Workflow Comparison

Load Libraries

```
library(ShortRead)
```

```
## 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:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: BiocParallel

## Loading required package: Biostrings

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'
```

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

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit

## Loading required package: Rsamtools

## Loading required package: GenomeInfoDb

## Loading required package: GenomicRanges

## Loading required package: GenomicAlignments

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
```

```
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## 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")'.


##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

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

```
library(qckitfastq)
library(ggplot2)
library(ggpubr)
library(dplyr)
```

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

## The following object is masked from 'package:ShortRead':
##
##     id

## The following objects are masked from 'package:GenomicAlignments':
##
##     first, last

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

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

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

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union

## The following object is masked from 'package:XVector':
##
##     slice

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

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

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

```
library(ggthemes)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.6     v purrr   0.3.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse()   masks Biostrings::collapse(), IRanges::collapse()
## x dplyr::combine()    masks Biobase::combine(), BiocGenerics::combine()
## x purrr::compact()    masks XVector::compact()
## x purrr::compose()    masks ShortRead::compose()
## x dplyr::count()      masks matrixStats::count()
```

```
## x dplyr::desc()       masks IRanges::desc()
## x tidyr::expand()     masks S4Vectors::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks GenomicAlignments::first(), S4Vectors::first()
## x dplyr::id()         masks ShortRead::id()
## x dplyr::lag()        masks stats::lag()
## x dplyr::last()       masks GenomicAlignments::last()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()     masks S4Vectors::rename()
## x dplyr::slice()      masks XVector::slice(), IRanges::slice()
## x tibble::view()      masks ShortRead::view()
```

Load Data

Set paths

```
graph_path=("C:/Users/voro/OneDrive/9_Studium/MasterThesis/R_scripts/Graphs")
EPI2ME_QC_D4_Path=("EPI2ME_Data_Reports/basecalling_1d_barcode-v1_D4.csv")
EPI2ME_QC_D5_Path=("EPI2ME_Data_Reports/basecalling_1d_barcode-v1_D5.csv")
EPI2ME_Tax_D4_Path=("EPI2ME_Data_Reports/classification_16s_barcode-v1_D4.csv")
EPI2ME_Tax_D5_Path=("EPI2ME_Data_Reports/classification_16s_barcode-v1_D5.csv")
Demultiplexing_qcat_D4_Path=("Workflow_Data_Reports/D4/demultiplexing_qcat_tax_wf_comp_01/")
Demultiplexing_qcat_D4_List=list.files(Demultiplexing_qcat_D4_Path,pattern = ".fastq")
Demultiplexing_qcat_D5_Path=("Workflow_Data_Reports/D5/demultiplexing_qcat_tax_wf_comp_01/")
Demultiplexing_qcat_D5_List=list.files(Demultiplexing_qcat_D5_Path,pattern = ".fastq")
Filter_Nanofilt_D4_Path=("Workflow_Data_Reports/D4/filter_trim_nanofilt_tax_wf_comp_01/")
Filter_Nanofilt_D4_List=list.files(Filter_Nanofilt_D4_Path,pattern = ".fastq")
Filter_Nanofilt_D5_Path=("Workflow_Data_Reports/D5/filter_trim_nanofilt_tax_wf_comp_01/")
Filter_Nanofilt_D5_List=list.files(Filter_Nanofilt_D5_Path,pattern = ".fastq")
Kraken2_Classification_D4_Path=("Workflow_Data_Reports/D4/classification_kraken2_nanofilt_tax_wf_comp_0
Kraken2_Classification_D5_Path=("Workflow_Data_Reports/D5/classification_kraken2_nanofilt_tax_wf_comp_0
Kraken2_Report_D4_Path=("Workflow_Data_Reports/D4/r_kraken2_report_nanofilt_tax_wf_comp_01/")
Kraken2_Report_D5_Path=("Workflow_Data_Reports/D5/r_kraken2_report_nanofilt_tax_wf_comp_01/")

graph_path=("Graphs/")
```

Load EPI2ME files

```
library(readr)
EPI2ME_QC_D4 <- read_csv(EPI2ME_QC_D4_Path)
EPI2ME_QC_D5 <- read_csv(EPI2ME_QC_D5_Path)
EPI2ME_Tax_D4 <- read_csv(EPI2ME_Tax_D4_Path)
EPI2ME_Tax_D5 <- read_csv(EPI2ME_Tax_D5_Path)
```

Comparison EPI2ME Fastq 16S vs. Kraken2 Workflow

Get total read amounts

Use of the ShortRead library (works) The total amount of reads for D4 before and after filtering with POND Workflow

```
read_count_df=countFastq(Demultiplexing_qcat_D4_Path)
sum(read_count_df$records)
```

```
## [1] 185241
```

```
read_count_df=countFastq(Filter_Nanofilt_D4_Path)
sum(read_count_df$records)
```

```
## [1] 110444
```

The total amount of reads for D5 before and after filtering with POND Workflow

```
read_count_df=countFastq(Demultiplexing_qcat_D5_Path)
sum(read_count_df$records)
```

```
## [1] 398466
```

```
read_count_df=countFastq(Filter_Nanofilt_D5_Path)
sum(read_count_df$records)
```

```
## [1] 191390
```

Plot Read quality

```
# NanoR need to have summary file and others

# Glist<-NanoPrepareG(DataFastq=Demultiplexing_qcat_D4_Path)
# Passed FASTQ: 25
# Error in file.path(DataSummary) :
#   argument "DataSummary" is missing, with no default
```

NanoR (not working)

```
# MinIONQC also needs summary file
```

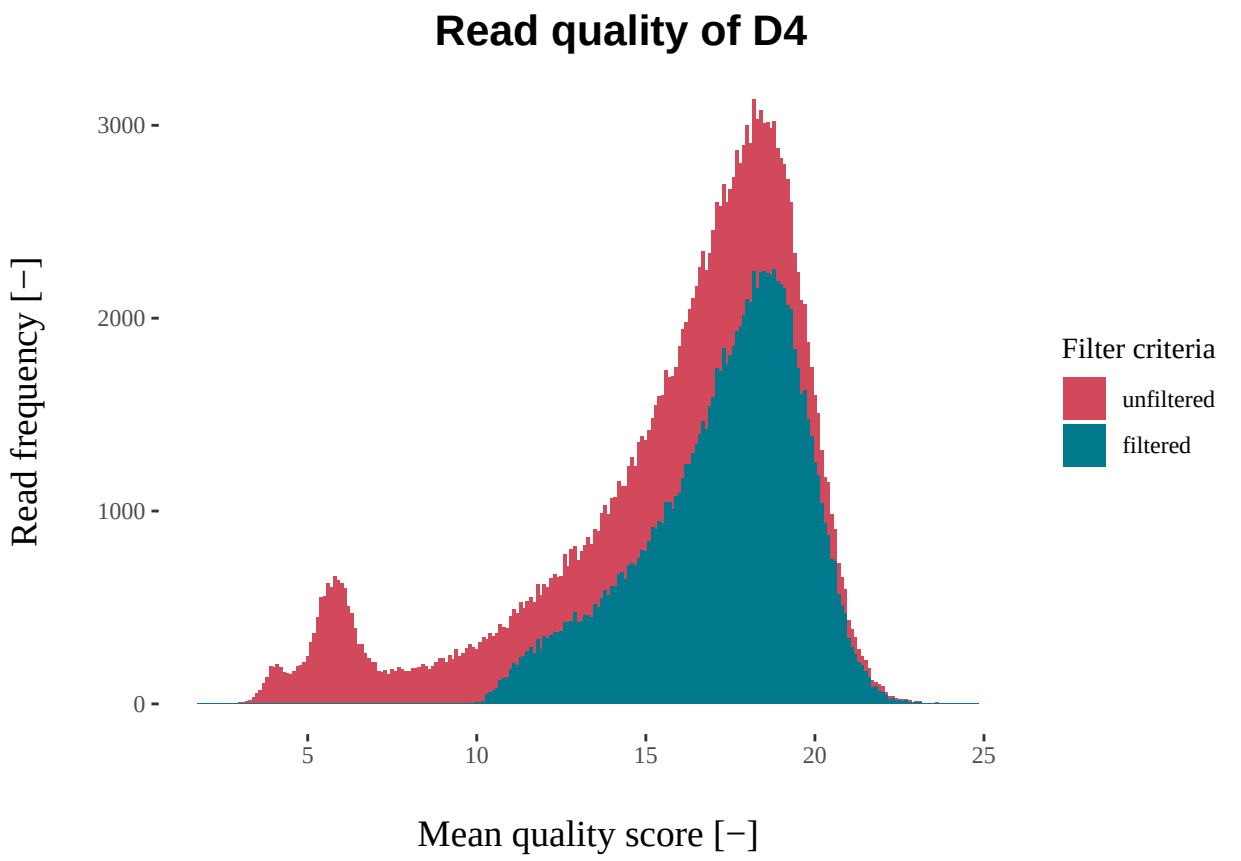
MinIONQC (not working)

qckitfastq (works) Plot the read quality of the D4 before and after filtering by length and quality

```
# First the bifile is create with the bash command("cats *.fastq > bigfile.fastq)
# Load the aggregated fastq files (bigfile)
curr_fastq_path=paste(Demultiplexing_qcat_D4_Path,"bigfile/bigfile.fastq",sep = "")
quality_dataframe_1 <- per_read_quality(curr_fastq_path)
curr_fastq_path=paste(Filter_Nanofilt_D4_Path,"/bigfile/bigfile.fastq",sep = "")
quality_dataframe_2 <- per_read_quality(curr_fastq_path)

# Put the filtered and unfiltered dataframes into one
quality_dataframe_1$read="unfiltered"
quality_dataframe_2$read="filtered"
prq=rbind(quality_dataframe_1,quality_dataframe_2)
prq$read=as.factor(prq$read)
colnames(prq)=c("filter_criteria","sequence_mean")

# Define names for the plot
title="Read quality of D4"
xlabel="Mean quality score [-]"
ylabel="Read frequency [-]"
legendlabel="Filter criteria"
filename="Section_5.2_ Read_Quality_D4.jpg"

# Define the data and order of the color filling
q=ggplot(prq,aes(x=sequence_mean,  fill = reorder(filter_criteria, desc(filter_criteria))))+
  # Define it to be a histogram plot with specific binwidth
  geom_histogram(position="identity",binwidth=0.1)+
  # Set a theme
  theme_tufte() +
  # Define formatting and load defined texts
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
# Set a color scheme
q=q+scale_fill_manual(values=c("#d1495b","#00798c"))
# Annotate the title
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
# Save and print the plot
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```



Plot the read quality of the D5 before and after filtering by length and quality

```
# First the bifile is create with the bash command("cats *.fastq > bigfile.fastq)
# Load the aggregated fastq files (bigfile)
curr_fastq_path=paste(Demultiplexing_qcat_D5_Path,"bigfile/bigfile.fastq",sep = "")
quality_dataframe_1 <- per_read_quality(curr_fastq_path)
curr_fastq_path=paste(Filter_Nanofilt_D5_Path,"/bigfile/bigfile.fastq",sep = "")
quality_dataframe_2 <- per_read_quality(curr_fastq_path)

# Put the filtered and unfiltered dataframes into one
quality_dataframe_1$read="unfiltered"
quality_dataframe_2$read="filtered"
prq=rbind(quality_dataframe_1,quality_dataframe_2)
prq$read=as.factor(prq$read)
colnames(prq)=c("filter_criteria","sequence_mean")

# Define names for the plot
title="Read quality of D5"
xlabel="Mean quality score [-]"
ylabel="Read frequency [-]"
legendlabel="Filter criteria"
filename="Section_5.2_ Read_Quality_D5.jpg"

# Define the data and order of the color filling
q=ggplot(prq,aes(x=sequence_mean,  fill = reorder(filter_criteria, desc(filter_criteria))))+
  # Define it to be a histogram plot with specific binwidth
```

```
  geom_histogram(position="identity",binwidth=0.1)+
  # Set a theme
  theme_tufte() +
  # Define formatting and load defined texts
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),

        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
# Set a color scheme
q=q+scale_fill_manual(values=c("#d1495b","#00798c"))
# Annotate the title
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
# Save and print the plot
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```

Plot Read length distribution Read Length Distribution for D4

```
# for loading single fastq file it is not working
# fseq <- seqTools::fastqq(fastq_path)

# Create a dataframe to which the read length counts can be added
store_dataframe_1=data.frame( matrix(0, ncol = 2, nrow = 10000))
# Read lengths up 10000 can be stored
colnames(store_dataframe_1)=c("read_length","num_reads")
store_dataframe_1$read_length=seq(1,10000)

for (i in Demultiplexing_qcat_D4_List){
  fastq_path=paste(Demultiplexing_qcat_D4_Path,i,sep = "")
  # print(fastq_path)
  sink("nul")
  fseq <- seqTools::fastqq(fastq_path)
  sink()
  read_len <- read_length(fseq)
  # print(sum(read_len$num_reads))

  zero=numeric(abs(nrow(store_dataframe_1)-nrow(read_len)))
  col_add=c(read_len$num_reads,zero)
  store_dataframe_1$num_reads=store_dataframe_1$num_read+col_add
}
# Create a dataframe to which the read length counts can be added
store_dataframe_2=data.frame( matrix(0, ncol = 2, nrow = 10000))
# Read lengths up 10000 can be stored
colnames(store_dataframe_2)=c("read_length","num_reads")
store_dataframe_2$read_length=seq(1,10000)

for (i in Filter_Nanofilt_D4_List){
  fastq_path=paste(Filter_Nanofilt_D4_Path,i,sep = "")
  # print(fastq_path)
  sink("nul")
  fseq <- seqTools::fastqq(fastq_path)
  sink()
  read_len <- read_length(fseq)
  # print(sum(read_len$num_reads))

  zero=numeric(abs(nrow(store_dataframe_2)-nrow(read_len)))
  col_add=c(read_len$num_reads,zero)
  store_dataframe_2$num_reads=store_dataframe_2$num_read+col_add
}


# Not all of the reads were loaded, print
read_count_df=countFastq(Demultiplexing_qcat_D4_Path)
sum(read_count_df$records)
```

```
## [1] 185241
```

```
sum(store_dataframe_1$num_reads)
```

```
## [1] 147453
```

```
read_count_df=countFastq(Filter_Nanofilt_D4_Path)
sum(read_count_df$records)
```

```
## [1] 110444
```

```
sum(store_dataframe_2$num_reads)
```

```
## [1] 110444
```

```
# Create filter criteria and store them in one dataframe
store_dataframe_1$filter_criteria="unfiltered"
store_dataframe_2$filter_criteria="filtered"
store_dataframe=rbind(store_dataframe_1,store_dataframe_2)

title="Readlength_Distribution of D4"
xlabel="Read length [bp]"
ylabel="Read frequency [-]"
legendlabel="Filter criteria"
filename="Section_5.2_Read_Length_Distribution_D4.jpg"

q=ggplot(store_dataframe, aes(y=num_reads, x=read_length,  fill = reorder(filter_criteria, desc(filter_
  geom_bar(stat = 'identity')+
  xlim(0, 2500)+
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
q=q+scale_fill_manual(values=c("#d1495b","#00798c"))
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
```

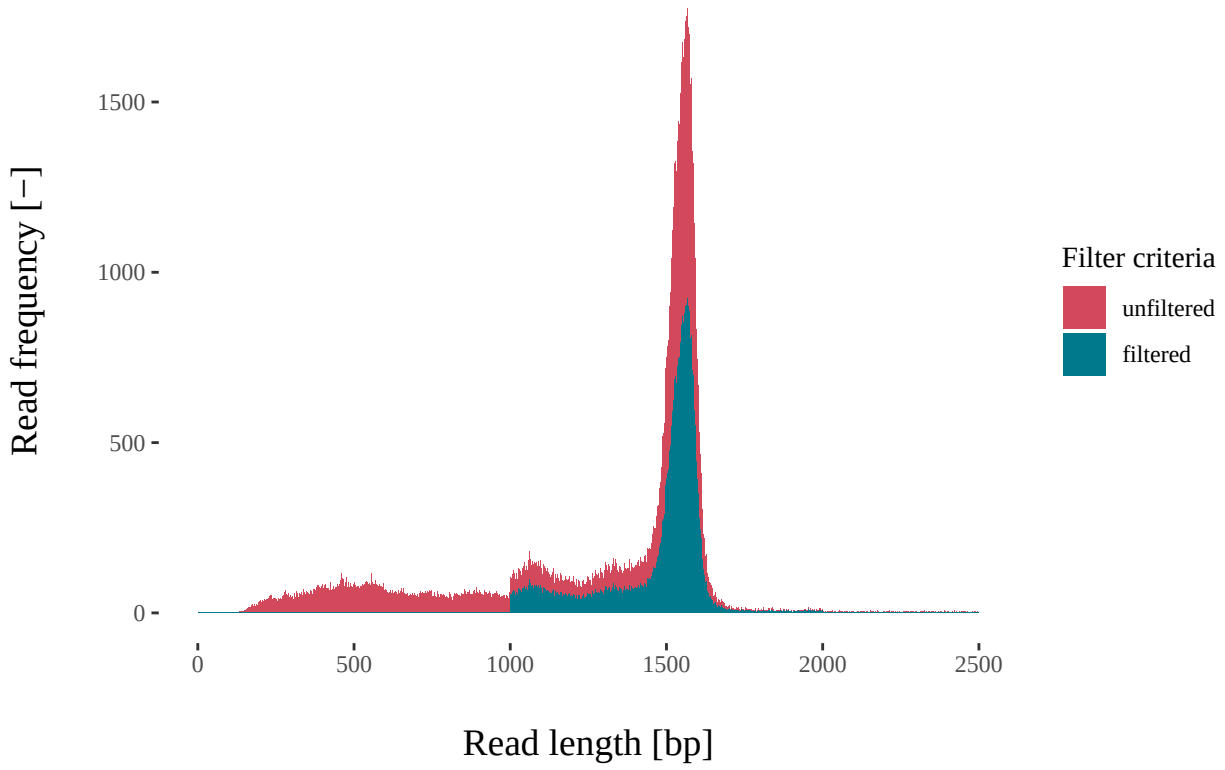
```
## Warning: Removed 15000 rows containing missing values (position_stack).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```



Read Length Distribution for D5

```
# for loading single fastq file it is not working
# fseq <- seqTools::fastqq(fastq_path)

# Create a dataframe to which the read length counts can be added
store_dataframe_1=data.frame( matrix(0, ncol = 2, nrow = 10000))
# Read lengths up 10000 can be stored
colnames(store_dataframe_1)=c("read_length","num_reads")
store_dataframe_1$read_length=seq(1,10000)

for (i in Demultiplexing_qcat_D5_List){
  fastq_path=paste(Demultiplexing_qcat_D4_Path,i,sep = "")
  # print(fastq_path)
  sink("nul")
  fseq <- seqTools::fastqq(fastq_path)
  sink()
  read_len <- read_length(fseq)
  # print(sum(read_len$num_reads))

  zero=numeric(abs(nrow(store_dataframe_1)-nrow(read_len)))
  col_add=c(read_len$num_reads,zero)
  store_dataframe_1$num_reads=store_dataframe_1$num_read+col_add
}
# Create a dataframe to which the read length counts can be added
store_dataframe_2=data.frame( matrix(0, ncol = 2, nrow = 10000))
# Read lengths up 10000 can be stored
```

```
colnames(store_dataframe_2)=c("read_length","num_reads")
store_dataframe_2$read_length=seq(1,10000)

for (i in Filter_Nanofilt_D5_List){
  fastq_path=paste(Filter_Nanofilt_D5_Path,i,sep = "")
  # print(fastq_path)
  sink("nul")
  fseq <- seqTools::fastqq(fastq_path)
  sink()
  read_len <- read_length(fseq)
  # print(sum(read_len$num_reads))

  zero=numeric(abs(nrow(store_dataframe_2)-nrow(read_len)))
  col_add=c(read_len$num_reads,zero)
  store_dataframe_2$num_reads=store_dataframe_2$num_read+col_add
}


# Not all of the reads were loaded, print
read_count_df=countFastq(Demultiplexing_qcat_D5_Path)
sum(read_count_df$records)
```

```
## [1] 398466
```

```
sum(store_dataframe_1$num_reads)
```

```
## [1] 147453
```

```
read_count_df=countFastq(Filter_Nanofilt_D5_Path)
sum(read_count_df$records)
```

```
## [1] 191390
```

```
sum(store_dataframe_2$num_reads)
```

```
## [1] 191390
```

```
# Create filter criteria and store them in one dataframe
store_dataframe_1$filter_criteria="unfiltered"
store_dataframe_2$filter_criteria="filtered"
store_dataframe=rbind(store_dataframe_1,store_dataframe_2)

title="Readlength_Distribution of D5"
xlabel="Read length [bp]"
ylabel="Read frequency [-]"
legendlabel="Filter criteria"
filename="Section_5.2_Read_Length_Distribution_D5.jpg"

q=ggplot(store_dataframe, aes(y=num_reads, x=read_length,  fill = reorder(filter_criteria, desc(filter_
  geom_bar(stat = 'identity')+
```

```
  xlim(0, 2500)+
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
q=q+scale_fill_manual(values=c("#d1495b","#00798c"))
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
```

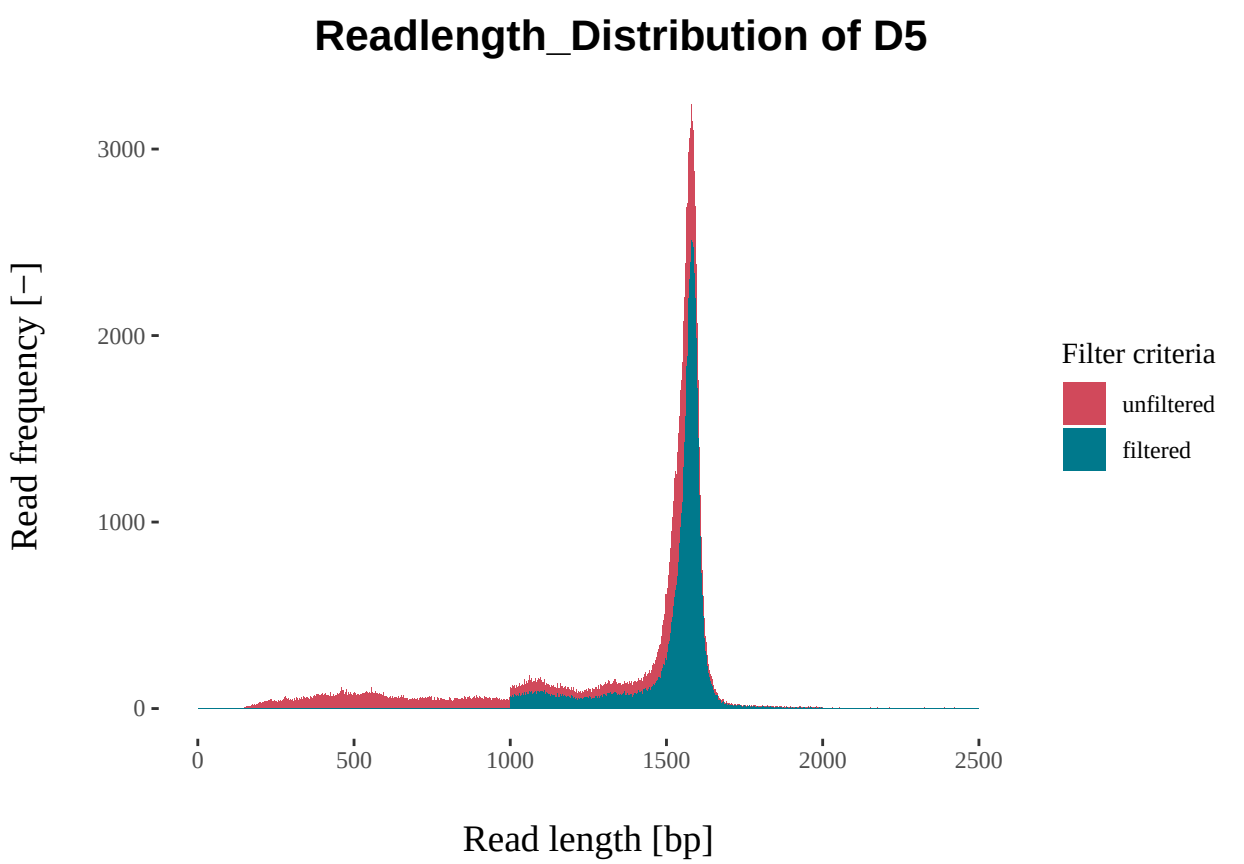
```
## Warning: Removed 15000 rows containing missing values (position_stack).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```

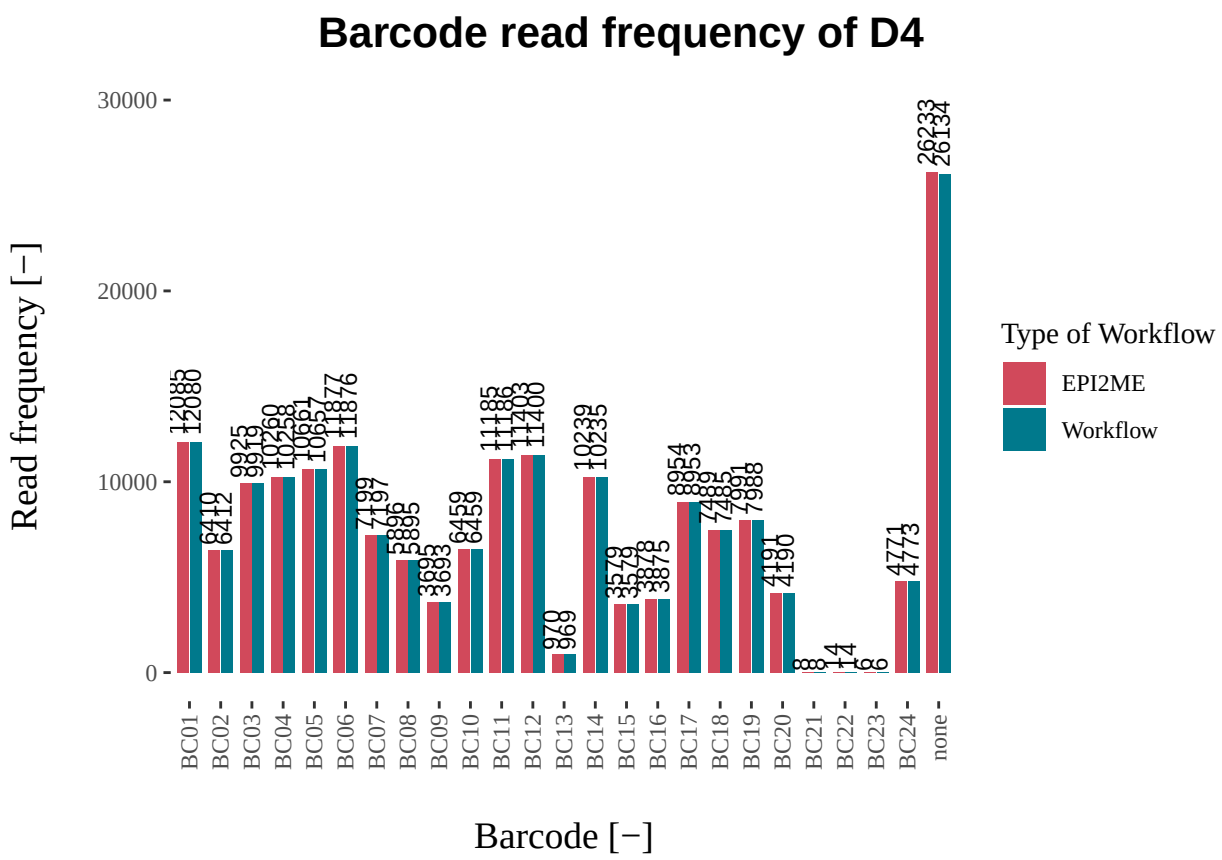


Compare Barcode Read Counts Compare Read Counts of EPI2ME vs Workflow for D4

```
# EPI2ME Barcode Counts
EPI2ME_QC_D4$barcode[is.na(EPI2ME_QC_D4$barcode)] <- "none"
EPI2ME_QC_D4_counts=as.data.frame(table(EPI2ME_QC_D4$barcode))
EPI2ME_QC_D4_counts$workflow="EPI2ME"
colnames(EPI2ME_QC_D4_counts)=c("barcode","counts","workflow")


# Workflow Barcode Counts
Workflow_QC_D4_counts=as.data.frame(unique(Demultiplexing_qcat_D4_List))
colnames(Workflow_QC_D4_counts)="barcode"
Workflow_QC_D4_counts$counts=0
for (i in Workflow_QC_D4_counts$barcode){
  fastq_path=paste(Demultiplexing_qcat_D4_Path,i,sep = "")
  current_count=countFastq(fastq_path)
  # print(current_count$records)
  Workflow_QC_D4_counts$counts[which(Workflow_QC_D4_counts$barcode==i)]=current_count$records
}
Workflow_QC_D4_counts$workflow="Workflow"
Workflow_QC_D4_counts$barcode=str_remove(Workflow_QC_D4_counts$barcode, ".fastq")

#Create one dataframe
barcode_comp=rbind(EPI2ME_QC_D4_counts,Workflow_QC_D4_counts)

title="Barcode read frequency of D4"
xlabel="Barcode [-]"
ylabel="Read frequency [-]"
legendlabel="Type of Workflow"
filename="Section_5.2_Barcode_Read_Frequency_D4.jpg"

q=ggplot(data = barcode_comp, aes(x = barcode, y = counts, fill = workflow),) +
geom_bar(position = position_dodge(width = 0.8), stat = "identity", width = 0.75)+
   geom_text(aes(label=counts), position = position_dodge(width = 1),size=3,hjust=-0.1,vjust=0.2,angle=
  ylim(0, 30000)+
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
q=q + scale_fill_manual(values=c("#d1495b","#00798c"))
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
ggsave(plot = q, width = 10, height = 5, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```



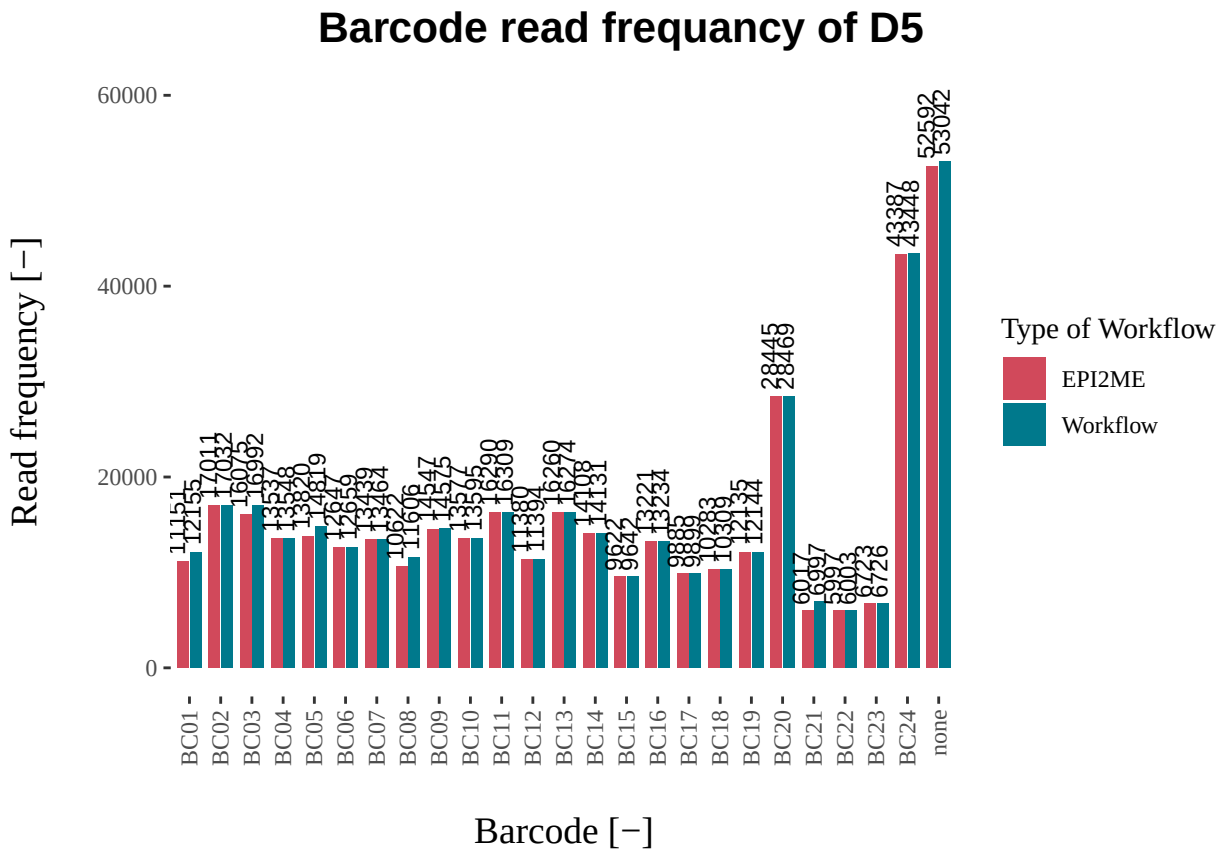
Compare Read Counts of EPI2ME vs Workflow for D5

```
# EPI2ME Barcode Counts
EPI2ME_QC_D5$barcode[is.na(EPI2ME_QC_D5$barcode)] <- "none"
EPI2ME_QC_D5_counts=as.data.frame(table(EPI2ME_QC_D5$barcode))
EPI2ME_QC_D5_counts$workflow="EPI2ME"
colnames(EPI2ME_QC_D5_counts)=c("barcode","counts","workflow")


# Workflow Barcode Counts
Workflow_QC_D5_counts=as.data.frame(unique(Demultiplexing_qcat_D5_List))
colnames(Workflow_QC_D5_counts)="barcode"
Workflow_QC_D5_counts$counts=0
for (i in Workflow_QC_D5_counts$barcode){
  fastq_path=paste(Demultiplexing_qcat_D5_Path,i,sep = "")
  current_count=countFastq(fastq_path)
  # print(current_count$records)
  Workflow_QC_D5_counts$counts[which(Workflow_QC_D5_counts$barcode==i)]=current_count$records
}
Workflow_QC_D5_counts$workflow="Workflow"
Workflow_QC_D5_counts$barcode=str_remove(Workflow_QC_D5_counts$barcode, ".fastq")

#Create one dataframe
barcode_comp=rbind(EPI2ME_QC_D5_counts,Workflow_QC_D5_counts)

title="Barcode read frequancy of D5"
xlabel="Barcode [-]"
```

```
ylabel="Read frequency [-]"
legendlabel="Type of Workflow"
filename="Section_5.2_Barcode_Read_Frequency_D5.jpg"

q=ggplot(data = barcode_comp, aes(x = barcode, y = counts, fill = workflow),) +
geom_bar(position = position_dodge(width = 0.8), stat = "identity", width = 0.75)+
   geom_text(aes(label=counts), position = position_dodge(width = 1),size=3,hjust=-0.1,vjust=0.2,angle=9
  ylim(0, 60000)+
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
q=q + scale_fill_manual(values=c("#d1495b","#00798c"))
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
ggsave(plot = q, width = 10, height = 5, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```



Classification Comparison The read classification success on the rank of species is plotted of EPI2ME against Workflow

```r
# Load the kraken2 report files for D4
filenames <- list.files(path=Kraken2_Report_D4_Path,pattern = "REPORT")
# Store the filenames in shortened form (BCXX)
T0_IDs <- substr(filenames,1,4)
# Create empty list
T0_df_list=list()
for (i in 1:length(filenames)){
  # Get the sample name of the barcode
  # sample_name=as.character(sample_id$sample_id[which(sample_id$barcode_t_0==substr(filenames[i],1,4))]
  sample_name=substr(filenames[i],1,4)
  # Check that the sample_name exitst
  if (length(sample_name)!=0){
    # Create the load path
    load_path=paste(Kraken2_Report_D4_Path,filenames[i], sep = "")
    # Store the df in the list
    df=read.csv(load_path, header=TRUE)
    # df=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE,trim_ws = TRUE)
    # Add the df to the list with the corresponding sample name
    T0_df_list[[sample_name]]<- df
    # Print whats going on
    print(paste(substr(filenames[i],1,4),sample_name,sep = " --> "))
  }
}
```

```
## [1] "BC01 --> BC01"
## [1] "BC02 --> BC02"
## [1] "BC03 --> BC03"
## [1] "BC04 --> BC04"
## [1] "BC05 --> BC05"
## [1] "BC06 --> BC06"
## [1] "BC07 --> BC07"
## [1] "BC08 --> BC08"
## [1] "BC09 --> BC09"
## [1] "BC10 --> BC10"
## [1] "BC11 --> BC11"
## [1] "BC12 --> BC12"
## [1] "BC13 --> BC13"
## [1] "BC14 --> BC14"
## [1] "BC15 --> BC15"
## [1] "BC16 --> BC16"
## [1] "BC17 --> BC17"
## [1] "BC18 --> BC18"
## [1] "BC19 --> BC19"
## [1] "BC20 --> BC20"
## [1] "BC21 --> BC21"
## [1] "BC22 --> BC22"
## [1] "BC23 --> BC23"
## [1] "BC24 --> BC24"
## [1] "none --> none"
```

```r
# Put all of the barcode dataframes form the list into one dataframe
D4_Total_Kraken2_Report <- bind_rows(T0_df_list, .id = "column_label")
# Adapt colnames
```

```r
colnames(D4_Total_Kraken2_Report)[1]<-"barcode"
colnames(D4_Total_Kraken2_Report)[2]<-"level"
#_______________________________________________________________________________

# Load the kraken2 report files for D5
filenames <- list.files(path=Kraken2_Report_D5_Path,pattern = "REPORT")
# Store the filenames in shortened form (BCXX)
T1_IDs <- substr(filenames,1,4)
# Create empty list
T1_df_list=list()
for (i in 1:length(filenames)){
  # Get the sample name of the barcode
  # sample_name=as.character(sample_id$sample_id[which(sample_id$barcode_t_0==substr(filenames[i],1,4))]
  sample_name=substr(filenames[i],1,4)
  # Check that the sample_name exitst
  if (length(sample_name)!=0){
    # Create the load path
    load_path=paste(Kraken2_Report_D5_Path,filenames[i], sep = "")
    # Store the df in the list
    df=read.csv(load_path, header=TRUE)
    # df=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE,trim_ws = TRUE)
    # Add the df to the list with the corresponding sample name
    T1_df_list[[sample_name]]<- df
    # Print whats going on
    print(paste(substr(filenames[i],1,4),sample_name,sep = " --> "))
  }
}
```

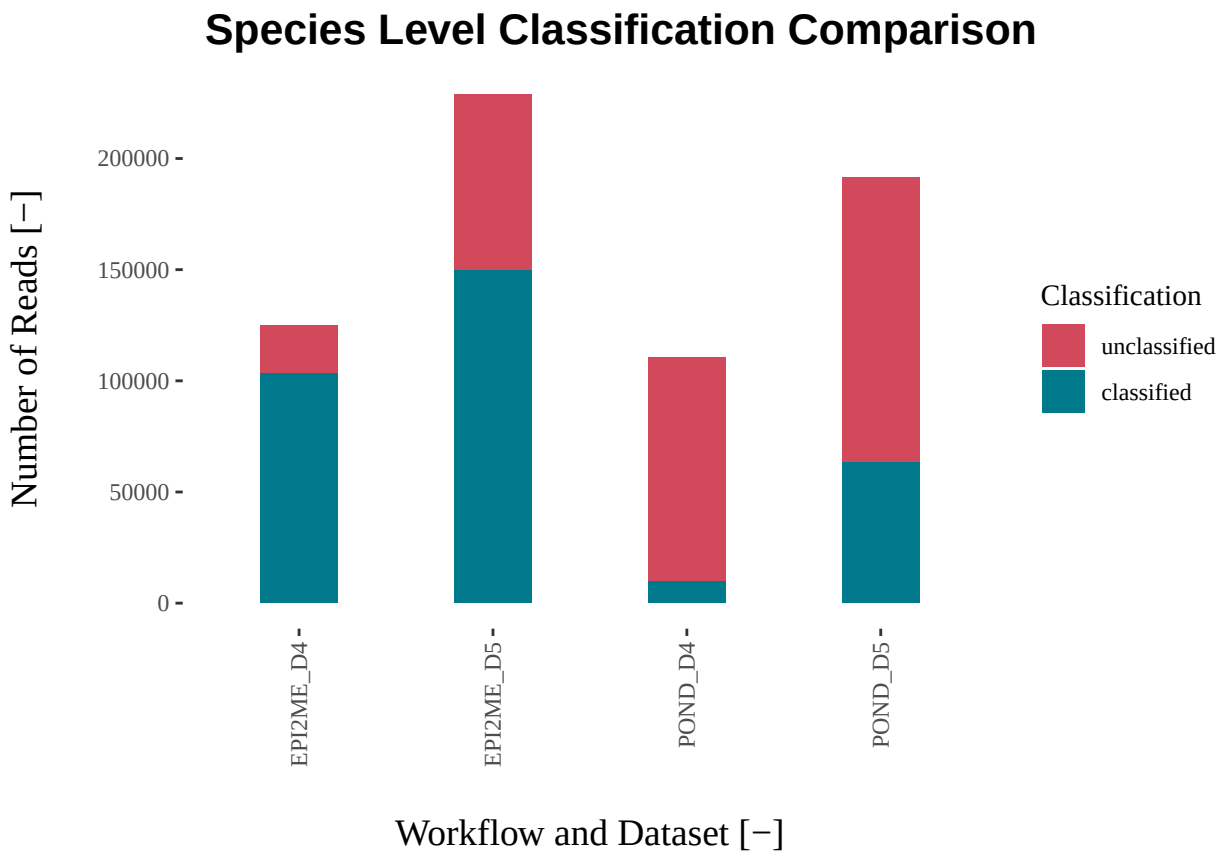
```
## [1] "BC01 --> BC01"
## [1] "BC02 --> BC02"
## [1] "BC03 --> BC03"
## [1] "BC04 --> BC04"
## [1] "BC05 --> BC05"
## [1] "BC06 --> BC06"
## [1] "BC07 --> BC07"
## [1] "BC08 --> BC08"
## [1] "BC09 --> BC09"
## [1] "BC10 --> BC10"
## [1] "BC11 --> BC11"
## [1] "BC12 --> BC12"
## [1] "BC13 --> BC13"
## [1] "BC14 --> BC14"
## [1] "BC15 --> BC15"
## [1] "BC16 --> BC16"
## [1] "BC17 --> BC17"
## [1] "BC18 --> BC18"
## [1] "BC19 --> BC19"
## [1] "BC20 --> BC20"
## [1] "BC21 --> BC21"
## [1] "BC22 --> BC22"
## [1] "BC23 --> BC23"
## [1] "BC24 --> BC24"
## [1] "none --> none"
```

```
# Put all of the barcode dataframes form the list into one dataframe
D5_Total_Kraken2_Report <- bind_rows(T1_df_list, .id = "column_label")
# Adapt colnames
colnames(D5_Total_Kraken2_Report)[1]<-"barcode"
colnames(D5_Total_Kraken2_Report)[2]<-"level"
#_____________________________________________________________________________

# EPI2ME Data D4
# Exchange NA with none
EPI2ME_Tax_D4$barcode[is.na(EPI2ME_Tax_D4$barcode)] <- "none"
# Subset the dataframe
df1=as.data.frame(table(EPI2ME_Tax_D4$exit_status,EPI2ME_Tax_D4$barcode))
# Remove the "Error" parameter
df1=df1[df1$Var1!="Error",]
# Add Column with the name of the workflow and the Dataset
df1$workflow="EPI2ME_D4"
# Rename Columns
colnames(df1)=c("classification","barcode","readnum","workflow_type")
# Change column type
df1$classification=as.character(df1$classification)
# Harmonize dataset
df1$classification[which(df1$classification=="Classification successful")]<-"classified"
# Harmonize dataset
df1$classification[which(df1$classification=="Unclassified")]<-"unclassified"
# Change column type
df1$barcode=as.character(df1$barcode)
#_____________________________________________________________________________

# POND D4 Data
df2=data.frame()
for (i in unique(D4_Total_Kraken2_Report$barcode)){
  # Get the total read number for the current barcode
  total_read_num=D4_Total_Kraken2_Report$total_read_number[which(D4_Total_Kraken2_Report$barcode==i)][1]
  # Get the percentage amount of classifed reads at the species level
  species_rank_perc=D4_Total_Kraken2_Report$s_cov_perc[which(D4_Total_Kraken2_Report$barcode==i)][1]
  # Calculat the number of classified reads
  num_classified_reads=total_read_num*species_rank_perc/100
  # Calculat the number of unclassified reads
  num_unclassified_reads=total_read_num-total_read_num*species_rank_perc/100
  # Define column and store them in new dataframe
  c1=c("classified",i,num_classified_reads)
  c2=c("unclassified",i, num_unclassified_reads)
  df2=rbind(df2,c1)
  df2=rbind(df2,c2)
}
# Add Column with the name of the workflow and the Dataset
df2$workflow="POND_D4"
# Rename Columns
colnames(df2)=c("classification","barcode","readnum","workflow_type")
# Harmonize data type
df2$readnum=round(as.numeric(df2$readnum))

#_____________________________________________________________________________
```

```
# EPI2ME Data D5
EPI2ME_Tax_D5$barcode[is.na(EPI2ME_Tax_D5$barcode)] <- "none"
df3=as.data.frame(table(EPI2ME_Tax_D5$exit_status,EPI2ME_Tax_D5$barcode))
df3=df3[df3$Var1!="Error",]
df3$workflow="EPI2ME_D5"
colnames(df3)=c("classification","barcode","readnum","workflow_type")
df3$classification=as.character(df3$classification)
df3$classification[which(df3$classification=="Classification successful")]<-"classified"
df3$classification[which(df3$classification=="Unclassified")]<-"unclassified"
df3$barcode=as.character(df3$barcode)
#-------------------------------------------------------------------------------

# POND D5 Data
df4=data.frame()
for (i in unique(D5_Total_Kraken2_Report$barcode)){
  total_read_num=D5_Total_Kraken2_Report$total_read_number[which(D5_Total_Kraken2_Report$barcode==i)][1]
  species_rank_perc=D5_Total_Kraken2_Report$s_cov_perc[which(D5_Total_Kraken2_Report$barcode==i)][1]
  num_classified_reads=total_read_num*species_rank_perc/100
  num_unclassified_reads=total_read_num-total_read_num*species_rank_perc/100
  c1=c("classified",i,num_classified_reads)
  c2=c("unclassified",i, num_unclassified_reads)
  df4=rbind(df4,c1)
  df4=rbind(df4,c2)
}
df4$workflow="POND_D5"
colnames(df4)=c("classification","barcode","readnum","workflow_type")
df4$readnum=round(as.numeric(df4$readnum))
#-------------------------------------------------------------------------------

# Combine all datasets for plotting
data=rbind(df1,df2,df3,df4)
#-------------------------------------------------------------------------------

title=paste("Species Level Classification Comparison")
xlabel="Workflow and Dataset [-]"
ylabel="Number of Reads [-]"
legendlabel="Classification"
filename="Section_5.2_Species_Level_Classification_Comparison.jpg"

q=ggplot(data, aes(x = workflow_type, y = readnum, fill=reorder(classification, desc(classification))))
  geom_bar(stat = "identity", width=0.4)+
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  xlab(xlabel) +
  ylab(ylabel) +
  guides(fill = guide_legend(title = legendlabel))
q=q + scale_fill_manual(values=c("#d1495b","#00798c"))
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Show Graph

```
q
```



Species Abundance Comparison The classified species is comapred based on its ambundance - Abundance Graphs - Pavian Graph

```
Barcode = "BC01"
topx=10

BC_EPI2ME_Tax_D4=as.data.frame(table(EPI2ME_Tax_D4$species[which(EPI2ME_Tax_D4$barcode==Barcode)]))
colnames(BC_EPI2ME_Tax_D4)=c("species_name", "read_freq")
BC_EPI2ME_Tax_D4$workflow="EPI2ME_D4"

BC_EPI2ME_Tax_D5=as.data.frame(table(EPI2ME_Tax_D5$species[which(EPI2ME_Tax_D5$barcode==Barcode)]))
colnames(BC_EPI2ME_Tax_D5)=c("species_name", "read_freq")
BC_EPI2ME_Tax_D5$workflow="EPI2ME_D5"

load_path=paste(Kraken2_Classification_D4_Path, Barcode,".fastq_REPORT", sep = "")
BC_POND_Tax_D4=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE, trim_ws = TRUE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
```

```
##   X3 = col_double(),
##   X4 = col_character(),
##   X5 = col_double(),
##   X6 = col_character()
## )
```

```
BC_POND_Tax_D4=BC_POND_Tax_D4[which(BC_POND_Tax_D4$X4=="S"),]
BC_POND_Tax_D4=as.data.frame(cbind(BC_POND_Tax_D4$X6, BC_POND_Tax_D4$X3))
colnames(BC_POND_Tax_D4)=c( "species_name","read_freq")
BC_POND_Tax_D4$workflow="POND_D4"
BC_POND_Tax_D4$read_freq=as.numeric(BC_POND_Tax_D4$read_freq)

load_path=paste(Kraken2_Classification_D5_Path, Barcode,".fastq_REPORT", sep = "")
BC_POND_Tax_D5=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE, trim_ws = TRUE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_character(),
##   X5 = col_double(),
##   X6 = col_character()
## )
```

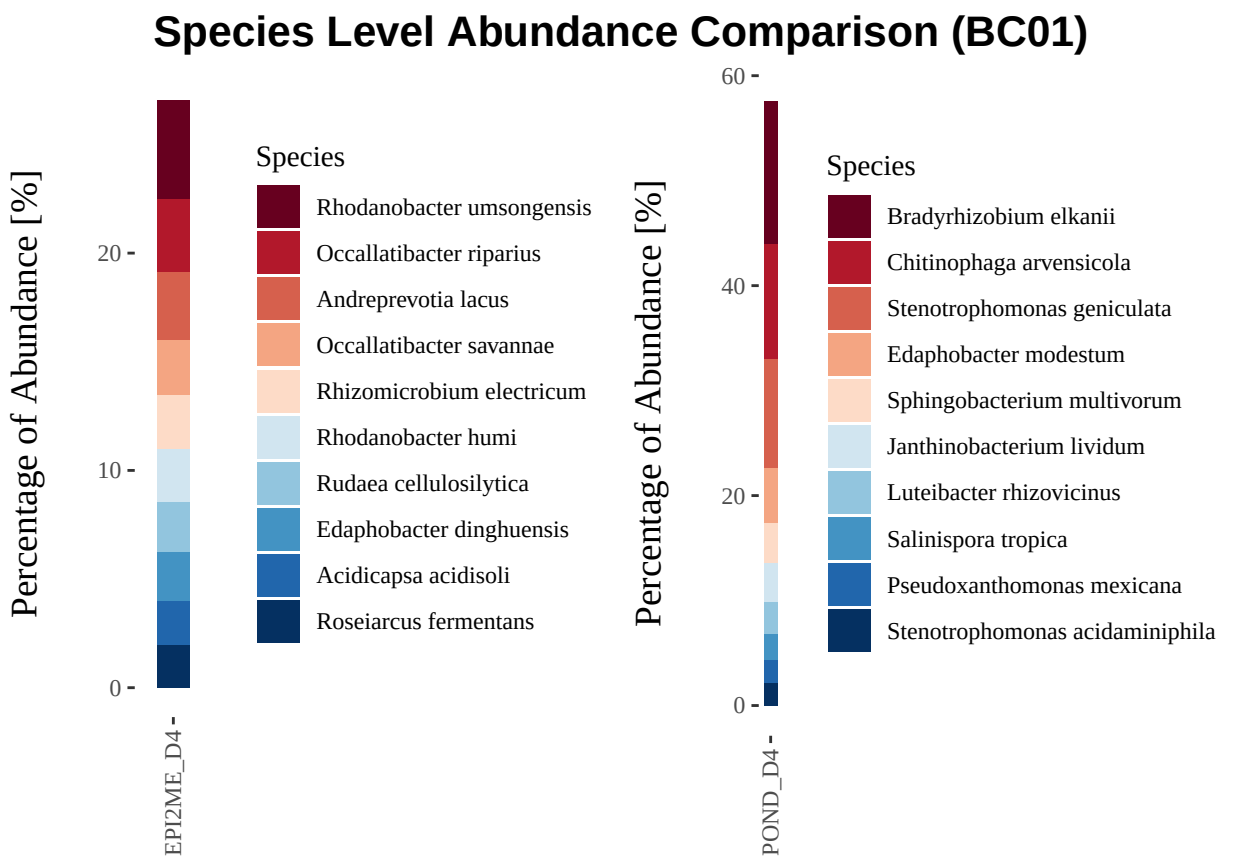
```
BC_POND_Tax_D5=BC_POND_Tax_D5[which(BC_POND_Tax_D5$X4=="S"),]
BC_POND_Tax_D5=as.data.frame(cbind(BC_POND_Tax_D5$X6, BC_POND_Tax_D5$X3))
colnames(BC_POND_Tax_D5)=c( "species_name","read_freq")
BC_POND_Tax_D5$workflow="POND_D5"
BC_POND_Tax_D5$read_freq=as.numeric(BC_POND_Tax_D5$read_freq)

# Create percentage Abundance
BC_EPI2ME_Tax_D4$read_freq=BC_EPI2ME_Tax_D4$read_freq/sum(BC_EPI2ME_Tax_D4$read_freq)*100
BC_EPI2ME_Tax_D5$read_freq=BC_EPI2ME_Tax_D5$read_freq/sum(BC_EPI2ME_Tax_D5$read_freq)*100
BC_POND_Tax_D4$read_freq=BC_POND_Tax_D4$read_freq/sum(BC_POND_Tax_D4$read_freq)*100
BC_POND_Tax_D5$read_freq=BC_POND_Tax_D5$read_freq/sum(BC_POND_Tax_D5$read_freq)*100

# Order the dataframes and only take top 10 abundance
BC_EPI2ME_Tax_D4<-BC_EPI2ME_Tax_D4[order(BC_EPI2ME_Tax_D4$read_freq, decreasing = TRUE),]
BC_EPI2ME_Tax_D4=head(BC_EPI2ME_Tax_D4,topx)

BC_EPI2ME_Tax_D5<-BC_EPI2ME_Tax_D5[order(BC_EPI2ME_Tax_D5$read_freq, decreasing = TRUE),]
BC_EPI2ME_Tax_D5=head(BC_EPI2ME_Tax_D5,topx)

BC_POND_Tax_D4<-BC_POND_Tax_D4[order(BC_POND_Tax_D4$read_freq, decreasing = TRUE),]
BC_POND_Tax_D4=head(BC_POND_Tax_D4,topx)

BC_POND_Tax_D5<-BC_POND_Tax_D5[order(BC_POND_Tax_D5$read_freq, decreasing = TRUE),]
BC_POND_Tax_D5=head(BC_POND_Tax_D5,topx)


# Combine all datasets for plotting
```

```
data=rbind(BC_EPI2ME_Tax_D4,BC_POND_Tax_D4,BC_EPI2ME_Tax_D5,BC_POND_Tax_D5)

# title=paste("Species Level Abundance Comparison", "(", Barcode, ")", sep = "")
# xlabel="Workflow and Dataset [-]"
# ylabel="Percentage of Abundance [%]"
# legendlabel="Species Name"
# filename="species_level_abundance_comparison.jpg"
# # Compare all of Datasets at once
# q=ggplot(data, aes(x = workflow, y = read_freq, fill=species_name)) +
#   geom_bar(stat = "identity", width=0.4)+
#   theme_tufte() +
#   theme(plot.title = element_text(color="Black", size=12, face="bold")) +
#   theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
#         axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
#         axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
#   xlab(xlabel) +
#   ylab(ylabel) +
#   guides(fill = guide_legend(title = legendlabel))
# q=q + scale_fill_brewer(palette="RdBu")
# q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
# ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)

# Define title and filename
title=paste("Species Level Abundance Comparison ", "(", Barcode, ")", sep = "")
xlabel="Workflow and Dataset [-]"
ylabel="Percentage of Abundance [%]"
legendlabel="Species Name"

filename=paste("Section_5.2_Speciel_Level_Abundance_Comparison_D4_",Barcode,".jpg", sep = "")

plot_sample_1 <- ggplot(data=BC_EPI2ME_Tax_D4, aes(x=workflow , y=read_freq, fill=reorder(species_name,
  geom_bar(stat="identity", width = 0.5) +
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  guides(fill = guide_legend(title = "Species")) +
  ylab(ylabel) +
  xlab(NULL)
plot_sample_1=plot_sample_1 + scale_fill_brewer(palette="RdBu")

plot_sample_2 <- ggplot(data=BC_POND_Tax_D4, aes(x=workflow , y=read_freq, fill=reorder(species_name, d
  geom_bar(stat="identity", width = 0.66) +
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  guides(fill = guide_legend(title = "Species")) +
  ylab(ylabel) +
  xlab(NULL)
plot_sample_2=plot_sample_2 + scale_fill_brewer(palette="RdBu")
```

```
q=ggplot <- ggarrange(plot_sample_1, plot_sample_2,ncol = 2, nrow = 1)
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Print the Graph

```
print(q)
```




```
topx=10

BC_EPI2ME_Tax_D4=as.data.frame(table(EPI2ME_Tax_D4$genus[which(EPI2ME_Tax_D4$barcode==Barcode)]))
colnames(BC_EPI2ME_Tax_D4)=c("genus_name", "read_freq")
BC_EPI2ME_Tax_D4$workflow="EPI2ME_D4"

BC_EPI2ME_Tax_D5=as.data.frame(table(EPI2ME_Tax_D5$genus[which(EPI2ME_Tax_D5$barcode==Barcode)]))
colnames(BC_EPI2ME_Tax_D5)=c("genus_name", "read_freq")
BC_EPI2ME_Tax_D5$workflow="EPI2ME_D5"

load_path=paste(Kraken2_Classification_D4_Path, Barcode,".fastq_REPORT", sep = "")
BC_POND_Tax_D4=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE, trim_ws = TRUE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
```

```
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_character(),
##   X5 = col_double(),
##   X6 = col_character()
## )
```

```
BC_POND_Tax_D4=BC_POND_Tax_D4[which(BC_POND_Tax_D4$X4=="G"),]
BC_POND_Tax_D4=as.data.frame(cbind(BC_POND_Tax_D4$X6, BC_POND_Tax_D4$X3))
colnames(BC_POND_Tax_D4)=c( "genus_name","read_freq")
BC_POND_Tax_D4$workflow="POND_D4"
BC_POND_Tax_D4$read_freq=as.numeric(BC_POND_Tax_D4$read_freq)

load_path=paste(Kraken2_Classification_D5_Path, Barcode,".fastq_REPORT", sep = "")
BC_POND_Tax_D5=read_delim(load_path,"\t", escape_double = FALSE, col_names = FALSE, trim_ws = TRUE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_character(),
##   X5 = col_double(),
##   X6 = col_character()
## )
```

```
BC_POND_Tax_D5=BC_POND_Tax_D5[which(BC_POND_Tax_D5$X4=="G"),]
BC_POND_Tax_D5=as.data.frame(cbind(BC_POND_Tax_D5$X6, BC_POND_Tax_D5$X3))
colnames(BC_POND_Tax_D5)=c( "genus_name","read_freq")
BC_POND_Tax_D5$workflow="POND_D5"
BC_POND_Tax_D5$read_freq=as.numeric(BC_POND_Tax_D5$read_freq)

# Create percentage Abundance
BC_EPI2ME_Tax_D4$read_freq=BC_EPI2ME_Tax_D4$read_freq/sum(BC_EPI2ME_Tax_D4$read_freq)*100
BC_EPI2ME_Tax_D5$read_freq=BC_EPI2ME_Tax_D5$read_freq/sum(BC_EPI2ME_Tax_D5$read_freq)*100
BC_POND_Tax_D4$read_freq=BC_POND_Tax_D4$read_freq/sum(BC_POND_Tax_D4$read_freq)*100
BC_POND_Tax_D5$read_freq=BC_POND_Tax_D5$read_freq/sum(BC_POND_Tax_D5$read_freq)*100

# Order the dataframes and only take top 10 abundance
BC_EPI2ME_Tax_D4<-BC_EPI2ME_Tax_D4[order(BC_EPI2ME_Tax_D4$read_freq, decreasing = TRUE),]
BC_EPI2ME_Tax_D4=head(BC_EPI2ME_Tax_D4,topx)

BC_EPI2ME_Tax_D5<-BC_EPI2ME_Tax_D5[order(BC_EPI2ME_Tax_D5$read_freq, decreasing = TRUE),]
BC_EPI2ME_Tax_D5=head(BC_EPI2ME_Tax_D5,topx)

BC_POND_Tax_D4<-BC_POND_Tax_D4[order(BC_POND_Tax_D4$read_freq, decreasing = TRUE),]
BC_POND_Tax_D4=head(BC_POND_Tax_D4,topx)

BC_POND_Tax_D5<-BC_POND_Tax_D5[order(BC_POND_Tax_D5$read_freq, decreasing = TRUE),]
BC_POND_Tax_D5=head(BC_POND_Tax_D5,topx)
```

```
# # Combine all datasets for plotting
# data=rbind(BC_EPI2ME_Tax_D4,BC_POND_Tax_D4,BC_EPI2ME_Tax_D5,BC_POND_Tax_D5)
#
# title=paste("Genus Level Abundance Comparison", "(", Barcode, ")", sep = "")
# xlabel="Workflow and Dataset [-]"
# ylabel="Percentage of Abundance [%]"
# legendlabel="Genus Name"
# filename="genus_level_abundance_comparison.jpg"
#
# q=ggplot(data, aes(x = workflow, y = read_freq, fill=genus_name)) +
#   geom_bar(stat = "identity", width=0.4)+
#   theme_tufte() +
#   theme(plot.title = element_text(color="Black", size=12, face="bold")) +
#   theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
#         axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
#         axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
#   xlab(xlabel) +
#   ylab(ylabel) +
#   guides(fill = guide_legend(title = legendlabel))
# q=q + scale_fill_brewer(palette="RdBu")
# q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
# ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)


# Define title and filename
title=paste("Genus Level Abundance Comparison ", "(", Barcode, ")", sep = "")
xlabel="Workflow and Dataset [-]"
ylabel="Percentage of Abundance [%]"
legendlabel="Genus Name"

filename=paste("Section_5.2_Genus_Level_Abundance_Comparison_D4_",Barcode,".jpg", sep = "")

plot_sample_1 <- ggplot(data=BC_EPI2ME_Tax_D4, aes(x=workflow , y=read_freq, fill=reorder(genus_name, de
  geom_bar(stat="identity", width = 0.37) +
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  guides(fill = guide_legend(title = "Genus")) +
  ylab(ylabel) +
  xlab(NULL)
plot_sample_1=plot_sample_1 + scale_fill_brewer(palette="RdBu")

plot_sample_2 <- ggplot(data=BC_POND_Tax_D4, aes(x=workflow , y=read_freq, fill=reorder(genus_name, des
  geom_bar(stat="identity", width = 0.37) +
  theme_tufte() +
  theme(plot.title = element_text(color="Black", size=12, face="bold")) +
  theme(axis.title.y = element_text(size=14,margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(size=14,margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  guides(fill = guide_legend(title = "Genus")) +
  ylab(ylabel) +
```

```
  xlab(NULL)
plot_sample_2=plot_sample_2 + scale_fill_brewer(palette="RdBu")

q=ggplot <- ggarrange(plot_sample_1, plot_sample_2,ncol = 2, nrow = 1)
q=annotate_figure(q,top = text_grob(title, color = "black", face = "bold", size = 16))
ggsave(plot = q, width = 8, height = 6, dpi = 300, filename = filename, path = graph_path)
```

Print the Graph

```
q
```