PCA plotting of RNASeq reads after mapping to human versus species-specific genomes

Load required libraries

```
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
```

```
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
```

```
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
```

```
library(ggplot2)
library(RColorBrewer)
library(stringr)
library(devtools)
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
```

```
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
```

```
library(ape)
```

Purpose

To compare by PCA the outputs of mapping reads from the dermal fibroblasts +/- poly(I:C) to the human genome versus their species-specific genome as it currently exists on ENSEMBL.

```
##Read in the previously generated rlog(dds, blind = TRUE) outputs

##From mapping to human genome
##dds1 <- DESeqDataSetFromMatrix(countData = dd,
                               ##colData = sampleTable_setup_again,
                               ##design = ~species + species:donor.n + species:treatment)
##rld_dds1 <- rlog(dds1, blind = TRUE)
rld_dds1_human <-
  get(load("dds_outputs_humanalignments/2019-05-27_rld_dds1_mappedTohuman.Rdata"))

##From mapping to species-specific genomes
##dds1 <- DESeqDataSetFromMatrix(countData = dd,
                               ##colData = sampleTable_setup_again,
                               ##design = ~species + species:donor.n + species:treatment)
```

```
##rld_dds1 <- rlog(dds1, blind = TRUE)
rld_dds1_species <-
  get(load("dds_outputs_speciesspecificalignments/2019-05-26_rld_dds1_mappedTospecies.Rdata"))

rld_dds1_mouse <-
  get(load("Expanded_Design_Factor_Mice/2019-07-10_allGenes_MouseMapped_dds1_rlogdds_MouseGenes.Rdata"))

##Folder for putting generated plots into
output_dir <- "PCA_output"

##Function to make rld outputs into format acceptable for further analysis
matrix_df <- function(input) {
matrix_made <- assay(input)
matrix_df <- as.data.frame(matrix_made)
matrix_df
}

species_df <- matrix_df(rld_dds1_species)
human_df <- matrix_df(rld_dds1_human)
mouse_df <- matrix_df(rld_dds1_mouse)
##Getting column order to be in alphabetical order (so the same) in both data frames
human_df <- human_df[,order(colnames(human_df))]
species_df <- species_df[,order(colnames(species_df))]
```

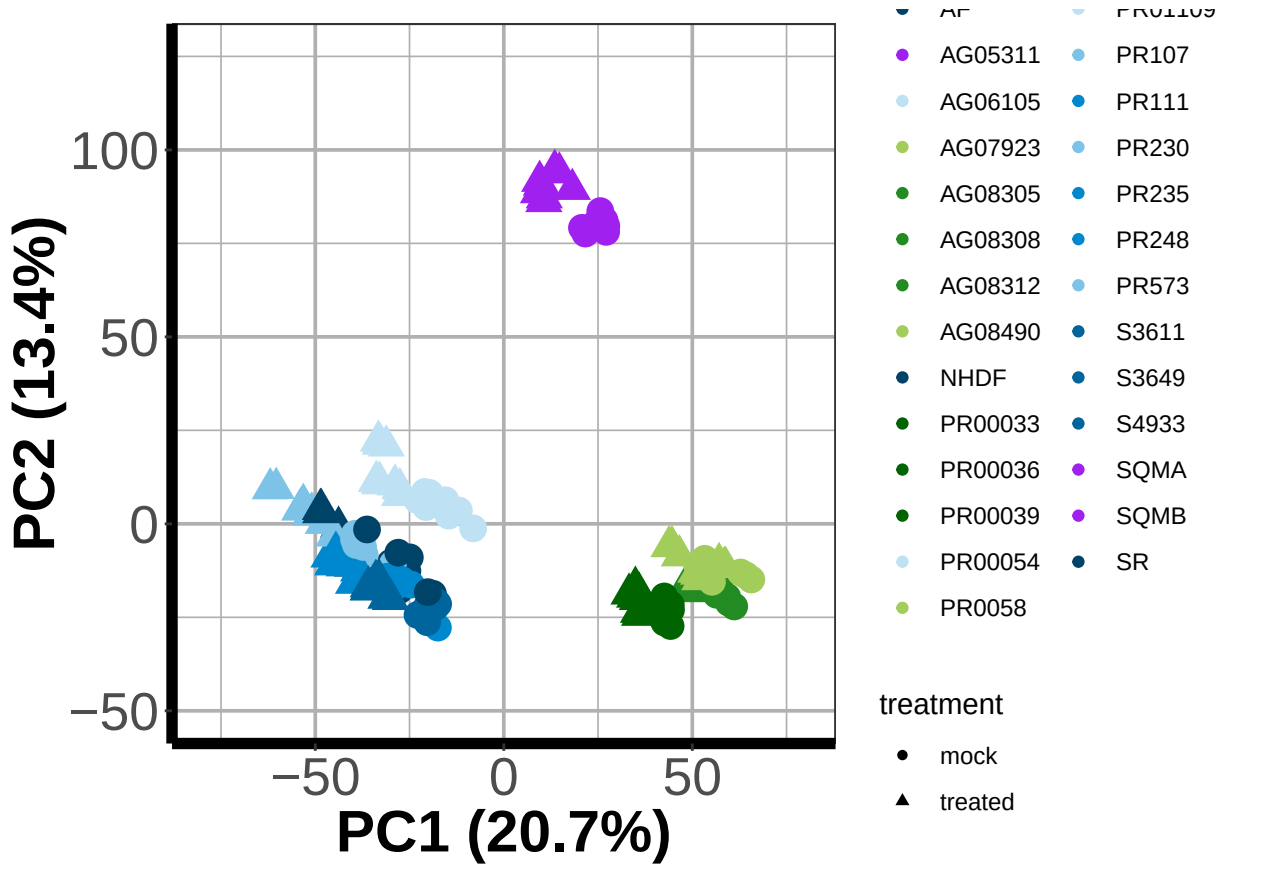
```
##Turn the species_df and human_df so that the column names are the ENSEMBL IDs and the rows
##are the samples
turned_human_df <- t(human_df)
turned_species_df <- t(species_df)
turned_mouse_df <- t(mouse_df)

PCA_calc_human <- prcomp(turned_human_df)
PCA_calc_species <- prcomp(turned_species_df)
PCA_calc_mouse <- prcomp(turned_mouse_df)

##For including on the final plot, I want the percent variance values
perc_var_human=round(100*PCA_calc_human$sdev^2/sum(PCA_calc_human$sdev^2),1)
perc_var_species=round(100*PCA_calc_species$sdev^2/sum(PCA_calc_species$sdev^2),1)
perc_var_mouse=round(100*PCA_calc_mouse$sdev^2/sum(PCA_calc_mouse$sdev^2),1)

##Making a data frame of the information for making the PCA plot
##Mapped to human genome
PCA_human_df <- data.frame(PCA1=PCA_calc_human$x[,1], PCA2=PCA_calc_human$x[,2],
                           sample = rownames(turned_human_df),
                           donor = str_extract(rownames(turned_human_df),
                                               "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|AF|NHDF|SR"),
   treatment = ifelse(grepl("mock|*M\\d|*M\\d\\d", rownames(turned_human_df)), "mock", "treated"),
   replicate = ifelse(grepl("_A_|*24A|*mockA|treatedA|M01|M1|T1|T01|mock A|treated A",
                            rownames(turned_human_df)), "A",
                      ifelse(grepl("_B_|*24B|*mockB|treatedB|M02|M2|T2|T02|mock B|treated B",
                                   rownames(turned_human_df)), "B", "C")))

##Setting the color scheme for the donors
colcoloring = function(donor) {
  ifelse(grepl("AG07923|AG08490|PR0058", donor), "darkolivegreen3",
```

```
      ifelse(grepl("PR00033|PR00036|PR00039", donor), "darkgreen",
        ifelse(grepl("AG05311|SQMA|SQMB", donor), "purple",
          ifelse(grepl("AG06105|PR00054|PR01109", donor), "#bee1f4",
            ifelse(grepl("PR230|PR0230|PR573|PR00573|PR107|PR00107", donor), "#7dc3e8",
              ifelse(grepl("PR111|PR235|PR248|PR00248", donor), "#0088ce",
                ifelse(grepl("S4933|S004933|S3611|S003611|S3649|S003649", donor), "#00659c",
                  ifelse(grepl("AG08308|AG08312|AG08305", donor), "forestgreen",
                    ifelse(grepl("NHDF|AF|SR", donor), "#004368", "grey")))))))))
}

colcolors_human <- unlist(lapply(PCA_human_df$donor, colcoloring))
names(colcolors_human) <- PCA_human_df$donor

##Mapped to species genome
PCA_species_df <- data.frame(PCA1=PCA_calc_species$x[,1], PCA2=PCA_calc_species$x[,2],
                             sample = rownames(turned_species_df),
                             donor = str_extract(rownames(turned_species_df),
                                                 "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|AF|NHDF|SR"),
   treatment = ifelse(grepl("mock|*M\\d|*M\\d\\d", rownames(turned_species_df)), "mock", "treated"),
   replicate = ifelse(grepl("_A_|*24A|*mockA|treatedA|M01|M1|T1|T01|mock A|treated A",
                            rownames(turned_species_df)), "A",
                      ifelse(grepl("_B_|*24B|*mockB|treatedB|M02|M2|T2|T02|mock B|treated B",
                                   rownames(turned_species_df)), "B", "C")))


colcolors_species <- unlist(lapply(PCA_species_df$donor, colcoloring))
names(colcolors_species) <- PCA_species_df$donor

##Mouse samples
PCA_mouse_df <- data.frame(PCA1=PCA_calc_mouse$x[,1], PCA2=PCA_calc_mouse$x[,2],
                           sample = rownames(turned_mouse_df),
                           donor = str_extract(rownames(turned_mouse_df),
                                               "C57\\w"),
   treatment = ifelse(grepl("mock", rownames(turned_mouse_df)), "mock", "treated"),
   replicate = ifelse(grepl("treated A",
                            rownames(turned_mouse_df)), "A",
                      ifelse(grepl("treated B",
                                   rownames(turned_mouse_df)), "B", "C")))

colcoloring_mouse = function(donor) {
  ifelse(donor == "C57A", "deeppink4",
                       ifelse(donor == "C57B", "deeppink2",
                              ifelse(donor == "C57C", "lightpink", "grey")))

}

colcolors_mouse <- unlist(lapply(PCA_mouse_df$donor, colcoloring_mouse))
names(colcolors_mouse) <- PCA_mouse_df$donor
```

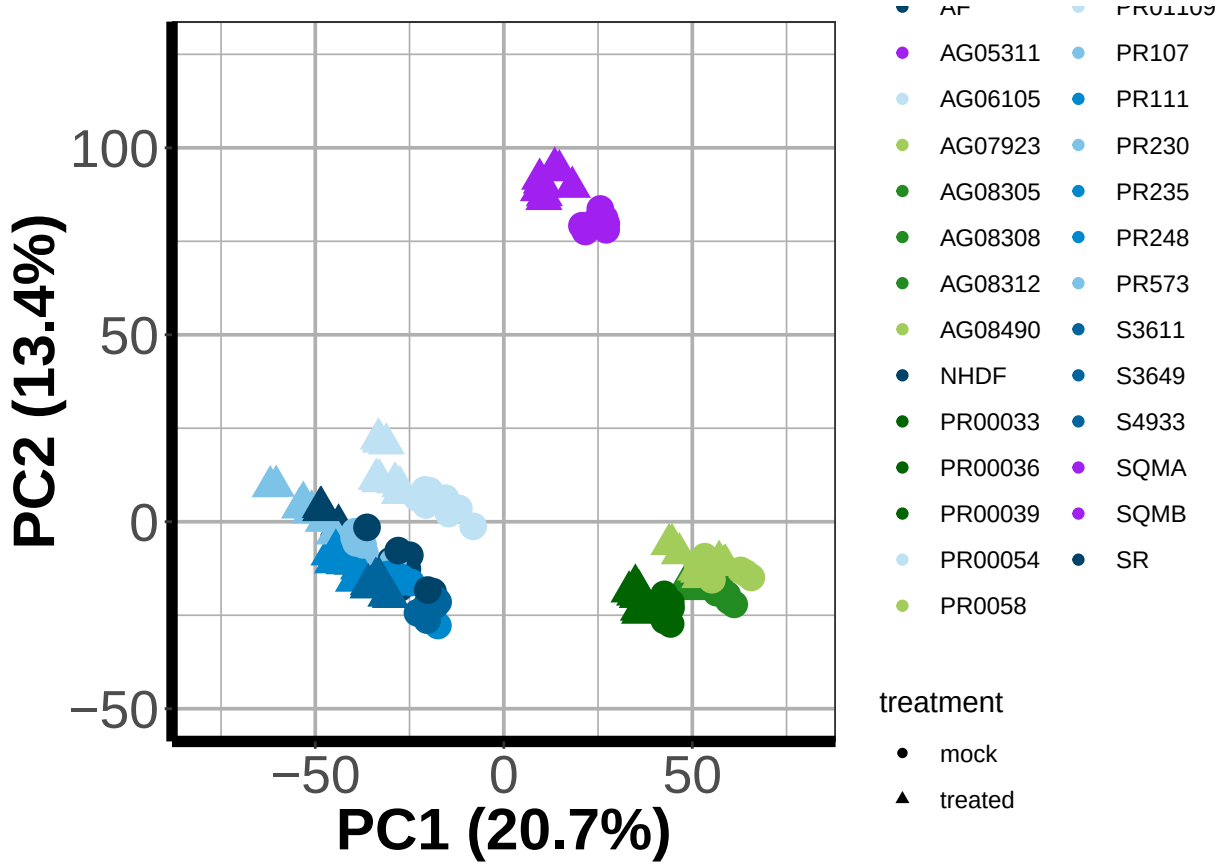
Now the actual PCA plotting

```
##Mapped to human genome
p_PCA_human <- ggplot(PCA_human_df, aes(x = PCA1, y = PCA2, fill = donor, shape = treatment))
p_PCA_human <- p_PCA_human +
```

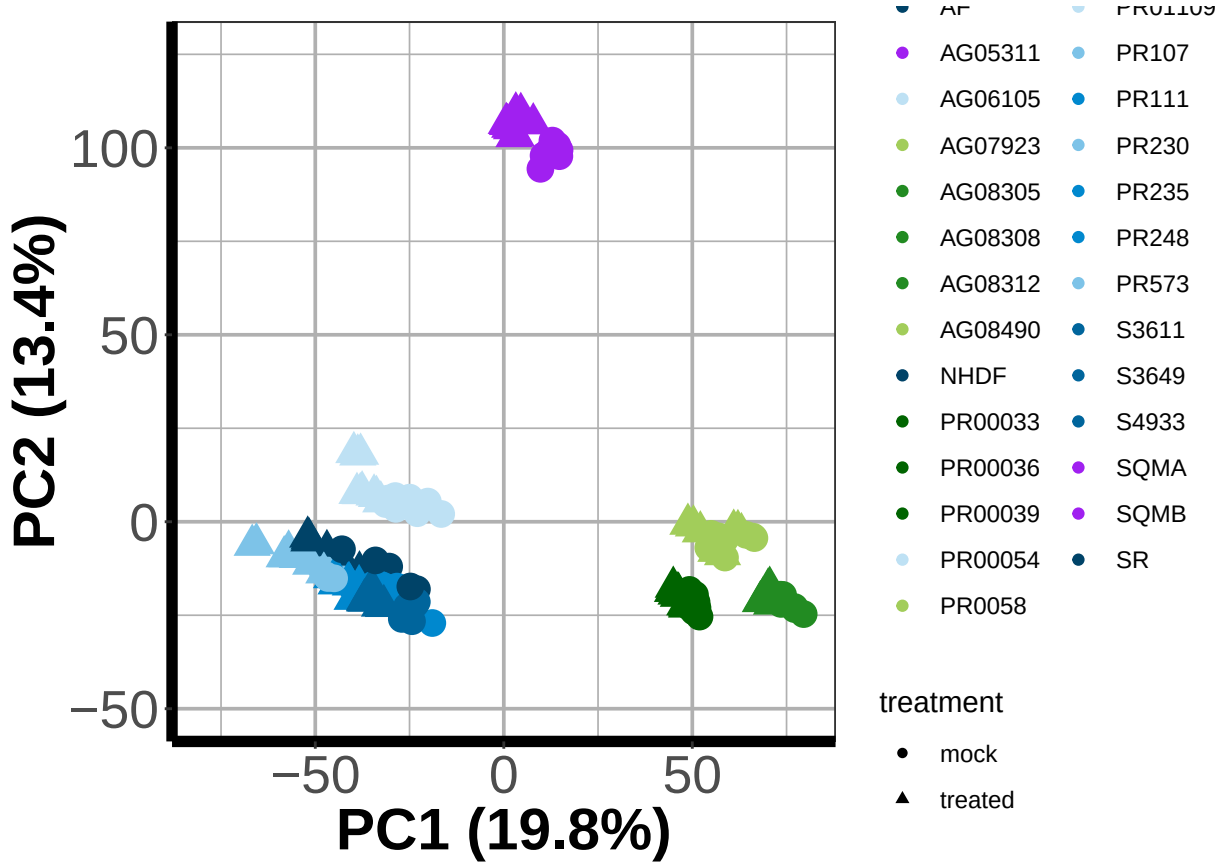
```
  geom_point(aes(colour = donor, shape = treatment, size = 3)) +
    scale_color_manual(values = colcolors_human) +
  labs(x=paste0("PC1 (",perc_var_human[1],"%)"), y=paste0("PC2 (",perc_var_human[2],"%)")) +
  theme_bw() +
  coord_cartesian(xlim = c(-80, 80), ylim = c(-50, 125)) +
  scale_size(guide = "none") +
  theme(axis.title.x = element_text(face = "bold", size = 22),
        axis.text = element_text(size = 20),
        panel.grid.major = element_line(size = 0.65, color = "gray69"),
        panel.grid.minor = element_line(size = 0.3, color = "gray69"),
        axis.line = element_line(size = 2),
        axis.title.y = element_text(face = "bold", size = 22))
p_PCA_human
```



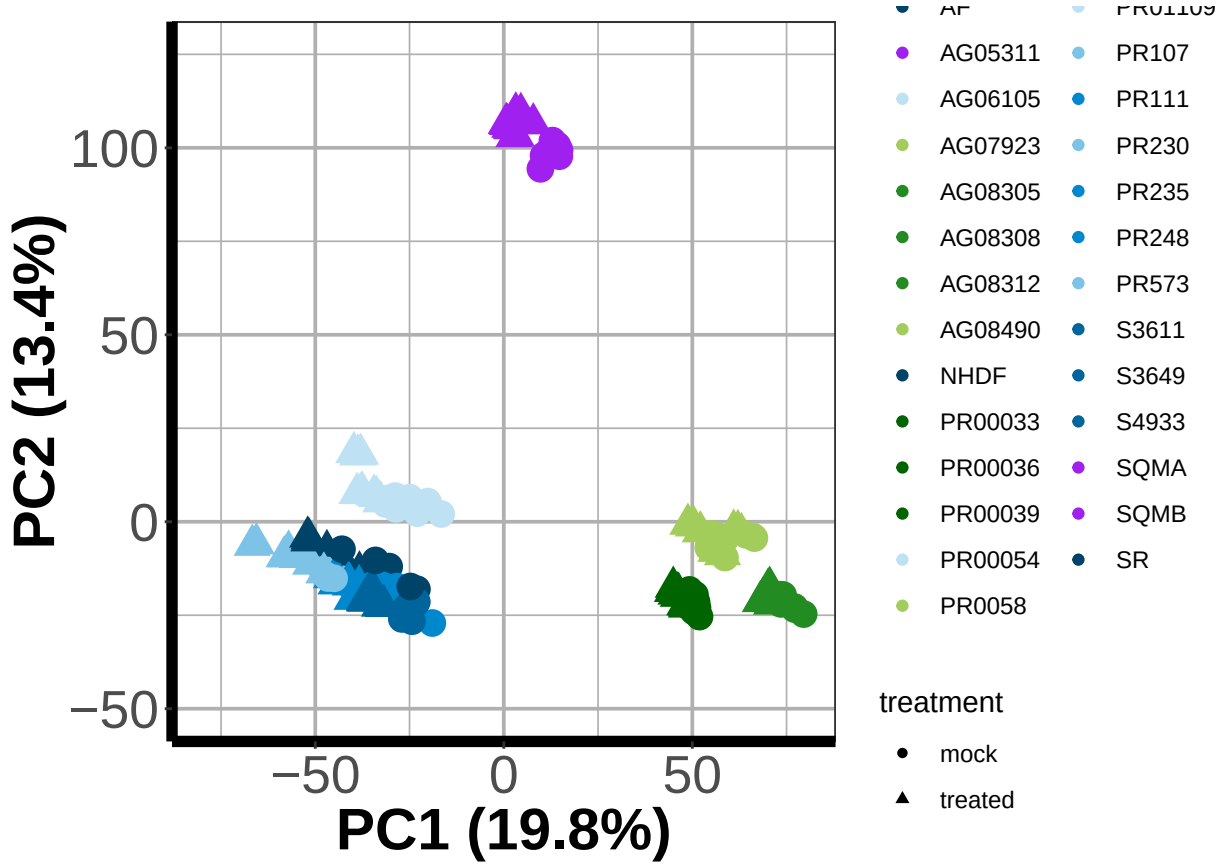

```
ggsave(file = file.path(output_dir, paste(Sys.Date(), "human_mapped_PCA_dds1_one2onehomologs.pdf")),
       plot = p_PCA_human, units = 'in', height = 8, width = 10, dpi = 300, device = "pdf")
print(p_PCA_human)
```



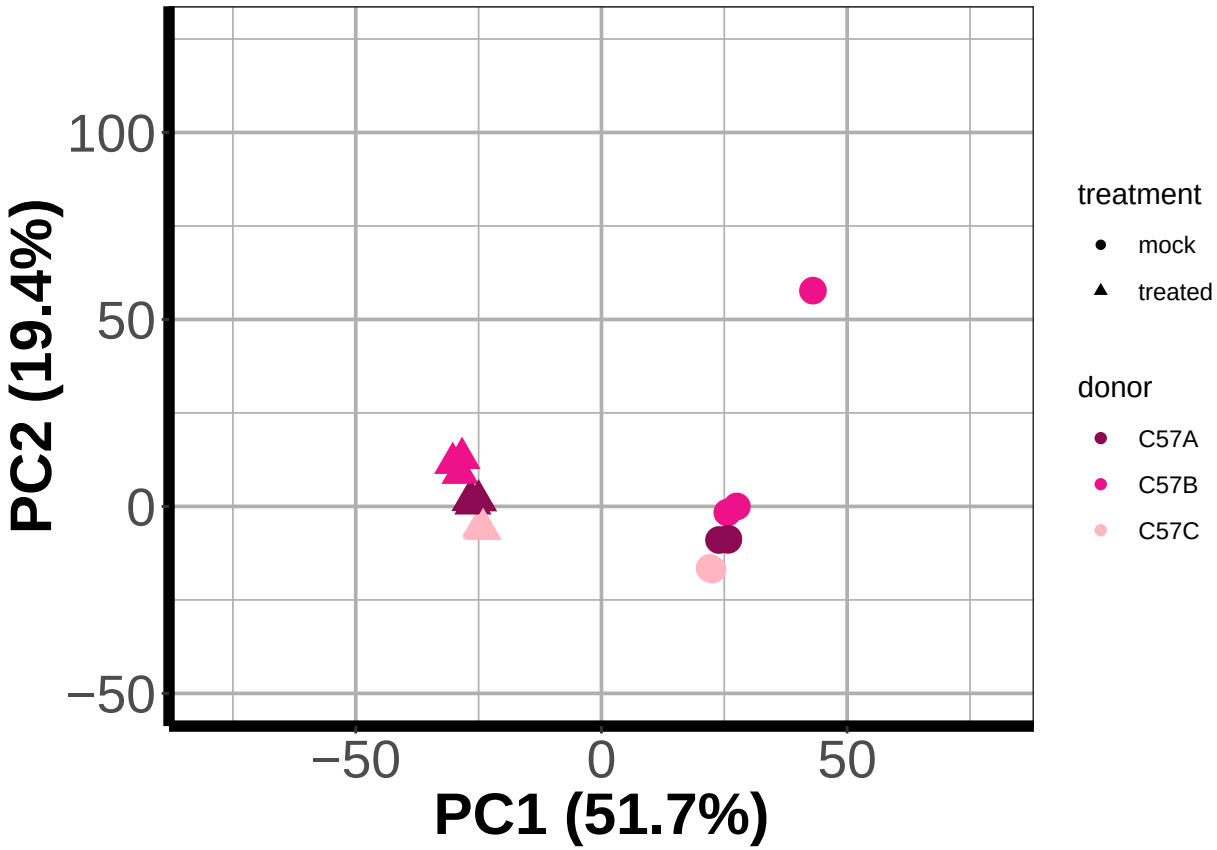

```
p_PCA_species <- ggplot(PCA_species_df, aes(x = PCA1, y = PCA2, fill = donor, shape = treatment))
p_PCA_species <- p_PCA_species +
  geom_point(aes(colour = donor, shape = treatment, size = 3)) +
    scale_color_manual(values = colcolors_species) +
  labs(x=paste0("PC1 (",perc_var_species[1],"%)"), y=paste0("PC2 (",perc_var_species[2],"%)")) +
  theme_bw() +
  coord_cartesian(xlim = c(-80, 80), ylim = c(-50, 125)) +
  scale_size(guide = "none") +
  theme(axis.title.x = element_text(face = "bold", size = 22),
        axis.text = element_text(size = 20),
        panel.grid.major = element_line(size = 0.65, color = "gray69"),
        panel.grid.minor = element_line(size = 0.3, color = "gray69"),
        axis.line = element_line(size = 2),
        axis.title.y = element_text(face = "bold", size = 22))
p_PCA_species
```




```
ggsave(file = file.path(output_dir, paste(Sys.Date(), "species_mapped_PCA_dds1_one2onehomologs.pdf")),
       plot = p_PCA_species, units = 'in', height = 8, width = 10, dpi = 300, device = "pdf")
print(p_PCA_species)
```




```
p_PCA_mouse <- ggplot(PCA_mouse_df, aes(x = PCA1, y = PCA2, fill = donor, shape = treatment))
p_PCA_mouse <- p_PCA_mouse +
  geom_point(aes(colour = donor, shape = treatment, size = 3)) +
    scale_color_manual(values = colcolors_mouse) +
  labs(x=paste0("PC1 (",perc_var_mouse[1],"%)"), y=paste0("PC2 (",perc_var_mouse[2],"%)")) +
  theme_bw() +
  coord_cartesian(xlim = c(-80, 80), ylim = c(-50, 125)) +
  scale_size(guide = "none") +
  theme(axis.title.x = element_text(face = "bold", size = 22),
        axis.text = element_text(size = 20),
        panel.grid.major = element_line(size = 0.65, color = "gray69"),
        panel.grid.minor = element_line(size = 0.3, color = "gray69"),
        axis.line = element_line(size = 2),
        axis.title.y = element_text(face = "bold", size = 22))
p_PCA_mouse
```




```
ggsave(file = file.path(output_dir, paste(Sys.Date(), "mouse_dds1.pdf")),
       plot = p_PCA_mouse, height = 8, width = 10, device = "pdf")
```

```
##Download the InnateDB file of human innate gene symbols
innate <- read.csv(file = "innatedb_curated_genes.csv") %>%
  filter(Species == "9606") %>%
  distinct(Gene.Symbol, .keep_all = TRUE)

innate_mouse <- read.csv(file = "innatedb_curated_genes.csv") %>%
  filter(Species == "10090") %>%
  distinct(Gene.Symbol, .keep_all = TRUE)

##Specifying that we want to work with the ENSEMBL database -- want to use ENSEMBL 96
##since this was the  version we used for processing our RNASeq reads.
#ensembl <- useMart("ENSEMBL_MART_ENSEMBL",
 #         host = "http://apr2019.archive.ensembl.org",
  #        ensemblRedirect = FALSE)
#human_ensembl <- useDataset("hsapiens_gene_ensembl", mart = ensembl)

##For mouse, we need an even older ENSEMBL database.
#ensembl2016 <- useMart("ENSEMBL_MART_ENSEMBL",
 #         host = "http://jul2016.archive.ensembl.org")

#mouse_ensembl <- useDataset("mmusculus_gene_ensembl", mart = ensembl2016)

##The innate gene list that was downloaded from innateDB only listed the gene symbol
##and gave no other identifier. Here, we wrote a function to pull in the ENSEMBL ID, biotype, gene name
```

```
##and description for each of the innate gene symbols in our document we read in as "innate"
#featurepage_symbol <- function(species_ensembl) {
 # getBM(attributes = c('ensembl_gene_id', 'description',
  #                           'external_gene_name', 'gene_biotype'),
   #               filters = 'external_gene_name',
    #                values = innate[,2],
     #             mart = species_ensembl)
#}

#featurepage_symbol_mouse <- function(species_ensembl) {
 # getBM(attributes = c('ensembl_gene_id', 'description',
  #                           'external_gene_name', 'gene_biotype'),
               #   filters = 'external_gene_name',
                #    values = innate_mouse[,2],
                 # mart = species_ensembl)
#}
#feature_mouse_innate <- featurepage_symbol_mouse(mouse_ensembl)

##Using the "featurepage_symbol" function to find specifically the human gene information
##for each of the InnateDB symbols in "innate."
#feature_human_innate <- featurepage_symbol(human_ensembl) %>%
 #  dplyr::rename(., hsapiens_homolog_ensembl_gene = ensembl_gene_id) %>%
  ###So when you pull the symbols from the human ENSEMBL mart that was set up, you get
  ## a class of genes known as LRG_gene which is from the Locus Reference Genomic
  ##record which is a way to distinguish between a gene that has mutliple
  ##sequence variants. Thus, all the entries where the biotype = LRG_gene can be
  ##removed for our purposes.
  #dplyr::filter_at(., vars(contains("biotype")), any_vars((. != "LRG_gene"))) %>%
  #unique() ##there were four rows that were identical
##Note that there are in some cases multiple ENSEMBL IDs for a given gene symbol. Hence,
##there are more rows in this than in the original list of innate immunity genes.
##The number of distinct gene symbols is found by this:
#distinct(feature_human_innate, external_gene_name, .keep_all = TRUE) %>%
  #nrow()
#write.csv(feature_human_innate, paste(Sys.Date(), "InnateDBGeneFeatures.csv"))

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

##With all this information fleshed out for the innateDB genes, we now want to limit
##the data frames we made from the rlog(dds) to these genes.
 innate_human_df <- human_df[rownames(human_df) %in% feature_human_innate[,2],]
 innate_species_df <- species_df[rownames(species_df) %in% feature_human_innate[,2],]
 #innate_mouse_df <- mouse_df[rownames(mouse_df) %in% feature_mouse_innate[,1],]

#write.csv(rownames(innate_human_df),
 #         paste(Sys.Date(), "InnateDB_ENSEMBL_IDs_One2OneOrthosAllSpecies.csv"))
```

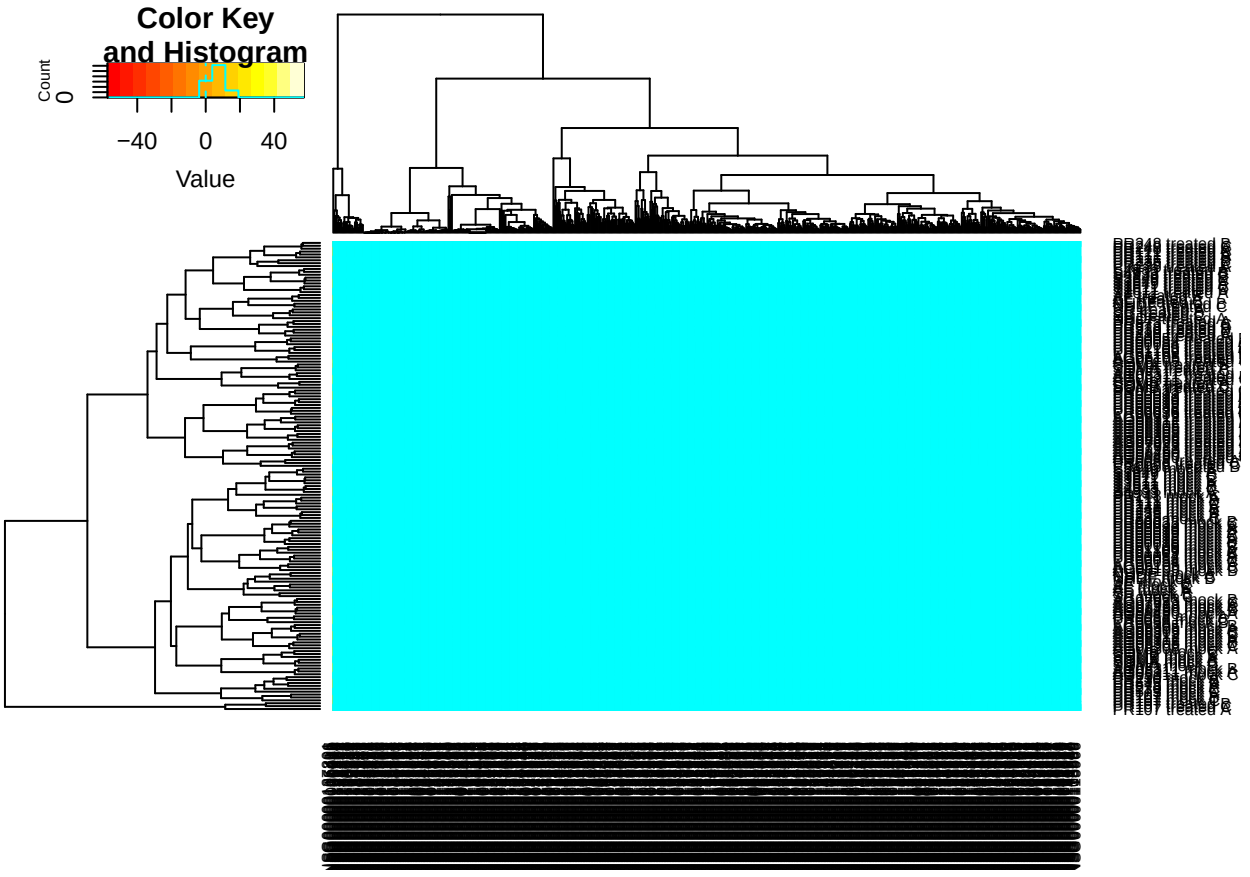
```
##Turn the species_df and human_df so that the column names are the ENSEMBL IDs and the rows
##are the samples
turned_innate_human_df <- t(innate_human_df)
turned_innate_species_df <- t(innate_species_df)
#turned_innate_mouse_df <- t(innate_mouse_df)

PCA_calc_human_innate <- prcomp(turned_innate_human_df)
```

```
PCA_calc_species_innate <- prcomp(turned_innate_species_df)
#PCA_calc_mouse_innate <- prcomp(turned_innate_mouse_df)

##For including on the final plot, I want the percent variance values
perc_var_human_innate=round(100*PCA_calc_human_innate$sdev^2/sum(PCA_calc_human_innate$sdev^2),1)
perc_var_species_innate=round(100*PCA_calc_species_innate$sdev^2/sum(PCA_calc_species_innate$sdev^2),1)
#perc_var_mouse_innate=round(100*PCA_calc_mouse_innate$sdev^2/sum(PCA_calc_mouse_innate$sdev^2),1)

##Making a data frame of the information for making the PCA plot
##Mapped to human genome
PCA_human_innate_df <- data.frame(PCA1=PCA_calc_human_innate$x[,1], PCA2=PCA_calc_human_innate$x[,2],
                          sample = rownames(turned_innate_human_df),
                          donor = str_extract(rownames(turned_innate_human_df),
                                              "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|AF|NHDF|SR"),
   treatment = ifelse(grepl("mock|*M\\d|*M\\d\\d", rownames(turned_innate_human_df)), "mock", "treated"),
   replicate = ifelse(grepl("_A_|*24A|*mockA|treatedA|MO1|M1|T1|TO1|mock A|treated A",
                            rownames(turned_innate_human_df)), "A",
                      ifelse(grepl("_B_|*24B|*mockB|treatedB|MO2|M2|T2|TO2|mock B|treated B",
                                   rownames(turned_innate_human_df)), "B", "C")))

##Setting the color scheme for the donors
colcolors_human_innate <- unlist(lapply(PCA_human_innate_df$donor, colcoloring))
names(colcolors_human_innate) <- PCA_human_innate_df$donor

##Mapped to species-specific genome
PCA_species_innate_df <- data.frame(PCA1=PCA_calc_species_innate$x[,1], PCA2=PCA_calc_species_innate$x[
                          sample = rownames(turned_innate_species_df),
                          donor = str_extract(rownames(turned_innate_species_df),
                                              "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|AF|NHDF|SR"),
   treatment = ifelse(grepl("mock|*M\\d|*M\\d\\d", rownames(turned_innate_species_df)), "mock", "treate
   replicate = ifelse(grepl("_A_|*24A|*mockA|treatedA|MO1|M1|T1|TO1|mock A|treated A",
                            rownames(turned_innate_species_df)), "A",
                      ifelse(grepl("_B_|*24B|*mockB|treatedB|MO2|M2|T2|TO2|mock B|treated B",
                                   rownames(turned_innate_species_df)), "B", "C")))


colcolors_species_innate <- unlist(lapply(PCA_species_innate_df$donor, colcoloring))
names(colcolors_species_innate) <- PCA_species_innate_df$donor

#Mouse
#PCA_mouse_innate_df <- data.frame(PCA1=PCA_calc_mouse_innate$x[,1], PCA2=PCA_calc_mouse_innate$x[,2],
 #                          sample = rownames(turned_innate_mouse_df),
  #                         donor = str_extract(rownames(turned_innate_mouse_df),
  #                                             "C57\\w"),
   #treatment = ifelse(grepl("mock", rownames(turned_innate_mouse_df)), "mock", "treated"),
  # replicate = ifelse(grepl("treated A",
   #                          rownames(turned_innate_mouse_df)), "A",
    #             ifelse(grepl("treated B",
     #                         rownames(turned_innate_mouse_df)), "B", "C")))

#colcoloring_mouse = function(donor) {
 # ifelse(donor == "C57A", "deeppink4",
  #                   ifelse(donor == "C57B", "deeppink2",
```

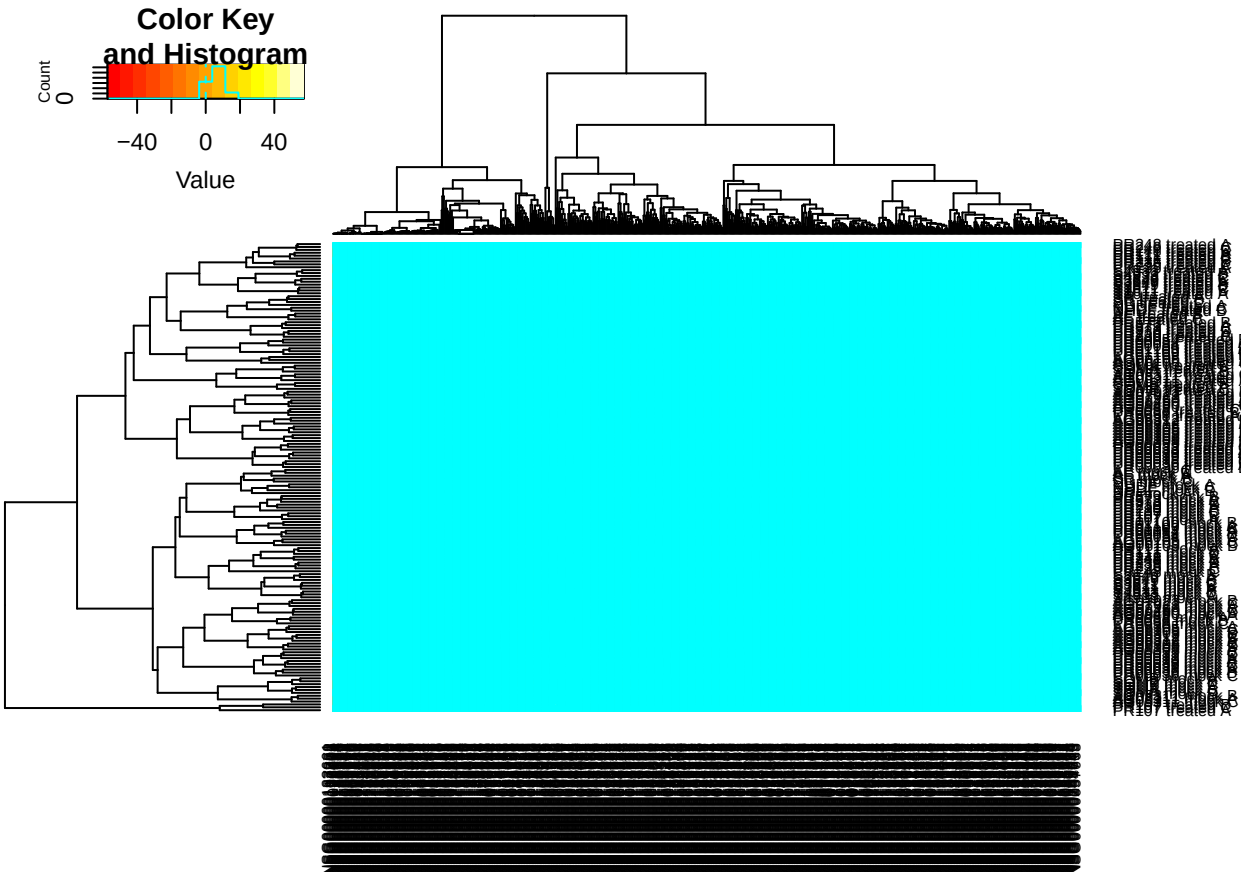
```
   #                              ifelse(donor == "C57C", "lightpink", "grey")))

#}

#colcolors_mouse_innate <- unlist(lapply(PCA_mouse_innate_df$donor, colcoloring_mouse))
#names(colcolors_mouse_innate) <- PCA_mouse_innate_df$donor
```

For a different visualization that takes into account all sources of variance and not just the first two principal components, I will look at a dendrogram of the rlog RNASeq read counts for each of the innate genes when aligning to either the human or the species-specific reference genome to see how the samples cluster.

```
##Heatmap data of the sample clustering can be pulled after using the heatmap.2 function
##Human genome alignment
hm_human_innate <- heatmap.2(turned_innate_human_df)
```




```
hc_human_innate <- as.hclust(hm_human_innate$rowDendrogram )

##Species-specific alignment
hm_species_innate <- heatmap.2(turned_innate_species_df)
```




```
hc_species_innate <- as.hclust(hm_species_innate$rowDendrogram )

##Setting up color schemes
cluster_labels <- function(input) {
  str_extract(input$labels, "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|AF|NHDF|SR")
}

cluster_labels_species <- cluster_labels(hc_species_innate)
cluster_labels_human <- cluster_labels(hc_human_innate)

treatment_labels <- function(input) {
  ifelse(grepl("*mock|*M\\d|*M\\d\\d", input$labels), "mock", "treated")
}

treatment_labels_species <- treatment_labels(hc_species_innate)
treatment_labels_human <- treatment_labels(hc_human_innate)

treatmentcoloring = function(treatment) {
  ifelse(treatment == "mock", "orange", "red")
}

treatmentcolors_species <- unlist(lapply(treatment_labels_species, treatmentcoloring))
treatmentcolors_human <- unlist(lapply(treatment_labels_human, treatmentcoloring))

clustcolors_human <- unlist(lapply(cluster_labels_human, colcoloring))
clustcolors_human
```

```
##   [1] "#004368"         "#004368"         "#004368"
##   [4] "#004368"         "#004368"         "#004368"
##   [7] "purple"          "purple"          "purple"
##  [10] "purple"          "purple"          "purple"
##  [13] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [16] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [19] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [22] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [25] "forestgreen"     "forestgreen"     "forestgreen"
##  [28] "forestgreen"     "forestgreen"     "forestgreen"
##  [31] "forestgreen"     "forestgreen"     "forestgreen"
##  [34] "forestgreen"     "forestgreen"     "forestgreen"
##  [37] "forestgreen"     "forestgreen"     "forestgreen"
##  [40] "forestgreen"     "forestgreen"     "forestgreen"
##  [43] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [46] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [49] "#004368"         "#004368"         "#004368"
##  [52] "#004368"         "#004368"         "#004368"
##  [55] "darkgreen"       "darkgreen"       "darkgreen"
##  [58] "darkgreen"       "darkgreen"       "darkgreen"
##  [61] "darkgreen"       "darkgreen"       "darkgreen"
##  [64] "darkgreen"       "darkgreen"       "darkgreen"
##  [67] "darkgreen"       "darkgreen"       "darkgreen"
##  [70] "darkgreen"       "darkgreen"       "darkgreen"
##  [73] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [76] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [79] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [82] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [85] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [88] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [91] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
##  [94] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
##  [97] "#0088ce"         "#0088ce"         "#0088ce"
## [100] "#0088ce"         "#0088ce"         "#0088ce"
## [103] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [106] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [109] "#0088ce"         "#0088ce"         "#0088ce"
## [112] "#0088ce"         "#0088ce"         "#0088ce"
## [115] "#0088ce"         "#0088ce"         "#0088ce"
## [118] "#0088ce"         "#0088ce"         "#0088ce"
## [121] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [124] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [127] "#00659c"         "#00659c"         "#00659c"
## [130] "#00659c"         "#00659c"         "#00659c"
## [133] "#00659c"         "#00659c"         "#00659c"
## [136] "#00659c"         "#00659c"         "#00659c"
## [139] "#00659c"         "#00659c"         "#00659c"
## [142] "#00659c"         "#00659c"         "#00659c"
## [145] "purple"          "purple"          "purple"
## [148] "purple"          "purple"          "purple"
## [151] "purple"          "purple"          "purple"
## [154] "purple"          "purple"          "purple"
## [157] "#004368"         "#004368"         "#004368"
## [160] "#004368"         "#004368"         "#004368"
```

```
clustcolors_species <- unlist(lapply(cluster_labels_species, colcoloring))
clustcolors_species
```

```
##   [1] "#004368"         "#004368"         "#004368"
##   [4] "#004368"         "#004368"         "#004368"
##   [7] "purple"          "purple"          "purple"
##  [10] "purple"          "purple"          "purple"
##  [13] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [16] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [19] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [22] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [25] "forestgreen"     "forestgreen"     "forestgreen"
##  [28] "forestgreen"     "forestgreen"     "forestgreen"
##  [31] "forestgreen"     "forestgreen"     "forestgreen"
##  [34] "forestgreen"     "forestgreen"     "forestgreen"
##  [37] "forestgreen"     "forestgreen"     "forestgreen"
##  [40] "forestgreen"     "forestgreen"     "forestgreen"
##  [43] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [46] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [49] "#004368"         "#004368"         "#004368"
##  [52] "#004368"         "#004368"         "#004368"
##  [55] "darkgreen"       "darkgreen"       "darkgreen"
##  [58] "darkgreen"       "darkgreen"       "darkgreen"
##  [61] "darkgreen"       "darkgreen"       "darkgreen"
##  [64] "darkgreen"       "darkgreen"       "darkgreen"
##  [67] "darkgreen"       "darkgreen"       "darkgreen"
##  [70] "darkgreen"       "darkgreen"       "darkgreen"
##  [73] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [76] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [79] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [82] "darkolivegreen3" "darkolivegreen3" "darkolivegreen3"
##  [85] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [88] "#bee1f4"         "#bee1f4"         "#bee1f4"
##  [91] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
##  [94] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
##  [97] "#0088ce"         "#0088ce"         "#0088ce"
## [100] "#0088ce"         "#0088ce"         "#0088ce"
## [103] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [106] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [109] "#0088ce"         "#0088ce"         "#0088ce"
## [112] "#0088ce"         "#0088ce"         "#0088ce"
## [115] "#0088ce"         "#0088ce"         "#0088ce"
## [118] "#0088ce"         "#0088ce"         "#0088ce"
## [121] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [124] "#7dc3e8"         "#7dc3e8"         "#7dc3e8"
## [127] "#00659c"         "#00659c"         "#00659c"
## [130] "#00659c"         "#00659c"         "#00659c"
## [133] "#00659c"         "#00659c"         "#00659c"
## [136] "#00659c"         "#00659c"         "#00659c"
## [139] "#00659c"         "#00659c"         "#00659c"
## [142] "#00659c"         "#00659c"         "#00659c"
## [145] "purple"          "purple"          "purple"
## [148] "purple"          "purple"          "purple"
## [151] "purple"          "purple"          "purple"
```

```
## [154] "purple"          "purple"          "purple"
## [157] "#004368"         "#004368"         "#004368"
## [160] "#004368"         "#004368"         "#004368"
```

```
##Alignment of all species with the human genome
png(file = file.path(output_dir, paste(Sys.Date(), "humanalignment_innateDBgenes_phylo.png")), units =
plot(as.phylo(hc_human_innate), tip.color = clustcolors_human, cex = 0.75, label.offset = 1, edge.lty =
     font = 1, no.margin = TRUE, direction = "downwards")
tiplabels(pch = 19, col = treatmentcolors_human)
nodelabels(pch = 15, col = "grey")
add.scale.bar()
dev.off()
```

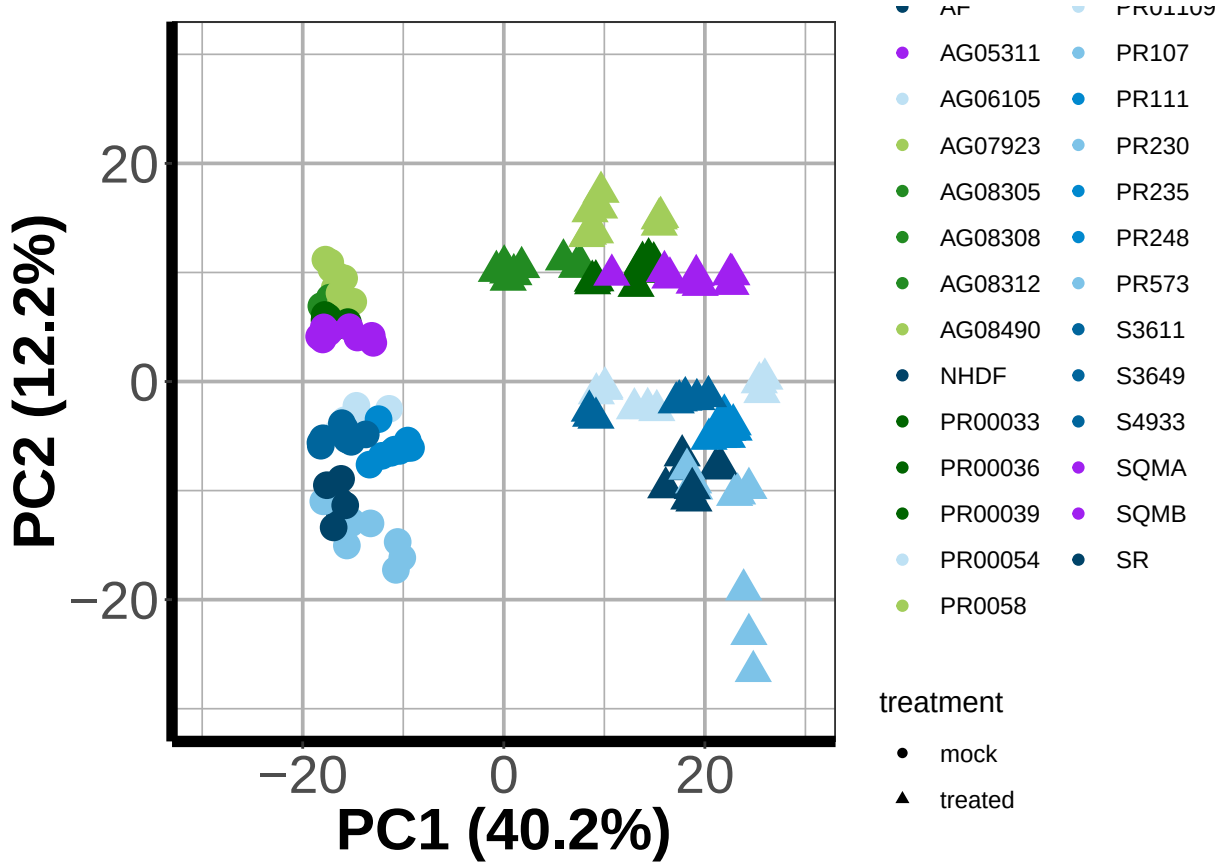
```
## pdf
##   2
```

```
png(file = file.path(output_dir, paste(Sys.Date(), "speciesalignment_innateDBgenes_phylo.png")),
    units = 'in', height = 7,
    width = 20, res = 300)
plot(as.phylo(hc_species_innate), tip.color = clustcolors_species, cex = 0.75, label.offset = 1, edge.l
     font = 1, no.margin = TRUE, direction = "downwards")
tiplabels(pch = 19, col = treatmentcolors_species)
nodelabels(pch = 15, col = "grey")
add.scale.bar()
dev.off()
```

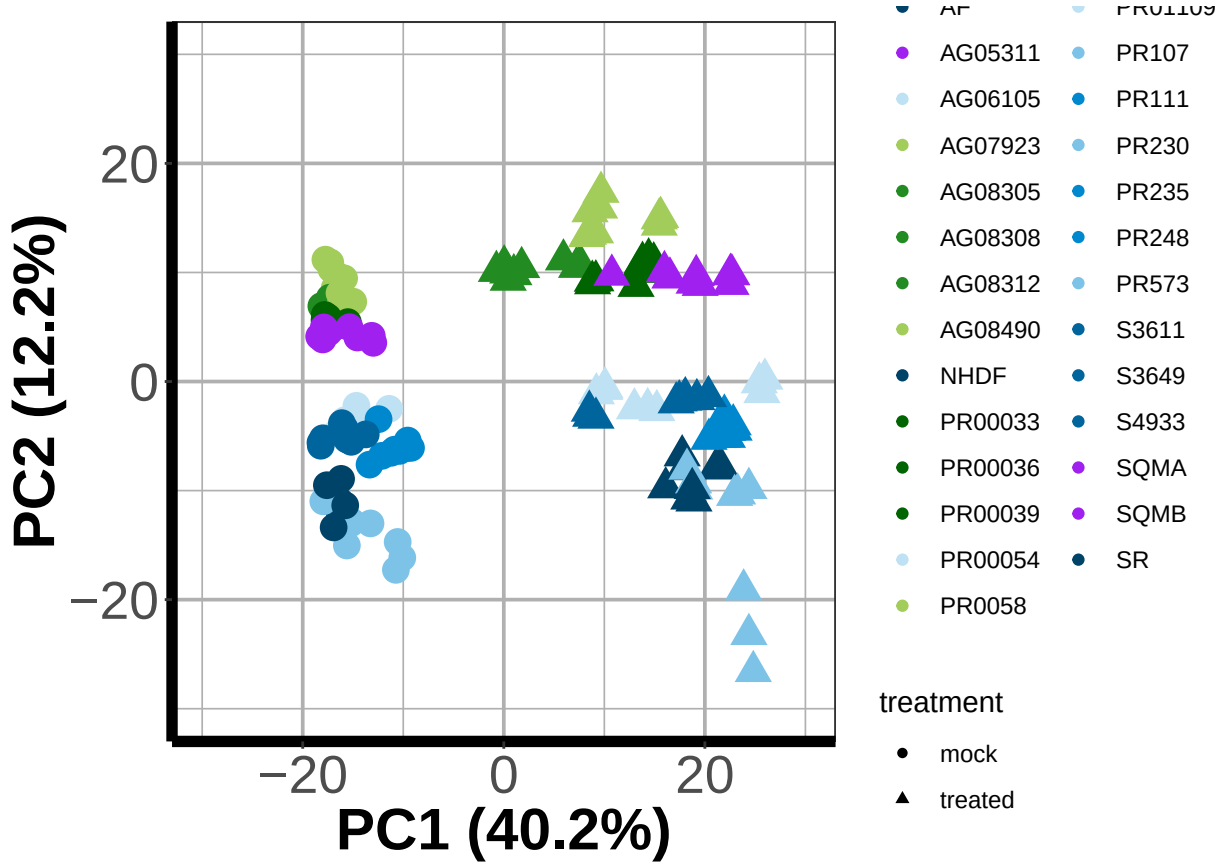
```
## pdf
##   2
```

Now the actual PCA plotting of the innate immune-limited genes

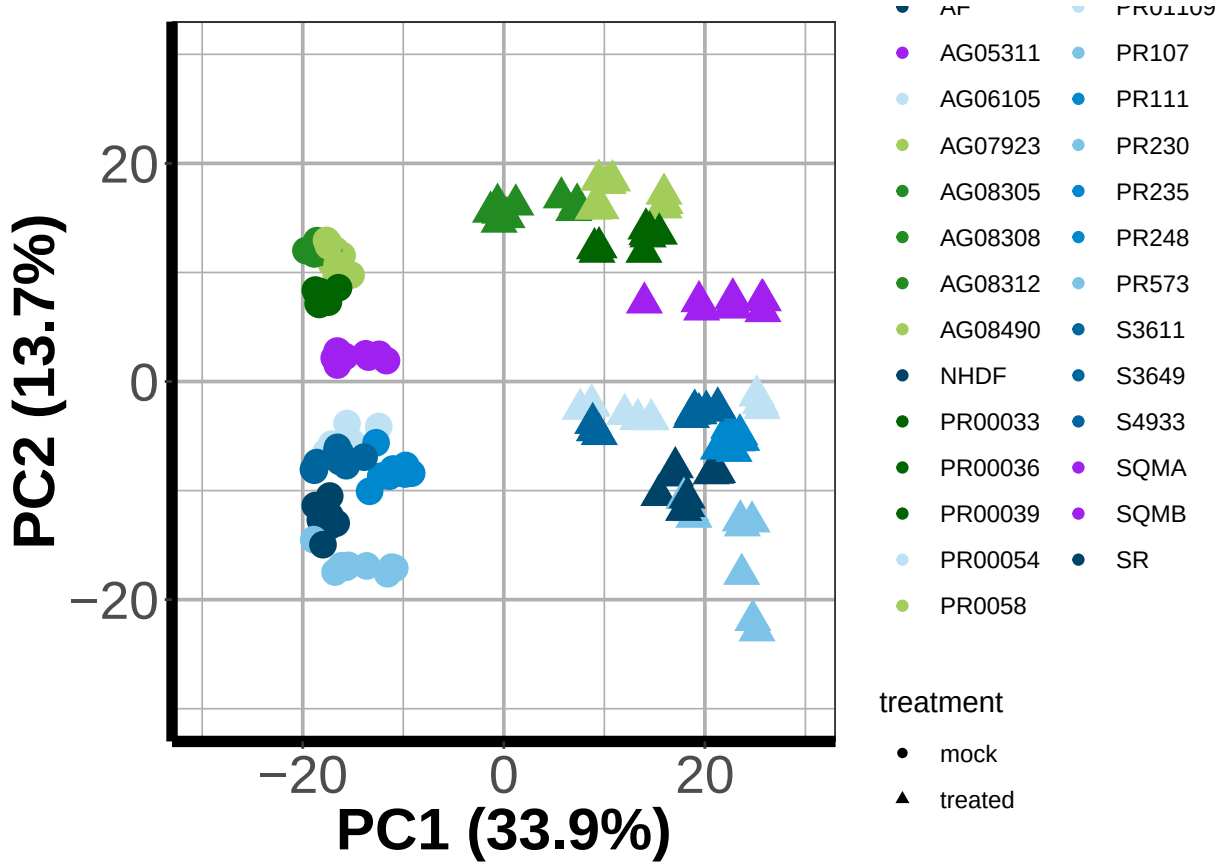
```
##Mapped to human genome
p_PCA_human_innate <- ggplot(PCA_human_innate_df, aes(x = PCA1, y = PCA2, fill = donor, shape = treatme
p_PCA_human_innate <- p_PCA_human_innate +
  geom_point(aes(colour = donor, shape = treatment, size = 3)) +
    scale_color_manual(values = colcolors_human_innate) +
  labs(x=paste0("PC1 (",perc_var_human_innate[1],"%)"), y=paste0("PC2 (",perc_var_human_innate[2],"%)")
  theme_bw() +
  coord_cartesian(xlim = c(-30, 30), ylim = c(-30, 30)) +
  ##geom_text_repel(aes(label = ifelse(grepl("*107|230|573", donor), paste(donor), ""))) +
  scale_size(guide = "none") +
  theme(axis.title.x = element_text(face = "bold", size = 22),
        axis.text = element_text(size = 20),
        panel.grid.major = element_line(size = 0.65, color = "gray69"),
        panel.grid.minor = element_line(size = 0.3, color = "gray69"),
        axis.line = element_line(size = 2),
        axis.title.y = element_text(face = "bold", size = 22))
p_PCA_human_innate
```



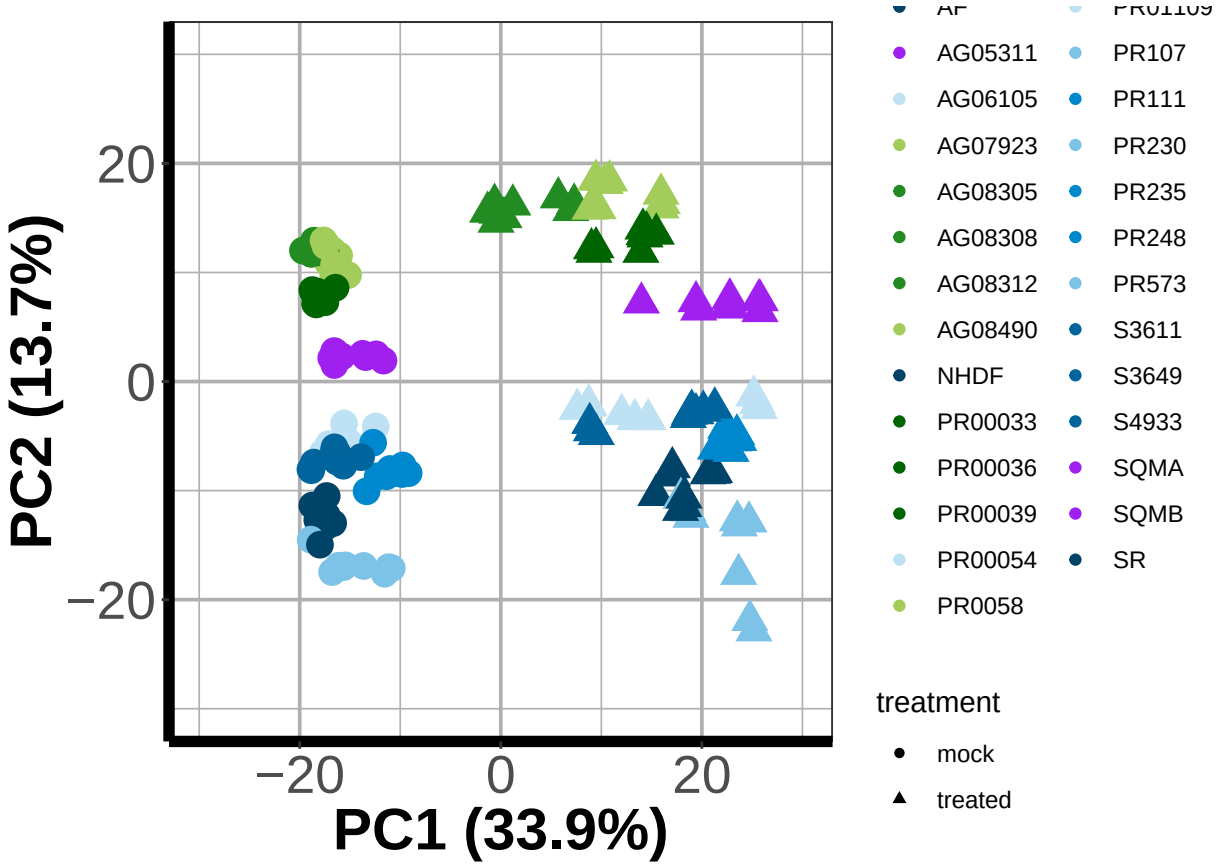

```
ggsave(file = file.path(output_dir, paste(Sys.Date(), "human_mapped_PCA_dds1_innate.pdf")), plot = p_PCA
       units = 'in', height = 8, width = 10, dpi = 300, device = "pdf")
print(p_PCA_human_innate)
```




```
p_PCA_species_innate <- ggplot(PCA_species_innate_df, aes(x = PCA1, y = PCA2, fill = donor, shape = trea
p_PCA_species_innate <- p_PCA_species_innate +
  geom_point(aes(colour = donor, shape = treatment, size = 3)) +
    scale_color_manual(values = colcolors_species_innate) +
  labs(x=paste0("PC1 (",perc_var_species_innate[1],"%)"), y=paste0("PC2 (",perc_var_species_innate[2],"%
  ##geom_text_repel(aes(label = ifelse(grepl("*107|230|573", donor), paste(donor), ""))) +
  theme_bw() +
  coord_cartesian(xlim = c(-30, 30), ylim = c(-30, 30)) +
  scale_size(guide = "none") +
  theme(axis.title.x = element_text(face = "bold", size = 22),
        axis.text = element_text(size = 20),
        panel.grid.major = element_line(size = 0.65, color = "gray69"),
        panel.grid.minor = element_line(size = 0.3, color = "gray69"),
        axis.line = element_line(size = 2),
        axis.title.y = element_text(face = "bold", size = 22))
p_PCA_species_innate
```




```
ggsave(file = file.path(output_dir, paste(Sys.Date(), "species_mapped_PCA_dds1_innate.pdf")),
       plot = p_PCA_species_innate, units = 'in', height = 8, width = 10, dpi = 300, device = "pdf")
print(p_PCA_species_innate)
```




```
#p_PCA_mouse_innate <- ggplot(PCA_mouse_innate_df, aes(x = PCA1, y = PCA2, fill = donor, shape =
  #                                                    treatment))
#p_PCA_mouse_innate <- p_PCA_mouse_innate +
 # geom_point(aes(colour = donor, shape = treatment, size = 3)) +
  #  scale_color_manual(values = colcolors_mouse_innate) +
  #labs(x=paste0("PC1 (",perc_var_mouse_innate[1],"%)"),
   #    y=paste0("PC2 (",perc_var_mouse_innate[2],"%)")) +
  ##geom_text_repel(aes(label = ifelse(grepl("*107|230|573", donor), paste(donor), ""))) +
  #theme_bw() +
  #coord_cartesian(xlim = c(-30, 30), ylim = c(-30, 30)) +
  #scale_size(guide = "none") +
  #theme(axis.title.x = element_text(face = "bold", size = 22),
   #     axis.text = element_text(size = 20),
    #    panel.grid.major = element_line(size = 0.65, color = "gray69"),
     #   panel.grid.minor = element_line(size = 0.3, color = "gray69"),
      #  axis.line = element_line(size = 2),
       # axis.title.y = element_text(face = "bold", size = 22))
#p_PCA_mouse_innate
#ggsave(file = file.path(output_dir, paste(Sys.Date(), "mouse_innate.pdf")),
 #      plot = p_PCA_mouse_innate,  height = 8, width = 10,device = "pdf")
#print(p_PCA_mouse_innate)
```

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] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2              ape_5.3
##  [3] purrr_0.2.5                 data.table_1.12.0
##  [5] reshape2_1.4.3              usethis_1.4.0
##  [7] devtools_2.0.1              RColorBrewer_1.1-2
##  [9] ggplot2_3.1.0               gplots_3.0.1
## [11] DESeq2_1.22.2               SummarizedExperiment_1.12.0
## [13] DelayedArray_0.8.0          BiocParallel_1.16.5
## [15] matrixStats_0.54.0          Biobase_2.42.0
## [17] GenomicRanges_1.34.0        GenomeInfoDb_1.18.1
## [19] IRanges_2.16.0              S4Vectors_0.20.1
## [21] BiocGenerics_0.28.0         genefilter_1.64.0
## [23] biomaRt_2.38.0              stringr_1.3.1
## [25] tibble_2.0.1                dplyr_0.7.8
## [27] plyr_1.8.4
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-137           fs_1.2.6               bitops_1.0-6
##  [4] bit64_0.9-7            progress_1.2.0         httr_1.4.0
##  [7] rprojroot_1.3-2        tools_3.5.2            backports_1.1.3
## [10] R6_2.3.0               rpart_4.1-13           KernSmooth_2.23-15
## [13] Hmisc_4.1-1            DBI_1.0.0              lazyeval_0.2.1
## [16] colorspace_1.4-0       nnet_7.3-12            withr_2.1.2
## [19] processx_3.2.1         tidyselect_0.2.5       gridExtra_2.3
## [22] prettyunits_1.0.2      bit_1.1-14             compiler_3.5.2
## [25] cli_1.0.1              htmlTable_1.13.1       desc_1.2.0
## [28] labeling_0.3           caTools_1.17.1.1       scales_1.0.0
## [31] checkmate_1.9.1        callr_3.1.1            digest_0.6.18
## [34] foreign_0.8-71         rmarkdown_1.11         XVector_0.22.0
## [37] base64enc_0.1-3        pkgconfig_2.0.2        htmltools_0.3.6
## [40] sessioninfo_1.1.1      htmlwidgets_1.3        rlang_0.3.1
## [43] rstudioapi_0.9.0       RSQLite_2.1.1          bindr_0.1.1
## [46] gtools_3.8.1           acepack_1.4.1          RCurl_1.95-4.11
## [49] magrittr_1.5           GenomeInfoDbData_1.2.0 Formula_1.2-3
## [52] Matrix_1.2-15          Rcpp_1.0.0             munsell_0.5.0
## [55] stringi_1.2.4          yaml_2.2.0             zlibbioc_1.28.0
## [58] pkgbuild_1.0.2         grid_3.5.2             blob_1.1.1
## [61] gdata_2.18.0           crayon_1.3.4           lattice_0.20-38
## [64] splines_3.5.2          annotate_1.60.0        hms_0.4.2
## [67] locfit_1.5-9.1         ps_1.3.0               knitr_1.21
```

```
## [70] pillar_1.3.1           pkgload_1.0.2          geneplotter_1.60.0
## [73] XML_3.98-1.16          glue_1.3.0             evaluate_0.12
## [76] latticeExtra_0.6-28    remotes_2.0.2          gtable_0.2.0
## [79] assertthat_0.2.0       xfun_0.4               xtable_1.8-3
## [82] survival_2.43-3        AnnotationDbi_1.44.0   memoise_1.1.0
## [85] cluster_2.0.7-1
```