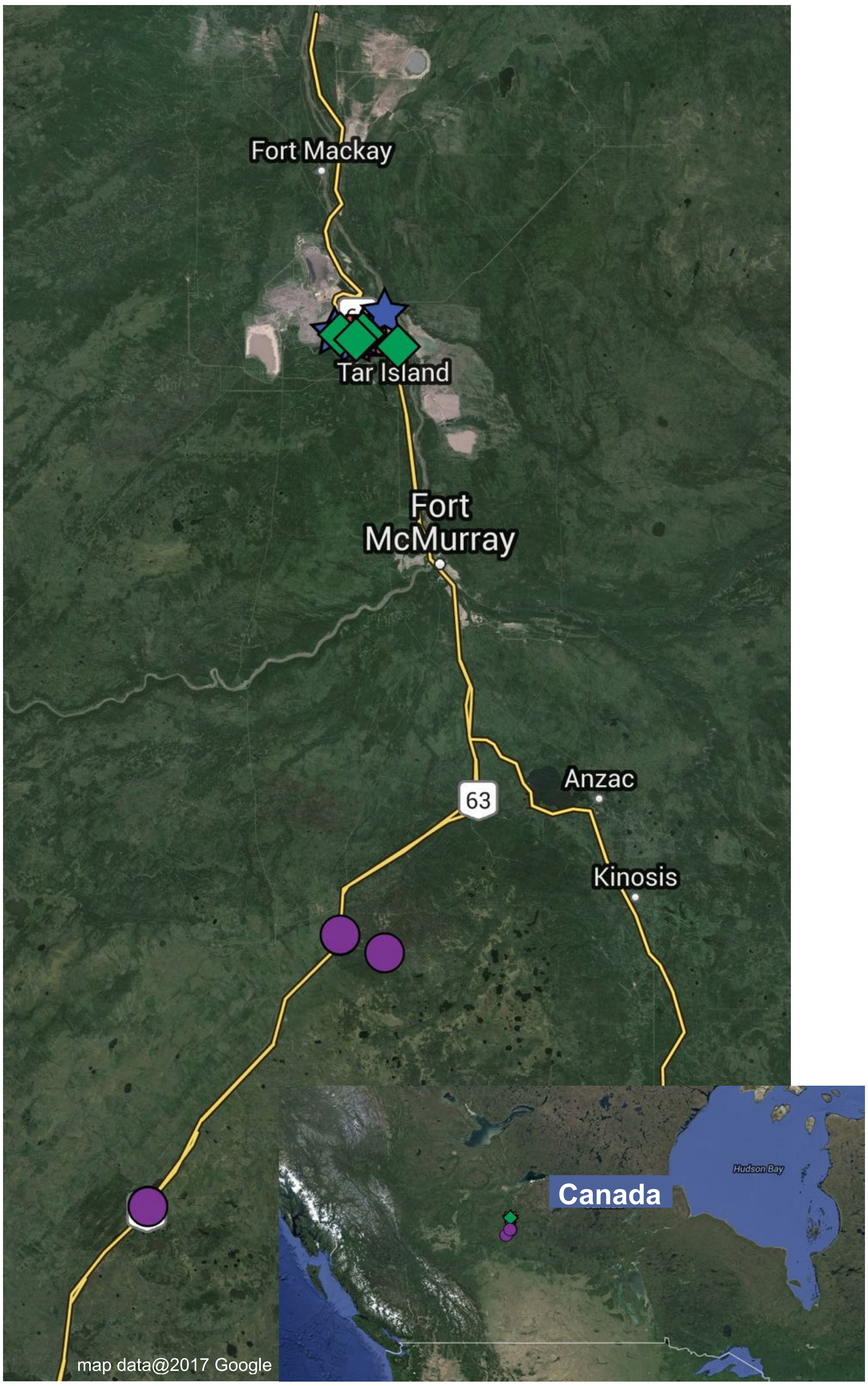


🟥 Reconstructed planted with coniferous trees

★ Reconstructed planted with deciduous trees

◆ Reconstructed planted with grasses

● Natural

Fig S1. Map of sampling sites

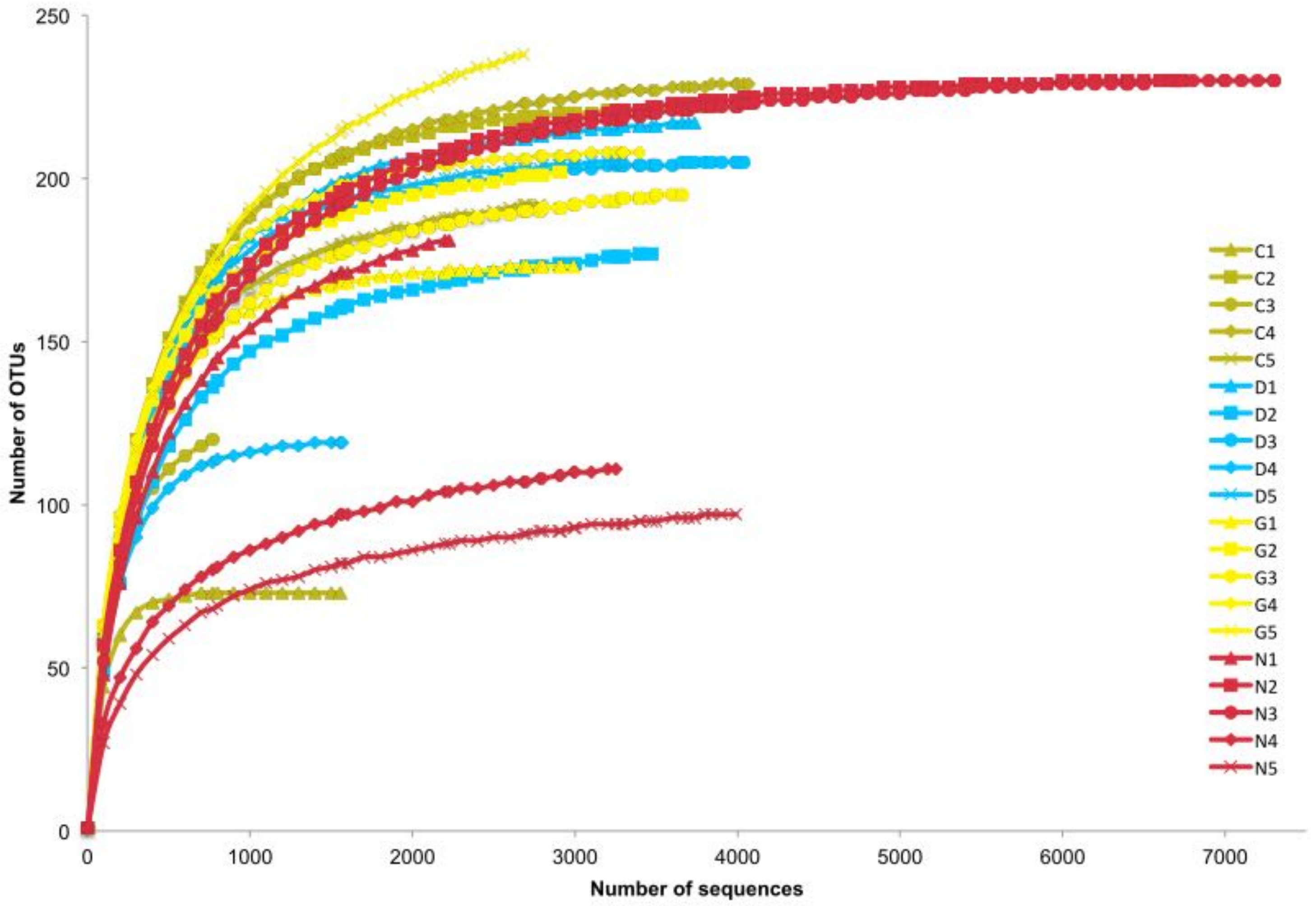


Fig S2. Rarefaction curves for reconstructed soils planted using coniferous species (C1 to C5), reconstructed soils planted using deciduous species (D1 to D5), reconstructed soils planted using grass species (G1 to G5) and in natural forest soils (N1 to N5)

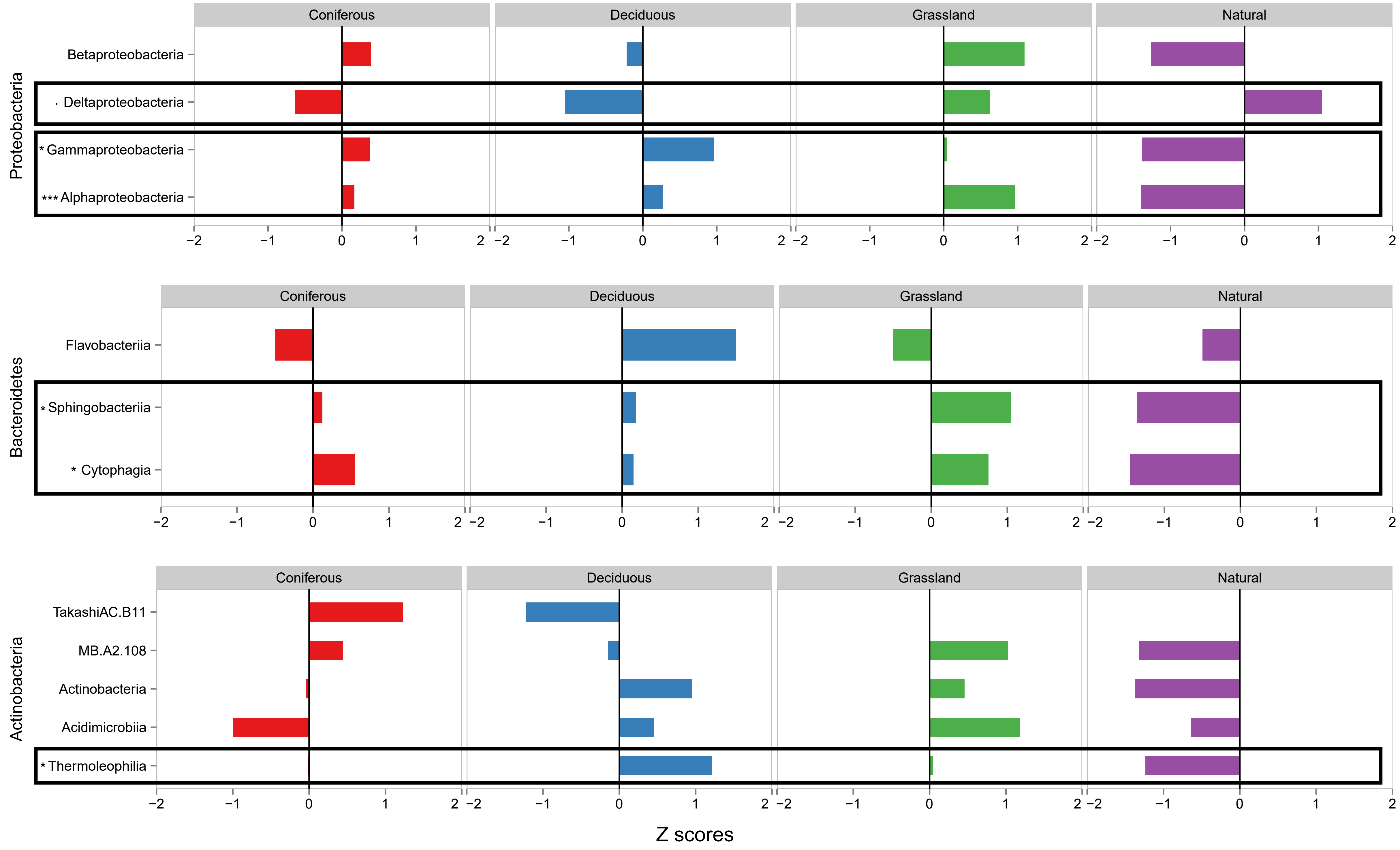


Fig S3. Z-scores of bacterial classes belonging to phyla more abundant in reconstructed soils (p<0.001: ***; p<0.01:**; p<0.05:*, p<0.1: .)

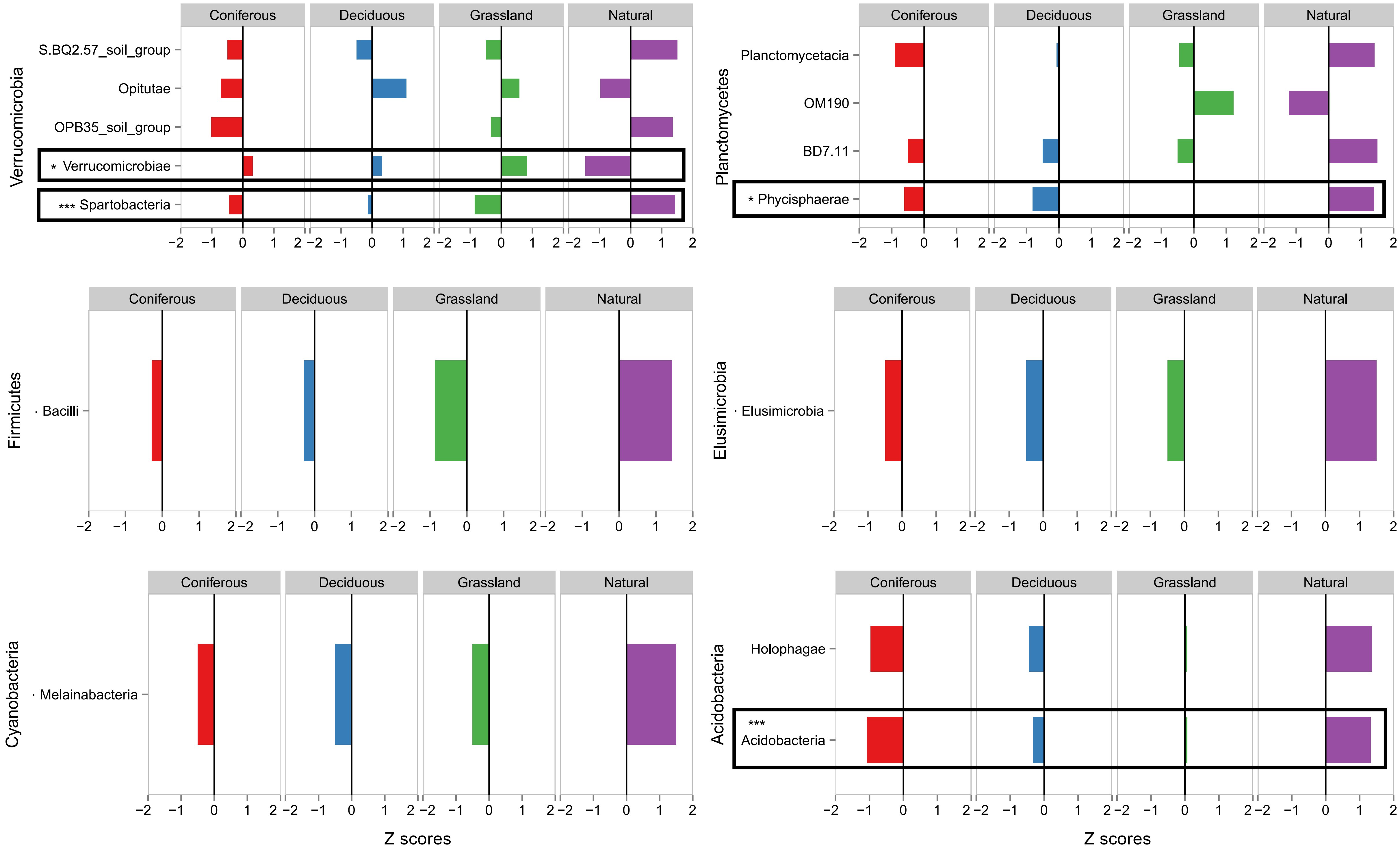


FIG S4 Z-scores of bacterial classes belonging to phyla more abundant in natural forest soils (p<0.001:***; p<0.01: **; p<0.05:*; p<0.1: .)

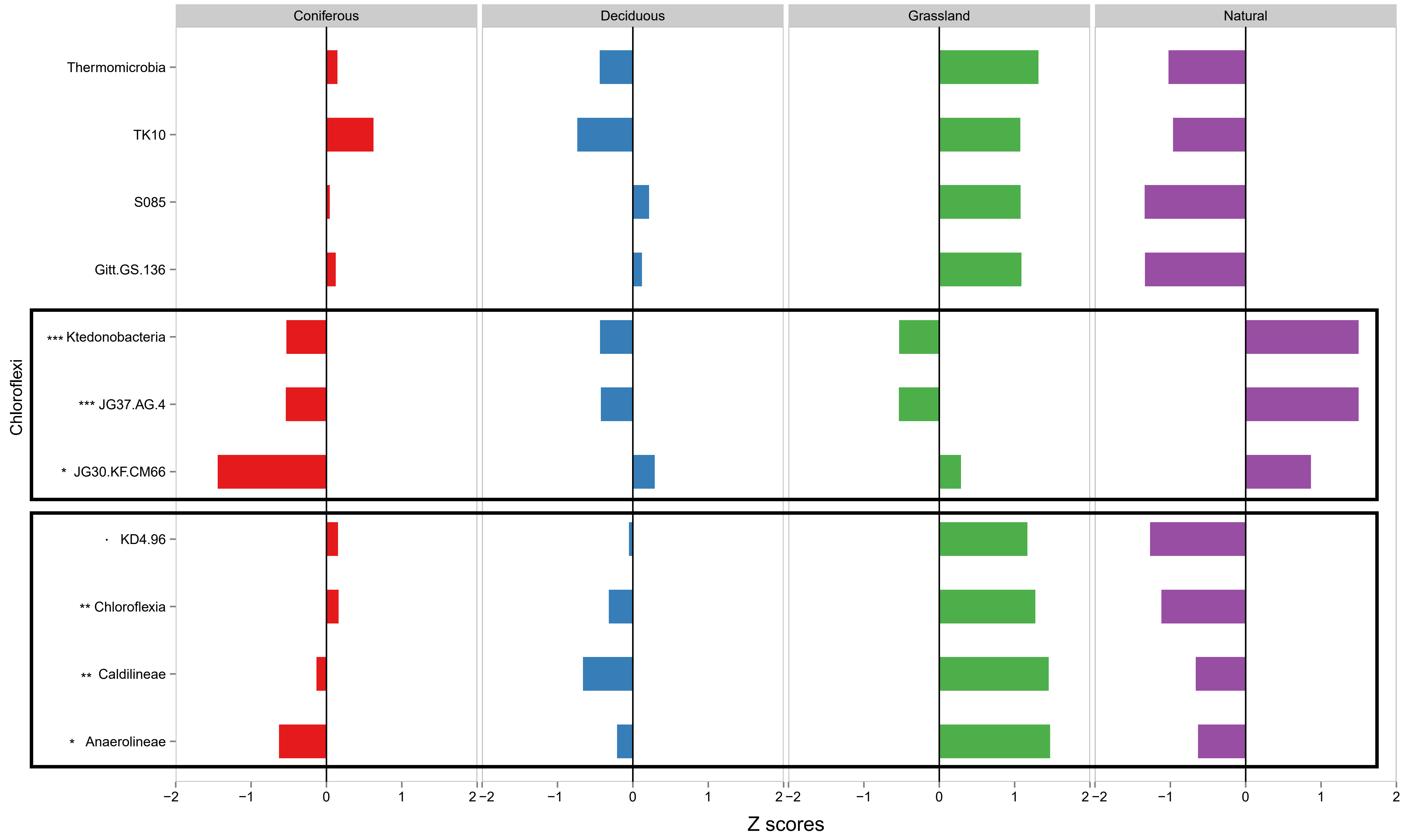


FIG S5 Z-scores of bacterial classes belonging to phyla more abundant in reconstructed soils planted with grasses (p<0.001: ***; p<0.01: **; p<0.05:*; p<0.1: .)

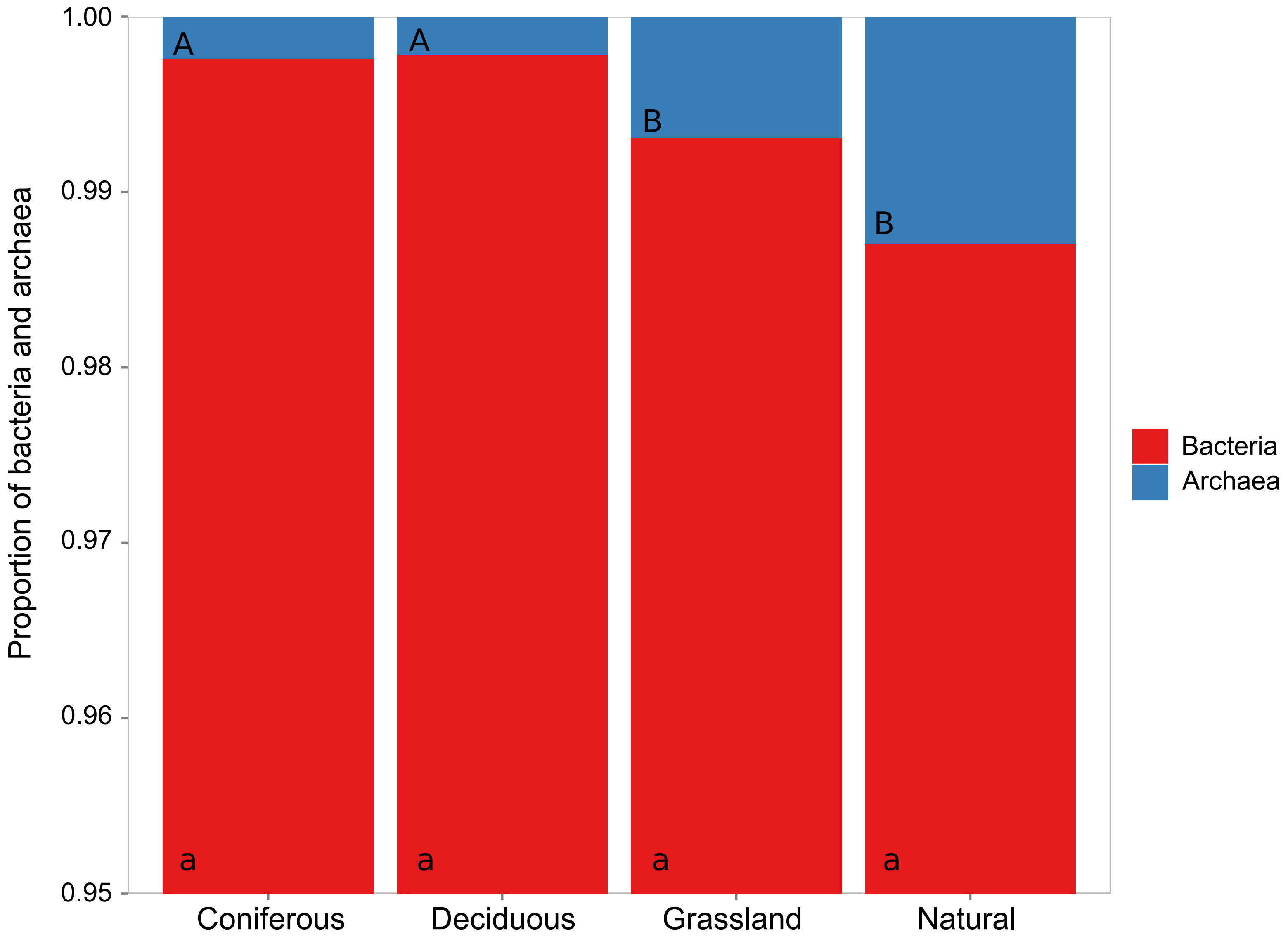
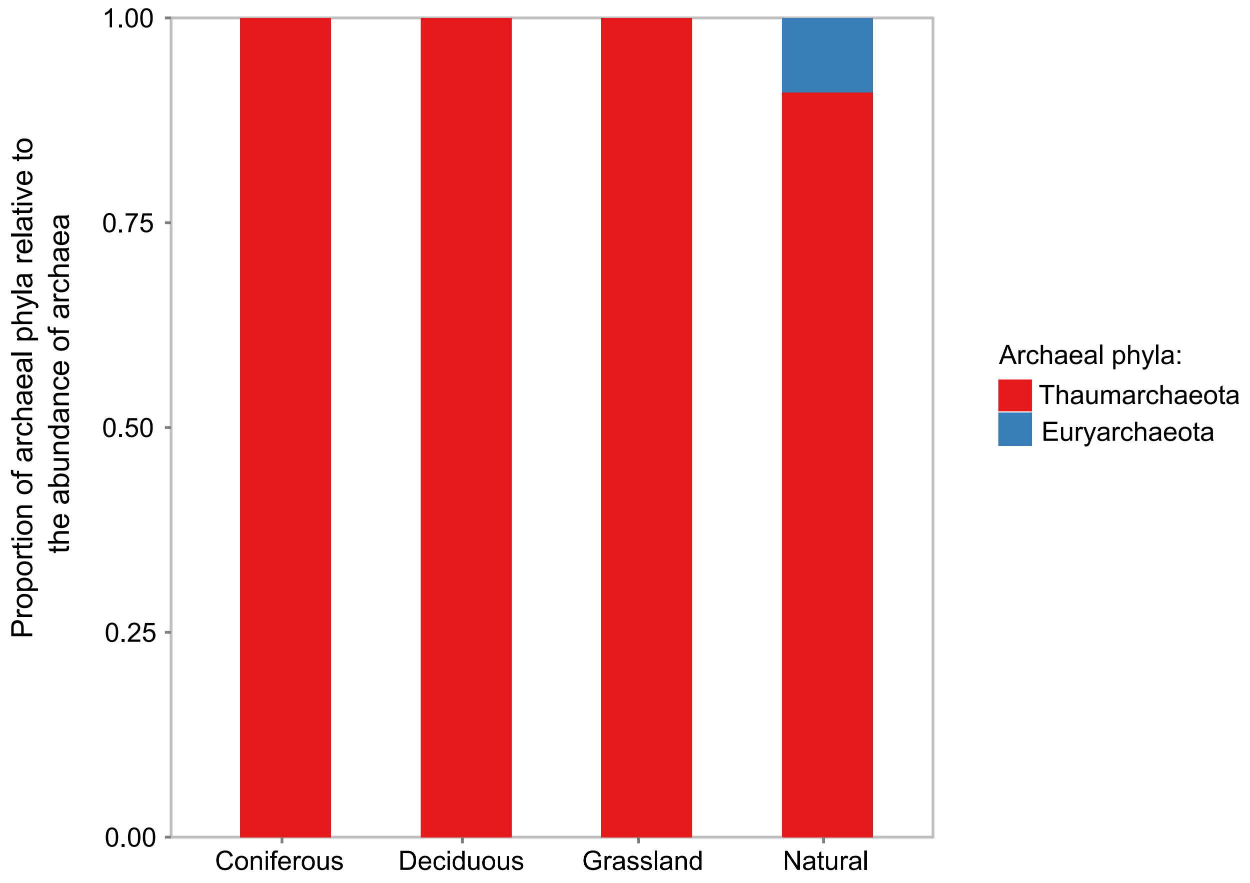


FIG S6 Relative abundance of bacterial and archaeal classes in the studied soils. Statistical differences among bacteria are identified with lower-case characters ($p>0.1$) and statistical differences among archaea are identified with upper-case characters ($p<0.01$)



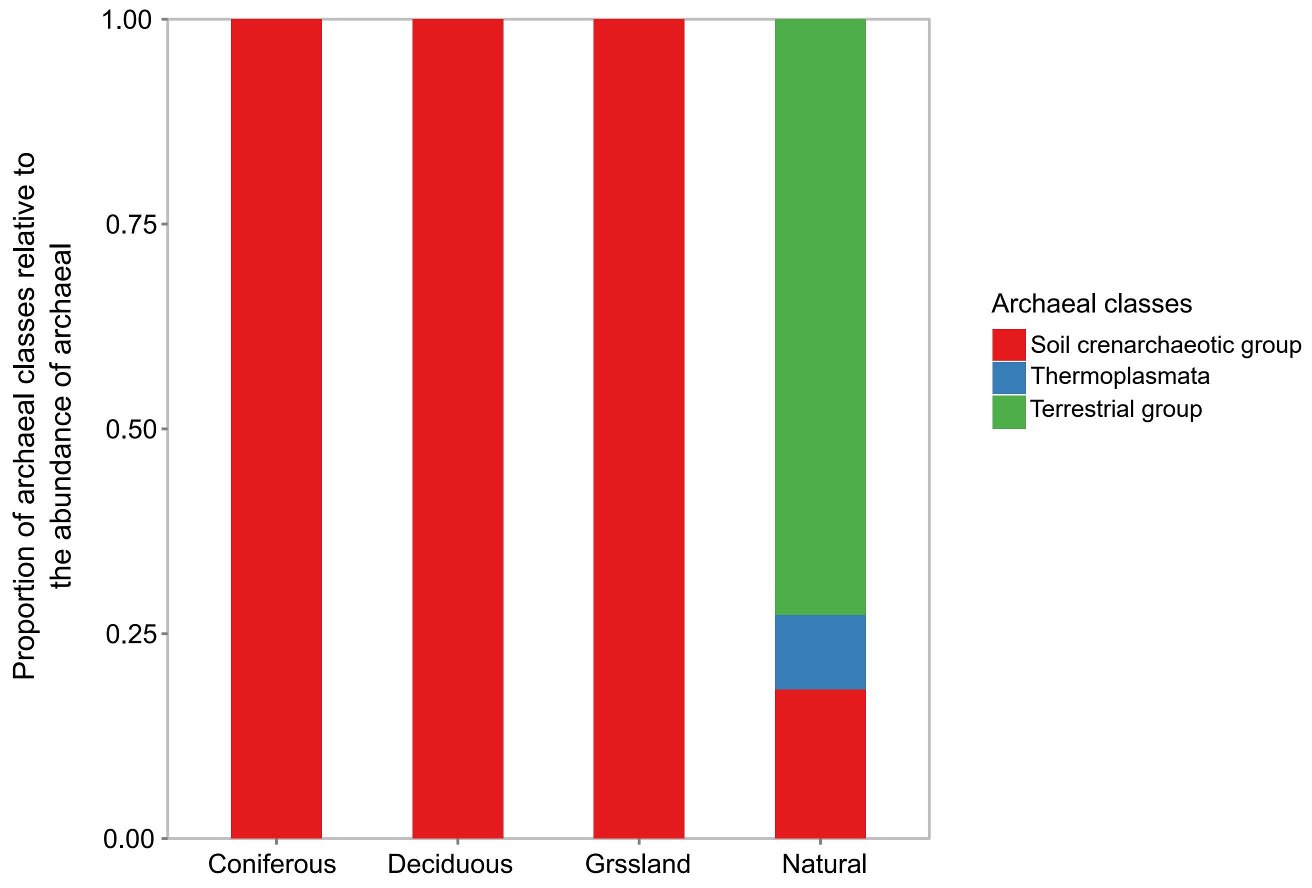


FIG S7 Relative abundance of archaeal phyla (top) and classes (bottom) in reconstructed soils planted using coniferous, deciduous and grass species and in natural forest soils.

Table S1. Number of sequences and OTU at each of the study sites

Site	Original number of reads	Reads after quality trimming and alignment	OTU
Coniferous (mean±sd)	34,646±2,068	13,456±1,219	167±68
C1	28,066	12,164	73
C2	31,026	13,100	222
C3	10,802	3,234	120
C4	34,838	14,536	229
C5	34,547	14,877	192
Deciduous (mean±sd)	27,856±9,930	11,582±4,794	185±39
D1	34,711	12,357	217
D2	31,425	12,084	177
D3	35,663	14,934	205
D4	37,021	13,657	119
D5	34,410	14,247	205
Grassland (mean±sd)	39,197±5,817	15,662±1,986	203±23
G1	30,085	13,860	173
G2	35,865	14,119	202
G3	41,313	15,588	195
G4	42,490	15,906	208
G5	45,464	18,838	238
Natural (mean±sd)	45,644±5,976	24,753±4,107	170±63
N1	39,957	19,327	181
N2	40,293	22,476	230
N3	44,740	24,546	230
N4	53,823	27,843	111
N5	49,405	29,574	97

Table S2. F values, p values and significance level of one-way ANOVA testing differences in abundance of bacterial phyla among reconstructed soils planted using coniferous, deciduous and grass species and in natural forest soils ***p<0.001; **p<0.01; *p<0.05; . p<0.1, NS: not significant (p>0.1)

Bacterial phyla	F value	p value	Level of significance
Acidobacteria	9.445	0.001	***
Actinobacteria	3.685	0.034	*
Armatimonadetes	1.000	0.418	NS
BHI80.139	1.000	0.418	NS
Bacteroidetes	8.806	0.001	**
Candidate_division_OD1	1.000	0.418	NS
Candidate_division_WS3	2.447	0.101	NS
Chloroflexi	3.072	0.058	.
Cyanobacteria	4.269	0.021	*
Elusimicrobia	2.667	0.083	.
Firmicutes	2.858	0.070	.
Gemmatimonadetes	0.717	0.556	NS
Nitrospirae	0.907	0.460	NS
Planctomycetes	4.549	0.017	*
Proteobacteria	14.381	0.000	***
TM6	1.000	0.418	NS
Verrucomicrobia	2.997	0.062	.
WD272	6.598	0.004	**
Unclassified	2.861	0.070	.

Table S3. F values, p values and significance of one-way ANOVA testing differences in abundance of bacterial classes among reconstructed soils planted using coniferous, deciduous and grass species and in natural forest soils (***p<0.001; **p<0.01; *p<0.05; . p<0.1, NS: not significant (p>0.1)

Bacterial classes	F value	p value	Level of significance
Acidobacteria	9.570	0.001	***
Acidimicrobiia	0.599	0.625	NS
Actinobacteria	2.042	0.148	NS
Alphaproteobacteria	10.367	0.000	***
Anaerolineae	3.782	0.032	*
Bacilli	2.858	0.070	.
BD7.11	1.000	0.418	NS
Betaproteobacteria	1.932	0.165	NS
Caldilineae	7.569	0.002	**
Chloroflexia	6.283	0.005	**
Chthonomonadetes	1.000	0.418	NS
Cytophagia	3.878	0.029	*
Deltaproteobacteria	2.844	0.071	.
Elusimicrobia	2.667	0.083	.
Flavobacteriia	1.000	0.418	NS
Gammaproteobacteria	4.806	0.014	*
Gemmatimonadetes	0.717	0.556	NS
Gitt.GS.136	1.340	0.296	NS
Holophagae	1.921	0.167	NS
JG30.KF.CM66	3.265	0.049	*
JG37.AG.4	10.209	0.001	***
KD4.96	2.948	0.064	.
Ktedonobacteria	13.168	0.000	***
Melainabacteria	2.664	0.083	.
Nitrospira	0.907	0.460	NS
OM190	1.346	0.295	NS
Opitutae	1.357	0.292	NS
Phycisphaerae	5.137	0.011	*
Planctomycetacia	1.680	0.211	NS
S.BQ2.57 soil group	1.000	0.418	NS
Spartobacteria	6.815	0.004	**
Sphingobacteriia	4.858	0.014	*
TakashiAC.B11	0.614	0.616	NS
Thermoleophilia	3.456	0.042	*
Thermomicrobia	1.490	0.255	NS
TK10	1.220	0.335	NS
unclassified	6.695	0.004	**
Verrucomicrobia Incertae Sedis	1.001	0.418	NS
Verrucomicrobiae	3.969	0.027	*

Plant community, soil pH and nitrogen deposition as drivers of α- and β-prokaryotic diversity in reconstructed soils and natural boreal forest soils

Masse J., Prescott, C.E., Renaut S., Terrat, Y., Grayston S.J.

Bioinformatic script using Mothur

```
###July 17th 2015
##Treatment of sequences using mothur
##Author: Jacynthe Masse
##Last update: December 3rd 2015
##Done using this tutorial: http://www.mothur.org/wiki/MiSeq_SOP


#Make contigs (using a doc txt have the forward and reverse sequences
for each sample)
make.contigs(file=StabilityFile, processors=4)


#Screening sequences to keep only sequences shorter than 500 bp and
without any ambiguity
screen.seqs(fasta=MI.M00833_0252.001.FLD0289.Micro--Com-
_007_SG_R1.trim.contigs.fasta, group=MI.M00833_0252.001.FLD0289.Micro--
Com-_007_SG_R1.contigs.groups, maxambig=0, maxlength=500)

####################Go to qiime for one step ########################

##I've combined the fasta files with qiime ##############
add_qiime_labels.py -m map16S.txt -i FastaToMerge -c InputFileName -o
combined_seq_fasta

##You'll need to create a folder "FastaToMerge" with all the fasta you
want to merge

##Change the name of the combined_seq_fasta to combined_seq.fasta

####################Go back to mothur ########################

####################Pre-treatment ########################

##Running unique sequences on the combined fasta; this reduced the time
needed for analyses and creates a .name file:
unique.seqs(fasta=combined_seqs.fasta)

##Outputs:
#combined_seqs.names
#combined_seqs.unique.fasta


##Running a count.seqs. This will create a table file that will be used
for subsequent analysis
##To create a count file with group. You first need to take your names
file and associate a group to each sequences
count.seqs(name=combined_seqs.names, group=combined_seqs.groups,
processors=4)
```

```
##It took 8 secs to create a table for 871,743 sequences
##Output: combined_seqs.count_table


summary.seqs(fasta=combined_seqs.unique.fasta,
count=combined_seqs.count_table)


##Aligned sequences
##I used silva (complete) reference database
###Again we can make our lives a bit easier by making a database
#customized to our region of interest
#(V4 here) using the pcr.seqs command. To run this command you need to
have the reference database (silva.bacteria.fasta) and know where in
that alignment your sequences start and end.
# To remove the leading and trailing dots we will set keepdots to
false. You could also run this command using your primers of interest.:

pcr.seqs(fasta=silva.nr_v119.align, start=11894, end=25319, keepdots=F,
processors=4)
##Output
### silva.nr_v119.pcr.align

#I renamed silva.nr_v119.pcr.align to silva.nr_v119.V4.align

align.seqs(fasta=combined_seqs.unique.fasta,
reference=silva.nr_v119.V4.align, flip=T, processors=4)

#It took 3762 secs to align 757592 sequences
#Outputs:
###combined_seqs.unique.align,
###combined_seqs.unique.align.report;
###combined_seqs.unique.flip.accnos

summary.seqs(fasta=combined_seqs.unique.align,
name=combined_seqs.names, processors=4)

##Screen sequences in order to remove the sequences that were not
#aligned correctly
screen.seqs(fasta=combined_seqs.unique.align,
count=combined_seqs.count_table, summary=combined_seqs.unique.summary,
start=1, end=13424, maxhomop=10, processors=4, minlength=290,
maxlength=293)

#It took 285 secs to screen 757592 sequences.
###combined_seqs.unique.good.summary
###combined_seqs.unique.good.align
###combined_seqs.unique.bad.accnos
###combined_seqs.good.count_table

summary.seqs(fasta=combined_seqs.unique.good.align,
count=combined_seqs.good.count_table, processors=4)


###Filter sequences.
```

```
##Now we know our sequences overlap the same alignment coordinates, we
##want to make sure they only overlap that region. So we'll filter the
##sequences to remove the overhangs at both ends. Since we've done
##paired-end sequencing, this shouldn't be much of an issue, but
whatever. In addition, there are many columns in the alignment that
only contain gap characters (i.e. "-").

##These can be pulled out without losing any information.
filter.seqs(fasta=combined_seqs.unique.good.align, vertical=T, trump=.)


#Output:
####combined_seqs.filter
####combined_seqs.unique.good.filter.fasta

summary.seqs(fasta=combined_seqs.unique.good.filter.fasta,
count=combined_seqs.good.count_table, processors=4)

#seems good. The final alignment has 668 columns. We might have
#introduced some redundancy in the sequences now that we have trimmed
#the sequences.

unique.seqs(fasta=combined_seqs.unique.good.filter.fasta,
count=combined_seqs.good.count_table)
#Output
####combined_seqs.unique.good.filter.count_table
####combined_seqs.unique.good.filter.unique.fasta

summary.seqs(fasta=combined_seqs.unique.good.filter.unique.fasta,
count=combined_seqs.unique.good.filter.count_table)
```

##**Let's remove more noise**:

```
pre.cluster(fasta=combined_seqs.unique.good.filter.unique.fasta,
count=combined_seqs.unique.good.filter.count_table, diffs=2)

#Output:
####combined_seqs.unique.good.filter.unique.precluster.fasta
####All the .map files for each group

summary.seqs(fasta=current, count=current)

###The majority of the cluster had 1 sequence, which is likely to be an
##artefact from sequencing. So let’s remove singletons
```

###**Removing singletons from precluster**

```
split.abund(fasta=combined_seqs.unique.good.filter.unique.precluster.fa
sta,
count=combined_seqs.unique.good.filter.unique.precluster.count_table,
cutoff=1)

## Outputs:
###combined_seqs.unique.good.filter.unique.precluster.rare.count_table
###combined_seqs.unique.good.filter.unique.precluster.abund.count_table
###combined_seqs.unique.good.filter.unique.precluster.rare.fasta
###combined_seqs.unique.good.filter.unique.precluster.abund.fasta
```

```
summary.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.a
bund.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.count_ta
ble, processors=4)
```

###**Removing chimera**

```
chimera.uchime(fasta=combined_seqs.unique.good.filter.unique.precluster
.abund.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.count_ta
ble, dereplicate=t, processors=4)

#****make sure your count does have group information***


#Outputs:
###combined_seqs.unique.good.filter.unique.precluster.abund.uchime.pick
.count_table
###combined_seqs.unique.good.filter.unique.precluster.abund.uchime.chim
eras
###combined_seqs.unique.good.filter.unique.precluster.abund.uchime.accn
os

#Running chimera.uchime with the count file will remove the chimeric
sequences from the count file, but not from the fasta

remove.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.ab
und.fasta,
accnos=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.
accnos)
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.fasta

summary.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.a
bund.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.count_table)
```

##**Classify sequences**

```
#As a final quality control step, we need to see if there are any
#"undesirables" in our dataset.
classify.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.
abund.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.count_table, reference=silva.nr_v119.V4.align,
taxonomy=silva.nr_v119.tax, cutoff=80, processors=4)

#Output:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.nr_v11
9.wang.taxonomy
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.nr_v11
9.wang.tax.summary


#Now that everything is classified we want to remove our undesirables.
#We do this with the remove.lineage command:
remove.lineage(fasta=combined_seqs.unique.good.filter.unique.precluster
.abund.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.count_table,
taxonomy=combined_seqs.unique.good.filter.unique.precluster.abund.pick.
nr_v119.wang.taxonomy, taxon=Chloroplast-Mitochondria)

#Outputs:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.nr_v11
9.wang.pick.taxonomy
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.f
asta
###combined_seqs.unique.good.filter.unique.precluster.abund.uchime.pick
.pick.count_table


summary.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.a
bund.pick.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.pick.count_table)


##So there were 8 unique sequences (159 total sequences) that were
##classify as chloroplast or mitochondria

#use lineage to remove mithochondria, chloroplast and unknown
remove.lineage(fasta=combined_seqs.unique.good.filter.unique.precluster
.abund.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.count_table,
taxonomy=combined_seqs.unique.good.filter.unique.precluster.abund.pick.
nr_v119.wang.taxonomy, taxon=Chloroplast-Mitochondria-unknown)

#Outputs:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.nr_v11
9.wang.pick.taxonomy
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.f
asta
###combined_seqs.unique.good.filter.unique.precluster.abund.uchime.pick
.pick.count_table

summary.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.a
bund.pick.pick.fasta,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.pick.count_table)

###################Cluster sequences into OTU ########################

## We can now cluster the sequences into OTUs to see how many OTUs we
have:

dist.seqs(fasta=combined_seqs.unique.good.filter.unique.precluster.abun
d.pick.pick.fasta, cutoff=0.20, processors=4)

#Output:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.d
ist
```

```
cluster(column=combined_seqs.unique.good.filter.unique.precluster.abund
.pick.pick.dist,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.pick.count_table)

##Output:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.a
n.unique_list.list


#To know how many sequences per OTU.
make.shared(list=combined_seqs.unique.good.filter.unique.precluster.abu
nd.pick.pick.an.unique_list.list,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.pick.count_table, label=0.03)

##Outputs:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.a
n.unique_list.shared
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.a
n.unique_list.12.rabund ***For all the samples


#To know the taxonomy for each of our OTUs.
classify.otu(list=combined_seqs.unique.good.filter.unique.precluster.ab
und.pick.pick.an.unique_list.list,
count=combined_seqs.unique.good.filter.unique.precluster.abund.uchime.p
ick.pick.count_table,
taxonomy=combined_seqs.unique.good.filter.unique.precluster.abund.pick.
nr_v119.wang.pick.taxonomy, label=0.03)

##Outputs:
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.a
n.unique_list.0.03.cons.tax.summary
###combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.a
n.unique_list.0.03.cons.taxonomy


###****************************Analyses*******************************

#First steps: I transformed the names for simpler ones.
system(mv
combined_seqs.unique.good.filter.unique.precluster.abund.pick.pick.an.u
nique_list.shared 16S_3rdtry.an.shared)
system(mv
stability.trim.contigs.good.unique.good.filter.unique.precluster.pick.p
ick.pick.an.unique_list.0.03.cons.taxonomy 16S_3rdtry.an.cons.taxonomy)

#We now want to do is see how many sequences we have in each sample.
We'll do this with the count.groups command:
count.groups(shared=16S_3rdtry.an.shared)


##Output:
###count.summary
```

```
#We see that our smallest sample had 769 sequences in it. That is a
#reasonable number. Despite what some say,subsampling and rarefying
#your data is an important thing to do. We'll generate a subsampled
#file for our analyses with the sub.sample command:

sub.sample(shared=16S_3rdtry.an.shared, size=769)
#Sampling 769 from each group.
#0.03

##Output:
###16S_3rdtry.an.0.03.subsample.shared


####******* OTU-based: alpha diversity******
#Let's start our analysis by analyzing the alpha diversity of the
#samples. First we will generate collector's curve of the
#Chao1 richness estimators and the inverse Simpson diversity index. To
#do this we will use the collect.single command:

collect.single(shared=16S_3rdtry.an.shared, calc=chao-invsimpson,
freq=100)
##Outputs:
###16S_3rdtry.an.NS6.invsimpson  ***We have these file for each sample
###
# We'll do this with the rarefaction.single command:

rarefaction.single(shared=16S_3rdtry.an.shared, calc=sobs, freq=100,
processors=4)
##Output:
###16S_3rdtry.an.groups.rarefaction


#Finally, let's get a table containing the number of sequences, the
#sample coverage, the number of observed OTUs, and the Inverse Simpson
#diversity estimate using the summary.single command. To standardize
#everything, let's randomly select 2441 sequences from each sample 1000
#times and calculate the average (note: that if we set subsample=T,
#then it would use the size of the smallest library):

summary.single(shared=16S_3rdtry.an.shared, calc=nseqs-coverage-sobs-
invsimpson, subsample=769)
 ##Outputs:
 ###16S_3rdtry.an.groups.ave-std.summary
 ###16S_3rdtry.an.groups.summary


 ####******* OTU-based: beta diversity

dist.shared(shared=16S_3rdtry.an.shared, calc=thetayc-jclass,
subsample=769, processors=4)

##Outputs:
combined_seqs.an.thetayc.unique.lt.dist
##16S_3rdtry.an.thetayc.0.03.lt.dist
##16S_3rdtry.an.jclass.0.03.lt.dist
##16S_3rdtry.an.thetayc.0.03.lt.ave.dist
##16S_3rdtry.an.thetayc.0.03.lt.std.dist
```

```
##16S_3rdtry.an.jclass.0.03.lt.ave.dist
##16S_3rdtry.an.jclass.0.03.lt.std.dist

amova(phylip=16S_3rdtry.an.thetayc.0.03.lt.ave.dist,
design=combined_seqs_essai2.design)

#Output:
###16S_3rdtry.an.thetayc.0.03.lt.ave.amova
```