), function(i)
    c("X", "SYMBOL", paste("padj_species"), names(monkeys_DGE[[i]][1]),
      paste("padj_human"), names(monkeys_DGE[[i]][2])))
for (i in 1:length(monkeys_joined)) {
  names(monkeys_joined[[i]]) <- colnames[[i]]
}

##Order the symbols as desired, with the human DGEs related to a given species in
## order from highest to lowest log2FoldChange.
  significance <- function(x) {
  y <- dplyr::arrange(x, desc(x[,4]))
  y$padj_species <- ifelse((y$padj_species > 0.05), "ns", "sig")
  y$padj_human <- ifelse((y$padj_human > 0.05), "ns", "sig")
  y
}
monkeys_ordered <- llply(monkeys_joined, significance)
names(monkeys_ordered) <- names(monkeys_DGE)

##Melt it all down.
monkeys.m <- melt(monkeys_ordered)
```

```
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
```

```
leveler_monkeys <- function(DGE_set, X) {
species.m <- filter(monkeys.m, L1 == !!X)
species.m$SYMBOL <- factor(species.m$SYMBOL, levels = DGE_set$SYMBOL[order(DGE_set[,4])])
species.m
}

monkeys <- mapply(leveler_monkeys, monkeys_ordered, names(monkeys_ordered), SIMPLIFY = FALSE) %>%
  do.call("rbind", .)
rownames(monkeys) <- c()
monkeys$padj_species <- ifelse(grepl("related", monkeys$variable), "", monkeys$padj_species)
monkeys$padj_human <- ifelse(!grepl("related", monkeys$variable), "", monkeys$padj_human)

write.csv(monkeys, file.path(output_dir,
                             paste(Sys.Date(), "DifferFromHuman_HumanMapped_AllMonkeys.csv")))

monkeys_text <- monkeys
monkeys_text$faces <-
  ifelse((monkeys_text$X %in% feature_human_innate$hsapiens_homolog_ensembl_gene), 'bold.italic',
         'plain')
monkeys_text <- dplyr::filter(monkeys_text, !grepl(".*related",variable))


#################################

##Only Old World monkeys
OWMs_cts <- list(rhmac_cts, ptmac_cts, bab_cts)
names(OWMs_cts) <- c("rhesus_macaque", "pigtailed_macaque", "olive_baboon")

OWMs_DGEs <- llply(OWMs_cts, llply, function(x) (x[x$X %in% OWMs$X, ])) %>%
  llply(., llply, function(x) dplyr::select(x, X, SYMBOL, padj, log2FoldChange))

##Could not for the life of me find an easier way to do this, just joining each list of data frames
##for each species so that you now have one data frame (instead of the original two) per species,
##containing the DGE for that species AND the DGE for the related human file.
OWMs_joined <- list((OWMs_DGEs[[1]] %>%
              purrr::reduce(full_join, by = "X")), (OWMs_DGEs[[2]] %>%
  purrr::reduce(full_join, by = "X")), OWMs_DGEs[[3]] %>% purrr::reduce(full_join, by = "X")) %>%
  llply(., function(x) x[c(1:4, 6:7)]) ##then select columns we care about (corresponds to X, SYMBOL,
                                        ##padj and log2FoldChange for the NHP species and the related
                                        ##human file) for each data frame within the list

##Now rename the columns within the data frames in our list so that they make sense once we melt it
##down for plotting.
colnames <- lapply(seq_along(OWMs_joined), function(i)
    c("X", "SYMBOL", paste("padj_species"), names(OWMs_DGEs[[i]][1]),
      paste("padj_human"), names(OWMs_DGEs[[i]][2])))
for (i in 1:length(OWMs_joined)) {
  names(OWMs_joined[[i]]) <- colnames[[i]]
}
```

```r
##Order the symbols as desired, with the human DGEs related to a given species in
## order from highest to lowest log2FoldChange.
  significance <- function(x) {
  y <- dplyr::arrange(x, desc(x[,4]))
  y$padj_species <- ifelse((y$padj_species > 0.05), "ns", "sig")
  y$padj_human <- ifelse((y$padj_human > 0.05), "ns", "sig")
  y
  }

OWMs_ordered <- llply(OWMs_joined, significance)
names(OWMs_ordered) <- names(OWMs_DGEs)

##Melt it all down.
OWMs.m <- melt(OWMs_ordered)
```

```
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
```

```r
leveler_OWMs <- function(DGE_set, X) {
species.m <- filter(OWMs.m, L1 == !!X)
species.m$SYMBOL <- factor(species.m$SYMBOL, levels = DGE_set$SYMBOL[order(DGE_set[,4])])
species.m
}

OWMs_all <- mapply(leveler_OWMs, OWMs_ordered, names(OWMs_ordered), SIMPLIFY = FALSE) %>%
  do.call("rbind", .)
rownames(OWMs_all) <- c()

OWMs_all$padj_species <- ifelse(grepl("related", OWMs_all$variable), "", OWMs_all$padj_species)
OWMs_all$padj_human <- ifelse(!grepl("related", OWMs_all$variable), "", OWMs_all$padj_human)

##Break apart OWMs.m into the constituent melted species so we can then re-set the symbol order to
##what we want based on what we did for OWMs_ordered above.

write.csv(OWMs_all, file.path(output_dir, paste(Sys.Date(),
                                                "DifferFromHuman_HumanMapped_OWMsOnly.csv")))

OWMs_all_text <- OWMs_all
OWMs_all_text$faces <- ifelse((OWMs_all_text$X %in%
            feature_human_innate$hsapiens_homolog_ensembl_gene), 'bold.italic', 'plain')
OWMs_all_text <- dplyr::filter(OWMs_all_text, !grepl(".*related",variable))

###################################
##Pulling out the DGEs relative to human that are found in only one species at a time as determined
##above in the lines getting "chimp <-, "bonobo <-", etc

ind_differ_DGEs <- function(ctsX, speciesX) {
species_DGE <- llply(ctsX, function(x) x[x$X %in% speciesX$X, ]) %>%
  llply(., function(x) dplyr::select(x, X, SYMBOL, padj, log2FoldChange)) %>%
  purrr::reduce(full_join, by = "X") %>%
    dplyr::select(X, SYMBOL.x, padj.x, padj.y, log2FoldChange.x, log2FoldChange.y)
colnames(species_DGE) <- c("X", "SYMBOL", "padj_species", "padj_human", names(ctsX))
species_DGE$SYMBOL <- str_replace(species_DGE$SYMBOL, ",.*", "")
species_DGE_ordered <- dplyr::arrange(species_DGE, desc(species_DGE[,6])) ##order by last column -
                                                        ##the log2FC values for the human samples
species_DGE_ordered$padj_species <- ifelse((species_DGE_ordered$padj_species > 0.05), "ns", "sig")
species_DGE_ordered$padj_human <- ifelse((species_DGE_ordered$padj_human > 0.05), "ns", "sig")
species_DGE_ordered
}

all_DGEs <- mapply(ind_differ_DGEs, all_cts, zoo_list, SIMPLIFY = FALSE)

all_DGEs.m <- melt(all_DGEs)
```

```
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
```

```r
Na_Values <- all_DGEs.m[is.na(all_DGEs.m$SYMBOL),] ##get positions of NA values in the SYMBOL column

Na_Values_chimp <- all_DGEs$chimpanzee[is.na(all_DGEs$chimpanzee$SYMBOL),] ##get positions of NA
                                                                  ##values in the SYMBOL column
Na_Values_chimp
```

```
##                    X SYMBOL padj_species padj_human
## 36 ENSG00000275558   <NA>           ns        sig
##    human_related_chimpanzee  chimpanzee
## 36               -0.8048947    2.410545
```

```r
levels(all_DGEs$chimpanzee$SYMBOL) <- c(levels(all_DGEs$chimpanzee$SYMBOL), "RN7SKP175")

all_DGEs$chimpanzee[36,2] = c("RN7SKP175")

Na_Values_baboon <- all_DGEs$olive_baboon[is.na(all_DGEs$olive_baboon$SYMBOL),] ##get positions of NA
                                                                        ##values in the SYMBOL column
Na_Values_baboon
```

```
##                    X SYMBOL padj_species padj_human
## 84 ENSG00000286190   <NA>          sig         ns
##    human_related_olive_baboon  olive_baboon
## 84                   -1.82912      1.316882
```

```r
levels(all_DGEs$olive_baboon$SYMBOL) <- c(levels(all_DGEs$olive_baboon$SYMBOL),"LOC728392")

all_DGEs$olive_baboon[84,2] = c("LOC728392")

Na_Values_rhmac <- all_DGEs$rhesus_macaque[is.na(all_DGEs$rhesus_macaque$SYMBOL),] ##get positions of
                                                                  ##NA values in the SYMBOL column
Na_Values_rhmac
```

```
##                    X SYMBOL padj_species padj_human
## 43 ENSG00000267281   <NA>          sig         ns
##    human_related_rhesus_macaque  rhesus_macaque
## 43                    -3.139422     -0.03248839
```

```
levels(all_DGEs$rhesus_macaque$SYMBOL) <- c(levels(all_DGEs$rhesus_macaque$SYMBOL),"ATF7")

all_DGEs$rhesus_macaque[43,2] = c("ATF7")

all_DGEs.m <- melt(all_DGEs)
```

```
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
## Using X, SYMBOL, padj_species, padj_human as id variables
```

```
all_DGEs.m$padj_species <- ifelse(grepl("related", all_DGEs.m$variable), "", all_DGEs.m$padj_species)
all_DGEs.m$padj_human <- ifelse(!grepl("related", all_DGEs.m$variable), "", all_DGEs.m$padj_human)


write.csv(all_DGEs.m, file.path(output_dir, paste(Sys.Date(),
                              "DifferFromHuman_HumanMapped_EachSpeciesUnique.csv")))
##Break apart all_DGEs.m into the constituent melted species so we can then re-set the symbol order
##to what we want based on what we did for all_DGEs above.
leveler <- function(DGE_set, X) {
species.m <- filter(all_DGEs.m, L1 == !!X)
species.m$SYMBOL <- factor(species.m$SYMBOL, levels = DGE_set$SYMBOL[order(DGE_set[,5])])
species.m
}

fewer <- mapply(leveler, all_DGEs, names(all_DGEs), SIMPLIFY = FALSE) %>%
  do.call("rbind", .)
rownames(fewer) <- c()

##Making a fewer_text as done above
fewer_text <- fewer
fewer_text$faces <- ifelse((fewer_text$X %in% feature_human_innate$hsapiens_homolog_ensembl_gene),
                           'bold.italic', 'plain')
fewer_text <- dplyr::filter(fewer_text, !grepl(".*related",variable))
```

```
output_dir <- "Expanded_Design_Factor_SpeciesHomologs_Outputs_HumanMapped"

##Genes for 6/8 species that were uniquely (i.e. only observed in each of these species
##and no other) differentially expressed upon poly(I:C) treatment compared to human.
fewer_filtered <- dplyr::filter(fewer, !grepl("baboon|squirrel", variable)) %>% droplevels()
fewer_text_filtered <- dplyr::filter(fewer_text, !grepl("baboon|squirrel", variable)) %>% droplevels()
fewer_filtered$variable <- with(fewer_filtered,
    factor(variable,levels = c("pigtailed_macaque","human_related_pigtailed_macaque ",
                                "rhesus_macaque", "human_related_rhesus_macaque ",
                               "gorilla", "human_related_gorilla ",
                              "orangutan",   "human_related_orangutan ",
                              "bonobo",  "human_related_bonobo ",
                              "chimpanzee", "human_related_chimpanzee ")))
fewer_filtered$L1 <- with(fewer_filtered, factor(L1,
    levels = c("chimpanzee", "bonobo", "gorilla","orangutan", "rhesus_macaque", "pigtailed_macaque")))
```

```r
pdf(file = file.path(output_dir, paste(Sys.Date(), "differFromHuman_shortList_HumanMapped.pdf")),
    height =4.7, width = 9.3)
ggplot(fewer_filtered,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black') +
  theme_bw(base_size = 6) +
  theme(panel.spacing = unit(2.5, "lines")) +
  geom_text(data = fewer_text_filtered,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05, size = rel(2), hjust = 1,
            angle = 90) +
  ylab("") +
  xlab("\n") +
  facet_wrap(~L1, scales = "free", nrow = 6) +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white", midpoint = 0,
                       limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_blank(), strip.text = element_blank())
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from all the monkeys (Old and New World) that were uniquely (i.e. only observed in this
##group of species and no other) differentially expressed upon poly(I:C) treatment compared to human.
##Unwrapped plot
monkeys_filtered <- filter(monkeys,
grepl("^human_related_olive_baboon|^squirrel_monkey|^rhesus_macaque|^olive_baboon|^pigtailed_macaque",
        variable)) %>%
  droplevels(.)
monkeys_filtered$variable <- with(monkeys_filtered,
                        factor(variable,levels = c("squirrel_monkey", "rhesus_macaque",
                                                   "pigtailed_macaque", "olive_baboon",
                                                   "human_related_olive_baboon ")))
pdf(file = file.path(output_dir, paste(Sys.Date(),
        "differfromHuman_Monkeys_HumanMapped_unwrapped.pdf")), height = 1.5, width = 5)
ggplot(monkeys_filtered,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height =1)) +
  theme_bw(base_size = 6) +
  geom_text(data = monkeys_text,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
            size = rel(2),
            hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white",
                       midpoint = 0, limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
    coord_cartesian(clip = "off")
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from all the monkeys (Old and New World) that were uniquely (i.e. only observed in this
##group of species and no other) differentially expressed upon poly(I:C) treatment compared to human.
##Wrapped plot
pdf(file = file.path(output_dir, paste(Sys.Date(), "differfromHuman_Monkeys_HumanMapped.pdf")),
    height = 4, width = 5)
ggplot(monkeys,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height =1)) +
  theme_bw(base_size = 6) +
  theme(panel.spacing = unit(2.2, "lines")) +
  geom_text(data = monkeys_text,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
            size = 0.8*6/.pt, # match font size to theme
            hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  facet_wrap(~L1, scales = "free", nrow = 4) +
  scale_fill_gradient2(low = "#e66101", high = "#542788") +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_rect(colour="black", fill="white", linetype="solid"))
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from baboon that were uniquely (i.e. only observed in baboon
##and no other) differentially expressed upon poly(I:C) treatment compared to human.
fewer_baboon <- dplyr::filter(fewer, grepl("baboon", variable)) %>% droplevels()
fewer_text_baboon <- dplyr::filter(fewer_text, grepl("baboon", variable)) %>% droplevels()
fewer_baboon$variable <- with(fewer_baboon,
                        factor(variable,levels = c("olive_baboon",
                                                   "human_related_olive_baboon ")))
pdf(file = file.path(output_dir, paste(Sys.Date(), "differfromHuman_olivebaboon_HumanMapped.pdf")),
    height =0.8, width = 15)
ggplot(fewer_baboon,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black') +
  theme_bw(base_size = 6) +
  geom_text(data = fewer_text_baboon,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.02,
            size = rel(2), hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  theme(legend.position = "none") +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white",
                       midpoint = 0, limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_blank(), strip.text = element_blank())
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from squirrel monkey that were uniquely (i.e. only observed in baboon
##and no other) differentially expressed upon poly(I:C) treatment compared to human.
fewer_sqm <- dplyr::filter(fewer, grepl("squirrel", variable)) %>% droplevels()
fewer_text_sqm <- dplyr::filter(fewer_text, grepl("squirrel", variable)) %>% droplevels()
fewer_sqm$variable <- with(fewer_sqm,
                        factor(variable,levels = c("squirrel_monkey",
                                                   "human_related_squirrel_monkey ")))
pdf(file = file.path(output_dir, paste(Sys.Date(),
  "differfromHuman_squirrelmonkey_HumanMapped.pdf")), height =0.8, width = 20)
ggplot(fewer_sqm,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black') +
  theme_bw(base_size = 6) +
  geom_text(data = fewer_text_sqm,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.02,
            size = rel(2), hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  theme(legend.position = "none") +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white",
                      midpoint = 0, limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_blank(), strip.text = element_blank())
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from all the Old World monkey that were uniquely (i.e. only observed in this
##group of species and no other) differentially expressed upon poly(I:C) treatment compared to human.
##Wrapped plot
pdf(file = file.path(output_dir, paste(Sys.Date(), "differfromHuman_OWMs_HumanMapped.pdf")),
    height = 3, width = 5)
ggplot(OWMs_all,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height =1)) +
  theme_bw(base_size = 6) +
  theme(panel.spacing = unit(2.2, "lines")) +
  geom_text(data = OWMs_all_text,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
            size = 0.8*6/.pt, # match font size to theme
            hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  facet_wrap(~L1, scales = "free", nrow = 3) +
  scale_fill_gradient2(low = "#e66101", high = "#542788") +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_rect(colour="black", fill="white", linetype="solid"))
dev.off()
```

```
## pdf
##   2
```

```r
##Genes from all the Old World monkey that were uniquely (i.e. only observed in this
##group of species and no other) differentially expressed upon poly(I:C) treatment compared to human.
##Unwrapped plot
OWMs_all_filtered <- filter(OWMs_all,
 grepl("^human_related_olive_baboon|^rhesus_macaque|^olive_baboon|^pigtailed_macaque", variable))
OWMs_all_filtered$variable <- with(OWMs_all_filtered,
                        factor(variable,levels = c("rhesus_macaque",
                                                   "pigtailed_macaque", "olive_baboon",
                                                   "human_related_olive_baboon ")))
pdf(file = file.path(output_dir, paste(Sys.Date(),
           "differfromHuman_OWMs_HumanMapped_unwrapped.pdf")),  height = 1, width = 5)
ggplot(OWMs_all_filtered,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height =1)) +
  theme_bw(base_size = 6) +
  geom_text(data = OWMs_all_text,
           aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
           size = rel(2),
           hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white",
                      midpoint = 0, limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
    coord_cartesian(clip = "off")
dev.off()
```

```
## pdf
##   2
```

```r
##Genes common to all the NHP species that were differentially expressed upon poly(I:C) treatment
##compared to human.
##Wrapped plot
common_species_filtered <- filter(common_species,
grepl("^human_related_olive|^rhesus|^olive|^pigtailed|^bonobo|^orangutan|^squirrel|^chimp|^gorilla",
      variable)) %>%
      droplevels()
common_species_filtered$variable <- with(common_species_filtered,
              factor(variable,levels = c("squirrel_monkey", "pigtailed_macaque","rhesus_macaque",
                                         "olive_baboon", "orangutan", "gorilla", "bonobo",
                                          "chimpanzee", "human_related_olive_baboon ")))
pdf(file = file.path(output_dir, paste(Sys.Date(),
    "differfromHuman_all_species_HumanMapped_unwrapped.pdf")), height = 1.5, width = 2.7)
ggplot(common_species_filtered, aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height = 1)) +
  theme_bw(base_size = 6) +
  geom_text(data = common_species_text,
           aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
           size = rel(2),
           hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  scale_fill_gradient2(low = "#e66101", high = "#542788", mid = "white", midpoint = 0,
                      limits = c(-6, 11)) +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
    coord_cartesian(clip = "off")
dev.off()
```

```
## pdf
##   2
```

```
##Genes common to all the NHP species that were differentially expressed upon poly(I:C) treatment
##compared to human.
##Wrapped plot
pdf(file = file.path(output_dir, paste(Sys.Date(),
 "differfromHuman_all_species_HumanMapped_wrapped.pdf")), height = 8, width = 4)
ggplot(common_species,  aes(SYMBOL, variable, fill = value)) +
  geom_tile(color = 'black', aes(height =1)) +
  theme_bw(base_size = 6) +
  theme(panel.spacing = unit(2.2, "lines")) +
  geom_text(data = common_species_text,
            aes(fontface = faces, label = SYMBOL, x = SYMBOL), y = 0.05,
            size = 0.8*6/.pt, # match font size to theme
            hjust = 1, angle = 90) +
  ylab("") +
  xlab("\n") +
  facet_wrap(~L1, scales = "free", nrow = 8) +
  scale_fill_gradient2(low = "#e66101", high = "#542788") +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_text(margin = margin(b = 25))) +
  coord_cartesian(clip = "off") +
  theme(strip.background = element_rect(colour="black", fill="white", linetype="solid"))
dev.off()
```

```
## pdf
##   2
```

```
## Default upset() arguments
sets_of_interest <- list(list("squirrel_monkey"), list("baboon"), list("orangutan"), list("rhesus"),
                    list("bonobo"), list("chimp"), list("gorilla"), list("pigtailed"),
                    list("pigtailed", "rhesus", "baboon"),
list("pigtailed", "rhesus", "baboon", "squirrel_monkey"),
list("pigtailed", "rhesus", "baboon", "squirrel_monkey", "gorilla", "orangutan", "chimp", "bonobo"),
list("pigtailed", "rhesus"), list("orangutan", "squirrel_monkey"),
list("pigtailed", "rhesus", "squirrel_monkey"),
list("orangutan", "pigtailed", "rhesus", "baboon", "squirrel_monkey"))

dot_colors <- c("purple", "darkgreen", "#bee1f4", "forestgreen", "#0088ce",
   "#7dc3e8", "#00659c", "darkolivegreen3", "orange", "orange", "grey55", "grey55", "grey55",
  "grey55", "orange")
nsets = 8; nintersects = 40; sets = NULL; keep.order = F; set.metadata = NULL;
intersections = sets_of_interest; matrix.color = "grey24"; main.bar.color = "gray55";
mainbar.y.label = ""; mainbar.y.max = NULL; sets.bar.color = "grey24"; sets.x.label = "Set Size";
point.size = 2.5; line.size = 0.4; mb.ratio = c(0.70,0.30); expression = NULL; att.pos = NULL;
att.color = dot_colors; order.by = 'freq'; decreasing = T; show.numbers = "yes";
number.angles = 0; group.by = "degree"; cutoff = NULL; queries = NULL; query.legend = "none";
shade.color = "gray88"; shade.alpha = 0.25; matrix.dot.alpha =0.5; sempty.intersections = NULL;
color.pal = 1; boxplot.summary = c("Average Expression"); attribute.plots = NULL;
```

```r
scale.intersections = "identity"; scale.sets = "identity"; text.scale = 1.4; set_size.angles = 0 ;
set_size.show = FALSE;  set_size.numbers_size = NULL; set_size.scale_max = NULL


data = as.data.frame(zoo_avg_df[2:10]); matrix.color="blue";
sets.bar.color = c("purple", "darkgreen", "forestgreen", "darkolivegreen3", "#bee1f4", "#0088ce",
                   "#00659c", "#7dc3e8")

## Modified internal upset() code

startend <- UpSetR:::FindStartEnd(data)
  first.col <- startend[1]
  last.col <- startend[2]

  if(color.pal == 1){
    palette <- c("#1F77B4", "#FF7F0E", "#2CA02C", "#D62728", "#9467BD", "#8C564B", "#E377C2",
                 "#7F7F7F", "#BCBD22", "#17BECF")
  } else{
    palette <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
                 "#CC79A7")
  }

  if(is.null(intersections) == F){
    Set_names <- unique((unlist(intersections)))
    Sets_to_remove <- UpSetR:::Remove(data, first.col, last.col, Set_names)
    New_data <- UpSetR:::Wanted(data, Sets_to_remove)
    Num_of_set <- UpSetR:::Number_of_sets(Set_names)
    if(keep.order == F){
      Set_names <- UpSetR:::order_sets(New_data, Set_names)
    }
    All_Freqs <- UpSetR:::specific_intersections(data, first.col, last.col, intersections, order.by, gro
                                    cutoff, main.bar.color, Set_names)
  } else if(is.null(intersections) == T){
    Set_names <- sets
    if(is.null(Set_names) == T || length(Set_names) == 0 ){
      Set_names <- UpSetR:::FindMostFreq(data, first.col, last.col, nsets)
    }
    Sets_to_remove <- UpSetR:::Remove(data, first.col, last.col, Set_names)
    New_data <- UpSetR:::Wanted(data, Sets_to_remove)
    Num_of_set <- UpSetR:::Number_of_sets(Set_names)
    if(keep.order == F){
    Set_names <- UpSetR:::order_sets(New_data, Set_names)
    }
    All_Freqs <- UpSetR:::Counter(New_data, Num_of_set, first.col, Set_names, nintersects, main.bar.colo
                         order.by, group.by, cutoff, empty.intersections, decreasing)
  }
  Matrix_setup <- UpSetR:::Create_matrix(All_Freqs)
  labels <- UpSetR:::Make_labels(Matrix_setup)
  #Chose NA to represent NULL case as result of NA being inserted when at least one contained both x an
  #i.e. if one custom plot had both x and y, and others had only x, the y's for the other plots were NA
  #if I decided to make the NULL case (all x and no y, or vice versa), there would have been alot more
  #NA can be indexed so that we still get the non NA y aesthetics on correct plot. NULL cant be indexed
  att.x <- c(); att.y <- c();
  if(is.null(attribute.plots) == F){
    for(i in seq_along(attribute.plots$plots)){
      if(length(attribute.plots$plots[[i]]$x) != 0){
        att.x[i] <- attribute.plots$plots[[i]]$x
      }
      else if(length(attribute.plots$plots[[i]]$x) == 0){
        att.x[i] <- NA
      }
      if(length(attribute.plots$plots[[i]]$y) != 0){
        att.y[i] <- attribute.plots$plots[[i]]$y
      }
      else if(length(attribute.plots$plots[[i]]$y) == 0){
        att.y[i] <- NA
      }
    }
  }

## Generate boxplot summary plots
BoxPlotsPlot <- function(bdat, att, att_color){
  ylab <- element_blank()
  col <- match(att, colnames(bdat))
  colnames(bdat)[col] <- "attribute"
  upper_xlim <- as.numeric((max(bdat$x) + 1))
  #upper_ylim <- max((data$`Average Expression`) + 1)
  #lower_ylim <- min((data$`Average Expression`) - 1)
  plot_lims <- as.numeric(0:upper_xlim)
  bdat$x <- as.factor(bdat$x)
  boxplotter <- ggplot(data = bdat, aes(x=x, y=attribute, fill=x, color = x)) +
    geom_point()
boxplots <- boxplotter +
  geom_dotplot(binaxis = "y", stackdir="center", method = "histodot", binwidth = 0.3) +
  theme_bw() +
  scale_x_discrete(limits = plot_lims, expand = c(0,0)) +
                         #+ scale_y_continuous(breaks = seq(lower_ylim, upper_ylim, 2), limits = c(lowe
                         theme(plot.margin = unit(c(-0.7,0,0,3.15), "cm"),
                                axis.title.y = element_blank(),
                                axis.ticks.x = element_blank(),
                                axis.text.x = element_blank(),
                                panel.border = element_blank(),
                                panel.background = element_rect(fill = NA),
                                panel.grid.major = element_line(colour = "gray"),
                                legend.position = "none",
                                axis.title.x = element_blank()) +
                scale_fill_manual(values = att_color) + scale_color_manual(values = att_color)
  return(boxplots)
}

  BoxPlots <- NULL
  if(is.null(boxplot.summary) == F){
    BoxData <- UpSetR:::IntersectionBoxPlot(All_Freqs, New_data, first.col, Set_names)
    BoxPlots <- list()
    for(i in seq_along(boxplot.summary)){
      BoxPlots[[i]] <- BoxPlotsPlot(BoxData, boxplot.summary[i], att.color)
    }
  }

  customAttDat <- NULL
  customQBar <- NULL
  Intersection <- NULL
  Element <- NULL
  legend <- NULL
  EBar_data <- NULL
  if(is.null(queries) == F){
    custom.queries <- UpSetR:::SeperateQueries(queries, 2, palette)
    customDat <- UpSetR:::customQueries(New_data, custom.queries, Set_names)
    legend <- UpSetR:::GuideGenerator(queries, palette)
    legend <- UpSetR:::Make_legend(legend)
    if(is.null(att.x) == F && is.null(customDat) == F){
      customAttDat <- UpSetR:::CustomAttData(customDat, Set_names)
    }
    customQBar <- UpSetR:::customQueriesBar(customDat, Set_names, All_Freqs, custom.queries)
  }
  if(is.null(queries) == F){
    Intersection <- UpSetR:::SeperateQueries(queries, 1, palette)
    Matrix_col <- UpSetR:::intersects(QuerieInterData, Intersection, New_data, first.col, Num_of_set,
                                   All_Freqs, expression, Set_names, palette)
    Element <- UpSetR:::SeperateQueries(queries, 1, palette)
    EBar_data <-UpSetR:::ElemBarDat(Element, New_data, first.col, expression, Set_names,palette, All_Fr
  } else{
    Matrix_col <- NULL
  }

 Matrix_layout <- UpSetR:::Create_layout(Matrix_setup, matrix.color, Matrix_col, matrix.dot.alpha)


  for(i in 1:8) {
      j <- which(Matrix_layout$y == i & Matrix_layout$value == 1)
      if(length(j) > 0) Matrix_layout$color[j] <- c("purple", "darkgreen", "forestgreen", "darkolivegre
            "#0088ce", "#00659c", "#7dc3e8")[i]
  }
## continuing with upset()

  Set_sizes <- UpSetR:::FindSetFreqs(New_data, first.col, Num_of_set, Set_names, keep.order)
  Bar_Q <- NULL
  if(is.null(queries) == F){
    Bar_Q <- UpSetR:::intersects(QuerieInterBar, Intersection, New_data, first.col, Num_of_set, All_Fre
  }
  QInter_att_data <- NULL
  QElem_att_data <- NULL
  if((is.null(queries) == F) & (is.null(att.x) == F)){
    QInter_att_data <- UpSetR:::intersects(QuerieInterAtt, Intersection, New_data, first.col, Num_of_se
                                    expression, Set_names, palette)
    QElem_att_data <- UpSetR:::elements(QuerieElemAtt, Element, New_data, first.col, expression, Set_na
                                 palette)
  }
  AllQueryData <- UpSetR:::combineQueriesData(QInter_att_data, QElem_att_data, customAttDat, att.x, att
  ShadingData <- NULL

  if(is.null(set.metadata) == F){
    ShadingData <- get_shade_groups(set.metadata, Set_names, Matrix_layout, shade.alpha)
    output <- Make_set_metadata_plot(set.metadata, Set_names)
    set.metadata.plots <- output[[1]]
    set.metadata <- output[[2]]

    if(is.null(ShadingData) == FALSE){
    shade.alpha <- unique(ShadingData$alpha)
    }
  } else {
    set.metadata.plots <- NULL
  }
  if(is.null(ShadingData) == TRUE){
  ShadingData <- UpSetR:::MakeShading(Matrix_layout, shade.color)
  }
  Main_bar <- suppressMessages(UpSetR:::Make_main_bar(All_Freqs, Bar_Q, show.numbers, mb.ratio, customQ
                                                      mainbar.y.label, mainbar.y.max, scale.intersectio
Make_matrix_plot <- function(Mat_data,Set_size_data, Main_bar_data, point_size, line_size, text_scale,
                             shading_data, shade_alpha){

  if(length(text_scale) == 1){
    name_size_scale <- text_scale
  }
  if(length(text_scale) > 1 && length(text_scale) <= 6){
    name_size_scale <- text_scale[5]
  }

  Mat_data$line_col <- 'black'

  Matrix_plot <- (ggplot()
                  + theme(panel.background = element_rect(fill = "white"),
                          plot.margin=unit(c(-0.2,0.5,0.5,0.5), "lines"),
                          axis.text.x = element_blank(),
                          axis.ticks.x = element_blank(),
                          axis.ticks.y = element_blank(),
                          axis.text.y = element_text(colour = "gray0",
                                                     size = 8*name_size_scale, hjust = 1),
                          panel.grid.major = element_blank(),
                          panel.grid.minor = element_blank())
                  + xlab(NULL) + ylab("   ")
                  + scale_y_continuous(breaks = c(1:nrow(Set_size_data)),
                                       limits = c(0.5,(nrow(Set_size_data) +0.5)),
                                       labels = labels, expand = c(0,0))
                  + scale_x_continuous(limits = c(0,(nrow(Main_bar_data)+1 )), expand = c(0,0))
                  + geom_rect(data = shading_data, aes_string(xmin = "min", xmax = "max",
                                                              ymin = "y_min", ymax = "y_max"),
                              fill = shading_data$shade_color, alpha = shade_alpha)
                  + geom_line(data= Mat_data, aes_string(group = "Intersection", x="x", y="y",
                                                         colour = "line_col"), size = line_size)
                 + geom_point(data= Mat_data, aes_string(x= "x", y= "y"), colour = Mat_data$color,
                     size= point_size, alpha = Mat_data$alpha, shape=16)
```

```r
                  + scale_color_identity())
  Matrix_plot <- ggplot_gtable(ggplot_build(Matrix_plot))
  return(Matrix_plot)
}

Matrix <- Make_matrix_plot(Matrix_layout, Set_sizes, All_Freqs, point.size, line.size,
                             text.scale, labels, ShadingData, shade.alpha)
  Sizes <- UpSetR:::Make_size_plot(Set_sizes, sets.bar.color, mb.ratio, sets.x.label, scale.sets, text.s
                          set_size.scale_max, set_size.numbers_size)


  ## UpSetR:::Make_base_plot(Main_bar, Matrix, Sizes, labels, mb.ratio, att.x, att.y, New_data,
    ##              expression, att.pos, first.col, att.color, AllQueryData, attribute.plots,
      ##            legend, query.legend, BoxPlots, Set_names, set.metadata, set.metadata.plots)
pdf(file.path(DGE_source, paste(Sys.Date(), "UpSetPlot_DGE_HumanMapped_vs_Humans.pdf")), width = 15,
    height = 7)
structure(class = "upset",
    .Data=list(
      Main_bar = Main_bar,
      Matrix = Matrix,
      Sizes = Sizes,
      labels = labels,
      mb.ratio = mb.ratio,
      att.x = att.x,
      att.y = att.y,
      New_data = New_data,
      expression = expression,
      att.pos = att.pos,
      first.col = first.col,
      att.color = att.color,
      AllQueryData = AllQueryData,
      attribute.plots = attribute.plots,
      legend = legend,
      query.legend = query.legend,
      BoxPlots = BoxPlots,
      Set_names = Set_names,
      set.metadata = set.metadata,
      set.metadata.plots = set.metadata.plots)
  )
dev.off()
```

```
## pdf
##   2
```

Session Info

```r
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] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2              UpSetR_1.4.0
##  [3] gridExtra_2.3               purrr_0.2.5
##  [5] data.table_1.12.0           reshape2_1.4.3
##  [7] ggrepel_0.8.0               usethis_1.4.0
##  [9] devtools_2.0.1              viridis_0.5.1
## [11] viridisLite_0.3.0           RColorBrewer_1.1-2
## [13] ggplot2_3.1.0               gplots_3.0.1
## [15] DESeq2_1.22.2               SummarizedExperiment_1.12.0
## [17] DelayedArray_0.8.0          BiocParallel_1.16.5
## [19] matrixStats_0.54.0          Biobase_2.42.0
## [21] GenomicRanges_1.34.0        GenomeInfoDb_1.18.1
## [23] IRanges_2.16.0              S4Vectors_0.20.1
## [25] BiocGenerics_0.28.0         genefilter_1.64.0
## [27] biomaRt_2.38.0              stringr_1.3.1
## [29] tibble_2.0.1                dplyr_0.7.8
## [31] plyr_1.8.4
##
## loaded via a namespace (and not attached):
##  [1] fs_1.2.6               bitops_1.0-6           bit64_0.9-7
##  [4] progress_1.2.0         httr_1.4.0             rprojroot_1.3-2
##  [7] tools_3.5.2            backports_1.1.3        R6_2.3.0
## [10] rpart_4.1-13           KernSmooth_2.23-15     Hmisc_4.1-1
## [13] DBI_1.0.0              lazyeval_0.2.1         colorspace_1.4-0
## [16] nnet_7.3-12            withr_2.1.2            processx_3.2.1
## [19] tidyselect_0.2.5       prettyunits_1.0.2      bit_1.1-14
## [22] compiler_3.5.2         cli_1.0.1              htmlTable_1.13.1
## [25] desc_1.2.0             labeling_0.3           caTools_1.17.1.1
## [28] scales_1.0.0           checkmate_1.9.1        callr_3.1.1
## [31] digest_0.6.18          foreign_0.8-71         rmarkdown_1.11
## [34] XVector_0.22.0         base64enc_0.1-3        pkgconfig_2.0.2
## [37] htmltools_0.3.6        sessioninfo_1.1.1      htmlwidgets_1.3
## [40] rlang_0.3.1            rstudioapi_0.9.0       RSQLite_2.1.1
## [43] bindr_0.1.1            gtools_3.8.1           acepack_1.4.1
## [46] RCurl_1.95-4.11        magrittr_1.5           GenomeInfoDbData_1.2.0
## [49] Formula_1.2-3          Matrix_1.2-15          Rcpp_1.0.0
## [52] munsell_0.5.0          stringi_1.2.4          yaml_2.2.0
## [55] zlibbioc_1.28.0        pkgbuild_1.0.2         grid_3.5.2
## [58] blob_1.1.1             gdata_2.18.0           crayon_1.3.4
## [61] lattice_0.20-38        splines_3.5.2          annotate_1.60.0
## [64] hms_0.4.2              locfit_1.5-9.1         ps_1.3.0
## [67] knitr_1.21             pillar_1.3.1           pkgload_1.0.2
## [70] geneplotter_1.60.0     XML_3.98-1.16          glue_1.3.0
## [73] evaluate_0.12          latticeExtra_0.6-28    remotes_2.0.2
## [76] gtable_0.2.0           assertthat_0.2.0       xfun_0.4
## [79] xtable_1.8-3           survival_2.43-3        AnnotationDbi_1.44.0
## [82] memoise_1.1.0          cluster_2.0.7-1
```