Graphical Representation of Feature Mapping Using Different 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(gplots)
```

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

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

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

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

```
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 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:plyr':
##
##     compact
```

Purpose

To graphically compare the assignment of reads to features following mapping to either species-specific genomes or the human genome.

```r
feature_folder <- "FeatureCountsStats"
feature_stats <- basename(Sys.glob(file.path(feature_folder, "*.tabular")))

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

feature_stats_read <- llply(file.path(feature_folder, feature_stats), reader)
names(feature_stats_read) <- sub('.tabular', '', feature_stats)

##To get our bearings of what columns are what
colnames(feature_stats_read[[1]])
```

```
##  [1] "Sample"                         "Total"
##  [3] "Assigned"                       "Unassigned_Unmapped"
##  [5] "Unassigned_MappingQuality"     "Unassigned_Chimera"
##  [7] "Unassigned_FragmentLength"     "Unassigned_Duplicate"
##  [9] "Unassigned_MultiMapping"       "Unassigned_Secondary"
## [11] "Unassigned_Nonjunction"        "Unassigned_NoFeatures"
## [13] "Unassigned_Overlapping_Length" "Unassigned_Ambiguity"
## [15] "percent_assigned"
```

```r
##We want the sample name and the various mapping values for the feature counts expressed as
##percentages of the total number of reads
feature_stats_select <- llply(feature_stats_read, "[", c(1:5, 12))
feature_stats_mutate <- llply(feature_stats_select,
  function(x) dplyr::mutate(x, AssignedPercent = (x$Assigned/x$Total)*100,
              MappingQualityPercent = (x$Unassigned_MappingQuality/x$Total)*100,
              NoFeaturePercent = (x$Unassigned_NoFeatures/x$Total)*100)) %>%
  llply(., function(x) dplyr::select_at(x, vars("Sample", contains("Percent"))))

##Now we need to simplify the sample names of each row.
##Important note: The original sample names for gorilla, bonobo have a shortened donor ID --
##they say just "230" instead of "PR230". These were changed in the files after download from
##Galaxy to make the simplification of donor names easier. The squirrel monkey sample names were
##also adjusted -- the samples prefaced with "7_Barcode_Splitter_on_data_13_data_14_and_data_3_A"
```

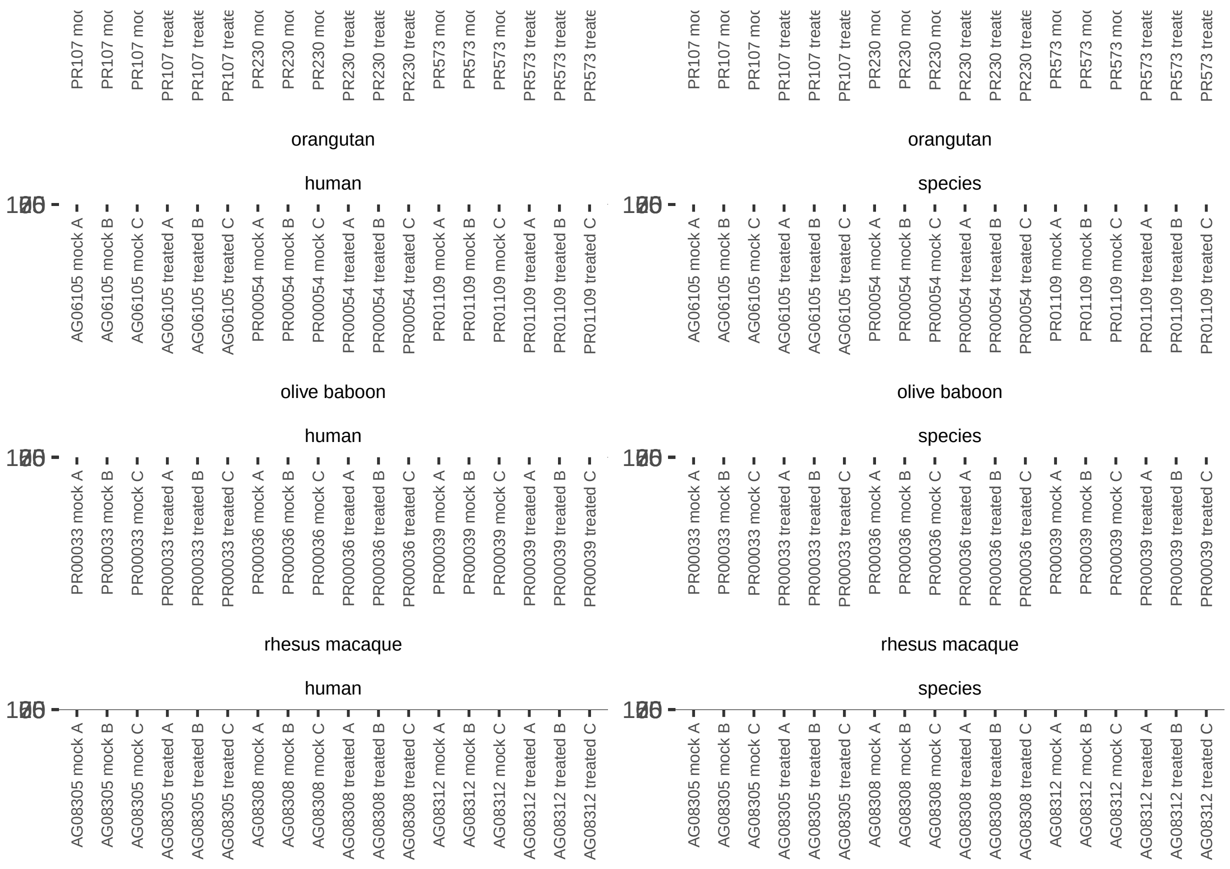
```
##were changed to read "data_3_SQMA" at the end; similarly
##"7_Barcode_Splitter_on_data_13_data_14_and_data_3_B" were changed to read "data_3_SQMB" at the
##end.The samples prefaced with ##"11_Barcode_Splitter_on_data_11_data_12_and_data_8_A_M" were
##changed to read "data_8_AG05311A_M" at the end.There was a typo in two lines of the pigtailed
##macaque where "PR00058" was missing the leading "P". The rhesus macaque also did not have clear
##"mock" and "treated" indications with the replicate letter, so these had to be manually fixed!
info_extract <- function(input) {
  donor <- str_extract(input$Sample, "PR\\d*|AG\\d*|S\\d{4,}|SQM\\w{1}|NHDF|AF|SR")
  treatment <- ifelse(grepl("mock|*M\\d|*M\\d\\d", input$Sample), "mock", "treated")
  replicate <- ifelse(grepl("_A_|24A|mockA|treatedA|M01|M1|T1|T01", input$Sample), "A",
                ifelse(grepl("_B_|24B|mockB|treatedB|M02|M2|T2|T02", input$Sample), "B", "C"))
  input$Sample <- paste(donor, treatment, replicate)
  input
  }

feature_stats_mutate_cleaner <- llply(feature_stats_mutate, info_extract) %>%
  melt()
```

```
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
## Using Sample as id variables
```

```
##Adding an additional column to designate what genome the reads were mapped to
feature_stats_mutate_cleaner$genome <- ifelse(grepl("_Human_Genome",
                      feature_stats_mutate_cleaner$L1), "human", "species")

##Making the L1 column more straightforward for using on our graph as a label
feature_stats_mutate_cleaner$L1 <- ifelse(grepl("AG07923|AG08490|PR0058",
                        feature_stats_mutate_cleaner$Sample), "pigtailed macaque",
 ifelse(grepl("PR00033|PR00036|PR00039", feature_stats_mutate_cleaner$Sample), "olive baboon",
 ifelse(grepl("AG05311|SQMA|SQMB", feature_stats_mutate_cleaner$Sample), "squirrel monkey",
  ifelse(grepl("AG06105|PR00054|PR01109", feature_stats_mutate_cleaner$Sample), "orangutan",
  ifelse(grepl("PR230|PR573|PR107", feature_stats_mutate_cleaner$Sample), "gorilla",
  ifelse(grepl("PR111|PR235|PR248", feature_stats_mutate_cleaner$Sample), "bonobo",
 ifelse(grepl("S4933|S3611|S3649", feature_stats_mutate_cleaner$Sample), "chimpanzee",
 ifelse(grepl("AG08308|AG08312|AG08305", feature_stats_mutate_cleaner$Sample), "rhesus macaque",
  ifelse(grepl("NHDF|AF|SR", feature_stats_mutate_cleaner$Sample), "human", "nothing")))))))))

##Double checking there are not any NAs
anyNA(feature_stats_mutate_cleaner)
```

```
## [1] FALSE
```

```
##Setting the levels in the desired order for the variable and L1 columns
feature_stats_mutate_cleaner$variable <- factor(feature_stats_mutate_cleaner$variable,
            levels = c("AssignedPercent", "NoFeaturePercent", "MappingQualityPercent"))
write.csv(feature_stats_mutate_cleaner, paste(Sys.Date(), "FeatureCountStats_data.csv"))
feature_stats_mutate_cleaner$L1 <- factor(feature_stats_mutate_cleaner$L1,
  levels = c("human", "chimpanzee", "bonobo", "gorilla", "orangutan", "olive baboon",
             "rhesus macaque", "pigtailed macaque", "squirrel monkey"))

##Plotting it all out
plot <- ggplot() +
geom_bar(data = filter(feature_stats_mutate_cleaner, L1 != "human"),
         mapping = aes(x = Sample, y = value, fill = variable),
         stat= "identity", position = "stack", colour = "black") +
  facet_wrap(~L1 + genome, scales = "free", ncol = 2) +
  scale_fill_manual(values = c("#d7191c", "#fdae61", "#2b83ba")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 6),
        panel.grid.major = element_line("black"),
        panel.grid.minor = element_line("black"), axis.title.y = element_blank(),
        axis.title.x = element_blank(),
        legend.title = element_blank(), legend.position = 'bottom') +
  theme(strip.background =element_rect(fill="white")) +
  theme(strip.text = element_text(colour = "black", size = 7))
ggsave(file = paste(Sys.Date(), "FeatureCountStats.pdf"), plot = plot,
       height = 20, width = 7, device = "pdf")
print(plot)
```



```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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2     purrr_0.2.5        data.table_1.12.0  reshape2_1.4.3
##  [5] ggrepel_0.8.0      usethis_1.4.0      devtools_2.0.1     viridis_0.5.1
##  [9] viridisLite_0.3.0  RColorBrewer_1.1-2 ggplot2_3.1.0      gplots_3.0.1
## [13] stringr_1.3.1      tibble_2.0.1       dplyr_0.7.8        plyr_1.8.4
##
## loaded via a namespace (and not attached):
##  [1] gtools_3.8.1       tidyselect_0.2.5   xfun_0.4           remotes_2.0.2
##  [5] colorspace_1.4-0   htmltools_0.3.6    yaml_2.2.0         rlang_0.3.1
##  [9] pkgbuild_1.0.2     pillar_1.3.1       glue_1.3.0         withr_2.1.2
## [13] sessioninfo_1.1.1  bindr_0.1.1        munsell_0.5.0      gtable_0.2.0
## [17] caTools_1.17.1.1   evaluate_0.12      memoise_1.1.0      labeling_0.3
## [21] knitr_1.21         callr_3.1.1        ps_1.3.0           Rcpp_1.0.0
## [25] KernSmooth_2.23-15 scales_1.0.0       backports_1.1.3    gdata_2.18.0
## [29] desc_1.2.0         pkgload_1.0.2      fs_1.2.6           gridExtra_2.3
## [33] digest_0.6.18      stringi_1.2.4      processx_3.2.1     grid_3.5.2
## [37] rprojroot_1.3-2    cli_1.0.1          tools_3.5.2        bitops_1.0-6
## [41] magrittr_1.5       lazyeval_0.2.1     crayon_1.3.4       pkgconfig_2.0.2
## [45] prettyunits_1.0.2  assertthat_0.2.0   rmarkdown_1.11     R6_2.3.0
## [49] compiler_3.5.2
```