

TB-Profiling Analysis

Geoffrey Olweny

2024-11-18

Contents

Install necessary packages required	2
Load the necessary packages	3
Reading in the Files	3
Inspect files	4
Convert each OTU, Taxonomy and Metadata file into a matrix	12
Create phyloseq object	12
Inspect the phyloseq object	12
Save phyloseq objects	13
Diversity metrics	13
Alpha diversity	13
Function for generating Mtb Infection status plots	13
Generate cow plot	14
View cow plot	14
Beta diversity	15
DMM	18
Cofounding factors between microbiome alpha diversity metrics and the different DMM microbial communities	29
Beta diversity of the different DMM microbial community types/pulmotypes	31
Relative abundance	34
Phylum	34
Genus	41
Differential abundance	48

Biomarker Identification	53
LEfSe analysis	53
Machine Learning	63
Compare machine learning models	63
Random Forest model	65

Install necessary packages required

These are packages to be used in the analysis.

```
# List of CRAN packages
cran_packages <- c(
  "ggplot2", "pgirmess", "reshape", "dplyr", "tidyverse", "rstatix",
  "ggpubr", "cowplot", "RColorBrewer", "fs", "vegan", "ggsignif", "ggtreeExtra",
  "ggstar", "ggalluvial", "ggtree", "patchwork", "randomForest",
  "caret", "ROCR", "e1071", "gridExtra", "fastDummies", "here", "pROC"
)

# List of Bioconductor packages
bioc_packages <- c(
  "metadeconfoundR", "DirichletMultinomial", "phyloseq", "edgeR", "MicrobiotaProcess"
)

# Function to check if a package is installed, and if not, install it
install_if_missing_cran <- function(pkg) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    install.packages(pkg, dependencies = TRUE)
  }
}

# Function to check if a Bioconductor package is installed, and if not, install it
install_if_missing_bioc <- function(pkg) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    BiocManager::install(pkg)
  }
}

# Apply the function to each CRAN package
lapply(cran_packages, install_if_missing_cran)

# Apply the function to each Bioconductor package
lapply(bioc_packages, install_if_missing_bioc)

# Install EasyStat from Git
if (!requireNamespace("EasyStat", quietly = TRUE)) {
  remotes::install_git('https://gitee.com/wentaomicro/EasyStat')
}

# Install ggClusterNet from GitHub
if (!requireNamespace("ggClusterNet", quietly = TRUE)) {
  remotes::install_github("taowenmicro/ggClusterNet")
}
```

```

}

# Load all the packages
all_packages <- c(cran_packages, bioc_packages, "EasyStat", "ggClusterNet")
lapply(all_packages, library, character.only = TRUE)

```

Load the necessary packages

These are to be used during analysis.

```

# Load required libraries
library(metadeconfoundR)
library(DirichletMultinomial)
library(phyloseq)
library(ggplot2)
library(pgirmess)
library(reshape)
library(dplyr)
library(tidyverse)
library(rstatix)
library(ggpubr)
library(cowplot)
library(fs)
library(vegan)
library(ggClusterNet)
library(EasyStat)
library(ggsignif)
library(ggtreeExtra)
library(ggstar)
library(ggalluvial)
library(ggtree)
library(patchwork)
library(edgeR)
library(randomForest)
library(caret)
library(ROCR)
library(e1071)
library(MicrobiotaProcess)
library(gridExtra)
library(fastDummies)
library(RColorBrewer)
library(here)

sessionInfo()

```

Reading in the Files

These are input files required for the protocol.

```

# Load OTU counts files
Genexpert_otu_df <- read.table("data/Genexpert_OTU.txt", header = TRUE, row.names = 1)
MtbInfectionStatus_otu_df <- read.table("data/MtbInfectionStatus_OTU.txt", header = TRUE, row.names = 1)
# Load taxonomy table
tax_df <- read.table("data/Tax.txt", header = TRUE, sep = "\t", row.names = 1, check.names = FALSE)
# Replace all "?" values in tax_df with "Unknown"
tax_df[tax_df == "?"] <- "Unknown"
# Load metadata files
Genexpert_metadata_df <- read.csv(file = "data/Genexpert_metadata.csv", header = TRUE, row.names = 1)
MtbInfectionStatus_metadata_df <- read.csv(file = "data/MtbInfectionStatus_metadata.csv", header = TRUE, row.names = 1)

```

Inspect files

```

# Inspect the first few rows of Genexpert OTU counts
head(Genexpert_otu_df)
#>      P01.A01.HLM069 P01.A02.HLM002 P01.A03.HLM038 P01.A04.HLM091 P01.A05.HLM061
#> OTU1              3535           29248           2826           1357           11970
#> OTU2              4420           27781           2856           2034           12041
#> OTU3                0                0                0                0                0
#> OTU4                0                0                0                0                0
#> OTU5              9383             185             428             6681             749
#> OTU6             15765             264             268             2504             706
#>      P01.A06.HLM094 P01.A07.HLM102 P01.A08.HLM056 P01.A09.HLM053 P01.A10.HLM019
#> OTU1              5904             5920             4279             11938             16310
#> OTU2              5417             5628             4227             11667             15288
#> OTU3                0                0                0                0                0
#> OTU4                0                0                0                0                0
#> OTU5              3587             206             4768             1778             1616
#> OTU6              2915              49             2194             1789             4876
#>      P01.A11.HLM137 P01.A12.HLM131 P01.B01.HLM062 P01.B02.HLM064 P01.B03.HLM037
#> OTU1              6244             8808             1805             12257             2761
#> OTU2              6404             9541             1705             11729             2663
#> OTU3                0                0                0                0                0
#> OTU4                0                0                0                0                0
#> OTU5              4518              25             1364             626             303
#> OTU6              1519             5310             15212             99             190
#>      P01.B04.HLM089 P01.B05.HLM107 P01.B06.HLM097 P01.B07.HLM098 P01.B08.HLM106
#> OTU1              1992             20050             8446             3482             391
#> OTU2              1918             20003             8019             3801             407
#> OTU3                0                0                0                0                0
#> OTU4                0                0                0                0                0
#> OTU5              1427             5259             544             2962             109
#> OTU6              1513             8138             712             1193             31
#>      P01.B09.HLM054 P01.B10.HLM020 P01.B11.HLM136 P01.B12.HLM122 P01.C01.HLM080
#> OTU1              1394             29611             9606             5960             1994
#> OTU2              1383             28680             8337             5348             1659
#> OTU3                0                0                0                0                0
#> OTU4                0                0                0                0                0
#> OTU5              4015             524             803             275             6881
#> OTU6              4596             137             106             56             9915
#>      P01.C02.HLM044 P01.C03.HLM045 P01.C04.HLM074 P01.C05.HLM114 P01.C06.HLM084

```

#> OTU1	5927	2344	3077	6969	5311
#> OTU2	5441	2197	2707	6727	5323
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	12482	505	1143	1521	12162
#> OTU6	14380	1087	3335	1141	6206
#>	P01.C07.HLM104	P01.C08.HLM096	P01.C09.HLM035	P01.C10.HLM021	P01.C11.HLM117
#> OTU1	9648	3421	10557	9749	7363
#> OTU2	9680	3252	10220	8320	6913
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	892	715	2895	379	3954
#> OTU6	409	584	3472	327	1042
#>	P01.C12.HLM128	P01.D01.HLM079	P01.D02.HLM043	P01.D03.HLM025	P01.D04.HLM073
#> OTU1	3212	4528	6698	5463	10157
#> OTU2	3085	4393	6366	5463	9607
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	2891	23	191	2169	6722
#> OTU6	504	79	232	1734	4159
#>	P01.D05.HLM090	P01.D06.HLM093	P01.D07.HLM105	P01.D08.HLM086	P01.D09.HLM055
#> OTU1	2113	4549	3833	4283	11894
#> OTU2	2012	4272	4645	3965	11934
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	699	1705	1681	771	4302
#> OTU6	1756	148	1843	646	3561
#>	P01.D10.HLM022	P01.D11.HLM101	P01.D12.HLM135	P01.E01.HLM065	P01.E02.HLM042
#> OTU1	4232	8691	2270	2412	11017
#> OTU2	4009	7372	2124	2355	11271
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	1186	911	204	1757	6181
#> OTU6	1057	238	75	4673	3215
#>	P01.E03.HLM049	P01.E05.HLM095	P01.E06.HLM100	P01.E07.HLM103	P01.E08.HLM112
#> OTU1	2504	3529	11085	13691	5748
#> OTU2	3111	4889	10099	13972	5486
#> OTU3	0	0	0	0	0
#> OTU4	1	0	0	0	0
#> OTU5	841	20327	470	12167	734
#> OTU6	8724	11277	870	2473	271
#>	P01.E09.HLM057	P01.E10.HLM023	P01.E11.HLM120	P01.E12.HLM138	P01.F01.HLM066
#> OTU1	3275	6643	5451	4505	362
#> OTU2	2989	7378	10802	4774	390
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	1719	10219	3525	5241	910
#> OTU6	2397	8074	11704	3297	668
#>	P01.F02.HLM041	P01.F03.HLM048	P01.F05.HLM092	P01.F06.HLM081	P01.F07.HLM087
#> OTU1	6514	2970	8410	5342	3853
#> OTU2	6392	2624	9607	5229	3572
#> OTU3	0	1	0	1	0
#> OTU4	0	0	0	0	0

#> OTU5	89	508	2519	65	290
#> OTU6	883	322	5987	110	3627
#>	P01.F08.HLM050	P01.F09.HLM058	P01.F10.HLM024	P01.F11.HLM116	P01.F12.HLM140
#> OTU1	6201	4984	5744	4787	9264
#> OTU2	6133	4805	6176	4457	9750
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	882	8251	1906	403	2061
#> OTU6	1187	1896	4528	340	1525
#>	P01.G01.HLM068	P01.G03.HLM067	P01.G04.HLM109	P01.G05.HLM083	P01.G06.HLM075
#> OTU1	4535	3191	11569	2189	15747
#> OTU2	3988	2811	10813	2108	16064
#> OTU3	0	1	0	1	0
#> OTU4	0	0	0	0	0
#> OTU5	372	3171	202	2343	1959
#> OTU6	25	964	121	5070	4084
#>	P01.G07.HLM072	P01.G08.HLM051	P01.G09.HLM059	P01.G10.HLM018	P01.G11.HLM123
#> OTU1	23475	14926	9789	3335	10761
#> OTU2	25045	14513	11781	3438	10048
#> OTU3	0	0	0	0	1
#> OTU4	0	0	0	0	0
#> OTU5	4407	522	1259	1606	13
#> OTU6	5502	369	4005	1657	311
#>	P01.G12.HLM178	P01.H01.HLM063	P01.H02.HLM039	P01.H03.HLM070	P01.H04.HLM110
#> OTU1	9978	1467	12026	6341	4345
#> OTU2	10178	1530	11612	6159	4122
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	4046	6545	2995	1156	1965
#> OTU6	3867	5590	4097	3068	1500
#>	P01.H05.HLM082	P01.H06.HLM078	P01.H07.HLM099	P01.H08.HLM052	P01.H09.HLM029
#> OTU1	2424	10655	12290	3938	13266
#> OTU2	2381	9713	12073	3937	12806
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	358	1935	5245	3445	946
#> OTU6	665	1273	6731	3095	574
#>	P01.H10.HLM017	P01.H11.HLM124	P01.H12.HLM148	P02.A01.HLM146	P02.A02.HLM016
#> OTU1	1563	7811	5929	7118	7418
#> OTU2	1531	7212	5206	6512	7269
#> OTU3	0	1	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	648	4747	232	1418	4758
#> OTU6	149	1415	582	702	494
#>	P02.A03.HLM171	P02.A04.HLM004	P02.A05.HLM167	P02.A06.HLM172	P02.A07.HLM154
#> OTU1	262	5488	10108	12980	2583
#> OTU2	30	4924	9531	12689	2749
#> OTU3	1	0	0	0	0
#> OTU4	0	0	1	0	0
#> OTU5	1	20	981	1000	205
#> OTU6	16	1656	5955	1365	875
#>	P02.A08.HLM085	P02.A09.HLM134	P02.A10.HLM405	P02.A11.HLM413	P02.A12.HLM421
#> OTU1	4325	1563	164	13534	376

#> OTU2	4017	1430	176	13427	389
#> OTU3	0	0	25330	18023	20357
#> OTU4	0	0	22786	16339	18961
#> OTU5	4829	3447	38	1	0
#> OTU6	2790	553	146	34	85
#>	P02.B01.HLM145	P02.B02.HLM177	P02.B03.HLM032	P02.B04.HLM001	P02.B05.HLM165
#> OTU1	266	3775	12538	10678	5331
#> OTU2	239	3203	11561	9855	5204
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	110	491	1056	903	2497
#> OTU6	82	662	5453	1617	817
#>	P02.B06.HLM013	P02.B07.HLM155	P02.B08.HLM003	P02.B09.HLM130	P02.B10.HLM406
#> OTU1	3882	5308	25768	12312	6874
#> OTU2	3692	5730	25621	11903	6450
#> OTU3	0	0	0	0	3394
#> OTU4	0	0	0	0	3109
#> OTU5	100	1139	4434	497	0
#> OTU6	196	1177	2407	222	58
#>	P02.B11.HLM414	P02.B12.HLM422	P02.C01.HLM143	P02.C02.HLM182	P02.C03.HLM033
#> OTU1	5550	324	3629	1377	4636
#> OTU2	5708	361	3453	935	4256
#> OTU3	1902	638	0	1	0
#> OTU4	1869	621	0	0	0
#> OTU5	0	0	315	490	2725
#> OTU6	30	83	283	678	3366
#>	P02.C04.HLM006	P02.C05.HLM129	P02.C06.HLM010	P02.C07.HLM157	P02.C08.HLM111
#> OTU1	14079	5316	7122	5123	24555
#> OTU2	13347	5437	6898	4150	24946
#> OTU3	0	0	0	0	0
#> OTU4	0	0	0	0	0
#> OTU5	1165	3756	238	594	607
#> OTU6	951	1291	282	147	342
#>	P02.C09.HLM127	P02.C10.HLM407	P02.C11.HLM415	P02.C12.HLM423	P02.D01.HLM115
#> OTU1	3263	949	2802	0	9134
#> OTU2	2864	900	2822	0	8126
#> OTU3	1	7896	1663	27594	0
#> OTU4	0	7246	1641	25516	0
#> OTU5	936	0	369	0	227
#> OTU6	598	41	41	1	537
#>	P02.D02.HLM186	P02.D03.HLM030	P02.D04.HLM173	P02.D05.HLM166	P02.D06.HLM189
#> OTU1	17092	7442	11468	9343	19794
#> OTU2	15934	7261	10746	9308	19297
#> OTU3	0	0	0	0	0
#> OTU4	0	0	1	0	0
#> OTU5	3830	94	3523	729	126
#> OTU6	2149	44	2671	370	3
#>	P02.D07.HLM158	P02.D08.HLM005	P02.D09.HLM400	P02.D10.HLM408	P02.D11.HLM416
#> OTU1	27441	7169	2	490	28
#> OTU2	25387	6632	0	443	20
#> OTU3	0	0	24451	13749	10195
#> OTU4	2	0	22516	13044	9262
#> OTU5	45	1439	0	0	20

#> OTU6	271	95	0	101	28
#>	P02.D12.HLM424	P02.E01.HLM144	P02.E02.HLM184	P02.E03.HLM014	P02.E04.HLM174
#> OTU1	9	3223	4669	13122	8730
#> OTU2	3	2912	4502	12316	7992
#> OTU3	49941	0	0	0	1
#> OTU4	46388	0	0	0	1
#> OTU5	0	67	1394	487	997
#> OTU6	1	411	203	1026	495
#>	P02.E05.HLM149	P02.E06.HLM190	P02.E07.HLM159	P02.E08.HLM132	P02.E09.HLM401
#> OTU1	8469	6610	5883	11059	0
#> OTU2	10078	6601	5622	11024	2
#> OTU3	0	0	0	0	35473
#> OTU4	0	0	0	0	32911
#> OTU5	8159	4653	121	3681	0
#> OTU6	4579	1015	72	5395	0
#>	P02.E10.HLM409	P02.E11.HLM417	P02.E12.HLM425	P02.F01.HLM179	P02.F02.HLM187
#> OTU1	76	2	1	163	23064
#> OTU2	20	1	0	177	22480
#> OTU3	3404	14098	861	0	0
#> OTU4	3296	13335	801	0	0
#> OTU5	0	1	1	8	837
#> OTU6	14	0	3	3	616
#>	P02.F03.HLM027	P02.F04.HLM175	P02.F05.HLM163	P02.F06.HLM126	P02.F07.HLM161
#> OTU1	5740	2661	5078	7113	15539
#> OTU2	5417	2670	5210	6454	15336
#> OTU3	0	0	0	0	0
#> OTU4	0	1	0	0	0
#> OTU5	1183	54	2256	2790	563
#> OTU6	1210	344	1359	84	1287
#>	P02.F08.HLM007	P02.F09.HLM402	P02.F10.HLM410	P02.F11.HLM418	P02.F12.HLM426
#> OTU1	5304	169	156	1462	303
#> OTU2	5351	169	123	1565	301
#> OTU3	0	7180	569	5554	3093
#> OTU4	0	6663	622	5022	2846
#> OTU5	2393	0	0	146	97
#> OTU6	2132	57	48	46	4
#>	P02.G01.HLM142	P02.G02.HLM188	P02.G03.HLM036	P02.G04.HLM147	P02.G05.HLM164
#> OTU1	6322	7875	12443	5244	4841
#> OTU2	6269	7410	11548	4421	4486
#> OTU3	0	0	0	1	0
#> OTU4	0	0	0	0	0
#> OTU5	388	415	136	210	313
#> OTU6	268	245	232	4124	718
#>	P02.G06.HLM141	P02.G07.HLM118	P02.G08.HLM008	P02.G09.HLM403	P02.G10.HLM411
#> OTU1	8205	8151	6264	0	78
#> OTU2	7789	9080	5792	1	100
#> OTU3	0	0	0	37534	477
#> OTU4	0	0	0	34974	488
#> OTU5	2173	8052	48	0	1
#> OTU6	73	5748	40	0	259
#>	P02.G11.HLM419	P02.G12.HLM427	P02.H01.HLM180	P02.H02.HLM176	P02.H03.HLM031
#> OTU1	3551	456	380	12289	933
#> OTU2	3401	454	352	11797	913


```

#> OTU3      10683      4538      0      0      0
#> OTU4      10204      4331      0      0      0
#> OTU5      0      23      6      555      342
#> OTU6      158      6      13      591      140
#>      P02.H04.HLM162 P02.H05.HLM009 P02.H06.HLM152 P02.H07.HLM119 P02.H08.HLM011
#> OTU1      6043      14173      9164      10575      1758
#> OTU2      5930      13656      8558      8710      1827
#> OTU3      0      2      1      0      0
#> OTU4      0      0      1      0      0
#> OTU5      4398      3933      2433      1728      1245
#> OTU6      2052      3379      870      657      583
#>      P02.H09.HLM404 P02.H10.HLM412 P02.H11.HLM420 P02.H12.HLM428 P03.A01.HLM429
#> OTU1      1      131      13      4      237
#> OTU2      0      180      145      1      260
#> OTU3      9189      22068      8178      6681      3386
#> OTU4      8597      20503      7712      6297      3218
#> OTU5      0      0      0      0      0
#> OTU6      0      69      410      12      6
#>      P03.B01.HLM430 P03.C01.HLM431 P03.D01.NEC
#> OTU1      1147      31      0
#> OTU2      1172      21      0
#> OTU3      17435      47      0
#> OTU4      16101      36      0
#> OTU5      0      1      0
#> OTU6      8      158      0

# Inspect the first few rows of Mtb Infection Status OTU counts
head(MtbInfectionStatus_otu_df)
#>      P01.A06.HLM094 P01.A07.HLM102 P01.A09.HLM053 P01.A11.HLM137 P01.A12.HLM131
#> OTU3      0      0      0      0      0
#> OTU4      0      0      0      0      0
#> OTU1      5904      5920      11938      6244      8808
#> OTU2      5417      5628      11667      6404      9541
#> OTU5      3587      206      1778      4518      25
#> OTU6      2915      49      1789      1519      5310
#>      P01.B10.HLM020 P01.C01.HLM080 P01.C09.HLM035 P01.D01.HLM079 P01.D04.HLM073
#> OTU3      0      0      0      0      0
#> OTU4      0      0      0      0      0
#> OTU1      29611      1994      10557      4528      10157
#> OTU2      28680      1659      10220      4393      9607
#> OTU5      524      6881      2895      23      6722
#> OTU6      137      9915      3472      79      4159
#>      P01.D06.HLM093 P01.D07.HLM105 P01.D08.HLM086 P01.E02.HLM042 P01.E06.HLM100
#> OTU3      0      0      0      0      0
#> OTU4      0      0      0      0      0
#> OTU1      4549      3833      4283      11017      11085
#> OTU2      4272      4645      3965      11271      10099
#> OTU5      1705      1681      771      6181      470
#> OTU6      148      1843      646      3215      870
#>      P01.E07.HLM103 P01.E10.HLM023 P01.E12.HLM138 P01.F01.HLM066 P01.F06.HLM081
#> OTU3      0      0      0      0      1
#> OTU4      0      0      0      0      0
#> OTU1      13691      6643      4505      362      5342

```

#> OTU2	13972	7378	4774	390	5229
#> OTU5	12167	10219	5241	910	65
#> OTU6	2473	8074	3297	668	110
#>	P01.F12.HLM140	P01.G03.HLM067	P01.G05.HLM083	P01.G06.HLM075	P01.H01.HLM063
#> OTU3	0	1	1	0	0
#> OTU4	0	0	0	0	0
#> OTU1	9264	3191	2189	15747	1467
#> OTU2	9750	2811	2108	16064	1530
#> OTU5	2061	3171	2343	1959	6545
#> OTU6	1525	964	5070	4084	5590
#>	P01.H04.HLM110	P01.H08.HLM052	P01.H11.HLM124	P01.H12.HLM148	P02.A01.HLM146
#> OTU3	0	0	1	0	0
#> OTU4	0	0	0	0	0
#> OTU1	4345	3938	7811	5929	7118
#> OTU2	4122	3937	7212	5206	6512
#> OTU5	1965	3445	4747	232	1418
#> OTU6	1500	3095	1415	582	702
#>	P02.A05.HLM167	P02.A09.HLM134	P02.A10.HLM405	P02.A11.HLM413	P02.A12.HLM421
#> OTU3	0	0	25330	18023	20357
#> OTU4	1	0	22786	16339	18961
#> OTU1	10108	1563	164	13534	376
#> OTU2	9531	1430	176	13427	389
#> OTU5	981	3447	38	1	0
#> OTU6	5955	553	146	34	85
#>	P02.B01.HLM145	P02.B04.HLM001	P02.B08.HLM003	P02.B10.HLM406	P02.B11.HLM414
#> OTU3	0	0	0	3394	1902
#> OTU4	0	0	0	3109	1869
#> OTU1	266	10678	25768	6874	5550
#> OTU2	239	9855	25621	6450	5708
#> OTU5	110	903	4434	0	0
#> OTU6	82	1617	2407	58	30
#>	P02.B12.HLM422	P02.C10.HLM407	P02.C11.HLM415	P02.C12.HLM423	P02.D09.HLM400
#> OTU3	638	7896	1663	27594	24451
#> OTU4	621	7246	1641	25516	22516
#> OTU1	324	949	2802	0	2
#> OTU2	361	900	2822	0	0
#> OTU5	0	0	369	0	0
#> OTU6	83	41	41	1	0
#>	P02.D10.HLM408	P02.D11.HLM416	P02.D12.HLM424	P02.E09.HLM401	P02.E10.HLM409
#> OTU3	13749	10195	49941	35473	3404
#> OTU4	13044	9262	46388	32911	3296
#> OTU1	490	28	9	0	76
#> OTU2	443	20	3	2	20
#> OTU5	0	20	0	0	0
#> OTU6	101	28	1	0	14
#>	P02.E11.HLM417	P02.E12.HLM425	P02.F09.HLM402	P02.F10.HLM410	P02.F11.HLM418
#> OTU3	14098	861	7180	569	5554
#> OTU4	13335	801	6663	622	5022
#> OTU1	2	1	169	156	1462
#> OTU2	1	0	169	123	1565
#> OTU5	1	1	0	0	146
#> OTU6	0	3	57	48	46
#>	P02.F12.HLM426	P02.G03.HLM036	P02.G09.HLM403	P02.G10.HLM411	P02.G11.HLM419

```

#> OTU3      3093      0      37534      477      10683
#> OTU4      2846      0      34974      488      10204
#> OTU1       303     12443      0       78      3551
#> OTU2       301     11548      1     100      3401
#> OTU5        97     136      0       1       0
#> OTU6         4     232      0     259      158
#>      P02.G12.HLM427 P02.H02.HLM176 P02.H04.HLM162 P02.H06.HLM152 P02.H09.HLM404
#> OTU3      4538      0      0      1      9189
#> OTU4      4331      0      0      1      8597
#> OTU1       456     12289     6043     9164      1
#> OTU2       454     11797     5930     8558      0
#> OTU5        23      555     4398     2433      0
#> OTU6         6      591     2052     870      0
#>      P02.H10.HLM412 P02.H11.HLM420 P02.H12.HLM428 P03.A01.HLM429 P03.B01.HLM430
#> OTU3      22068     8178     6681     3386     17435
#> OTU4      20503     7712     6297     3218     16101
#> OTU1       131      13      4      237     1147
#> OTU2       180      145      1      260     1172
#> OTU5         0       0      0       0       0
#> OTU6         69     410      12      6       8
#>      P03.C01.HLM431 P03.D01.NEC
#> OTU3        47      0
#> OTU4        36      0
#> OTU1        31      0
#> OTU2        21      0
#> OTU5         1      0
#> OTU6       158      0

```

Inspect the first few rows of the taxonomy table

```
head(tax_df)
```

```

#>      Kingdom      Phylum      Class      Order      Family
#> OTU120 Bacteria Firmicutes Bacilli Erysipelotrichales Erysipelotrichaceae
#> OTU524 Bacteria Firmicutes Bacilli Erysipelotrichales Erysipelotrichaceae
#> OTU598 Bacteria Firmicutes Bacilli Erysipelotrichales Erysipelotrichaceae
#> OTU662 Bacteria Firmicutes Bacilli Erysipelotrichales Erysipelatoclostridiaceae
#> OTU725 Bacteria Firmicutes Bacilli Erysipelotrichales Erysipelatoclostridiaceae
#> OTU460 Bacteria Firmicutes Bacilli Acholeplasmatales Acholeplasmataceae
#>      Genus      Species
#> OTU120      Unknown      Unknown
#> OTU524      Bulleidia      Bulleidia extructa
#> OTU598 Erysipelotrichaceae UCG-006      Unknown
#> OTU662      Unknown      Unknown
#> OTU725      Eggerthia      Eggerthia catenaformis
#> OTU460      Acholeplasma Mycoplasmataceae genomosp.

```

Inspect the first few rows of Genexpert metadata

```
head(Genexpert_metadata_df)
```

```

#>      Group
#> P01.A01.HLM069 Xpert-ve
#> P01.A02.HLM002 Xpert-ve
#> P01.A03.HLM038 Xpert-ve
#> P01.A04.HLM091 Xpert-ve
#> P01.A05.HLM061 Xpert-ve

```

```
#> P01.A06.HLM094 Xpert-ve

# Inspect the first few rows of Mtb Infection Status metadata
head(MtbInfectionStatus_metadata_df)
#>           Group
#> P01.A06.HLM094 M.tb_Uninfected
#> P01.A07.HLM102          LTBI
#> P01.A09.HLM053          LTBI
#> P01.A11.HLM137 M.tb_Uninfected
#> P01.A12.HLM131 M.tb_Uninfected
#> P01.B10.HLM020 M.tb_Uninfected
```

Convert each OTU, Taxonomy and Metadata file into a matrix

```
# Create the OTU table for Genexpert samples
Genexpert_OTU <- otu_table(Genexpert_otu_df, taxa_are_rows = TRUE)

# Create the OTU table for Mtb Infection Status samples
MtbInfectionStatus_OTU <- otu_table(MtbInfectionStatus_otu_df, taxa_are_rows = TRUE)

# Create the taxonomy table
TAX <- phyloseq::tax_table(as.matrix(tax_df))

# Create the sample data for Genexpert samples
Genexpert_SAM <- sample_data(Genexpert_metadata_df)

# Create the sample data for Mtb Infection Status samples
MtbInfectionStatus_SAM <- sample_data(MtbInfectionStatus_metadata_df)
```

Create phyloseq object

This is an S4 object required for downstream analysis.

```
# Create the phyloseq object for Genexpert samples
Genexpert_phyloseq <- phyloseq(Genexpert_OTU, TAX, Genexpert_SAM)

# Create the phyloseq object for Mtb Infection Status samples
MtbInfectionStatus_phyloseq <- phyloseq(MtbInfectionStatus_OTU, TAX, MtbInfectionStatus_SAM)
```

Inspect the phyloseq object

```
# Inspect the Genexpert phyloseq object
Genexpert_phyloseq
#> phyloseq-class experiment-level object
#> otu_table() OTU Table: [ 835 taxa and 193 samples ]
#> sample_data() Sample Data: [ 193 samples by 1 sample variables ]
#> tax_table() Taxonomy Table: [ 835 taxa by 7 taxonomic ranks ]
```

```
# Inspect the Mtb Infection Status phyloseq object
MtbInfectionStatus_phyloseq
#> phyloseq-class experiment-level object
#> otu_table() OTU Table:      [ 500 taxa and 72 samples ]
#> sample_data() Sample Data:  [ 72 samples by 1 sample variables ]
#> tax_table() Taxonomy Table:  [ 500 taxa by 7 taxonomic ranks ]
```

Save phyloseq objects

```
# Save Genexpert_physeq as an RDS file
saveRDS(Genexpert_phyloseq, file = "data/Genexpert_physeq.rds")

# Save MtbInfectionStatus_physeq as an RDS file
saveRDS(MtbInfectionStatus_phyloseq, file = "data/MtbInfectionStatus_physeq.rds")
```

Diversity metrics

Alpha diversity

```
metrics <- c("Shannon", "Observed", "Chao1", "Simpson", "Fisher", "ACE")
pal.Coll <- c("darkblue", "coral", "darkgreen")

# Estimate richness for Genexpert data
alpha_metrics_Group <- estimate_richness(Genexpert_phyloseq, measures = metrics)

# Estimate richness for MtbStatus data
alpha_metrics_MtbStatus <- estimate_richness(MtbInfectionStatus_phyloseq, measures = metrics)

# Combine Genexpert metadata with richness estimates and adjust the Group variable
alpha_metrics_df_Group <- cbind(Genexpert_metadata_df, alpha_metrics_Group)
alpha_metrics_df_Group$Group <- as.factor(alpha_metrics_df_Group$Group)

# Combine MtbStatus metadata with richness estimates and adjust the Group variable
alpha_metrics_df_MtbStatus <- cbind(MtbInfectionStatus_metadata_df, alpha_metrics_MtbStatus)
alpha_metrics_df_MtbStatus$Group <- as.factor(alpha_metrics_df_MtbStatus$Group)
```

Function for generating Mtb Infection status plots

```
# Function to generate a plot for a given metric
generate_plot <- function(metric) {
  # Perform Wilcoxon test on the metric grouped by 'Group', adjust p-values using BH method,
  # add significance and position information
  stats_test <- alpha_metrics_df_MtbStatus %>%
    rstatix::wilcox_test(reformulate("Group", metric)) %>%
    rstatix::adjust_pvalue(method = "BH") %>%
    rstatix::add_significance() %>%
```

```

rstatix::add_xy_position(x = "Group")

# Create the plot
plot <- alpha_metrics_df_MtbStatus %>%
  ggplot(aes(x = Group, y = .data[[metric]])) +
  geom_boxplot(color = "black", alpha = 0.5, outlier.shape = NA) + # Add boxplot
  geom_point(position = position_jitter(0.2), size = 3, aes(fill = Group, color = Group)) + # Add points
  scale_color_manual(values = pal.Coll) + # Set manual color scale
  xlab("Group") + # Label x-axis
  ylab(paste(metric, "Index")) + # Label y-axis
  labs(caption = get_pwc_label(stats_test)) + # Add caption with significance
  theme_bw() + # Set theme to black and white
  theme(
    text = element_text(size = 10), # Set text size
    axis.title.x = element_blank(), # Remove x-axis title
    axis.text.x = element_blank(), # Remove x-axis text
    axis.ticks.x = element_blank(), # Remove x-axis ticks
    axis.text.x.bottom = element_text(size = 10, angle = 90), # Set x-axis text at bottom with rotation
    legend.position = "none" # Remove legend
  ) +
  stat_pvalue_manual(stats_test, bracket.nudge.y = -2, step.increase = 0.05, hide.ns = TRUE, tip.length = 0.5)

# Return the plot and statistical test results as a list
return(list(plot = plot, stats_test = stats_test))
}

```

Generate cow plot

```

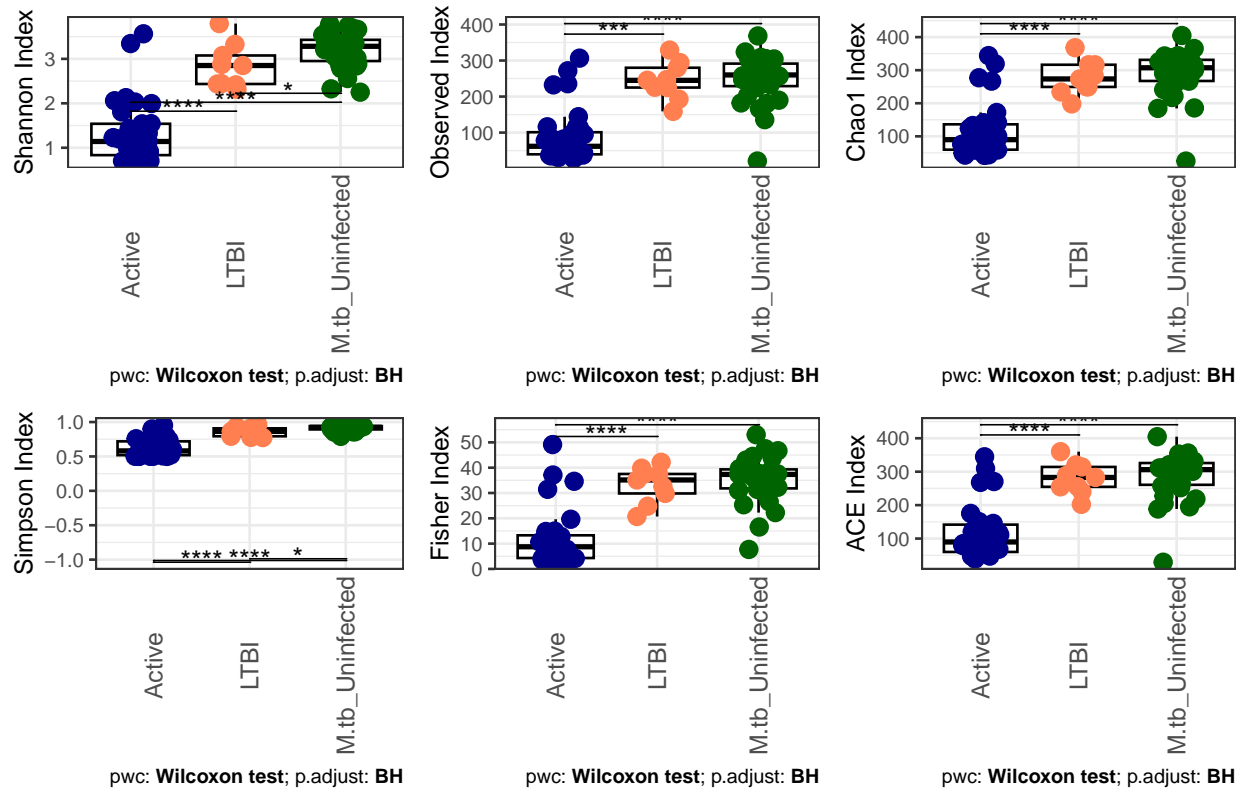
plots <- lapply(metrics, generate_plot)
names(plots) <- metrics
alpha_diversity_final <- cowplot::plot_grid(plotlist = lapply(plots, `[`, "plot"))
# Add a title to the alpha_diversity_final plot
alpha_diversity_final <- ggdraw() +
  draw_plot(alpha_diversity_final, 0, 0, 1, 0.9) + # Adjust plot size to leave space for title
  draw_label("Alpha Diversity According to Mtb Infection Status",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16, fontface = "bold")

```

View cow plot

```
print(alpha_diversity_final)
```

Alpha Diversity According to Mtb Infection Status



```
ggsave(filename = "Results/Figure 1.jpg",
       plot = alpha_diversity_final,
       width = 10, height = 8, dpi = 600)
```

Beta diversity

This is done using the Bray_Curtis dissimilarity matrix.

```
# Compute Bray-Curtis dissimilarity matrix and perform PCoA for Genexpert results
beta <- vegan::vegdist(t(Genexpert_otu_df), method = "bray", na.rm = TRUE)
beta[is.na(beta)] <- 0
pcoaE <- cmdscale(beta, k = 2)
pcoaE <- as.data.frame(pcoaE)
metadata_beta <- cbind(Genexpert_metadata_df, pcoaE)
Group_metadata_beta <- metadata_beta
centroids <- aggregate(cbind(V1, V2) ~ Group, metadata_beta, mean)

# Compute Bray-Curtis dissimilarity matrix and perform PCoA for Mtb Infection status
beta <- vegan::vegdist(t(MtbInfectionStatus_otu_df), method = "bray", na.rm = TRUE) # Compute Bray-Cur
beta[is.na(beta)] <- 0 # Replace any NA values with 0
pcoaE <- cmdscale(beta, k = 2) # Perform PCoA, reducing to 2 dimensions
pcoaE <- as.data.frame(pcoaE) # Convert PCoA result to a data frame
metadata_beta <- cbind(MtbInfectionStatus_metadata_df, pcoaE) # Combine metadata with PCoA results
centroids <- aggregate(cbind(V1, V2) ~ Group, metadata_beta, mean) # Calculate centroids for each group
```

```

# Plot PCoA with points colored by Group
metadata_beta %>%
  ggplot(aes(x = V1, y = V2, color = Group)) + # PCoA plot by M.tb infection status
  theme_classic() +
  scale_color_manual(values = pal.Coll) + # Set custom colors for groups
  geom_point(aes(color = Group), size = 5, alpha = 0.8) + # Plot points with transparency
  xlab("PCo 1") + ylab("PCo 2") + # Label axes
  theme(axis.title.x = element_text(size = 13),
        axis.text.x = element_text(size = 13),
        axis.text.y = element_text(size = 13),
        axis.title.y = element_text(size = 13),
        legend.position = "bottom", # Position the legend on the bottom
        legend.title = element_text(size = 12), # Set title size for legend
        legend.text = element_text(size = 10)) + # Set text size for legend
  labs(color = "M.tuberculosis Infection Status") + # Set legend title
  stat_ellipse(aes(color = Group)) + # Add ellipses to show group clusters
  geom_point(data = centroids, size = 5, shape = 16, color = "black") + # Add centroids in black
  geom_point(data = centroids, size = 4, shape = 16) -> B # Overlay centroids with group color

# Add density plot for V1 (first PCoA axis)
xdensity <- metadata_beta %>%
  ggplot(aes(x = V1)) +
  geom_density(alpha = 0.5, aes(fill = Group, color = Group)) + # Create density plot
  scale_fill_manual(values = pal.Coll) + # Set manual fill colors
  scale_color_manual(values = pal.Coll) + # Set manual line colors
  theme_classic() + # Apply classic theme for a clean look
  theme(
    axis.title.x = element_blank(), # Remove x-axis title
    axis.text.x = element_blank(), # Remove x-axis text
    axis.ticks.x = element_blank(), # Remove x-axis ticks
    axis.line.x = element_blank(), # Remove x-axis line
    axis.text.y = element_blank(), # Remove y-axis text
    axis.ticks.y = element_blank(), # Remove y-axis ticks
    legend.position = "none" # Hide legend
  )

# Add density plot for V2 (second PCoA axis)
ydensity <- metadata_beta %>%
  ggplot(aes(V2)) +
  geom_density(alpha = 0.5, aes(fill = Group, color = Group)) + # Create density plot for V2 colored b
  scale_fill_manual(values = pal.Coll) + # Set manual fill colors
  scale_color_manual(values = pal.Coll) + # Set manual line colors
  theme_classic() + # Apply classic theme for a clean look
  theme(
    axis.text.x = element_blank(), # Remove x-axis text
    axis.ticks.x = element_blank(), # Remove x-axis ticks
    axis.title.y = element_blank(), # Remove y-axis title
    axis.text.y = element_blank(), # Remove y-axis text
    axis.ticks.y = element_blank(), # Remove y-axis ticks
    axis.line.y = element_blank(), # Remove y-axis line
    legend.position = "none" # Hide legend
  ) +
  coord_flip() # Flip coordinates for a vertical plot

```



```

# Create a blank plot for spacing
blankPlot <- ggplot() + geom_blank(aes(1,1)) + theme_void()

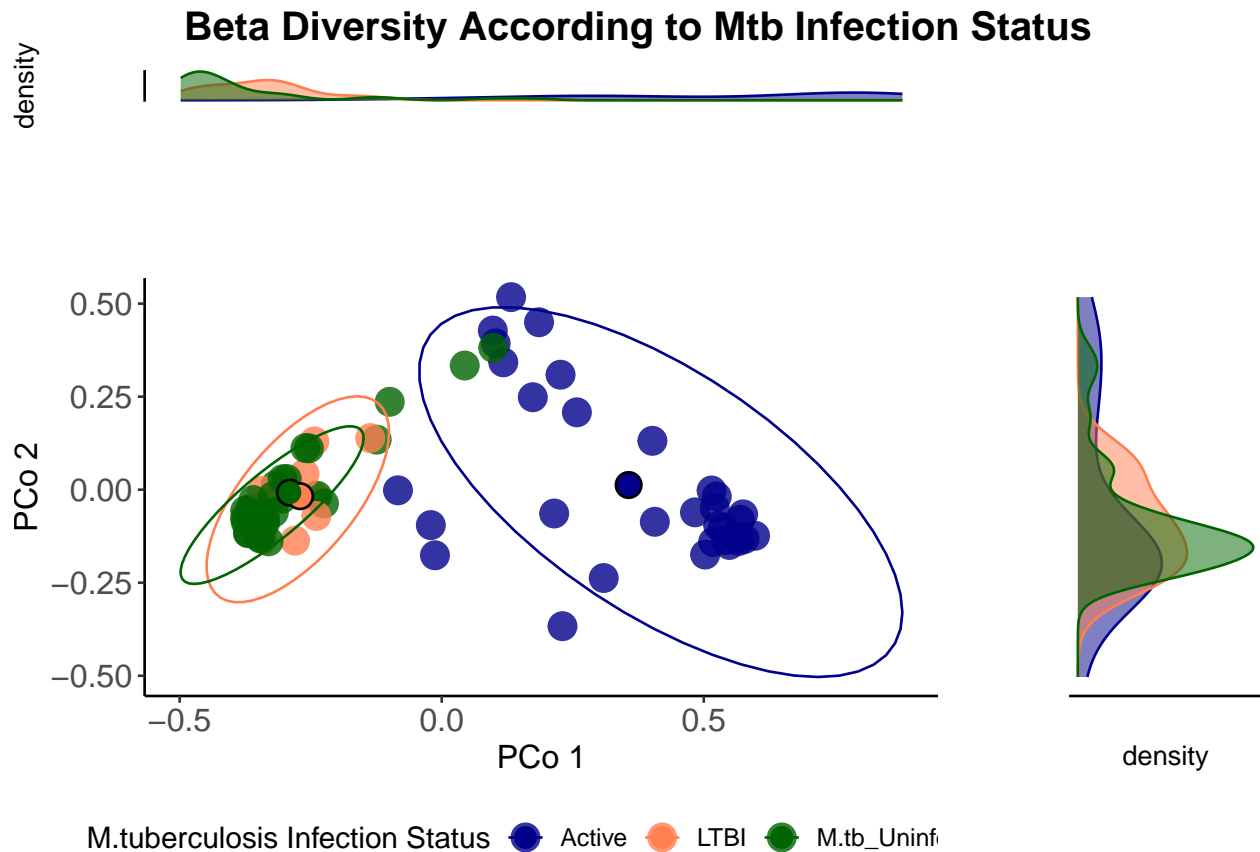
# Create a combined plot with legend at the bottom
beta_final <- plot_grid(
  xdensity + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")), # Add x-density plot with no margins
  blankPlot + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")), # Add a blank plot with no margins
  B + theme(legend.position = "bottom", # Add the main plot B with legend at the bottom
    plot.margin = unit(c(0, 0, 0, 0), "cm")),
  ydensity + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")), # Add y-density plot with no margins
  nrow = 2, # Arrange plots in a 2-row grid
  rel_widths = c(4, 1.4), # Set relative widths for columns
  rel_heights = c(1.4, 4), # Set relative heights for rows
  align = "hv" # Align plots both horizontally and vertically
)

# Load cowplot package
library(cowplot)

# Add a title to the beta_final plot
beta_final <- ggdraw() +
  draw_plot(beta_final, 0, 0, 1, 0.9) + # Adjust the plot size to leave space for the title
  draw_label("Beta Diversity According to Mtb Infection Status",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16, fontface = "bold")

# Display the final plot
print(beta_final)

```



```
# Perform PERMANOVA test to compare beta diversity between 'TB_history' groups
# Compute Bray-Curtis dissimilarity matrix from OTU data
permanova <- adonis2(vegdist(t(MtbInfectionStatus_otu_df), method = "bray", na.rm = TRUE) ~ as.factor(G),
                    permutations = 999)

# Convert PERMANOVA results to a data frame
permanova <- as.data.frame(permanova)

# Save the data frame to a CSV file
write.csv(permanova, file = "Results/Mtb_infection_status_permanova_results.csv", row.names = TRUE)

ggsave(filename = "Results/Figure 2.jpg",
        plot = beta_final,
        width = 10, height = 8, dpi = 600)
```

DMM

The models help identify underlying community structures by partitioning the data into distinct groups based on microbial abundance profiles. DMM utilizes an Infinite mixture model; hence, it can infer the optimal number of microbiome microbial community types.

```
# Read in the Genus abundance table
genus <- read.table("data/Genus.txt", header = TRUE, row.names = NULL, sep = "\t")
```

```

# Assign tax column as row names and remove the tax column from the data frame
rownames(genus) <- genus$Genus
genus <- subset(genus, select = -Genus)

# Ensure row names are unique
unique_row_names <- make.names(row.names(genus), unique = TRUE)
rownames(genus) <- unique_row_names

# Create a copy of genus called raw
raw <- genus

# Transpose the data frame and remove columns with zero sums
raw_values <- t(raw) # Transpose the data frame
raw_values <- raw_values[, colSums(raw_values) > 0] # Remove columns with zero sums
raw_values_t <- t(raw_values) # Transpose back to the original orientation

# Note that genus_table has samples as columns and genera as rows
genus_table <- as.data.frame(raw_values_t)

# Create a data frame with the row names of genus_table
genus_p <- as.data.frame(row.names(genus_table))

# Rename the column to 'long_names'
colnames(genus_p) <- "long_names"

# Separate the long names into multiple columns based on the dot separator
tax_names <- tidyr::separate(
  genus_p,
  long_names,
  into = c("A", "B", "C", "D", "E", "F"),
  sep = "\\.",
  fill = "right", # Fill missing values with NA on the right side
  extra = "drop"  # Drop any extra columns beyond what is specified
)

# Add the genus names (last part after the last dot) to genus_table
genus_table$short <- tax_names$F

# Make these genus names unique and set them as row names for genus_table
rownames(genus_table) <- make.names(genus_table$short, unique = TRUE)

# Remove the 'short' column from genus_table since it's now redundant
genus_table$short <- NULL

# Normalize genus_table
genus_table <- genus_table / min(genus_table[genus_table > 0])

# Set the maximum number of Dirichlet components to check
all_dmns <- 6

# Initialize a list to store the Dirichlet models
dmn_list <- vector("list", all_dmns)

```

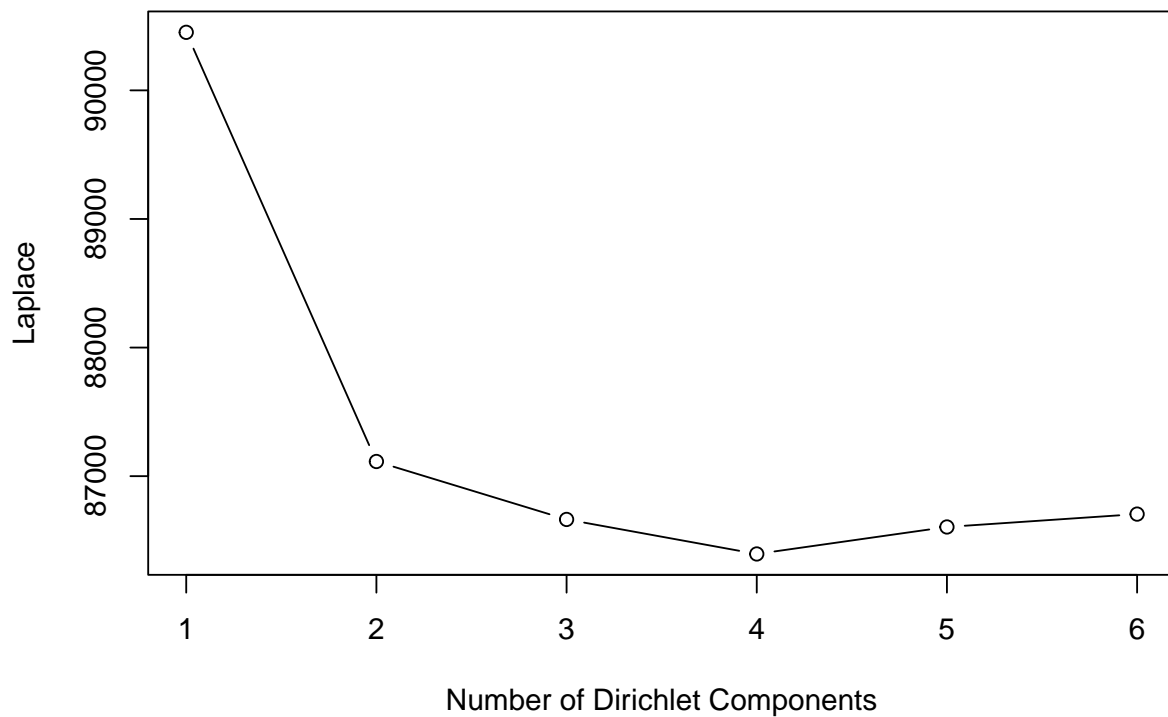
```

# Fit Dirichlet Multinomial models with components ranging from 1 to all_dmns
for (i in 1:all_dmns) {
  print(i) # Print the current number of components being processed
  dmn_list[[i]] <- dmn(as.matrix(t(genus_table)), i, verbose = FALSE) # Fit the model and store it in
}
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 4
#> [1] 5
#> [1] 6

# Calculate Laplace, BIC, and AIC for each Dirichlet Multinomial model
lplc <- sapply(dmn_list, laplace)
BIC <- sapply(dmn_list, BIC)
AIC <- sapply(dmn_list, AIC)

# Plot Laplace, BIC, and AIC values
plot(lplc, type = "b", xlab = "Number of Dirichlet Components", ylab = "Laplace")

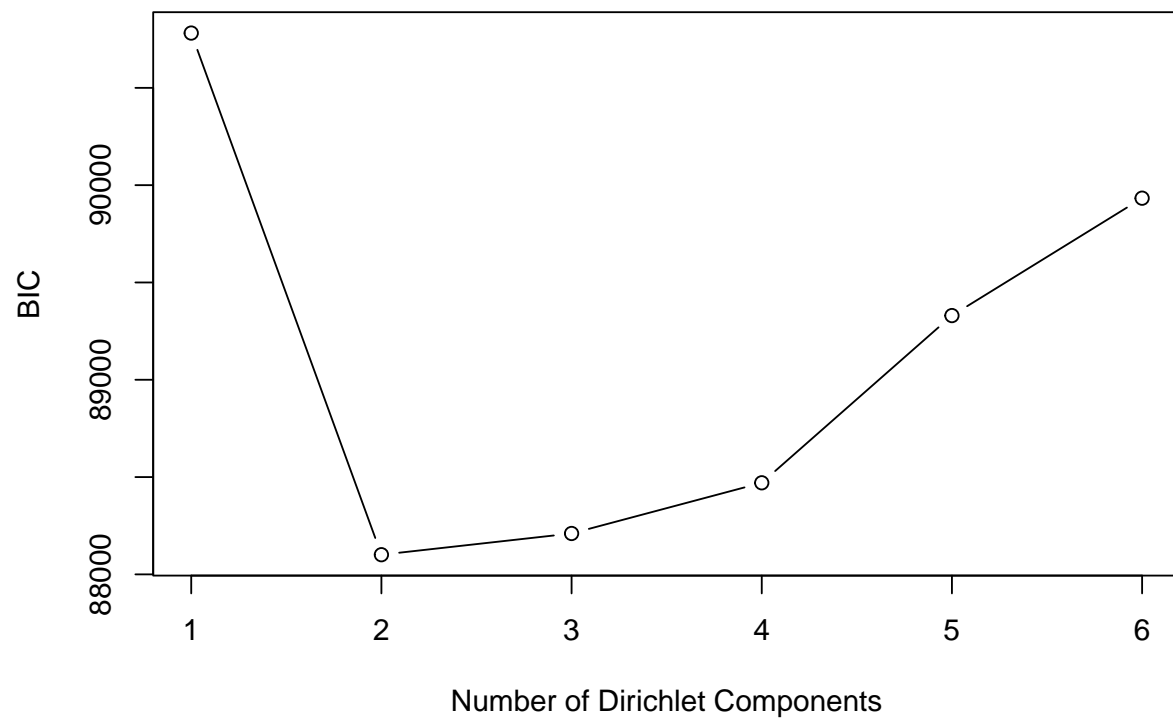
```



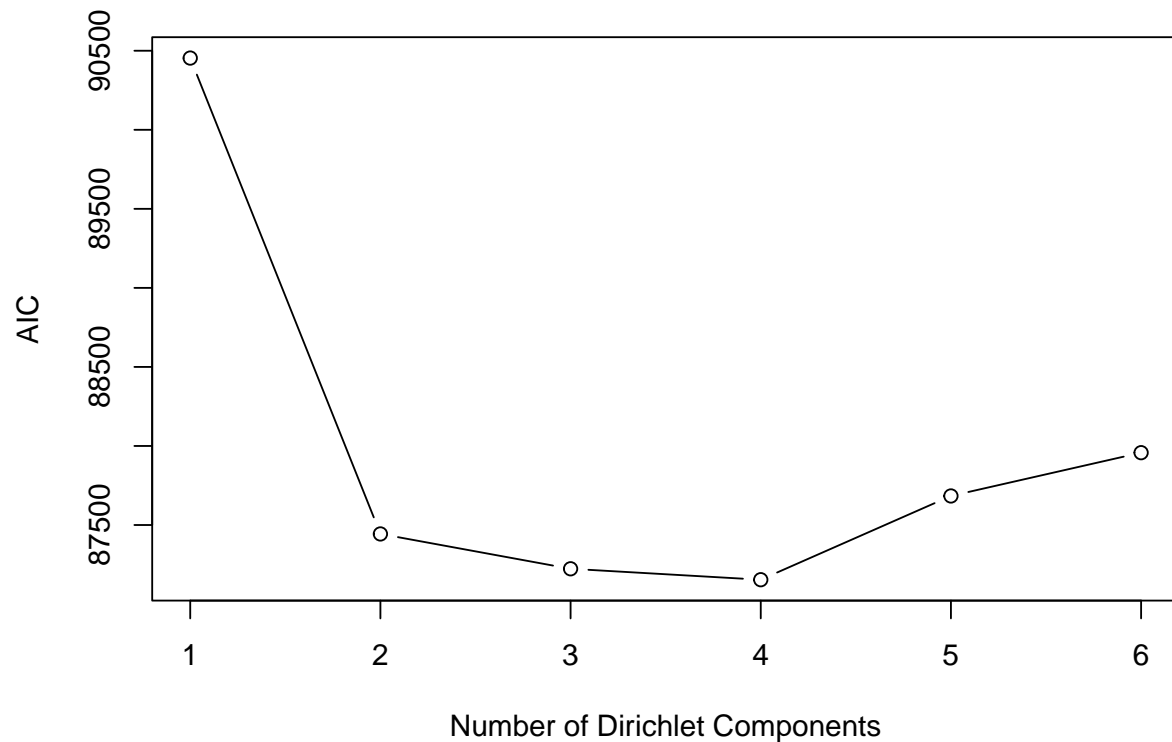
```

plot(BIC, type = "b", xlab = "Number of Dirichlet Components", ylab = "BIC")

```



```
plot(AIC, type = "b", xlab = "Number of Dirichlet Components", ylab = "AIC")
```



```

# Find the model with the minimum Laplace value
best_lplc_index <- which.min(lplc)
optimal_lplc_model <- dmn_list[[best_lplc_index]]
print(optimal_lplc_model)
#> class: DMN
#> k: 4
#> samples x taxa: 193 x 201
#> Laplace: 86394.88 BIC: 88470.49 AIC: 87153.99

# Find the model with the minimum BIC value
best_BIC_index <- which.min(BIC)
optimal_BIC_model <- dmn_list[[best_BIC_index]]
print(optimal_BIC_model)
#> class: DMN
#> k: 2
#> samples x taxa: 193 x 201
#> Laplace: 87114.06 BIC: 88100.98 AIC: 87443.55

# Find the model with the minimum AIC value
best_AIC_index <- which.min(AIC)
optimal_AIC_model <- dmn_list[[best_AIC_index]]
print(optimal_AIC_model)
#> class: DMN
#> k: 4

```

```

#> samples x taxa: 193 x 201
#> Laplace: 86394.88 BIC: 88470.49 AIC: 87153.99

# Assign the best model based on the minimum Laplace value as the final model
best_fit <- dmn_list[[best_lplc_index]]

# Get the cluster importance from the best model
cluster_imp <- fitted(best_fit) # Extract cluster importance

# Print the cluster importance
print(cluster_imp)
#>
#> X
#> X.1
#> X.2
#> Actinomyces
#> F0332
#> Mobiluncus
#> Alloscardovia
#> Bifidobacterium
#> Parascardovia
#> X.3
#> Corynebacterium
#> Mycobacterium
#> Tropheryma
#> Microbacterium
#> X.4
#> Rothia
#> X.5
#> Brooklawnia
#> Cutibacterium
#> Propionibacterium
#> Pseudopropionibacterium
#> Streptomyces
#> Sphaerimonospora
#> Atopobium
#> Olsenella
#> Cryptobacterium
#> DNF00809
#> Slackia
#> X.6
#> X.7
#> X.8
#> Bacteroides
#> Incertae
#> X.9
#> uncultured
#> Odoribacter
#> F0058
#> X.10
#> Porphyromonas
#> X.11
#> Alloprevotella

```

	[,1]	[,2]	[,3]	[,4]
X	1.267400e-02	2.357635e-02	0.0202613135	2.304540e-02
X.1	2.683456e-01	4.244991e-01	0.0493107301	2.834032e-01
X.2	2.182763e-03	7.286351e-03	0.0003917502	8.100961e-03
Actinomyces	7.819756e-01	1.642658e+00	0.1149445656	8.573296e-01
F0332	4.076938e-02	1.049263e-01	0.0043136107	8.494050e-02
Mobiluncus	1.009677e-02	4.170371e-02	0.0043277276	5.523869e-02
Alloscardovia	2.269703e-02	3.168365e-02	0.0122463031	3.868054e-02
Bifidobacterium	2.944869e-02	6.033423e-02	0.0206072476	6.287278e-02
Parascardovia	6.150895e-03	2.152269e-02	0.0003917502	8.213341e-03
X.3	1.424629e-02	1.688153e-02	0.0042968319	3.124187e-02
Corynebacterium	1.953910e-01	3.011177e-01	0.0491959624	1.806634e-01
Mycobacterium	1.955023e-02	2.153095e-02	1.8372815575	2.238604e-02
Tropheryma	6.150048e-03	1.471063e-02	0.0003917502	3.167897e-02
Microbacterium	4.034033e-02	8.569188e-02	0.0941305616	9.761927e-02
X.4	2.020101e-01	2.762384e-01	0.0620451152	1.584810e-01
Rothia	7.134038e-01	1.725998e+00	0.1098956488	5.413802e-01
X.5	4.051851e-02	8.401005e-02	0.0082820690	8.058611e-02
Brooklawnia	2.053154e-04	1.210232e-02	0.0042968319	8.075152e-03
Cutibacterium	2.222626e-02	2.389560e-02	0.1087956005	7.190937e-04
Propionibacterium	4.091491e-02	1.057362e-01	0.0121511718	7.462466e-02
Pseudopropionibacterium	1.101528e-01	2.462316e-01	0.0167620012	3.339437e-01
Streptomyces	2.996799e-02	2.563493e-02	0.0695788484	3.708785e-02
Sphaerimonospora	2.017647e-04	2.347866e-04	0.0303002456	7.190426e-04
Atopobium	2.995944e-01	6.207999e-01	0.0816597052	6.818366e-01
Olsenella	3.283793e-02	1.254349e-01	0.0204407102	1.549504e-01
Cryptobacterium	8.216449e-03	4.904866e-02	0.0082302636	8.764068e-02
DNF00809	1.014780e-02	1.911506e-02	0.0003917502	3.880741e-02
Slackia	2.448157e-02	7.953864e-02	0.0202092086	1.077279e-01
X.6	1.015633e-02	3.116590e-02	0.0003917502	7.942780e-03
X.7	2.182754e-03	7.337787e-03	0.0003917502	7.942701e-03
X.8	1.228832e-02	2.436727e-02	0.0003917502	2.310726e-02
Bacteroides	1.228703e-02	3.766817e-02	0.0003917502	3.813644e-02
Incertae	1.421901e-02	5.884036e-02	0.0003917502	4.637414e-02
X.9	2.169633e-03	3.225032e-02	0.0003917502	1.527265e-02
uncultured	4.166559e-03	3.461586e-02	0.0003917502	2.326688e-02
Odoribacter	2.182847e-03	3.266737e-02	0.0003917502	1.527272e-02
F0058	6.652119e-02	1.359599e-01	0.0082017992	9.647892e-02
X.10	1.615640e-02	1.685260e-02	0.0003917502	7.190879e-04
Porphyromonas	6.186845e-01	2.322410e+00	0.0240513087	4.780305e-01
X.11	5.065499e-01	9.291271e-01	0.0082498011	3.051310e-01
Alloprevotella	9.303453e-01	2.006748e+00	0.0204684336	5.260657e-01

#> Prevotella	7.844490e-01	2.280312e+00	0.0287039253	4.787608e-01
#> Prevotella_7	2.868407e+00	7.429251e+00	0.0506954306	8.158621e-01
#> Prevotellaceae	4.225701e-03	2.627367e-02	0.0003917502	8.000008e-03
#> Rikenellaceae	1.021434e-02	7.446176e-02	0.0003917502	6.136925e-02
#> Tannerella	1.728470e-01	3.048448e-01	0.0042968319	1.629841e-01
#> X2534	6.134412e-03	1.699069e-02	0.0003917502	8.123996e-03
#> Edaphobaculum	4.214320e-03	3.210277e-02	0.0043136225	4.222505e-02
#> Capnocytophaga	2.426140e-01	3.919293e-01	0.0043447340	2.610357e-01
#> Bergeyella	1.392463e-01	1.883802e-01	0.0003917502	1.114460e-01
#> X.12	4.200294e-03	1.994865e-02	0.0003917502	8.000008e-03
#> Lentimicrobium	5.245484e-02	1.093528e-01	0.0003917502	7.269270e-02
#> X.13	4.801453e-02	1.549551e-01	0.0003917502	5.513079e-02
#> Campylobacter	2.772846e-01	4.676766e-01	0.0161902190	2.502370e-01
#> Helicobacter	2.205570e-03	2.588148e-03	0.0003917502	7.190864e-04
#> Wolinella	6.141332e-03	2.180894e-02	0.0003917502	7.190861e-04
#> Flexilinea	2.180251e-04	1.444529e-02	0.0003917502	3.734962e-02
#> X.14	2.869691e-02	5.266948e-02	0.0286405695	4.793201e-02
#> aestivum	1.431187e-02	1.208774e-02	0.0082017992	2.324183e-02
#> Desulfomicrobium	2.193403e-03	1.927082e-02	0.0003917502	8.000008e-03
#> Acholeplasma	6.149082e-03	2.540791e-02	0.0003917502	7.190864e-04
#> Brevibacillus	4.190566e-03	7.320507e-03	0.0670564956	7.206520e-04
#> X.15	6.244768e-03	3.968529e-02	0.0003917502	7.190865e-04
#> Eggerthia	2.182754e-03	4.129549e-02	0.0003917502	3.748428e-02
#> X.16	1.946899e-01	3.477813e-01	0.0956661092	4.628730e-01
#> Bulleidia	1.018616e-02	6.225459e-02	0.0043136107	8.151298e-02
#> Erysipelotrichaceae	8.179205e-03	4.835274e-02	0.0003917502	5.595979e-02
#> X.17	1.387653e-01	2.395884e-01	0.0285296878	1.844611e-01
#> Abiotrophia	1.709094e-01	2.975747e-01	0.0502532116	3.260764e-01
#> Dolosigranulum	1.057562e-02	2.204217e-02	0.0042968319	8.128610e-03
#> Granulicatella	6.397943e-01	1.640976e+00	0.1151750337	1.296229e+00
#> X.18	2.274973e-02	5.884766e-02	0.0043136107	3.865165e-02
#> Lacticaseibacillus	2.017647e-04	2.588148e-03	0.0003917502	8.244471e-03
#> Lactobacillus	1.239291e-02	3.275632e-02	0.0042968319	1.566807e-02
#> Ligilactobacillus	3.227388e-02	4.901425e-02	0.0003917502	2.350252e-02
#> Limosilactobacillus	4.146164e-02	9.984502e-02	0.0043447340	7.158145e-02
#> Weissella	1.429285e-02	1.456931e-02	0.0043136107	2.275588e-02
#> X.19	1.485802e-01	2.316624e-01	0.0199151066	1.900650e-01
#> Streptococcus	1.168997e+01	4.623618e+01	0.2889505555	1.474116e+01
#> Mycoplasma	2.771031e-02	1.255431e-01	0.0003917502	6.395984e-02
#> Paenibacillus	2.017647e-04	2.347866e-04	0.0085223127	7.190965e-04
#> X.20	3.543826e-02	6.096956e-02	0.0003917502	6.182903e-02
#> oral	2.235171e-02	6.185062e-02	0.0003917502	3.869379e-02
#> X.21	1.383786e-01	1.940126e-01	0.0338686006	1.543555e-01
#> Gemella	5.810810e-01	1.364555e+00	0.1280232987	8.100994e-01
#> Staphylococcus	3.340168e-02	4.693917e-02	0.0741133784	8.881471e-02
#> X014	1.811686e-01	3.602658e-01	0.0208947219	2.800263e-01
#> X014.1	1.014003e-01	2.219055e-01	0.0164000235	2.273852e-01
#> group	2.017647e-04	1.220013e-02	0.0003917502	8.100961e-03
#> group.1	1.873074e-02	8.385377e-02	0.0042968367	1.547664e-02
#> Pseudoramibacter	8.124274e-03	3.822935e-02	0.0043220920	6.046650e-02
#> Defluviitaleaceae	1.630488e-02	1.154090e-01	0.0003917502	4.760584e-02
#> X.22	2.719545e-01	4.871638e-01	0.0672721678	3.539757e-01
#> Butyrivibrio	1.098569e-01	1.985416e-01	0.0003917502	1.576057e-01

#> <i>Catonella</i>	1.517480e-01	2.831082e-01	0.0003917502	1.770039e-01
#> <i>Howardella</i>	1.414414e-02	3.389399e-02	0.0003917502	5.438526e-02
#> <i>Johnsonella</i>	9.186975e-02	2.046459e-01	0.0290926896	1.684466e-01
#> <i>Lachnoanaerobaculum</i>	2.082114e-01	3.525172e-01	0.0699704425	2.691617e-01
#> <i>Moryella</i>	2.017647e-04	7.362863e-03	0.0003917502	7.190845e-04
#> <i>Oribacterium</i>	3.479229e-01	6.849567e-01	0.0779289817	5.184949e-01
#> <i>Roseburia</i>	8.135362e-03	4.489493e-02	0.0003917502	8.136908e-03
#> <i>Shuttleworthia</i>	7.245236e-02	1.892598e-01	0.0205079589	1.259956e-01
#> <i>Stomatobaculum</i>	2.477104e-01	4.280138e-01	0.0204388300	2.890799e-01
#> <i>X.23</i>	1.018752e-02	7.352636e-03	0.0003917502	7.190875e-04
#> <i>Eubacterium</i>	2.182754e-03	1.921084e-02	0.0043136107	2.346864e-02
#> <i>Peptococcus</i>	8.205079e-02	1.487257e-01	0.0003917502	1.176197e-01
#> <i>X.24</i>	1.004212e-02	7.776466e-02	0.0043136107	6.630811e-02
#> <i>Anaerovoracaceae</i>	8.058630e-03	8.118847e-02	0.0082605884	5.821499e-02
#> <i>Anaerovoracaceae.1</i>	3.554415e-02	8.558151e-02	0.0003917502	3.909215e-02
#> <i>Anaerovoracaceae.2</i>	2.017647e-04	4.949499e-03	0.0003917502	3.245332e-02
#> <i>Anaerovoracaceae.3</i>	2.424639e-02	9.675761e-02	0.0003917502	5.561159e-02
#> <i>Anaerovoracaceae.4</i>	9.292569e-02	1.879484e-01	0.0375881174	2.233484e-01
#> <i>Anaerovoracaceae.5</i>	4.432561e-02	1.242537e-01	0.0291489443	8.816024e-02
#> <i>Anaerovoracaceae.6</i>	2.308478e-01	4.245990e-01	0.0665916552	2.900216e-01
#> <i>Anaerovoracaceae.7</i>	1.424556e-02	1.100016e-01	0.0082336948	7.885789e-02
#> <i>Family</i>	2.017647e-04	1.203752e-02	0.0003917502	2.254526e-02
#> <i>Family.1</i>	2.187084e-03	2.716673e-02	0.0043136107	2.379066e-02
#> <i>Family.2</i>	1.510513e-01	4.233029e-01	0.0506596415	4.355458e-01
#> <i>Family.3</i>	6.179164e-03	4.191365e-02	0.0003917502	3.090384e-02
#> <i>Peptostreptococcaceae</i>	1.815468e-02	4.675017e-02	0.0003917502	2.992416e-02
#> <i>Peptostreptococcaceae.1</i>	5.640447e-02	1.749777e-01	0.0162721852	1.188730e-01
#> <i>Peptostreptococcaceae.2</i>	1.426687e-02	6.507982e-02	0.0003917502	4.562403e-02
#> <i>Peptostreptococcaceae.3</i>	1.843750e-01	3.944693e-01	0.0599717966	2.638948e-01
#> <i>Peptostreptococcaceae.4</i>	2.822453e-02	3.646838e-02	0.0003917502	2.310365e-02
#> <i>X.25</i>	1.842281e-02	4.911956e-02	0.0003917502	3.708785e-02
#> <i>Selenomonadaceae</i>	1.313265e-01	2.398495e-01	0.0043319840	1.748935e-01
#> <i>Selenomonadaceae.1</i>	2.464898e-01	4.621172e-01	0.0201047506	3.697639e-01
#> <i>Veillonellaceae</i>	5.937262e-02	1.473246e-01	0.0082017992	8.790478e-02
#> <i>Veillonellaceae.1</i>	3.994187e-02	1.286191e-01	0.0042968319	7.547675e-02
#> <i>Veillonellaceae.2</i>	1.458523e-01	2.708408e-01	0.0082605884	1.474717e-01
#> <i>Veillonellaceae.3</i>	3.126204e-01	5.435977e-01	0.0609892354	3.020251e-01
#> <i>Veillonellaceae.4</i>	2.024519e+00	4.498941e+00	0.1206327716	9.553768e-01
#> <i>Pelospora</i>	2.017647e-04	2.597295e-03	0.0003917502	8.119003e-03
#> <i>X.26</i>	1.615572e-02	6.231817e-02	0.0003917502	3.708785e-02
#> <i>Fusobacterium</i>	1.137125e+00	4.568576e+00	0.0449350273	6.931650e-01
#> <i>X.27</i>	6.784229e-01	1.621495e+00	0.1145091733	4.840957e-01
#> <i>Leptotrichia</i>	5.810622e-01	1.537388e+00	0.0786619156	5.489399e-01
#> <i>Streptobacillus</i>	1.285513e-01	2.287935e-01	0.0003917502	1.205521e-01
#> <i>SR1</i>	1.224582e-01	2.378592e-01	0.0003917502	1.845238e-01
#> <i>SR1.1</i>	1.083211e-01	1.894571e-01	0.0081981663	1.759436e-01
#> <i>oral.1</i>	2.399430e-02	4.713465e-02	0.0003917502	3.101454e-02
#> <i>P22</i>	1.214238e-02	4.988665e-03	0.0003917502	2.262660e-02
#> <i>X.28</i>	1.789054e-01	3.987653e-01	0.0341767455	4.460172e-01
#> <i>Saccharibacteria</i>	5.343908e-02	2.123049e-01	0.0124261782	2.247798e-01
#> <i>X.29</i>	3.756442e-02	1.422879e-01	0.0082929404	1.845060e-01
#> <i>Candidatus</i>	1.834348e-01	4.498723e-01	0.0295310230	3.627042e-01
#> <i>TM7a</i>	4.961472e-02	1.227443e-01	0.0043406878	1.501298e-01

#> TM7x	3.249440e-01	7.804067e-01	0.0732851498	6.935221e-01
#> uncultured.1	9.637833e-02	2.386589e-01	0.0083506662	2.425973e-01
#> Candidatus.1	1.414927e-02	4.572869e-02	0.0043277276	6.649950e-02
#> X.30	4.166645e-03	2.670659e-02	0.0003917502	2.262660e-02
#> X.31	4.167739e-03	4.712241e-02	0.0003917502	6.755952e-02
#> uncultured.2	2.195120e-03	3.162142e-02	0.0043462689	1.289841e-01
#> X.32	2.399257e-02	7.319264e-02	0.0283005892	2.084341e-01
#> Brevundimonas	2.359473e-02	6.243446e-02	0.0201736949	1.310589e-01
#> Caulobacter	2.106665e-02	4.023880e-02	0.0043218513	8.146059e-02
#> Reyranella	6.150038e-03	4.937918e-03	0.0281524457	7.191023e-04
#> X.33	6.173840e-03	3.704054e-02	0.0126820101	1.146006e-01
#> Bosea	4.182360e-03	2.434099e-02	0.0251104253	1.212434e-01
#> X.34	2.017647e-04	1.458019e-02	0.0003917502	3.879172e-02
#> Allorhizobium	5.653391e-02	1.375329e-01	0.0207043694	9.753402e-01
#> Mesorhizobium	1.845848e-02	5.273662e-02	0.0243402999	8.186311e-02
#> Bradyrhizobium	1.225847e-02	4.522319e-02	0.0278304772	7.528957e-02
#> X.35	1.816910e-02	1.198186e-02	0.0003917502	7.942701e-03
#> Sphingobium	3.909581e-02	8.687261e-02	0.0253186550	7.036068e-01
#> Sphingomonas	3.976850e-02	1.481282e-01	0.0245733659	3.256575e-01
#> Lautropia	1.643385e-01	2.400694e-01	0.0082605884	1.245722e-01
#> Ralstonia	2.456303e-02	5.096656e-02	0.0862578369	8.521433e-02
#> X.36	4.165715e-03	7.329145e-03	0.0484677502	7.190990e-04
#> Aquabacterium	2.240786e-03	1.916808e-02	0.0240038493	7.942701e-03
#> Comamonas	2.523008e-02	4.163742e-02	0.0003917502	2.335820e-02
#> Methylibium	1.027232e-02	7.215866e-03	0.0796459471	7.191970e-04
#> X.37	1.371187e-01	2.264467e-01	0.0042968319	1.057284e-01
#> Alysella	2.655848e-02	3.150848e-02	0.0003917502	7.190933e-04
#> Eikenella	6.092618e-02	8.825077e-02	0.0003917502	3.002895e-02
#> Kingella	4.888491e-02	1.126394e-01	0.0042968319	3.809172e-02
#> Neisseria	1.238717e+00	2.667701e+00	0.0509363422	5.599892e-01
#> Methyloversatilis	1.438687e-02	2.474984e-02	0.0083257161	4.698872e-02
#> Propionivibrio	4.153684e-03	7.341543e-03	0.0003917502	1.516403e-02
#> Cardiobacterium	4.358908e-02	9.217205e-02	0.0003917502	6.867813e-02
#> Enterobacter	1.464622e-02	2.203311e-02	0.0359406635	2.992416e-02
#> Escherichia	1.884007e-02	3.796622e-02	0.0399675422	4.017195e-02
#> Kosakonia	4.225716e-03	4.983658e-03	0.0121511725	8.093518e-03
#> Providencia	2.217485e-03	2.347866e-04	0.0003917502	7.190854e-04
#> Actinobacillus	1.436583e-01	1.769883e-01	0.0043136107	1.004448e-01
#> Aggregatibacter	2.904645e-01	4.406440e-01	0.0003917502	1.614565e-01
#> Haemophilus	1.110332e+00	1.629140e+00	0.0408973444	4.668234e-01
#> Acinetobacter	4.191266e-03	2.347865e-04	0.0617765593	7.190790e-04
#> Faucicola	2.691701e-02	2.859430e-02	0.0003917502	1.555390e-02
#> Moraxella	4.319993e-02	9.474747e-02	0.0043220673	4.088512e-02
#> Pseudomonas	1.280050e-02	1.435684e-02	0.0406520202	3.024386e-02
#> Stenotrophomonas	8.217457e-02	1.841372e-01	0.0291839448	3.163592e+01
#> X.38	3.711758e-02	1.575967e-01	0.0003917502	5.588069e-02
#> X.39	2.017647e-04	9.742732e-03	0.0003917502	7.190869e-04
#> Treponema	1.124269e-01	2.900500e-01	0.0082498011	1.687430e-01
#> Fretibacterium	4.149872e-02	1.179619e-01	0.0003917502	9.352883e-02
#> Pyramidobacter	2.214840e-03	2.633419e-02	0.0003917502	3.059968e-02
#> X.40	2.182754e-03	7.303402e-03	0.0042965807	2.981697e-02
#> Dioszegia	2.182754e-03	7.303402e-03	0.0003917502	7.190865e-04
#> Vishniacozyma	8.165497e-03	1.685132e-02	0.0160752402	1.538282e-02

```

#> NA. 3.587155e-02 8.205677e-02 0.1548986919 3.974514e-01

# Extract fitted values from the models
p1 <- fitted(dmn_list[[1]], scale = TRUE) # Fitted values from the first model
p5 <- fitted(best_fit, scale = TRUE) # Fitted values from the best model

# Compute the mean difference
meandiff <- colSums(abs(p5 - as.vector(p1)))

# Display the mean difference
print(meandiff)
#> [1] 0.4344801 0.6288452 0.9956168 0.8635993

# Extract the fitted values from the best model
x <- mixture(best_fit)

# Initialize an empty list to store plots
plot_list <- vector("list", ncol(fitted(best_fit)))

# Loop through each cluster
for (k in seq(ncol(fitted(best_fit)))) {
  # Melt the fitted values to long format
  d <- melt(fitted(best_fit))
  colnames(d) <- c("OTU", "cluster", "value") # Rename columns
  # Filter and process data for the current cluster
  d <- subset(d, cluster == k) %>%
    arrange(value) %>%
    mutate(OTU = factor(OTU, levels = unique(OTU))) %>%
    filter(abs(value) > quantile(abs(value), 0.8)) # Keep top 20% most significant OTUs

  # Define a function to generate color hues
  gg_color_hue <- function(n) {
    hues = seq(15, 375, length = n + 1)
    hcl(h = hues, l = 65, c = 100)[1:n]
  }

  # Get colors for the bars
  cols = gg_color_hue(ncol(fitted(best_fit)))

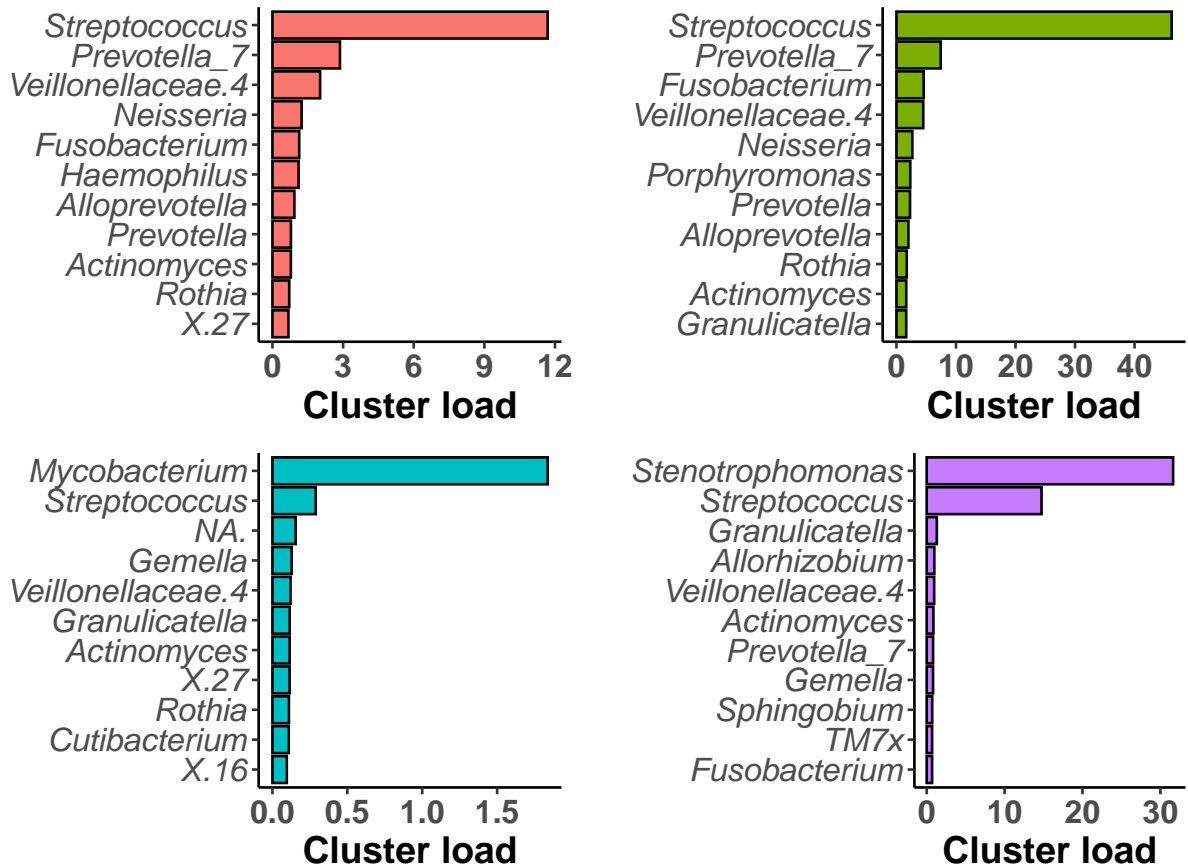
  # Create the plot for the current cluster
  p <- ggplot(d[(length(d$value) - 10):length(d$value), ],
    aes(x = OTU, y = value)) +
    xlab("") + # Remove x-axis label
    ylab("Cluster load") + # Label y-axis
    geom_bar(stat = "identity", fill = cols[k], colour = "black") + # Bar plot with color and border
    coord_flip() + # Flip coordinates for horizontal bars
    theme_classic() + # Apply classic theme
    theme(axis.text.y = element_text(face = "italic"), # Style y-axis text
      axis.text = element_text(size = 13, face = 'bold'), # Style axis text
      axis.title = element_text(size = 15, face = 'bold')) # Style axis titles

  # Store the plot in the list
  plot_list[[k]] <- p
}

```

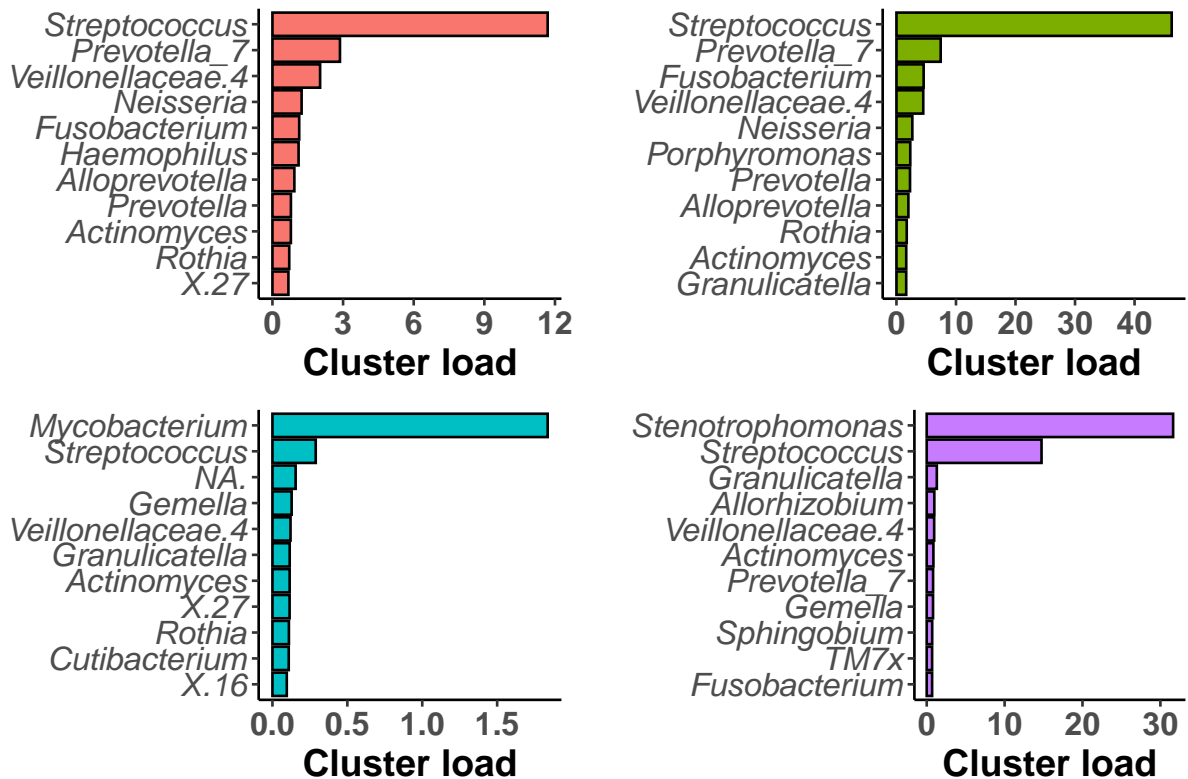
```
}
```

```
# Combine and display all plots in plot_list
combined_plot<- grid.arrange(grobs = plot_list, ncol = 2)
```



```
# Create the combined plot with cowplot and add a title
combined_plot <- ggdraw() +
  draw_plot(combined_plot, 0, 0, 1, 0.9) + # Adjust plot size to leave space for title
  draw_label("DMM Microbial Communities/Pulmotypes",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16, fontface = "bold")
print(combined_plot)
```

DMM Microbial Communities/Pulmotypes



```
ggsave("Results/Figure 3.jpg", plot = combined_plot, dpi = 600, width = 12, height = 8)
```

Cofounding factors between microbiome alpha diversity metrics and the different DMM microbial communities

```
alpha_features <- alpha_metrics_df_Group

meta_alpha_decon <- Genexpert_metadata_df

# Fit Dirichlet Multinomial models with components ranging from 1 to all_dmns
dmn_list <- lapply(1:all_dmns, function(k) dmn(as.matrix(t(genus_table)), k, verbose = FALSE))

# Determine the best model (based on Laplace)
best_fit_index <- best_lplc_index
best_fit <- dmn_list[[best_fit_index]]

# Assign clusters for the best fit Dirichlet model
cluster_result <- as.data.frame(mixture(best_fit, assign = TRUE))
cluster_col_name <- paste0("Dirichlet_multinomial_k=", best_fit_index)
colnames(cluster_result) <- cluster_col_name
```

```

# Add the cluster results to meta_alpha_decon
meta_alpha_decon <- cbind(meta_alpha_decon, cluster_result)

# Update cluster labels to descriptive names
cluster_levels <- unique(meta_alpha_decon[[cluster_col_name]])
cluster_labels <- paste("Type", cluster_levels)
meta_alpha_decon[[cluster_col_name]] <- factor(
  meta_alpha_decon[[cluster_col_name]],
  levels = cluster_levels,
  labels = cluster_labels
)

# Add a new column 'smpl' with row names
meta_alpha_decon$smpl <- row.names(meta_alpha_decon)

# Create dummy columns for the selected cluster column
meta_alpha_decon <- fastDummies::dummy_cols(.data = meta_alpha_decon, select_columns = cluster_col_name)

# Set the row names of 'meta_alpha_decon' to the 'smpl' column and clean up
row.names(meta_alpha_decon) <- meta_alpha_decon$smpl
meta_alpha_decon$smpl <- NULL
meta_alpha_decon[[cluster_col_name]] <- NULL

# Order the dataframes by row names
alpha_features <- alpha_features[order(rownames(alpha_features)), ]
meta_alpha_decon <- meta_alpha_decon[order(rownames(meta_alpha_decon)), ]
# Recode the values in the Group column in both dataframes
meta_alpha_decon$Group[meta_alpha_decon$Group == "Xpert-ve"] <- 0
meta_alpha_decon$Group[meta_alpha_decon$Group == "Xpert+ve"] <- 1

# Replace values in Group column
alpha_features$Group <- ifelse(alpha_features$Group == "Xpert-ve", 0,
                              ifelse(alpha_features$Group == "Xpert+ve", 1, alpha_features$Group))

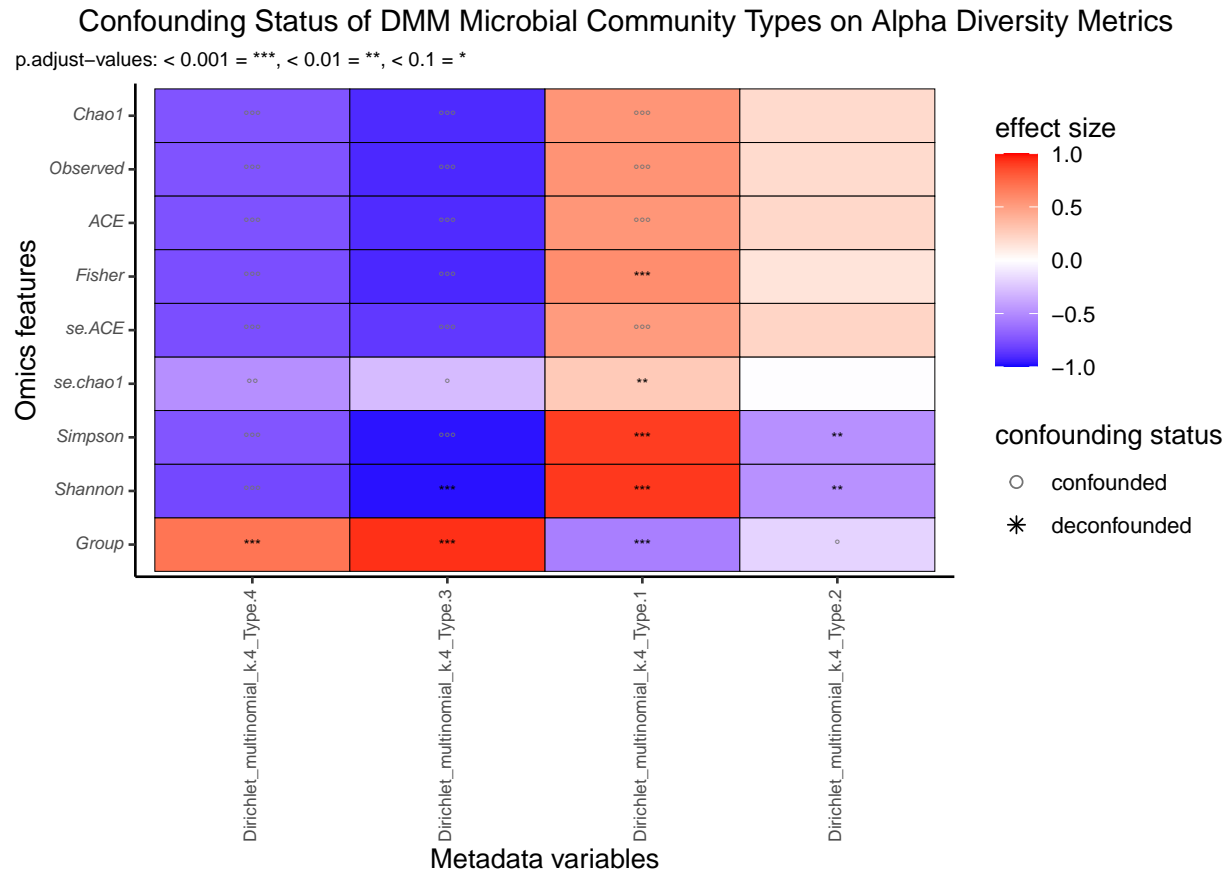
#Run metadecofound function
meta_alpha_out <- metadecofoundR::MetaDeconfound(
  featureMat = alpha_features,
  metaMat = meta_alpha_decon,
  nnodes = 4, # Number of nodes to use for parallel processing
  logfile = here::here("MetadecofoundR_feature_alpha.log") # Path to the log file
)

#Heatmap
alpha_metaDR <- metadecofoundR::BuildHeatmap(
  meta_alpha_out,
  d_col = c("blue", "white", "red"), # Color scheme for the heatmap
  d_range = "full" # Use the full range of the data for color scaling
)

alpha_metaDR1 <- alpha_metaDR +
  theme(axis.text.y = element_text(face = "italic"))
# Add a title to the alpha_metaDR1 plot
alpha_metaDR1 <- alpha_metaDR1 +
  ggtitle("Confounding Status of DMM Microbial Community Types on Alpha Diversity Metrics") +
  theme(plot.title = element_text(hjust = 0.5, size = 12)) # Center and size the title

```

```
print(alpha_metaDR1)
```



```
ggsave("Results/Figure 4.jpg",alpha_metaDR1, width = 8, height =8,limitsize = FALSE, dpi=600)
```

Beta diversity of the different DMM microbial community types/pulmotypes

```
# Create cluster results and combine with metadata

# Add the cluster results to metadata_beta
metadata_beta <- cbind(Group_metadata_beta, cluster_result)
metadata_beta[[cluster_col_name]] <- as.character(metadata_beta[[cluster_col_name]])

# Calculate centroids for the clusters
centroids <- aggregate(cbind(V1, V2) ~ ., data = metadata_beta, FUN = mean)

# Define custom colors for the plot
ComCol <- c("chartreuse3", "deepskyblue2", "red3", "purple", "orange", "violet")

# Create the main plot with points and ellipses
Bcluster <- metadata_beta %>%
```



```

ggplot(aes(x = V1, y = V2, color = .data[[cluster_col_name]])) +
  theme_classic() +
  scale_color_manual(values = ComCol) +
  geom_point(size = 5, alpha = 0.8) +
  xlab("PCo 1") + ylab("PCo 2") +
  labs(color = "Microbial Community types") +
  theme(axis.title.x = element_text(size = 13),
        axis.text.x = element_text(size = 13),
        axis.text.y = element_text(size = 13),
        axis.title.y = element_text(size = 13),
        legend.position = "bottom") +
  stat_ellipse() +
  geom_point(data = centroids, size = 5, shape = 16, color = "black") +
  geom_point(data = centroids, size = 4, shape = 21, fill = "white")

#Create the x-axis density plot
xdensity <- metadata_beta %>%
  ggplot(aes(x = V1, fill = .data[[cluster_col_name]], color = .data[[cluster_col_name]])) +
  geom_density(alpha = .5) +
  scale_fill_manual(values = ComCol) +
  scale_color_manual(values = ComCol) +
  theme_classic() +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.line.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none")

#Create the y-axis density plot with flipped coordinates
ydensity <- metadata_beta %>%
  ggplot(aes(x = V2, fill = .data[[cluster_col_name]], color = .data[[cluster_col_name]])) +
  geom_density(alpha = .5) +
  scale_fill_manual(values = ComCol) +
  scale_color_manual(values = ComCol) +
  theme_classic() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.line.y = element_blank(),
        legend.position = "none") +
  coord_flip()

#Create a blank plot for layout purposes
blankPlot <- ggplot() +
  geom_blank(aes(1, 1)) +
  theme_void()

#Combine the plots into a single grid layout
Beta_diversity <-

```

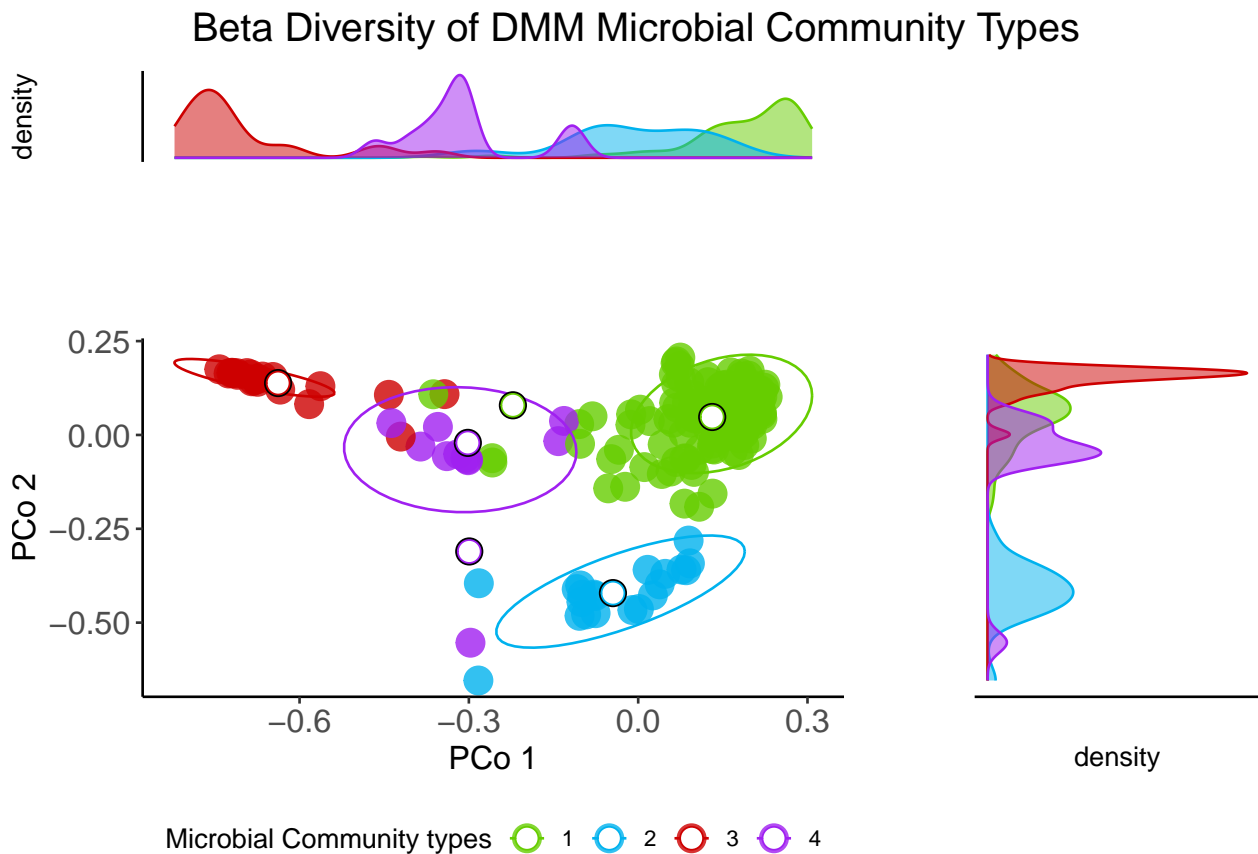


```

cowplot::plot_grid(
  xdensity + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")),
  blankPlot + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")),
  Bcluster + theme(legend.position = "bottom",
    plot.margin = unit(c(0, 0, 0, 0), "cm")),
  ydensity + theme(plot.margin = unit(c(0, 0, 0, 0), "cm")),
  nrow = 2,
  rel_widths = c(1, 0.5),
  rel_heights = c(0.5, 1),
  align = "hv"
)
# Add a title to the Beta_diversity plot
Beta_diversity <- cowplot::ggdraw() +
  cowplot::draw_plot(Beta_diversity, 0, 0, 1, 0.9) + # Adjust plot position and size
  cowplot::draw_label("Beta Diversity of DMM Microbial Community Types",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16)

# Display the combined plot
print(Beta_diversity)

```



```

ggsave("Results/Figure 5.jpg", Beta_diversity, width = 8, height = 8, limitsize = FALSE, dpi = 600)

```

Relative abundance

Phylum

```
# Custom theme for plots
custom_theme <- ggplot2::theme_bw() +
  ggplot2::theme(
    panel.background = ggplot2::element_blank(), # Remove background color
    panel.grid = ggplot2::element_blank(), # Remove grid lines
    legend.position = "right", # Position legend on the right
    legend.title = ggplot2::element_blank(), # Remove legend title
    legend.background = ggplot2::element_blank(), # Remove legend background
    legend.key = ggplot2::element_blank(), # Remove legend key background
    plot.title = ggplot2::element_text(vjust = -8.5, hjust = 0.1), # Adjust plot title position
    axis.title.y = ggplot2::element_text(colour = "black"), # Set Y axis title color
    axis.text = ggplot2::element_text() # Set axis text style
  )

# Color palette for plots
color_set <- c(RColorBrewer::brewer.pal(11, "Set1"), RColorBrewer::brewer.pal(9, "Pastel1"))

# Assign the phyloseq object to a variable
phyloseq_object <- MtbInfectionStatus_phyloseq
group_col <- "Group" # Define the group column name
rank <- "Phylum" # Define the taxonomic rank
add_labels <- TRUE # Flag to add labels
show_sd <- FALSE # Flag to show standard deviation
top_n <- 10 # Number of top taxa to include
transform_abundance <- TRUE # Flag to transform data to relative abundance

# Get unique levels for group column
axis_order <- phyloseq::sample_data(phyloseq_object)$Group %>% unique()

# Perform taxonomic glomming (grouping)
phyloseq_data <- ggClusterNet::tax_glom_wt(ps = phyloseq_object, ranks = rank)

# Transform to relative abundance if specified
if (transform_abundance == TRUE) {
  phyloseq_data <- phyloseq_data %>%
    phyloseq::transform_sample_counts(function(x) {x / sum(x)})
}

# Extract OTU and taxonomy tables
otu_table <- phyloseq::otu_table(phyloseq_data)
tax_table <- phyloseq::tax_table(phyloseq_data)

# Adjust taxonomy table to group non-top taxa as "others"
for (i in 1:dim(tax_table)[1]) {
  if (row.names(tax_table)[i] %in% names(sort(rowSums(otu_table), decreasing = TRUE)[1:top_n])) {
    tax_table[i, rank] <- tax_table[i, rank]
  } else {
    tax_table[i, rank] <- "others"
  }
}

# Update the phyloseq object with the modified taxonomy table
```

```

phyloseq::tax_table(phyloseq_data) <- tax_table

# Melt the phyloseq object to a data frame
taxonomies <- phyloseq_data %>%
  phyloseq::psmelt()

# Convert abundance to percentage
taxonomies$Abundance <- taxonomies$Abundance * 100
# Rename taxonomy column to "taxonomy"
colnames(taxonomies) <- gsub(rank, "taxonomy", colnames(taxonomies))

# Initialize an empty data frame
final_data <- c()
# Loop over each group to calculate relative abundance
for (i in 1:length(unique(phyloseq::sample_data(phyloseq_object)$Group))) {
  group_name <- as.data.frame(table(phyloseq::sample_data(phyloseq_object)$Group))[i, 1]
  group_size <- as.data.frame(table(phyloseq::sample_data(phyloseq_object)$Group))[i, 2]

  # Filter data by group
  group_data <- taxonomies %>%
    dplyr::filter(Group == group_name)
  group_data$Abundance <- group_data$Abundance / group_size

  # Create a temporary data frame for the current group
  temp_data <- data.frame(Sample = group_data$Sample,
                          Abundance = group_data$Abundance,
                          taxonomy = group_data$taxonomy,
                          Group = group_data$Group)

  # Combine data for all groups
  if (i == 1) {
    final_data <- temp_data
  } else {
    final_data <- rbind(final_data, temp_data)
  }
}

# Update taxonomies data frame with combined data
taxonomies <- final_data

# Group data by taxonomy and group columns
grouped_taxa <- dplyr::group_by(taxonomies, taxonomy, Group)
# Summarize abundance and standard deviation by group and taxonomy
summarized_data <- dplyr::summarize(grouped_taxa, sum(Abundance), sd(Abundance))

# Group data by taxonomy to calculate total abundance
taxonomy_groups <- dplyr::group_by(taxonomies, taxonomy)
total_abundance <- dplyr::summarize(taxonomy_groups, sum(Abundance))
head(total_abundance)
#> # A tibble: 6 x 2
#>   taxonomy      `sum(Abundance)`
#>   <chr>                <dbl>
#> 1 Actinobacteriota      84.3

```

```

#> 2 Bacteroidota 35.2
#> 3 Campylobacterota 0.272
#> 4 Firmicutes 120.
#> 5 Fusobacteriota 17.9
#> 6 Patescibacteria 2.19
# Rename columns for total abundance data
colnames(total_abundance) <- c("taxonomy", "total_sum")
# Arrange taxa by total abundance in descending order
total_abundance <- dplyr::arrange(total_abundance, desc(total_sum))

# Preview the summarized data
head(summarized_data)
#> # A tibble: 6 x 4
#> # Groups: taxonomy [2]
#> taxonomy Group `sum(Abundance)` `sd(Abundance)`
#> <chr> <chr> <dbl> <dbl>
#> 1 Actinobacteriota Active 73.5 1.08
#> 2 Actinobacteriota LTBI 5.07 0.382
#> 3 Actinobacteriota M.tb_Uninfected 5.70 0.124
#> 4 Bacteroidota Active 1.36 0.153
#> 5 Bacteroidota LTBI 14.7 1.26
#> 6 Bacteroidota M.tb_Uninfected 19.1 0.355
# Rename columns in summarized data
colnames(summarized_data) <- c("taxonomy", "group", "Abundance", "sd")
# Convert taxonomy to a factor and order by total abundance
summarized_data$taxonomy <- factor(summarized_data$taxonomy, order = TRUE, levels = total_abundance$taxonomy)

# Copy summarized data for further processing
summarized_data_2 <- summarized_data

# Calculate cumulative sums for plotting labels
plot_data <- plyr::ddply(summarized_data_2, "group", summarize, label_sd = cumsum(Abundance), label_y = cumsum(sd))
head(plot_data)
#> group label_sd label_y
#> 1 Active 73.51348 36.75674
#> 2 Active 74.87137 74.19242
#> 3 Active 74.88777 74.87957
#> 4 Active 94.77396 84.83087
#> 5 Active 95.60601 95.18999
#> 6 Active 95.89413 95.75007

# Combine summarized data with cumulative sum labels
plot_data <- cbind(as.data.frame(summarized_data_2), as.data.frame(plot_data)[, -1])

# Set label column to be the taxonomy
plot_data$label <- plot_data$taxonomy

# Order taxonomy levels by total abundance
plot_data$taxonomy <- factor(plot_data$taxonomy, order = TRUE, levels = c(as.character(total_abundance$taxonomy)[1:6]))

# Create a bar plot for relative abundance
bar_plot <- ggplot(plot_data, aes(x = group, y = Abundance, fill = taxonomy, order = taxonomy)) +
  geom_bar(stat = "identity", width = 0.5, color = "black") +

```

```

theme(axis.title.x = element_blank()) +
theme(legend.text = element_text(size = 6)) +
scale_y_continuous(name = "Relative abundance (%)") +
guides(fill = guide_legend(title = rank)) +
labs(x = "", y = "Relative abundance (%)", title = "")

# Adjust X-axis to follow the order of groups
bar_plot <- bar_plot + scale_x_discrete(limits = axis_order)

# Add error bars if the show_sd flag is TRUE
if (show_sd == TRUE) {
  bar_plot <- bar_plot +
    geom_errorbar(aes(ymin = label_sd - sd, ymax = label_sd + sd), width = 0.2)
}

# Convert sample data to a data frame
sample_data <- as.data.frame(phyloseq::sample_data(phyloseq_object))
# Adjust text angle on the X-axis if there are more than 3 groups
if (length(unique(sample_data$Group)) > 3) {
  bar_plot <- bar_plot + theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
}

# Get the taxonomy levels
taxa <- plot_data$taxonomy
# Calculate the number of factor levels
num_factors <- taxa %>% levels() %>% length()

# Summarize the number of occurrences of each factor level
factor_summary <- taxa %>%
  as.factor() %>%
  summary() %>%
  as.data.frame()

# Add row names as an ID column
factor_summary$id <- row.names(factor_summary)

# Arrange factors by ID and data by taxonomy
arranged_factors <- dplyr::arrange(factor_summary, id)
arranged_plot_data <- dplyr::arrange(plot_data, taxonomy)
head(arranged_plot_data)
#>      taxonomy      group Abundance      sd label_sd label_y
#> 1   Firmicutes      Active 19.886191 0.9281756 99.99162 99.991625
#> 2   Firmicutes      LTBI 51.785083 2.5829114 100.00000 99.995812
#> 3   Firmicutes M.tb_Uninfected 48.417191 0.4710692 5.06555 2.532775
#> 4 Actinobacteriota      Active 73.513479 1.0768376 73.51348 36.756740
#> 5 Actinobacteriota      LTBI 5.065550 0.3823114 74.87137 74.192424
#> 6 Actinobacteriota M.tb_Uninfected 5.702155 0.1238589 74.88777 74.879569
#>      label
#> 1   Firmicutes
#> 2   Firmicutes
#> 3   Firmicutes
#> 4 Actinobacteriota
#> 5 Actinobacteriota

```

```

#> 6 Actinobacteriota
# Assign an ID to each row in the arranged plot data
arranged_plot_data$ID <- factor(rep(c(1:num_factors), factor_summary$.))

head(arranged_plot_data)
#>      taxonomy      group Abundance      sd label_sd label_y
#> 1      Firmicutes      Active 19.886191 0.9281756 99.99162 99.991625
#> 2      Firmicutes      LTBI 51.785083 2.5829114 100.00000 99.995812
#> 3      Firmicutes M.tb_Uninfected 48.417191 0.4710692 5.06555 2.532775
#> 4 Actinobacteriota      Active 73.513479 1.0768376 73.51348 36.756740
#> 5 Actinobacteriota      LTBI 5.065550 0.3823114 74.87137 74.192424
#> 6 Actinobacteriota M.tb_Uninfected 5.702155 0.1238589 74.88777 74.879569
#>      label ID
#> 1      Firmicutes 1
#> 2      Firmicutes 1
#> 3      Firmicutes 1
#> 4 Actinobacteriota 2
#> 5 Actinobacteriota 2
#> 6 Actinobacteriota 2
arranged_plot_data$Abundance
#> [1] 19.886191413 51.785083383 48.417190971 73.513479241 5.065550426
#> [6] 5.702155363 3.943758692 17.908110417 17.947841619 1.357889101
#> [11] 14.722943432 19.116239320 0.832052493 9.472137561 7.595283834
#> [16] 0.288115774 0.871687299 1.032733374 0.016401643 0.146890748
#> [21] 0.108970667 0.151190044 0.005484613 0.016627884 0.002546348
#> [26] 0.012671308 0.029314897 0.000000000 0.000000000 0.029325513
#> [31] 0.008375249 0.009440813 0.004316559

# Create an alluvial plot for relative abundance
alluvial_plot <- ggplot(arranged_plot_data, aes(x = group, y = Abundance, fill = taxonomy, alluvium = t
ggalluvial::geom_flow(aes(fill = taxonomy, colour = taxonomy),
  stat = "alluvium", lode.guidance = "rightleft",
  color = "black", size = 0.2, width = 0.35, alpha = .2) +
geom_bar(width = 0.45, stat = "identity") +
labs(x = "", y = "Relative abundance (%)", title = "") +
guides(fill = guide_legend(title = rank), color = FALSE) +
scale_y_continuous(expand = c(0, 0))

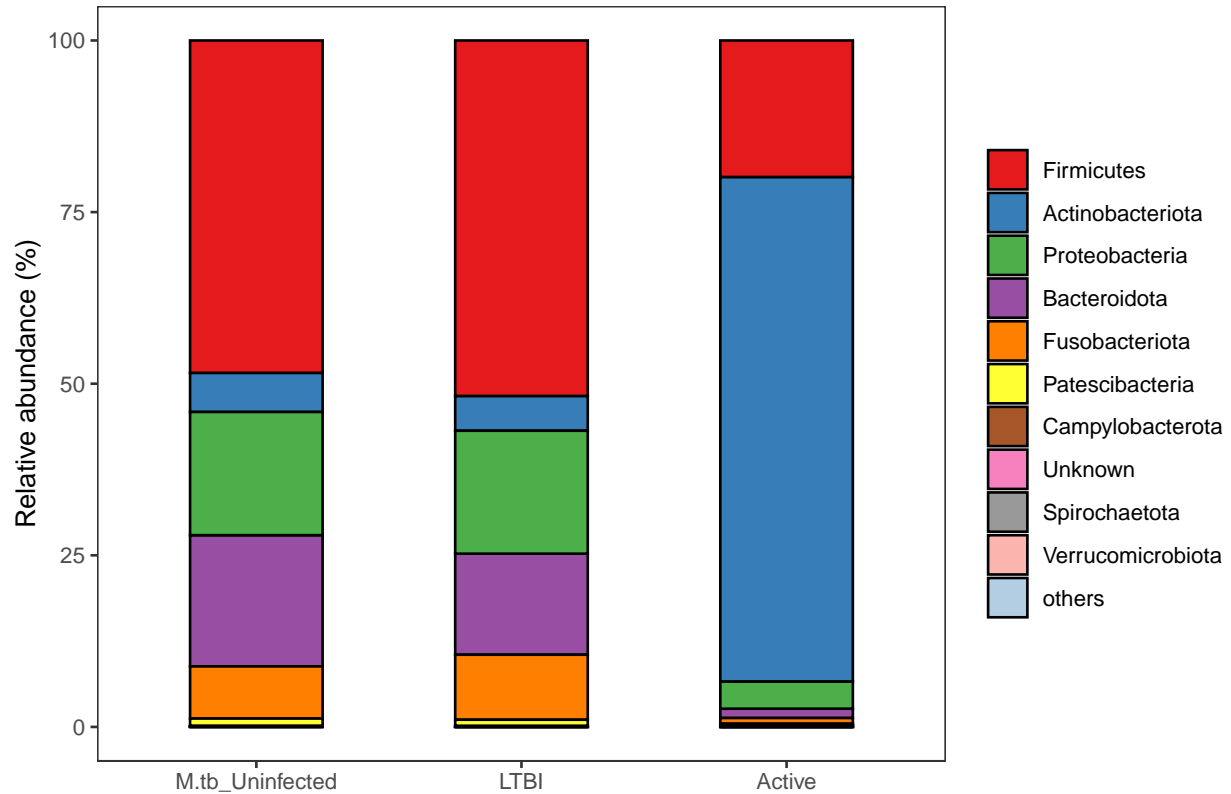
# Add error bars to the alluvial plot if the show_sd flag is TRUE
if (show_sd == TRUE) {
  alluvial_plot <- alluvial_plot +
    geom_errorbar(aes(ymin = label_sd - sd, ymax = label_sd + sd), width = 0.2)
}

# Adjust text angle on the X-axis if there are more than 3 groups
if (length(unique(sample_data$Group)) > 3) {
  alluvial_plot <- alluvial_plot + theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
}

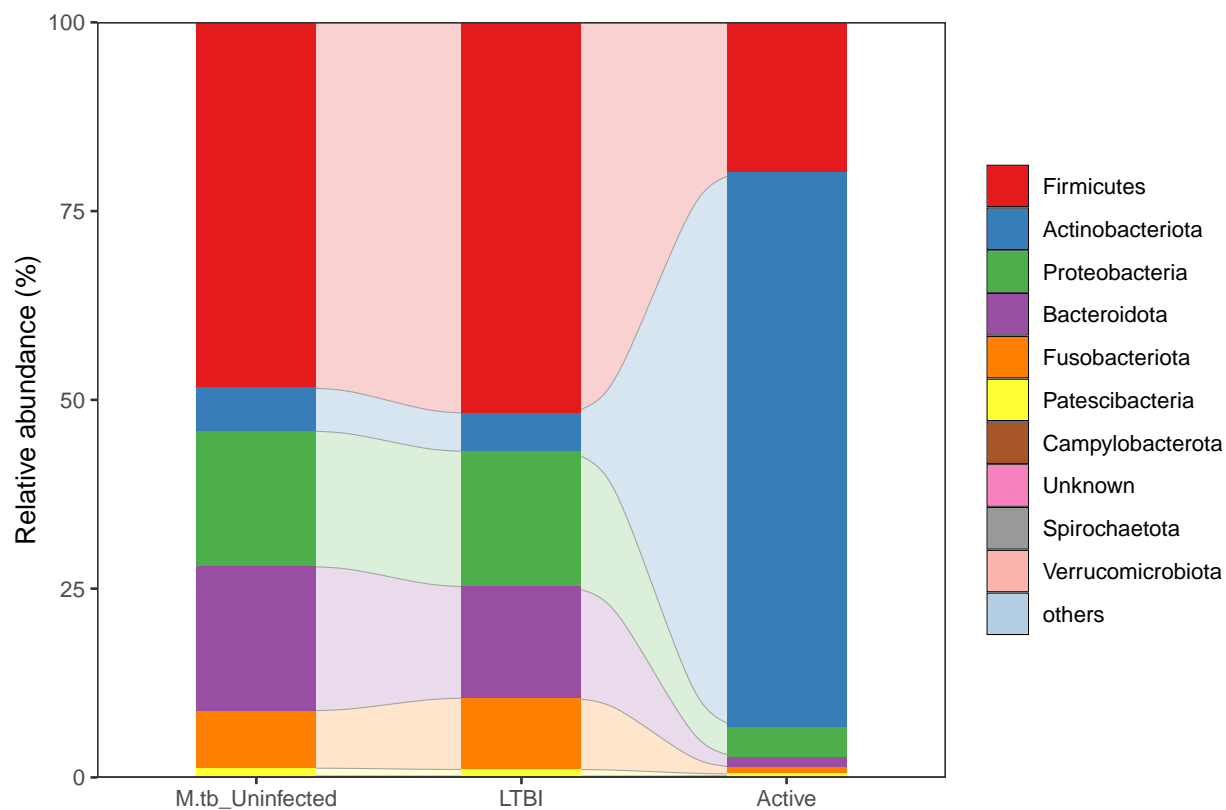
# Get rank names from the phyloseq object
phyloseq::rank_names(phyloseq_object)
#> [1] "Kingdom" "Phylum" "Class" "Order" "Family" "Genus" "Species"

```

```
# Customize the bar plot with a manual color scale and theme
bar_plot_1 <- bar_plot +
  scale_fill_manual(values = color_set) +
  scale_x_discrete(limits = axis_order) +
  custom_theme
bar_plot_1
```



```
# Customize the alluvial plot with a manual color scale and theme
bar_plot_2 <- alluvial_plot +
  scale_fill_manual(values = color_set) +
  scale_x_discrete(limits = axis_order) +
  custom_theme
bar_plot_2
```

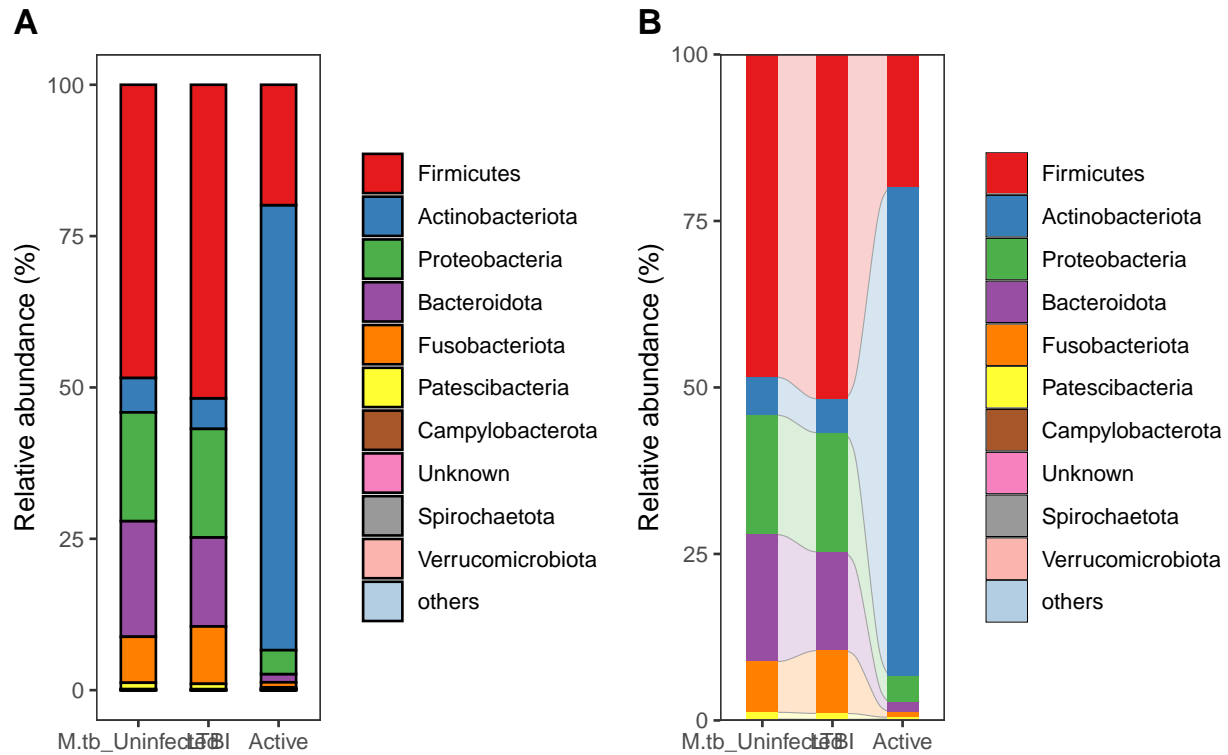


```
# Combine the bar plot and alluvial plot side by side using cowplot
combined_plot <- cowplot::plot_grid(bar_plot_1, bar_plot_2, labels = c("A", "B"))

# Add a title to the combined plot
combined_plot <- cowplot::ggdraw() +
  cowplot::draw_plot(combined_plot, 0, 0, 1, 0.9) + # Adjust plot position and size
  cowplot::draw_label("Phylum Relative Abundance According to Mtb Infection Status",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16)

# Print the combined plot with title
print(combined_plot)
```


Phylum Relative Abundance According to Mtb Infection Status



```
# Save the combined plot as a high-resolution JPG file
ggsave("Results/Figure 6A.jpg", plot = combined_plot, width = 12, height = 6, dpi = 600)
```

Genus

```
# Assign the phyloseq object to a variable
phyloseq_object <- MtbInfectionStatus_phyloseq
group_col <- "Group" # Define the group column name
rank <- "Genus" # Define the taxonomic rank
add_labels <- TRUE # Flag to add labels
show_sd <- FALSE # Flag to show standard deviation
top_n <- 10 # Number of top taxa to include
transform_abundance <- TRUE # Flag to transform data to relative abundance

# Get unique levels for group column
axis_order <- phyloseq::sample_data(phyloseq_object)$Group %>% unique()

# Perform taxonomic glomming (grouping)
phyloseq_data <- ggClusterNet::tax_glom_wt(ps = phyloseq_object, ranks = rank)

# Transform to relative abundance if specified
```

```

if (transform_abundance == TRUE) {
  phyloseq_data <- phyloseq_data %>%
    phyloseq::transform_sample_counts(function(x) {x / sum(x)})
}

# Extract OTU and taxonomy tables
otu_table <- phyloseq::otu_table(phyloseq_data)
tax_table <- phyloseq::tax_table(phyloseq_data)

# Adjust taxonomy table to group non-top taxa as "others"
for (i in 1:dim(tax_table)[1]) {
  if (row.names(tax_table)[i] %in% names(sort(rowSums(otu_table), decreasing = TRUE)[1:top_n])) {
    tax_table[i, rank] <- tax_table[i, rank]
  } else {
    tax_table[i, rank] <- "others"
  }
}

# Update the phyloseq object with the modified taxonomy table
phyloseq::tax_table(phyloseq_data) <- tax_table

# Melt the phyloseq object to a data frame
taxonomies <- phyloseq_data %>%
  phyloseq::psmelt()

# Convert abundance to percentage
taxonomies$Abundance <- taxonomies$Abundance * 100
# Rename taxonomy column to "taxonomy"
colnames(taxonomies) <- gsub(rank, "taxonomy", colnames(taxonomies))

# Initialize an empty data frame
final_data <- c()
# Loop over each group to calculate relative abundance
for (i in 1:length(unique(phyloseq::sample_data(phyloseq_object)$Group))) {
  group_name <- as.data.frame(table(phyloseq::sample_data(phyloseq_object)$Group))[i, 1]
  group_size <- as.data.frame(table(phyloseq::sample_data(phyloseq_object)$Group))[i, 2]

  # Filter data by group
  group_data <- taxonomies %>%
    dplyr::filter(Group == group_name)
  group_data$Abundance <- group_data$Abundance / group_size

  # Create a temporary data frame for the current group
  temp_data <- data.frame(Sample = group_data$Sample,
                          Abundance = group_data$Abundance,
                          taxonomy = group_data$taxonomy,
                          Group = group_data$Group)

  # Combine data for all groups
  if (i == 1) {
    final_data <- temp_data
  } else {
    final_data <- rbind(final_data, temp_data)
  }
}

```

```

}

# Update taxonomies data frame with combined data
taxonomies <- final_data

# Group data by taxonomy and group columns
grouped_taxa <- dplyr::group_by(taxonomies, taxonomy, Group)
# Summarize abundance and standard deviation by group and taxonomy
summarized_data <- dplyr::summarize(grouped_taxa, sum(Abundance), sd(Abundance))

# Group data by taxonomy to calculate total abundance
taxonomy_groups <- dplyr::group_by(taxonomies, taxonomy)
total_abundance <- dplyr::summarize(taxonomy_groups, sum(Abundance))
head(total_abundance)
#> # A tibble: 6 x 2
#>   taxonomy      `sum(Abundance)`
#>   <chr>          <dbl>
#> 1 Brevibacillus      7.76
#> 2 Fusobacterium     11.4
#> 3 Mycobacterium     64.1
#> 4 Neisseria         14.8
#> 5 Prevotella_7     20.7
#> 6 Stenotrophomonas  11.3
# Rename columns for total abundance data
colnames(total_abundance) <- c("taxonomy", "total_sum")
# Arrange taxa by total abundance in descending order
total_abundance <- dplyr::arrange(total_abundance, desc(total_sum))

# Preview the summarized data
head(summarized_data)
#> # A tibble: 6 x 4
#> # Groups:   taxonomy [2]
#>   taxonomy      Group      `sum(Abundance)` `sd(Abundance)`
#>   <chr>        <chr>          <dbl>          <dbl>
#> 1 Brevibacillus Active      7.76           0.769
#> 2 Brevibacillus LTBI        0              0
#> 3 Brevibacillus M.tb_Uninfected 0.0000414      0.00000744
#> 4 Fusobacterium Active      0.176          0.0186
#> 5 Fusobacterium LTBI        7.04           1.15
#> 6 Fusobacterium M.tb_Uninfected 4.19           0.139
# Rename columns in summarized data
colnames(summarized_data) <- c("taxonomy", "group", "Abundance", "sd")
# Convert taxonomy to a factor and order by total abundance
summarized_data$taxonomy <- factor(summarized_data$taxonomy, order = TRUE, levels = total_abundance$taxonomy)

# Copy summarized data for further processing
summarized_data_2 <- summarized_data

# Calculate cumulative sums for plotting labels
plot_data <- plyr::ddply(summarized_data_2, "group", summarize, label_sd = cumsum(Abundance), label_y =
head(plot_data)
#>   group label_sd label_y
#> 1 Active 7.764394 3.882197

```

```

#> 2 Active 7.939998 7.852196
#> 3 Active 72.007652 39.973825
#> 4 Active 72.599165 72.303408
#> 5 Active 72.761836 72.680500
#> 6 Active 72.762470 72.762153

# Combine summarized data with cumulative sum labels
plot_data <- cbind(as.data.frame(summarized_data_2), as.data.frame(plot_data)[, -1])

# Set label column to be the taxonomy
plot_data$label <- plot_data$taxonomy

# Order taxonomy levels by total abundance
plot_data$taxonomy <- factor(plot_data$taxonomy, order = TRUE, levels = c(as.character(total_abundance$

# Create a bar plot for relative abundance
bar_plot <- ggplot(plot_data, aes(x = group, y = Abundance, fill = taxonomy, order = taxonomy)) +
  geom_bar(stat = "identity", width = 0.5, color = "black") +
  theme(axis.title.x = element_blank()) +
  theme(legend.text = element_text(size = 6)) +
  scale_y_continuous(name = "Relative abundance (%)") +
  guides(fill = guide_legend(title = rank)) +
  labs(x = "", y = "Relative abundance (%)", title = "")

# Adjust X-axis to follow the order of groups
bar_plot <- bar_plot + scale_x_discrete(limits = axis_order)

# Add error bars if the show_sd flag is TRUE
if (show_sd == TRUE) {
  bar_plot <- bar_plot +
    geom_errorbar(aes(ymin = label_sd - sd, ymax = label_sd + sd), width = 0.2)
}

# Convert sample data to a data frame
sample_data <- as.data.frame(phyloseq::sample_data(phyloseq_object))
# Adjust text angle on the X-axis if there are more than 3 groups
if (length(unique(sample_data$Group)) > 3) {
  bar_plot <- bar_plot + theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
}

# Get the taxonomy levels
taxa <- plot_data$taxonomy
# Calculate the number of factor levels
num_factors <- taxa %>% levels() %>% length()

# Summarize the number of occurrences of each factor level
factor_summary <- taxa %>%
  as.factor() %>%
  summary() %>%
  as.data.frame()

# Add row names as an ID column
factor_summary$id <- row.names(factor_summary)

```

```

# Arrange factors by ID and data by taxonomy
arranged_factors <- dplyr::arrange(factor_summary, id)
arranged_plot_data <- dplyr::arrange(plot_data, taxonomy)
head(arranged_plot_data)
#>      taxonomy      group Abundance      sd label_sd label_y
#> 1 Streptococcus    Active 8.614375e+00 5.469314e-01 72.95060 72.95043
#> 2 Streptococcus    LTBI 4.240755e+01 2.700678e+00 76.43687 74.69373
#> 3 Streptococcus M.tb_Uninfected 3.740700e+01 4.756377e-01 80.54534 78.49111
#> 4 Mycobacterium    Active 6.406765e+01 1.247160e+00 81.37685 77.06966
#> 5 Mycobacterium    LTBI 6.984606e-04 2.328202e-04 85.82627 83.60156
#> 6 Mycobacterium M.tb_Uninfected 4.826456e-04 3.789528e-05 88.12495 86.97561
#>      label
#> 1 Streptococcus
#> 2 Streptococcus
#> 3 Streptococcus
#> 4 Mycobacterium
#> 5 Mycobacterium
#> 6 Mycobacterium
# Assign an ID to each row in the arranged plot data
arranged_plot_data$ID <- factor(rep(c(1:num_factors), factor_summary$.),)

head(arranged_plot_data)
#>      taxonomy      group Abundance      sd label_sd label_y
#> 1 Streptococcus    Active 8.614375e+00 5.469314e-01 72.95060 72.95043
#> 2 Streptococcus    LTBI 4.240755e+01 2.700678e+00 76.43687 74.69373
#> 3 Streptococcus M.tb_Uninfected 3.740700e+01 4.756377e-01 80.54534 78.49111
#> 4 Mycobacterium    Active 6.406765e+01 1.247160e+00 81.37685 77.06966
#> 5 Mycobacterium    LTBI 6.984606e-04 2.328202e-04 85.82627 83.60156
#> 6 Mycobacterium M.tb_Uninfected 4.826456e-04 3.789528e-05 88.12495 86.97561
#>      label ID
#> 1 Streptococcus 1
#> 2 Streptococcus 1
#> 3 Streptococcus 1
#> 4 Mycobacterium 2
#> 5 Mycobacterium 2
#> 6 Mycobacterium 2
arranged_plot_data$Abundance
#> [1] 8.614375e+00 4.240755e+01 3.740700e+01 6.406765e+01 6.984606e-04
#> [6] 4.826456e-04 1.139800e+01 1.945466e+01 2.529569e+01 1.626713e-01
#> [11] 7.788218e+00 1.273785e+01 5.915123e-01 8.364471e+00 5.872532e+00
#> [16] 1.756048e-01 7.036583e+00 4.189324e+00 6.342343e-04 7.352734e+00
#> [21] 3.957739e+00 4.770567e-01 4.108475e+00 6.049974e+00 2.298674e+00
#> [26] 3.486270e+00 4.459788e+00 7.764394e+00 0.000000e+00 4.141968e-05
#> [31] 4.449428e+00 3.458604e-04 2.958449e-02

# Create an alluvial plot for relative abundance
alluvial_plot <- ggplot(arranged_plot_data, aes(x = group, y = Abundance, fill = taxonomy, alluvium = t
ggalluvial::geom_flow(aes(fill = taxonomy, colour = taxonomy),
  stat = "alluvium", lode.guidance = "rightleft",
  color = "black", size = 0.2, width = 0.35, alpha = .2) +
geom_bar(width = 0.45, stat = "identity") +
labs(x = "", y = "Relative abundance (%)", title = "") +
guides(fill = guide_legend(title = rank), color = FALSE) +

```

```

scale_y_continuous(expand = c(0, 0))

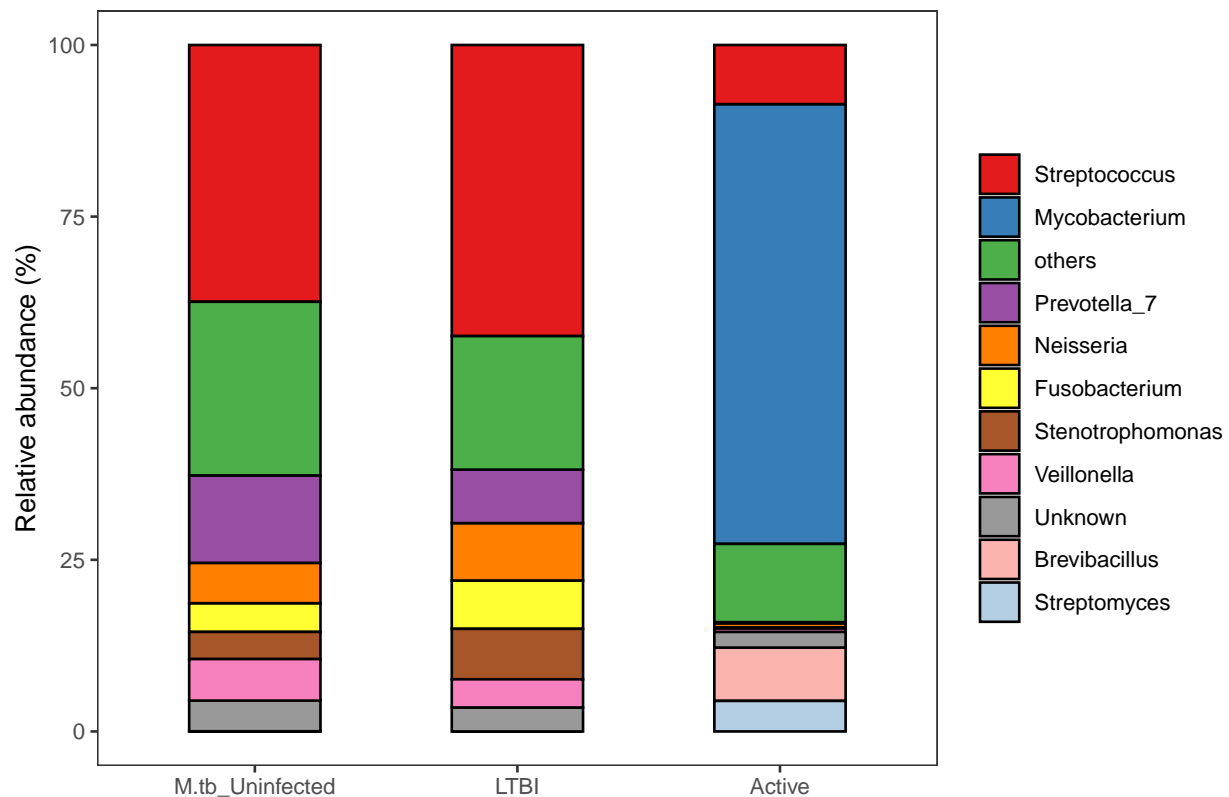
# Add error bars to the alluvial plot if the show_sd flag is TRUE
if (show_sd == TRUE) {
  alluvial_plot <- alluvial_plot +
    geom_errorbar(aes(ymin = label_sd - sd, ymax = label_sd + sd), width = 0.2)
}

# Adjust text angle on the X-axis if there are more than 3 groups
if (length(unique(sample_data$Group)) > 3) {
  alluvial_plot <- alluvial_plot + theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
}

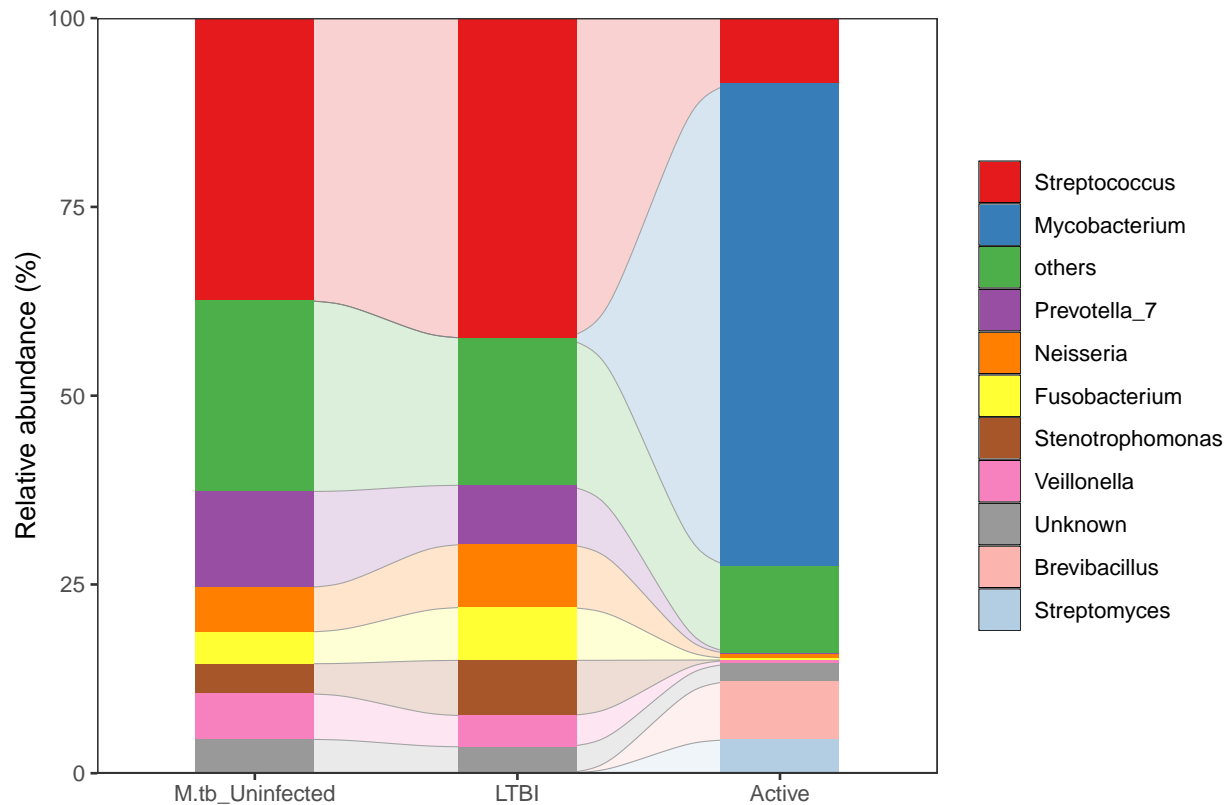
# Get rank names from the phyloseq object
phyloseq::rank_names(phyloseq_object)
#> [1] "Kingdom" "Phylum" "Class" "Order" "Family" "Genus" "Species"

# Customize the bar plot with a manual color scale and theme
bar_plot_1 <- bar_plot +
  scale_fill_manual(values = color_set) +
  scale_x_discrete(limits = axis_order) +
  custom_theme
bar_plot_1

```



```
# Customize the alluvial plot with a manual color scale and theme
bar_plot_2 <- alluvial_plot +
  scale_fill_manual(values = color_set) +
  scale_x_discrete(limits = axis_order) +
  custom_theme
bar_plot_2
```

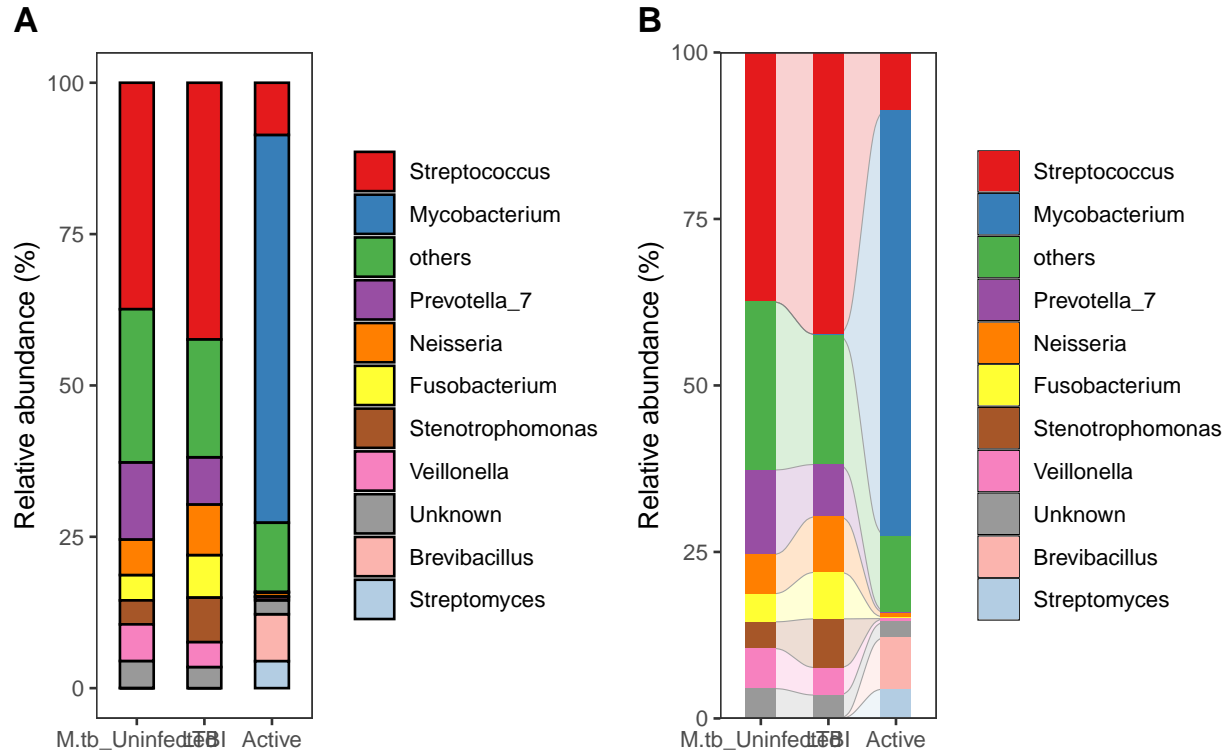


```
# Combine the bar plot and alluvial plot side by side using cowplot
combined_plot <- cowplot::plot_grid(bar_plot_1, bar_plot_2, labels = c("A", "B"))

# Add a title to the combined plot
combined_plot <- cowplot::ggdraw() +
  cowplot::draw_plot(combined_plot, 0, 0, 1, 0.9) + # Adjust plot position and size
  cowplot::draw_label("Genus Relative Abundance According to Mtb Infection Status",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16)

# Print the combined plot with title
print(combined_plot)
```

Genus Relative Abundance According to Mtb Infection Status



```
# Save the combined plot as a high-resolution JPG file
ggsave("Results/Figure 6B.jpg", plot = combined_plot, width = 12, height = 6, dpi = 600)
```

Differential abundance

```
# Load phyloseq object and process taxonomy
physeq_data <- MtbInfectionStatus_phyloseq
# Analysis parameters
analysis_group <- "Group" # Group variable for analysis
p_value_threshold <- 0.05 # Threshold for p-value significance
log_fold_change_threshold <- 0 # Threshold for log fold change
artificial_groups <- NULL # Placeholder for artificial group comparisons
normalization_method <- "TMM" # Method for normalization
rank_selection <- 6 # Taxonomic rank for analysis
contrast_matrix <- NULL # Placeholder for contrast matrix

# Initialize a list to store plots
plot_list <- list() # List to store individual plots

# Prepare phyloseq data based on rank selection
```



```

if (rank_selection %in% c("OTU", "gene", "meta")) {
  physeq_data <- physeq_data # No change needed
} else if (rank_selection %in% c(1:7)) {
  physeq_data <- physeq_data %>%
    ggClusterNet::tax_glom_wt(ranks = rank_selection) # Aggregate taxa at specified rank
} else if (rank_selection %in% c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species")) {
  # No action needed for standard taxonomic ranks
} else {
  print("Unknown rank_selection, please check") # Print error if rank selection is invalid
}

# Prepare design matrix and count data
sample_data <- as.data.frame(phyloseq::sample_data(physeq_data)) # Convert sample data to dataframe
descriptive_groups <- as.character(levels(as.factor(sample_data$Group))) # Extract group levels

if (is.null(artificial_groups)) {
  contrast_combinations <- combn(descriptive_groups, 2) # Generate all pairwise contrasts
} else if (!is.null(artificial_groups)) {
  contrast_combinations <- as.matrix(contrast_matrix) # Use provided contrast matrix
}

otu_table_data <- as.data.frame(ggClusterNet::vegan_otu(physeq_data)) # Extract OTU table
count_matrix <- as.matrix(otu_table_data) # Convert to matrix and transpose
count_matrix <- t(count_matrix)
sample_data$SampleType <- as.factor(sample_data$Group) # Convert Group to factor
dge_list <- edgeR::DGEList(counts = count_matrix, group = sample_data$SampleType) # Create DGEList object
dge_list <- edgeR::calcNormFactors(dge_list, method = normalization_method) # Normalize counts

# Define design matrix for GLM
design_matrix <- model.matrix(~ 0 + dge_list$samples$group) # Create design matrix without intercept
colnames(design_matrix) <- levels(sample_data$SampleType) # Set column names
dge_list <- edgeR::estimateGLMCommonDisp(dge_list, design_matrix) # Estimate common dispersion
dge_list <- edgeR::estimateGLMTagwiseDisp(dge_list, design_matrix) # Estimate tagwise dispersion
glm_fit <- edgeR::glmFit(dge_list, design_matrix) # Fit GLM

# Perform differential analysis and plot results
for (i in 1:dim(contrast_combinations)[2]) {
  comparison_groups <- contrast_combinations[, i] # Extract current comparison groups
  print(comparison_groups) # Print comparison groups

  contrast_name <- paste(comparison_groups[1], comparison_groups[2], sep = "-") # Create contrast name
  contrast_matrix <- limma::makeContrasts(contrasts = contrast_name, levels = c(as.character(levels(a
  glm_lrt <- edgeR::glmLRT(glm_fit, contrast = contrast_matrix) # Perform likelihood ratio test

  diff_test_result <- edgeR::decideTestsDGE(glm_lrt, adjust.method = "fdr", p.value = p_value_threshol
  summary(diff_test_result) # Summarize differential test results
  result_table <- glm_lrt$table # Extract results table
  result_table$sig <- diff_test_result # Add significance results
  row.names(count_matrix)[1:6] # Print row names for checking

  result_table <- cbind(result_table, padj = result_table$PValue) # Adjust p-values
  enriched_taxa <- row.names(subset(result_table, sig == 1)) # Identify enriched taxa
  depleted_taxa <- row.names(subset(result_table, sig == -1)) # Identify depleted taxa

```

```

result_table$level <- as.factor(ifelse(as.vector(result_table$sig) == 1, "enriched",
                                     ifelse(as.vector(result_table$sig) == -1, "depleted",
                                             ifelse(result_table$padj < 0.05, "unplot", "nosig"))))
result_table <- data.frame(row.names = row.names(result_table), logFC = result_table$logFC, level =
                           result_table$level)

filtered_table <- result_table %>%
  dplyr::filter(level %in% c("enriched", "depleted", "nosig")) # Filter based on significance

filtered_table$Genus <- row.names(filtered_table) # Add Genus names
if (nrow(filtered_table) <= 1) {
  next # Skip if no significant results
}

top_taxa <- filtered_table %>%
  dplyr::mutate(ord = logFC^2) %>%
  dplyr::filter(level != "nosig") %>%
  dplyr::arrange(desc(ord)) %>%
  head(n = 5) # Select top 5 taxa

# Store the top 5 enriched/depleted results in the list
plot_list[[i]] <- ggplot(filtered_table, aes(x = logFC, y = -log2(p), color = level)) +
  geom_point(size = 3.5, alpha = 0.7, shape = 16) + # Slightly larger point size with transparency
  geom_hline(yintercept = -log2(p_value_threshold), linetype = "dashed", color = 'black', size = 0.9) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = 'black', size = 0.9) + # Thicker dashed lines
  ggrepel::geom_text_repel(data = top_taxa, aes(x = logFC, y = -log2(p), label = Genus),
                           size = 4, box.padding = 0.5, max.overlaps = Inf,
                           segment.color = 'grey50', segment.size = 0.6,
                           nudge_x = 0.2, nudge_y = 0.2) + # Adjusted padding and overlap handling
  scale_color_manual(values = c("enriched" = "red", "depleted" = "blue", "nosig" = "grey")) + # Color by significance
  labs(title = contrast_name, x = "Log Fold Change", y = "-Log2 P-Value", color = "Significance") +
  theme_minimal(base_size = 14) + # Use a minimal theme for a cleaner look
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5), # Center and bold title
    axis.title = element_text(size = 15), # Larger axis titles
    axis.text = element_text(size = 12), # Larger axis text
    legend.position = "top", # Place legend at the top for better visibility
    legend.title = element_text(size = 12), # Larger legend title
    legend.text = element_text(size = 10), # Larger legend text
    panel.grid.major = element_line(size = 0.5, linetype = 'solid', color = "grey90"), # Lighter grid lines
    panel.grid.minor = element_blank(), # Remove minor grid lines for a cleaner look
    panel.border = element_rect(color = "grey80", fill = NA, size = 0.5) # Add border around the plot
  )

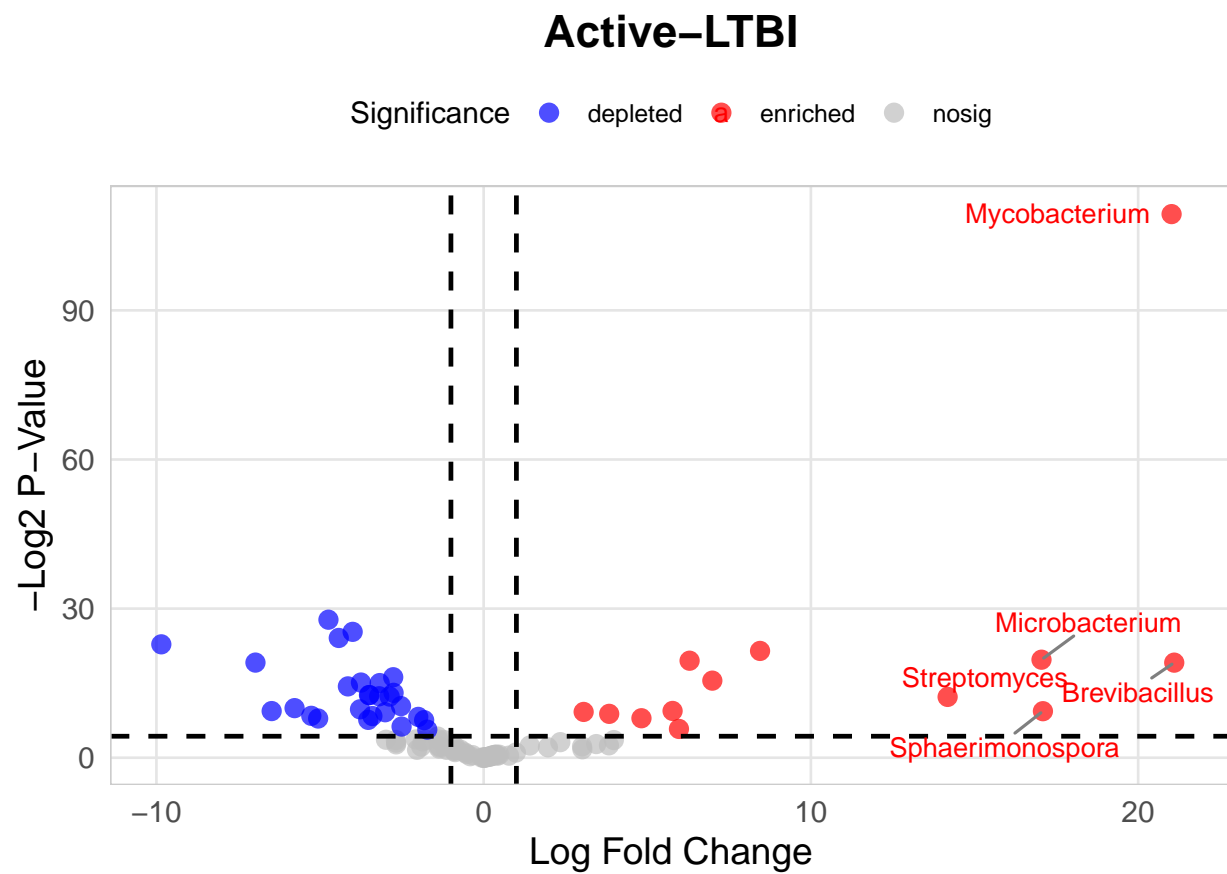
colnames(result_table) <- paste(contrast_name, colnames(result_table), sep = "_") # Rename columns

if (i == 1) {
  combined_results <- result_table # Initialize combined results
} else {
  combined_results <- cbind(combined_results, result_table) # Combine results
}
}
#> [1] "Active" "LTBI"

```

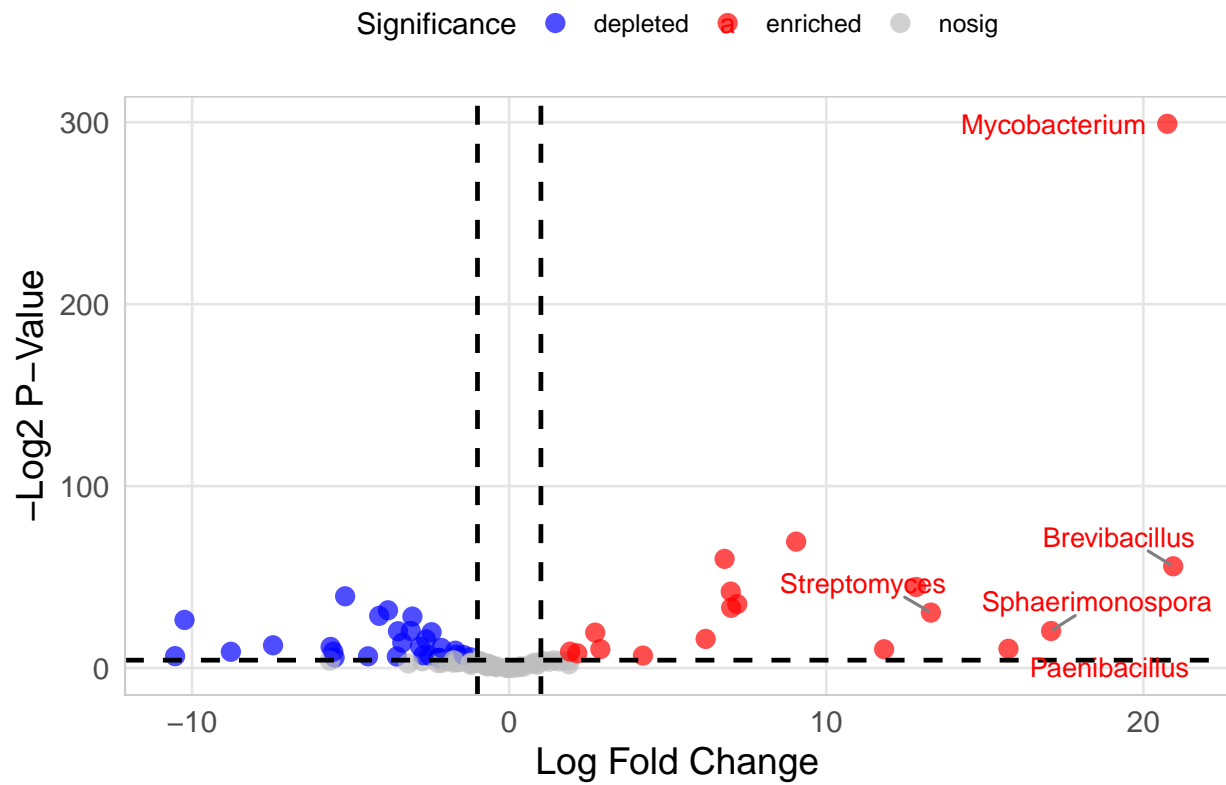
```
#> [1] "Active"      "M.tb_Uninfected"
#> [1] "LTBI"       "M.tb_Uninfected"

# Print all plots
print(plot_list) # Print all plots stored in the list
#> [[1]]
```



```
#>
#> [[2]]
```

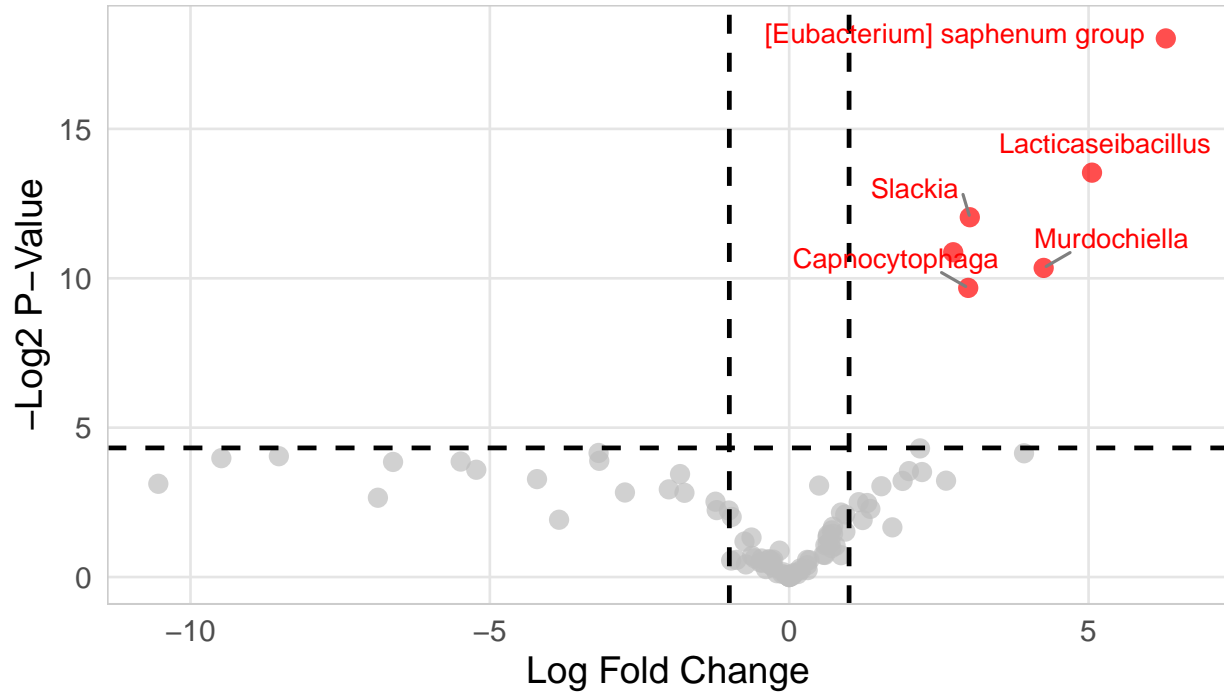
Active-M.tb_Uninfected



```
#>  
#> [[3]]
```

LTBI-M.tb_Uninfected

Significance ● enriched ● nosig



```
# Combine and save the plots
combined_plot <- wrap_plots(plot_list, ncol = 3)
ggsave(filename = "Results/Figure 7.jpg", plot = combined_plot, width = 15, height = 10, dpi = 600)
```

Biomarker Identification

LEfSe analysis

```
# Load the saved phyloseq object from an RDS file
ps <- MtbInfectionStatus_phyloseq
mytheme1 <- ggplot2::theme_bw() +
  ggplot2::theme(
    panel.background = ggplot2::element_blank(), # Remove background color
    panel.grid = ggplot2::element_blank(), # Remove grid lines
    legend.position = "right", # Position legend on the right
    legend.title = ggplot2::element_blank(), # Remove legend title
    legend.background = ggplot2::element_blank(), # Remove legend background
    legend.key = ggplot2::element_blank(), # Remove legend key background
    plot.title = ggplot2::element_text(hjust = 0.5), # Center plot title without vertical adjustment
    axis.title.y = ggplot2::element_text(colour = "black"), # Set Y axis title color
    axis.text = ggplot2::element_text() # Set axis text style
```

```

)

# Function to generate a base plot from phyloseq data
# Function to create a circular phylogenetic tree plot with a well-edited title
p_base = function(ps, Top = 100, ranks = 6, title = "Circular Phylogenetic Tree") {
  alltax = ps %>%
    ggClusterNet::tax_glom_wt(ranks = ranks) %>% # Collapse taxa at a given rank
    ggClusterNet::filter_OTU_ps(Top) %>% # Filter top OTUs
    ggClusterNet::vegan_tax() %>% # Extract taxonomy table
    as.data.frame() # Convert to data frame

  alltax$OTU = row.names(alltax) # Add OTU IDs as a column

  # Concatenate taxonomic ranks into a single string, separated by "_Rank_"
  alltax$Kingdom = paste(alltax$Kingdom, sep = "_Rank_")
  for (i in 2:ranks) {
    alltax[, i] = paste(alltax[, i - 1], alltax[, i], sep = "_Rank_")
  }

  alltax[is.na(alltax)] = "Unknown" # Replace NA values with "Unknown"
  trda <- MicrobiotaProcess::convert_to_treedata(alltax) # Convert to tree data format

  # Create a circular phylogenetic tree plot
  p <- ggtree(trda, layout = "circular", size = 0.2, xlim = c(30, NA)) +
    geom_point(
      pch = 21, # Set point shape
      size = 3, # Set point size
      alpha = 1, # Set point transparency
      fill = "#FFFFB3" # Set fill color for points
    ) +
    ggtitle(title) + # Add title with the given argument
    theme(plot.title = element_text(size = 18, face = "bold", hjust = 0.5), # Title appearance
          axis.text = element_blank(), # Hide axis text
          axis.title = element_blank(), # Hide axis titles
          panel.background = element_blank(), # Remove background grid
          panel.grid = element_blank(), # Remove panel grid
          legend.position = "none") # Remove legend

  # Extract the label for each node, using the last taxonomic rank in the string
  p$data$lab2 <- p$data$label %>% strsplit("_Rank_") %>%
    sapply(function(x) x[length(x)])

  # Clean up the labels
  p$data$lab2 = gsub("st__", "", p$data$lab2)
  p$data$nodeSize = 1 # Set node size for all nodes

  return(p) # Return the ggplot object
}

# Function to perform LDA (Linear Discriminant Analysis) on the phyloseq object
LDA_Micro = function(ps = ps,
                     Top = 100,

```

```

        ranks = 6,
        p.lvl = 0.05,
        lda.lvl = 2,
        seed = 11,
        adjust.p = F) {

# Process taxonomy data similarly to p_base function
alltax = ps %>%
  ggClusterNet::tax_glom_wt(ranks = ranks) %>%
  phyloseq::filter_taxa(function(x) sum(x) > 0, TRUE) %>%
  ggClusterNet::filter_OTU_ps(Top) %>%
  ggClusterNet::vegan_tax() %>%
  as.data.frame()
alltax$OTU = row.names(alltax)

# Concatenate taxonomic ranks similarly as in p_base
alltax$Kingdom = paste(alltax$Kingdom, sep = "_Rank_")
for (i in 2:ranks) {
  alltax[, i] = paste(alltax[, i - 1], alltax[, i], sep = "_Rank_")
}

# Prepare OTU table for LDA analysis
otu = ps %>%
  ggClusterNet::tax_glom_wt(ranks = ranks) %>%
  phyloseq::filter_taxa(function(x) sum(x) > 0, TRUE) %>%
  ggClusterNet::filter_OTU_ps(Top) %>%
  ggClusterNet::vegan_otu() %>%
  t() %>%
  as.data.frame()

# Merge OTU data with taxonomy data
otu_tax = merge(otu, alltax, by = "row.names", all = F)

# Summarize OTU counts at each taxonomic level
tem = colnames(alltax)[-length(colnames(alltax))]
i = 1
tem2 = c("k_", "p_", "c_", "o_", "f_", "g_", "s_", "st_")
for (i in 1:ranks) {
  rank1 <- otu_tax %>%
    dplyr::group_by(!sym(tem[i])) %>%
    dplyr::summarise_if(is.numeric, sum, na.rm = TRUE)
  colnames(rank1)[1] = "id"
  rank1$id = paste(tem2[i], rank1$id, sep = "")
  if (i == 1) {
    all = rank1
  }
  if (i != 1) {
    all = rbind(all, rank1)
  }
}

# Convert summarized data to phyloseq object for further analysis
data1 = as.data.frame(all)

```

```

row.names(data1) = data1$id
data1$id = NULL

ps_G_graphlan = phyloseq::phyloseq(
  phyloseq::otu_table(as.matrix(data1), taxa_are_rows = TRUE),
  phyloseq::sample_data(ps)
)

# Extract OTU and sample data for LDA
otu = as.data.frame((ggClusterNet::vegan_otu(ps_G_graphlan)))
map = as.data.frame(phyloseq::sample_data(ps_G_graphlan))
claslbl = map$Group %>% as.factor()

set.seed(seed) # Set seed for reproducibility

# Perform Kruskal-Wallis rank sum test on OTUs
rawpvalues <- apply(otu, 2, function(x) kruskal.test(x, claslbl)$p.value)
ord.inx <- order(rawpvalues)
rawpvalues <- rawpvalues[ord.inx]

# Adjust p-values if specified
clapvalues <- p.adjust(rawpvalues, method = "fdr")

# Prepare data for LDA
wil_datadf <- as.data.frame(otu[, ord.inx])

# Perform LDA and calculate LDAscore
ldares <- MASS::lda(claslbl ~ ., data = wil_datadf)
ldamean <- as.data.frame(t(ldares$means))
class_no <-< length(unique(claslbl))
ldamean$max <- apply(ldamean[, 1:class_no], 1, max)
ldamean$min <- apply(ldamean[, 1:class_no], 1, min)
ldamean$LDAscore <- signif(log10(1 + abs(ldamean$max - ldamean$min) / 2), digits = 3)

# Determine the class with the highest LDAscore
a = rep("A", length(ldamean$max))
for (i in 1:length(ldamean$max)) {
  name = colnames(ldamean[, 1:class_no])
  a[i] = name[ldamean[, 1:class_no][i, ] %in% ldamean$max[i]]
}
ldamean$class = a

# Add p-values and FDR to the results table
tem1 = row.names(ldamean)
ldamean$Pvalues <- signif(rawpvalues[match(row.names(ldamean), names(rawpvalues))], digits = 5)
ldamean$FDR <- signif(clapvalues, digits = 5)
resTable <- ldamean
rawNms <- rownames(resTable)
rownames(resTable) <- gsub("`", "'", rawNms)

# Count significant features based on criteria
if (adjust.p) {
  de.Num <- sum(clapvalues <= p.lvl & ldamean$LDAscore >= lda.lvl)

```



```

} else {
  de.Num <- sum(rawpvalues <= p.lvl & ldamean$LDAscore >= lda.lvl)
}

# Display the results message
if (de.Num == 0) {
  current.msg <- "No significant features were identified with given criteria."
} else {
  current.msg <- paste("A total of", de.Num, "significant features with given criteria.")
}
print(current.msg)

# Sort the results table by p-values and LDAscore
ord.inx <- order(resTable$Pvalues, resTable$LDAscore)
resTable <- resTable[ord.inx, , drop = FALSE]
resTable <- resTable[, c(ncol(resTable), 1:(ncol(resTable) - 1))]

# Filter the significant taxa
if (adjust.p) {
  taxtree = resTable[clapvalues <= p.lvl & ldamean$LDAscore >= lda.lvl, ]
} else {
  taxtree = resTable[ldamean$Pvalues <= p.lvl, ]
}

# Assign colors to significant taxa based on their class
colour = c('darkgreen', 'red', "blue", "#4DAF4A", "#984EA3", "#FF7F00", "#FFFF33", "#A65628", "#F781BF")
selececol = colour[1:length(levels(as.factor(taxtree$class)))]
names(selececol) = levels(as.factor(taxtree$class))
A = rep("a", length(row.names(taxtree)))
for (i in 1:length(row.names(taxtree))) {
  A[i] = selececol[taxtree$class[i]]
}

taxtree$color = A # Assign color to taxtree

# Prepare the final output as a list
lefse_lists = data.frame(node = row.names(taxtree),
                          color = A,
                          Group = taxtree$class,
                          stringsAsFactors = FALSE)

return(list(lefse_lists, taxtree))
}

# Function to annotate clades in the phylogenetic tree
clade.anno_wt <- function(gtree, anno.data, alpha = 0.2, anno.depth = 5, anno.x = 10,
                          anno.y = 40) {

  short.labs <- c(letters, paste(letters, 1:500, sep = ""))

  # Helper function to calculate offset for clade labels
  get_offset <- function(x) {
    (x * 0.2 + 0.2)^2
  }

```

```

}

# Helper function to calculate the angle for clade labels
get_angle <- function(node) {
  data <- gtree$data
  sp <- tidytree::offspring(data, node)$node
  sp2 <- c(sp, node)
  sp.df <- data[match(sp2, data$node), ]
  mean(range(sp.df$angle))
}

# Arrange annotation data and assign colors to clades
anno.data <- dplyr::arrange(anno.data, node)
highlight.color <- anno.data$color
node_list <- anno.data$node
node_ids <- (gtree$data %>% filter(label %in% node_list) %>%
  arrange(label))$node
anno <- rep("yellow", nrow(gtree$data))

# Highlight clades with specified colors
for (i in 1:length(node_ids)) {
  n <- node_ids[i]
  color <- highlight.color[i]
  anno[n] <- color
  mapping <- gtree$data %>% filter(node == n)
  nodeClass <- as.numeric(mapping$nodeDepth)
  offset <- get_offset(nodeClass)
  gtree <- gtree + geom_highlight(node = n, fill = color,
                                alpha = alpha, extend = offset)
}

# Annotate clades with labels
short.labs.anno <- NULL
for (i in 1:length(node_ids)) {
  n <- node_ids[i]
  mapping <- gtree$data %>% filter(node == n)
  nodeClass <- as.numeric(mapping$nodeDepth)
  if (nodeClass <= anno.depth) {
    lab <- short.labs[1]
    short.labs <- short.labs[-1]
    if (is.null(short.labs.anno)) {
      short.labs.anno = data.frame(lab = lab, annot = mapping$lab2,
                                   stringsAsFactors = F)
    } else {
      short.labs.anno = rbind(short.labs.anno, c(lab, mapping$lab2))
    }
  } else {
    lab <- mapping$lab2
  }
}

offset <- get_offset(nodeClass) - 0.4
angle <- get_angle(n) + 90
gtree <- gtree + geom_cladelabel(node = n, label = lab,

```

```

        angle = angle, fontsize = 1 + sqrt(nodeClass),
        offset = offset, barsize = NA, hjust = 0.5)
}

# Generate legend for clade colors
if (!is.null(short.labs.anno)) {
  anno_shapes = sapply(short.labs.anno$lab, utf8ToInt)
  stable.p <- ggpubr::ggtexttable(short.labs.anno, rows = NULL,
                                theme = ggpubr::ttheme(
                                  colnames.style = ggpubr::colnames_style(fill = "white"),
                                  tbody.style = ggpubr::tbody_style(fill = ggpubr::get_palette("RdBu"))
                                ))
}

y = (1:length(unique(anno.data$Group)))

pleg <- ggplot() + geom_point2(aes(
  y = y,
  x = rep(1, length(unique(anno.data$color))), fill = as.factor(1:length(unique(anno.data$Group)))
), pch = 21, size = 2) +
  geom_text(aes(y = y,
                x = rep(1, length(unique(anno.data$color))), label = unique(anno.data$Group)),
            hjust = -0.2
  ) + scale_fill_manual(values = unique(anno.data$color), guide = F) +
  theme_void()

layout <- "
AAAAAABB
AAAAAABB
AAAAAABB
CCCCCCCC
"

# Combine tree plot and legend layout
if (is.null(short.labs.anno)) {
  gtree <- gtree + pleg + plot_layout(design = layout)
} else {
  gtree <- gtree + stable.p + pleg + plot_layout(design = layout)
}
}

# Function to create a bar plot for LEfSe results, highlighting taxa with significant LDA scores
lefse_bar = function(taxtree = tablda[[2]]) {
  taxtree = tablda[[2]]
  taxtree$ID = row.names(taxtree)

  # Prepare LEfSe data for bar plot
  taxtree$ID = gsub("_Rank_", ";", taxtree$ID)
  taxtree <- taxtree %>%
    arrange(class, LDAscore)

```

```

taxtree$ID = factor(taxtree$ID, levels = taxtree$ID)
taxtree$class = factor(taxtree$class, levels = unique(taxtree$class))

# Create the bar plot with a customized title
pbar <- ggplot(taxtree) +
  geom_bar(aes(y = ID, x = LDAScore, fill = class), stat = "identity") +
  scale_fill_manual(values = unique(taxtree$color)) +
  mytheme1 +
  scale_x_continuous(limits = c(0, max(taxtree$LDAScore) * 1.2)) +
  labs(title = "LDA scores of significant taxa", # Add title
       x = "LDA Score", # x-axis label
       y = "Taxa") + # y-axis label
  theme(plot.title = element_text(size = 14, face = "bold", hjust = 0.5), # Adjust title size
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        legend.title = element_text(size = 12),
        legend.text = element_text(size = 10))
return(pbar)
}

# Initialize empty lists to store the plots and results
tree_plots <- list()
bar_plots <- list()
tree_lefse_data <- list()

# Loop through different taxonomic ranks and generate plots/results
for (j in 2:6) {

  # Generate base tree plot for the given rank
  p1 <- p_base(ps, Top = 200, ranks = j, title = paste("Circular Tree Plot"))

  # Perform LDA analysis and get the results
  tablda <- LDA_Micro(ps = ps,
                     Top = 200,
                     ranks = j,
                     p.lvl = 0.05,
                     lda.lvl = 2,
                     seed = 11,
                     adjust.p = F)

  # Annotate clades in the tree plot
  p2 <- clade.anno_wt(p1, tablda[[1]], alpha = 0.3, anno.depth = 2)
  tree_plots[[paste0("Rank_", j, "_tree_plot")]] <- p2 # Save tree plot in the list

  # Create a bar plot of LEfSe results
  p <- lefse_bar(taxtree = tablda[[2]])
  bar_plots[[paste0("Rank_", j, "_bar_plot")]] <- p # Save bar plot in the list

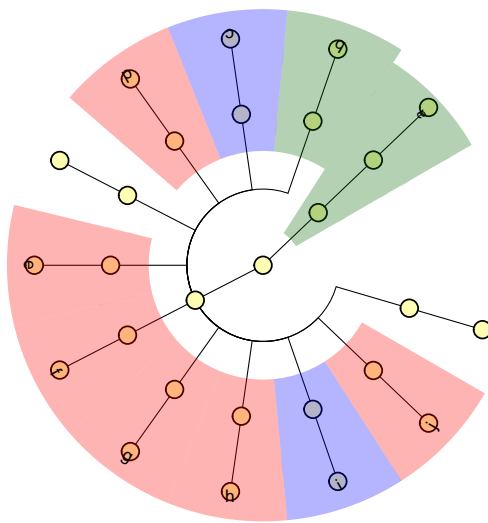
  # Save LEfSe data for the current rank
  res <- tablda[[2]]
  tree_lefse_data[[paste0("Rank_", j, "_tree_lefse_data")]] <- res # Save data in the list
}

```

```
#> [1] "A total of 6 significant features with given criteria."
#> [1] "A total of 16 significant features with given criteria."
#> [1] "A total of 35 significant features with given criteria."
#> [1] "A total of 63 significant features with given criteria."
#> [1] "A total of 89 significant features with given criteria."

# Access the plots and data for rank 2
tree_plots[["Rank_2_tree_plot"]]
```

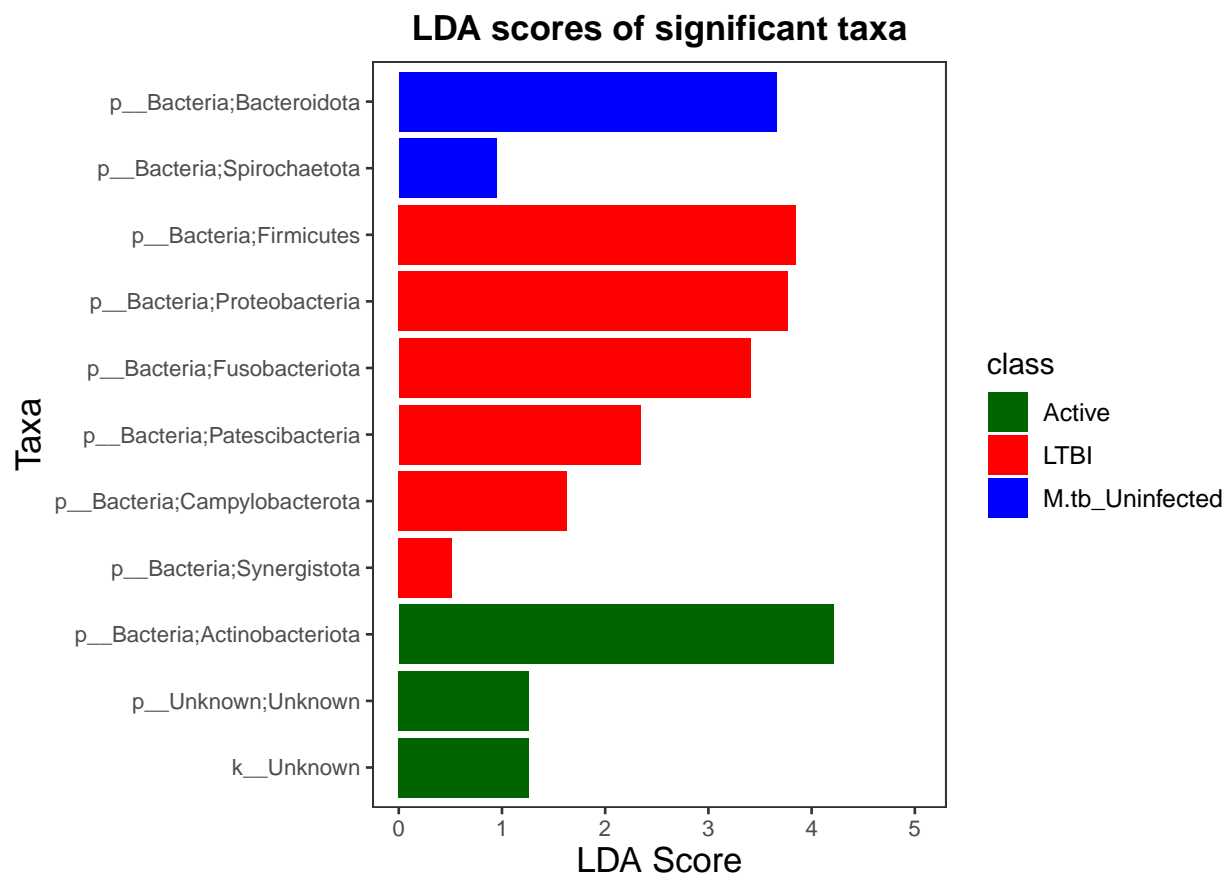
Circular Tree Plot



lab	annot
a	k__Unknown
b	Actinobacteriota
c	Bacteroidota
d	Campylobacterota
e	Firmicutes
f	Fusobacteriota
g	Patescibacteria
h	Proteobacteria
i	Spirochaetota
j	Synergistota

- LTBI
- M.tb_Uninfected
- Active

```
bar_plots[["Rank_2_bar_plot"]]
```



```
tree_lefse_data[["Rank_2_tree_lefse_data"]]
#>
#> FDR Active LTBI
#> p__Bacteria_Rank_Bacteroidota 5.7740e-09 420.90625 8494.11111
#> p__Bacteria_Rank_Fusobacteriota 9.2038e-09 231.53125 5374.22222
#> p__Bacteria_Rank_Campylobacterota 1.3014e-08 4.81250 88.88889
#> p__Bacteria_Rank_Proteobacteria 2.0776e-08 1883.03125 13637.88889
#> p__Bacteria_Rank_Actinobacteriota 3.7172e-08 34977.71875 2881.66667
#> k__Unknown 9.6045e-08 35.78125 1.44444
#> p__Unknown_Rank_Unknown 9.6045e-08 35.78125 1.44444
#> p__Bacteria_Rank_Firmicutes 5.4868e-06 11705.71875 25826.55556
#> p__Bacteria_Rank_Patescibacteria 8.2322e-06 78.53125 511.11111
#> p__Bacteria_Rank_Spirochaetota 5.6116e-04 0.96875 7.22222
#> p__Bacteria_Rank_Synergistota 1.4714e-02 0.96875 5.55556
#>
#> M.tb_Uninfected max min
#> p__Bacteria_Rank_Bacteroidota 9532.258065 9532.258065 420.906250
#> p__Bacteria_Rank_Fusobacteriota 3425.903226 5374.222222 231.531250
#> p__Bacteria_Rank_Campylobacterota 49.483871 88.888889 4.812500
#> p__Bacteria_Rank_Proteobacteria 7755.483871 13637.888889 1883.031250
#> p__Bacteria_Rank_Actinobacteriota 2464.387097 34977.718750 2464.387097
#> k__Unknown 12.419355 35.781250 1.444444
#> p__Unknown_Rank_Unknown 12.419355 35.781250 1.444444
#> p__Bacteria_Rank_Firmicutes 25587.935484 25826.555556 11705.718750
#> p__Bacteria_Rank_Patescibacteria 458.322581 511.111111 78.531250
#> p__Bacteria_Rank_Spirochaetota 16.580645 16.580645 0.968750
#> p__Bacteria_Rank_Synergistota 2.612903 5.555556 0.968750
```

```

#>                                LDA score      class      P values      color
#> p__Bacteria_Rank_Bacteroidota      3.660 M.tb_Uninfected 4.1243e-10      blue
#> p__Bacteria_Rank_Fusobacteriota      3.410          LTBI 1.3148e-09      red
#> p__Bacteria_Rank_Campylobacterota      1.630          LTBI 2.7886e-09      red
#> p__Bacteria_Rank_Proteobacteria      3.770          LTBI 5.9360e-09      red
#> p__Bacteria_Rank_Actinobacteriota      4.210          Active 1.3276e-08 darkgreen
#> k__Unknown                          1.260          Active 4.8022e-08 darkgreen
#> p__Unknown_Rank_Unknown              1.260          Active 4.8022e-08 darkgreen
#> p__Bacteria_Rank_Firmicutes           3.850          LTBI 3.1353e-06      red
#> p__Bacteria_Rank_Patescibacteria      2.340          LTBI 5.2922e-06      red
#> p__Bacteria_Rank_Spirochaetota        0.945 M.tb_Uninfected 4.0083e-04      blue
#> p__Bacteria_Rank_Synergistota         0.518          LTBI 1.1561e-02      red

# Save the Rank 2 tree plot as a high-resolution image
ggsave(filename = "Results/Figure 8A.jpg",
        plot = tree_plots[["Rank_2_tree_plot"]],
        width = 10, height = 8, dpi = 600)

# Save the Rank 2 bar plot as a high-resolution image
ggsave(filename = "Results/Figure 8B.jpg",
        plot = bar_plots[["Rank_2_bar_plot"]],
        width = 10, height = 8, dpi = 600)

```

Machine Learning

Compare machine learning models

```

MicroRoc <- function(otu = NULL, tax = NULL, map = NULL, tree = NULL,
                     ps = NULL, group_var = NULL, repnum = 5) {

  # Prepare the phyloseq object for analysis
  ps <- ggClusterNet::inputMicro(otu, tax, map, tree, ps, group = group_var)

  # Extract sample data and OTU table from the phyloseq object
  sample_mapping <- as.data.frame(phyloseq::sample_data(ps))

  # Ensure the group_var is treated as a factor with explicit levels
  sample_mapping[[group_var]] <- factor(sample_mapping[[group_var]])

  # Convert the factor levels to numeric: 0 for the first level, 1 for the second
  sample_mapping[[group_var]] <- as.numeric(sample_mapping[[group_var]]) - 1

  # Check the conversion
  table(sample_mapping[[group_var]]) # Should show counts of 0s and 1s

  otu_table <- as.data.frame(t(ggClusterNet::vegan_otu(ps)))
  colnames(otu_table) <- gsub("-", "_", colnames(otu_table))

  # Prepare the data for modeling
  data_for_modeling <- as.data.frame(t(otu_table))
}

```

```

data_for_modeling$group <- factor(sample_mapping[[group_var]]) # Ensure group is a factor

# Models to evaluate
models <- c("RF", "SVM", "GLM")
auc_scores <- list()
roc_data <- list()

# Define a common FPR grid
common_fpr <- seq(0, 1, length.out = 100)

set.seed(100) # Use the same seed value for reproducibility

for (model in models) {
  auc_values <- numeric(repnum) # Store AUC for each fold
  tpr_list <- matrix(NA, nrow = length(common_fpr), ncol = repnum) # Store interpolated TPR values

  folds <- createFolds(y = data_for_modeling$group, k = repnum)

  for (i in 1:repnum) {
    test_fold <- data_for_modeling[folds[[i]], ]
    train_fold <- data_for_modeling[-folds[[i]], ]

    if (model == "RF") {
      model_fit <- randomForest(group ~ ., data = train_fold, importance = TRUE)
      predictions <- predict(model_fit, newdata = test_fold, type = "prob")[, 2]

    } else if (model == "SVM") {
      model_fit <- svm(group ~ ., data = train_fold, probability = TRUE)
      predictions <- attr(predict(model_fit, test_fold, probability = TRUE), "probabilities")[, 2]

    } else if (model == "GLM") {
      model_fit <- glm(group ~ ., family = binomial, data = train_fold)
      predictions <- predict(model_fit, test_fold, type = "response")
    }

    # Compute ROC and interpolate TPR at common FPR
    # Ensure test_fold$group is numeric
roc_result <- roc(as.numeric(as.character(test_fold$group)), predictions)
    interpolated_tpr <- approx(x = roc_result$specificities,
                              y = roc_result$sensitivities,
                              xout = common_fpr,
                              method = "linear",
                              ties = "ordered")$y

    tpr_list[, i] <- interpolated_tpr
    auc_values[i] <- auc(roc_result)
  }

  # Calculate mean and CI for TPR at each FPR
mean_tpr <- rowMeans(tpr_list, na.rm = TRUE)
lower_ci <- apply(tpr_list, 1, quantile, probs = 0.025, na.rm = TRUE)
upper_ci <- apply(tpr_list, 1, quantile, probs = 0.975, na.rm = TRUE)

  # Store AUC results and ROC curve data

```



```

auc_scores[[model]] <- list(mean = mean(auc_values), ci = quantile(auc_values, c(0.025, 0.975)))
roc_data[[model]] <- data.frame(fpr = common_fpr, mean_tpr = mean_tpr, lower_ci = lower_ci, upper_ci = upper_ci)
}

# Function to create individual ROC plots with CI
plot_roc <- function(roc_data, auc, model_name, color) {
  ggplot(roc_data, aes(x = fpr)) +
    geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci), fill = color, alpha = 0.2) +
    geom_line(aes(y = mean_tpr), color = color, size = 1) +
    labs(x = "False Positive Rate",
         y = "True Positive Rate",
         title = sprintf("Performance of the %s Model", model_name)) +
    annotate("text", x = 0.75, y = 0.25,
            label = sprintf("AUC: %.3f (95%% CI: %.3f - %.3f)", auc$mean, auc$ci[1], auc$ci[2]), color = color) +
    theme_minimal(base_size = 14)
}

# Create plots for RF, SVM, and GLM
rf_plot <- plot_roc(roc_data$RF, auc_scores$RF, "Random Forest", "red")
svm_plot <- plot_roc(roc_data$SVM, auc_scores$SVM, "SVM", "blue")
glm_plot <- plot_roc(roc_data$GLM, auc_scores$GLM, "GLM", "black")

# Save each plot
ggsave(filename = "Results/Figure 9A.jpg", plot = rf_plot, width = 8, height = 6, dpi = 600)
ggsave(filename = "Results/Figure 9B.jpg", plot = svm_plot, width = 8, height = 6, dpi = 600)
ggsave(filename = "Results/Figure 9C.jpg", plot = glm_plot, width = 8, height = 6, dpi = 600)

return(list(rf_plot = rf_plot, svm_plot = svm_plot, glm_plot = glm_plot, auc_scores = auc_scores))
}

# Example usage
result <- MicroRoc(ps = Genexpert_phyloseq, group_var = "Group")
#> Error in approx(x = roc_result$specificities, y = roc_result$sensitivities, : need at least two non-
print(result$rf_plot)
#> Error: object 'result' not found
print(result$svm_plot)
#> Error: object 'result' not found
print(result$glm_plot)
#> Error: object 'result' not found
print(result$auc_scores)
#> Error: object 'result' not found

```

Random Forest model

```

# Function to estimate model accuracy and generate a confusion matrix table
estimate_accuracy <- function(rf_model) {
  # Extract confusion matrix from the Random Forest model
  confusion_matrix <- as.data.frame(rf_model$confusion)

  # Add a new column to calculate the class error
  confusion_matrix$class_error <- round(confusion_matrix$class.error, 3)
  confusion_matrix$Group <- row.names(confusion_matrix)
}

```

```

# Reorder columns for better readability
confusion_matrix <- dplyr::select(confusion_matrix, Group, everything())

# Calculate the overall accuracy rate of the model
accuracy_rate <- paste(round(100 - tail(rf_model$err.rate[, 1], 1) * 100, 2), "%", sep = "")
accuracy_rate_df <- data.frame(ID = "Model Accuracy Rate", Accuracy_Rate = accuracy_rate)
colnames(accuracy_rate_df) <- c("Random Forest", "Accuracy Rate")

# Create tables for the accuracy rate and confusion matrix
accuracy_table <- ggpubr::ggtexttable(accuracy_rate_df, rows = NULL)
confusion_table <- ggpubr::ggtexttable(confusion_matrix, rows = NULL)

# Combine both tables into one display
combined_tables <- accuracy_table / confusion_table

# Return the combined tables
return(combined_tables)
}

# Function to extract top important features (bacterial markers) based on Random Forest model
extract_top_features <- function(rf_model, ps, top_features = 20) {
  # Extract feature importance metrics from the Random Forest model
  feature_importance <- as.data.frame(round(randomForest::importance(rf_model), 2))

  # Assign feature IDs for better readability
  feature_importance$feature_id <- row.names(feature_importance)
  row.names(feature_importance) <- gsub("feature", "", row.names(feature_importance))
  feature_importance$feature_id <- gsub("feature", "", feature_importance$feature_id)

  # Order features by their importance (MeanDecreaseAccuracy) and select top features
  ordered_features <- dplyr::arrange(feature_importance, desc(MeanDecreaseAccuracy))
  top_features_df <- head(ordered_features, n = top_features)

  # Extract taxonomy information corresponding to the top features
  taxonomy_df <- as.data.frame(ggClusterNet::vegan_tax(ps))
  relevant_taxonomy <- taxonomy_df[row.names(top_features_df), ]

  # Merge top features with their corresponding taxonomy information
  top_features_df <- merge(top_features_df, relevant_taxonomy, by = "row.names", all = FALSE)
  row.names(top_features_df) <- top_features_df$Row.names
  top_features_df$Row.names <- NULL

  # Return the dataframe of top features along with their taxonomy
  return(top_features_df)
}

# Function to plot the importance of features and create a polar bar plot
plot_feature_importance <- function(top_features_df) {
  # Plot 1: Importance of features based on MeanDecreaseAccuracy from Random Forest
  importance_plot <- ggplot(top_features_df, aes(x = MeanDecreaseAccuracy, y = reorder(feature_id, MeanDecreaseAccuracy))) +
    geom_point(size = 6, pch = 21, fill = "#9ACD32", color = "#9ACD32") +
    geom_segment(aes(yend = feature_id, xend = 0, size = 3, color = "#9ACD32")) +
    geom_label(aes(x = MeanDecreaseAccuracy * 1.1, label = Genus), size = 3)

```

```

# Prepare data for the polar plot by arranging and indexing the top features
top_features_df <- dplyr::arrange(top_features_df, desc(MeanDecreaseAccuracy))
top_features_df$index <- paste(1:length(top_features_df$feature_id))
label_angle <- 90 - 360 * (as.numeric(top_features_df$index) - 0.5) / length(top_features_df$feature_id)
top_features_df$feature_id <- factor(top_features_df$feature_id, levels = top_features_df$feature_id)

# Plot 2: Polar bar plot of feature importance
polar_plot <- top_features_df %>%
  ggplot(aes(x = factor(feature_id), y = MeanDecreaseAccuracy, label = Genus)) +
  geom_bar(stat = 'identity', position = 'dodge', fill = "blue") +
  geom_text(hjust = 0, angle = label_angle, alpha = 1) +
  coord_polar() +
  theme_void()

# Return the plots
return(list(importance_plot = importance_plot, polar_plot = polar_plot))
}

# Function to perform recursive feature elimination cross-validation (RFCV) with statistics
run_recursive_feature_cv <- function(otu = NULL, tax = NULL, map = NULL, tree = NULL,
                                     ps = NULL, group = "Group", optimal_features = 20, num_folds = 5) {

  # Preprocess the phyloseq object and scale the microbial data
  ps <- ggClusterNet::inputMicro(otu, tax, map, tree, ps, group = group)

  # Prepare the OTU table and sample metadata
  otu_table <- as.data.frame(ggClusterNet::vegan_otu(ps))
  sample_metadata <- as.data.frame(phyloseq::sample_data(ps))

  # Set classification labels and clean up column names
  otu_table$group <- factor(sample_metadata$Group)
  colnames(otu_table) <- gsub("-", "_", colnames(otu_table))

  # Define feature data by removing the group column
  num_features <- ncol(otu_table) - 1
  feature_data <- otu_table[1:num_features]

  # Initial RFCV with a set seed
  set.seed(315)
  rfcv_result <- rfcv(feature_data, otu_table$group, cv.fold = num_folds, scale = "log", step = 0.9)

  # Store cross-validation error data for each fold
  error_cv_df <- data.frame(num_features = rfcv_result$n.var, error_rate_1 = rfcv_result$error.cv)

  # Repeat RFCV with different seeds to ensure stability
  for (seed in 316:(315 + num_folds - 1)) {
    set.seed(seed)
    rfcv_result <- rfcv(feature_data, otu_table$group, cv.fold = num_folds, scale = "log", step = 0.9)
    error_cv_df <- cbind(error_cv_df, rfcv_result$error.cv)
  }

  # Calculate mean, standard deviation, and confidence intervals for error rates
  error_cv_df <- error_cv_df[, -1] # Remove the num_features column

```

```

colnames(error_cv_df) <- paste('error_rate', 1:num_folds, sep = '_')
mean_error <- rowMeans(error_cv_df)
std_error <- apply(error_cv_df, 1, sd)

# Calculate 95% confidence intervals (assuming normal distribution)
ci_low <- mean_error - 1.96 * (std_error / sqrt(num_folds))
ci_high <- mean_error + 1.96 * (std_error / sqrt(num_folds))

# Prepare summary data for plotting and reporting
rfcv_summary <- data.frame(
  num_features = rfcv_result$n.var,
  mean_error = mean_error,
  std_error = std_error,
  ci_low = ci_low,
  ci_high = ci_high
)

# Transform the data for plotting
plot_data <- tidyr::gather(rfcv_summary, key = "metric", value = "value", -num_features)

# Generate a plot showing the RFCV results with error bars
rfcv_plot <- ggplot() +
  geom_line(data = plot_data, aes(x = num_features, y = value, group = metric, color = metric), linetype = "solid") +
  geom_ribbon(aes(x = rfcv_summary$num_features, ymin = rfcv_summary$ci_low, ymax = rfcv_summary$ci_high, fill = "black"), fill = "black", alpha = 0.1) +
  geom_line(aes(x = rfcv_summary$num_features, y = rfcv_summary$mean_error), colour = 'black') +
  coord_trans(x = "log2") +
  scale_x_continuous(breaks = c(1, 2, 5, 10, 20, 30, 50, 100, 200)) +
  labs(title = paste('Cross validation Training set (n =', nrow(otu_table), ')', sep = ''),
       x = 'Number of features',
       y = 'Cross-validation error rate') +
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5), # Customize title size and position
    axis.title.x = element_text(size = 14), # Customize x-axis title size
    axis.title.y = element_text(size = 14) # Customize y-axis title size
  ) +
  annotate("text", x = optimal_features, y = max(rfcv_summary$mean_error),
         label = paste("optimal = ", optimal_features, sep = ""))

# Return the plot and summary data
return(list(plot = rfcv_plot, summary_data = rfcv_summary))
}

# Main function to run the Random Forest analysis and generate reports and plots
micro_rf_analysis <- function(otu = NULL, tax = NULL, map = NULL, tree = NULL,
                             ps = NULL, group = "Group", top_features = 20, perform_rfcv = FALSE,
                             rfcv_folds = 5, min_ylim = -1, max_ylim = 5) {

  # Preprocess the phyloseq object and scale the microbial data
  ps <- ggClusterNet::inputMicro(otu, tax, map, tree, ps, group = group) %>% ggClusterNet::scale_micro(
    sample_data_df <- as.data.frame(phyloseq::sample_data(ps))

  # Prepare OTU table with scaled data and set up Random Forest model
  mapping_df <- as.data.frame(phyloseq::sample_data(ps))

```

```

otu_table <- as.data.frame((ggClusterNet::vegan_otu(ps)))
colnames(otu_table) <- paste("feature", colnames(otu_table), sep = "")
otu_table$group <- factor(mapping_df$Group)

# Fit the Random Forest model
rf_model <- randomForest::randomForest(group ~ ., data = otu_table, importance = TRUE, proximity = TRUE)
print(rf_model)

# Estimate the accuracy of the model and generate the confusion matrix
accuracy_table <- estimate_accuracy(rf_model)

# Extract the top important features (bacterial markers)
top_features_df <- extract_top_features(rf_model, ps, top_features)

# Generate plots for feature importance
plots <- plot_feature_importance(top_features_df)

# Perform RFCV if requested
if (perform_rfcv) {
  rfcv_result <- run_recursive_feature_cv(otu = NULL, tax = NULL, map = NULL, tree = NULL,
                                          ps = ps, group = group, optimal_features = top_features,
                                          num_folds = rfcv_folds)

  rfcv_plot <- rfcv_result[[1]]
  rfcv_table <- rfcv_result[[2]]
} else {
  rfcv_plot <- NULL
  rfcv_table <- NULL
}

# Return the results: importance plot, polar plot, RFCV plot, RFCV table, top features, accuracy table
return(list(plots$importance_plot, plots$polar_plot, rfcv_plot, rfcv_table, top_features_df, accuracy_table))
}

# Run the Random Forest analysis with specified parameters
result <- micro_rf_analysis(ps = MtbInfectionStatus_phyloseq,
                           group = "Group",
                           top_features = 40, perform_rfcv = TRUE, rfcv_folds = 5,
                           min_ylim = -1, max_ylim = 5)

#>
#> Call:
#> randomForest(formula = group ~ ., data = otu_table, importance = TRUE, proximity = TRUE)
#>
#> Type of random forest: classification
#>
#> Number of trees: 500
#> No. of variables tried at each split: 22
#>
#> OOB estimate of error rate: 19.44%
#> Confusion matrix:
#>
#> Active LTBI M.tb_Uninfected class.error
#> Active      30      0      2 0.06250000
#> LTBI        0      0      9 1.00000000
#> M.tb_Uninfected 2      1     28 0.09677419

# Access the generated plots and tables

```

```

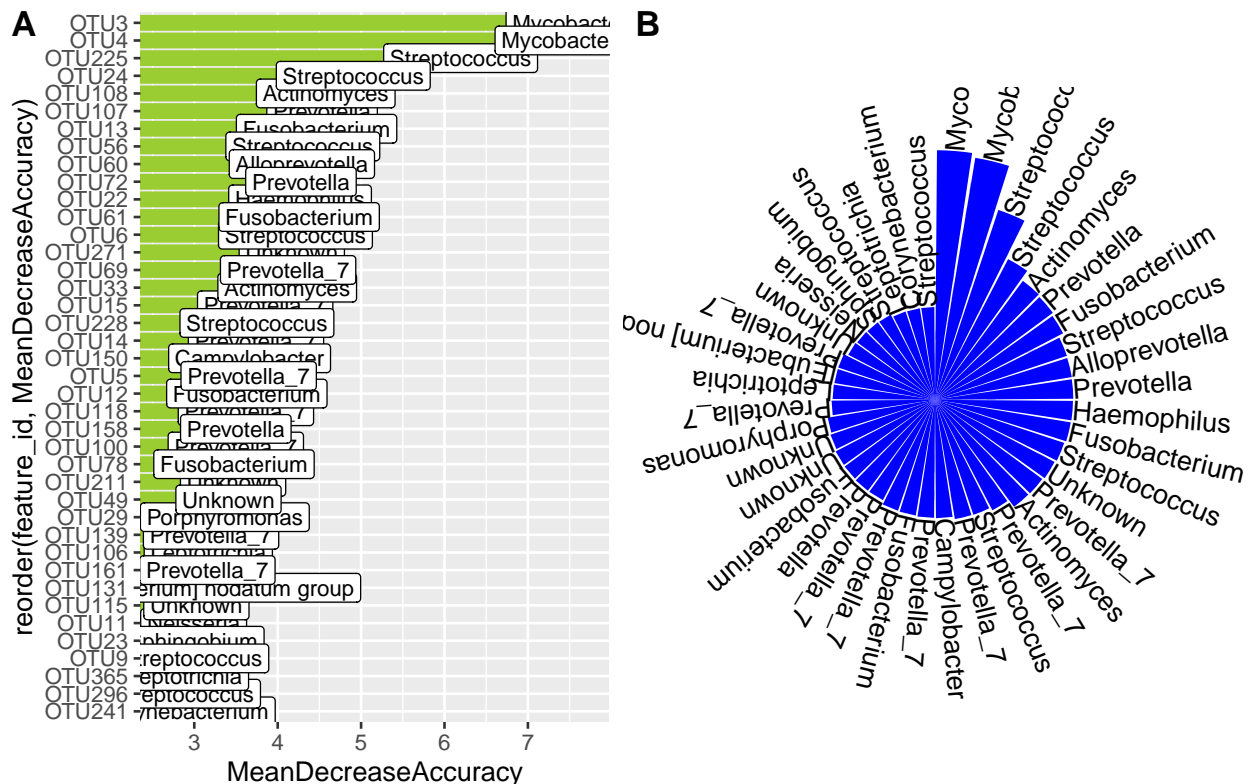
plot1 <- result[[1]] # Importance plot
plot2 <- result[[2]] # Polar plot
plot3 <- result[[3]] # RFCV plot (if perform_rfcv = TRUE)

combined_plot <- cowplot::plot_grid(plot1, plot2, labels = c("A", "B"))
# Add a title to the combined plot
combined_plot <- cowplot::ggdraw() +
  cowplot::draw_plot(combined_plot, 0, 0, 1, 0.9) + # Adjust plot position and size
  cowplot::draw_label("Combined Feature Importance and Polar Plot",
    x = 0.5, y = 0.95, hjust = 0.5, size = 16)

# Print the combined plot with title
print(combined_plot)

```

Combined Feature Importance and Polar Plot



```

# Save the plots as high-resolution images
ggsave(filename = "Results/Figure 9D.jpg", plot = combined_plot, width = 10, height = 8, dpi = 600)
ggsave(filename = "Results/Figure 9E.jpg", plot = plot3, width = 10, height = 8, dpi = 600)

```