Hierarchical-based bacterial population genomics analysis using R

Gomes-Neto JC, Pavlovikj N, Benson AK

This case study will be done using a Salmonella Newport dataset that is available on NCBI-SRA, and contains 2,365 genomes.

A list of genomes and datasets are all available here: https://figshare.com/account/home#/projects/116625 This link to Figshare requires login credentials.

How to install all packages. Packages should be installed only once. From time-to-time new versions will be available and the user is responsible for updating them accordingly. Make sure versions of programs are reported every time you use them.

When beginning running this script, you can remove the # that comes before the install.packages() function, but when running it the second time around, and subsequently, comment the function out using #. That way you avoid creating issues with dependencies, and different versions of the program.

```
# install tidyverse
# install.packages("tidyverse")
# install skimr
# install.packages("skimr")
# install vegan
# install.packages("vegan")
# install forcats
# install.packages("forcats")
# install naniar
# install.packages("naniar")
# install ggpubr
# install.packages("ggpubr")
# install ggrepel
# install.packages("ggrepel")
# install reshape2
# install.packages("reshape2")
# install reshape2
# install.packages("RColorBrewer")
# install ggtree
# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install("ggtree")
# installation of ggtree will prompt a question about installation - answer i
s "a" to install/update all dependencies
```

How to activate packages prior to utilization.

```r
# Load previously installed packages
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.
3.1 ──

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(skimr)
library(vegan)
```

```
## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.5-7
```

```r
library(forcats)
library(naniar)
```

```
##
## Attaching package: 'naniar'

## The following object is masked from 'package:skimr':
##
##     n_complete
```

```r
library(ggtree)
```

```
## ggtree v3.0.2  For help: https://yulab-smu.top/treedata-book/
##
## If you use ggtree in published research, please cite the most appropriate
paper(s):
##
## 1. Guangchuang Yu. Using ggtree to visualize data on tree-like structures.
Current Protocols in Bioinformatics, 2020, 69:e96. doi:10.1002/cpbi.96
## 2. Guangchuang Yu, Tommy Tsan-Yuk Lam, Huachen Zhu, Yi Guan. Two methods f
or mapping and visualizing associated data on phylogeny using ggtree. Molecul
ar Biology and Evolution 2018, 35(12):3041-3043. doi:10.1093/molbev/msy194
## 3. Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, Tommy Tsan-Yuk Lam.
ggtree: an R package for visualization and annotation of phylogenetic trees w
ith their covariates and other associated data. Methods in Ecology and Evolut
ion 2017, 8(1):28-36. doi:10.1111/2041-210X.12628
```

```
##
## Attaching package: 'ggtree'

## The following object is masked from 'package:tidyr':
##
##     expand

library(ggpubr)

##
## Attaching package: 'ggpubr'

## The following object is masked from 'package:ggtree':
##
##     rotate

library(ggrepel)
library(reshape2)

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

library(RColorBrewer)
```

Enter and quality control all genotypic data including: serovar-predictions (generated by SISTR), BAPS level 1 (generated by fastbaps), ST lineages (generated by mlst), and cgMLST variants (generated by SISTR). All input files were generated by the describe programs which are part of the computational platform called ProkEvo.

```
# enter the BAPS data
baps <- read_csv('~/Documents/jove_paper/data/fastbaps_partition_baps_prior_l
6.csv')

##
## ── Column specification ────────────────────────────────────────────────
──────
## cols(
##   Isolates = col_character(),
##   `Level 1` = col_double(),
##   `Level 2` = col_double(),
##   `Level 3` = col_double(),
##   `Level 4` = col_double(),
##   `Level 5` = col_double(),
##   `Level 6` = col_double()
## )

# check the first six rows of the dataset
head(baps)
```

```
## # A tibble: 6 x 7
##   Isolates   `Level 1` `Level 2` `Level 3` `Level 4` `Level 5` `Level 6`
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 SRR1002805         8        21        44        67        99       134
## 2 SRR1002816         8        25        60       103       167       235
## 3 SRR1002817         1         1         1         1         1         1
## 4 SRR1002827         1         1         1         1         1         1
## 5 SRR1002828         1         1         1         1         1         1
## 6 SRR1002830         8        25        60       103       167       235
```

```
# changing the first two column names because for hierarchical analysis we on
ly use BAPS1
colnames(baps)[1:2] <- c("id", "baps1")
# check the first six rows of the datset again to see the change in column na
mes
head(baps)
```

```
## # A tibble: 6 x 7
##   id         baps1 `Level 2` `Level 3` `Level 4` `Level 5` `Level 6`
##   <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 SRR1002805     8        21        44        67        99       134
## 2 SRR1002816     8        25        60       103       167       235
## 3 SRR1002817     1         1         1         1         1         1
## 4 SRR1002827     1         1         1         1         1         1
## 5 SRR1002828     1         1         1         1         1         1
## 6 SRR1002830     8        25        60       103       167       235
```

```
# select columns id and baps_1
baps1 <- baps %>%
              select(id, baps1)
# check the first six rows of the dataset
head(baps1)
```

```
## # A tibble: 6 x 2
##   id         baps1
##   <chr>      <dbl>
## 1 SRR1002805     8
## 2 SRR1002816     8
## 3 SRR1002817     1
## 4 SRR1002827     1
## 5 SRR1002828     1
## 6 SRR1002830     8
```

```
# quality control baps1 data
skim(baps1)
```

Data summary

Name	baps1
Number of rows	2365

Number of columns	2
_____	
Column type frequency:	
character	1
numeric	1
_____	
Group variables	None

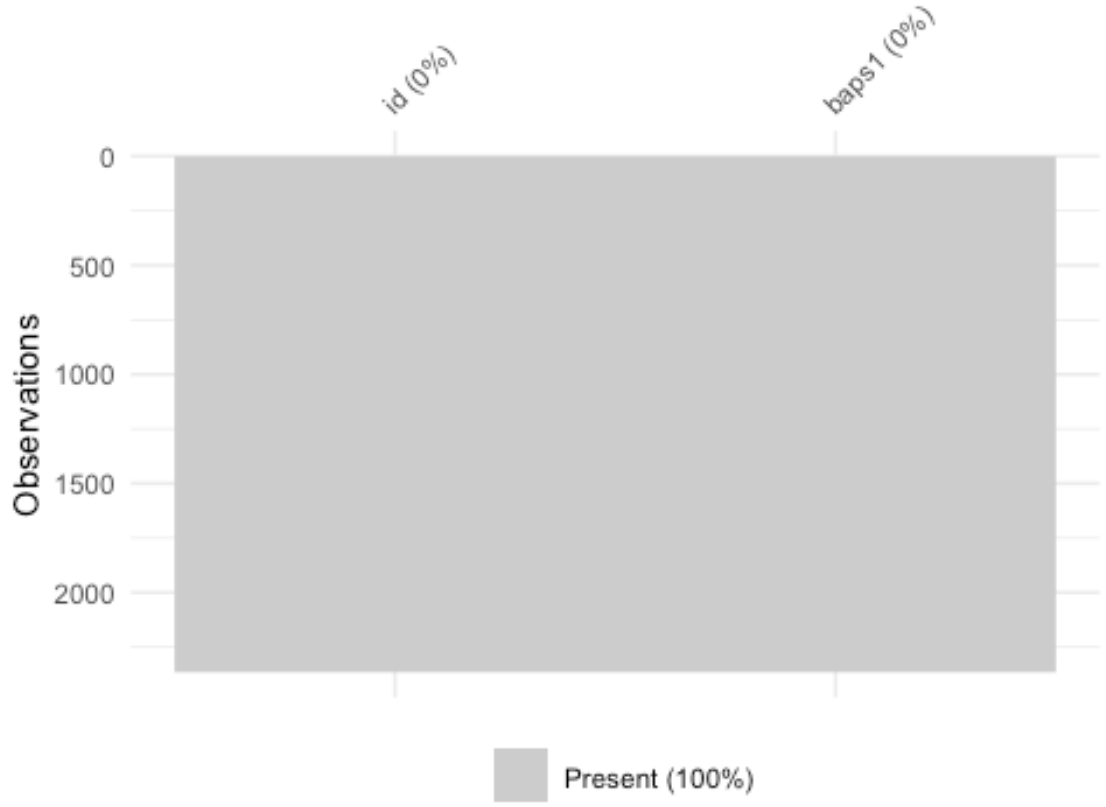
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
baps1	0	1	5.64	3.11	1	1	8	8	9	▆▁▁▁▁▆

```
# using a plotting strategy to check for missing values
vis_miss(baps1)
```




```
# rename BAPS level 1 sub-groups or haplotypes
baps1$baps_1 <- ifelse(baps1$baps1 == 1, "BAPS1 sub-group 1", # the ifelse fu
nctions tests for different conditions prior to assigning groups to one categ
ory or another
                    ifelse(baps1$baps1 == 2, "BAPS1 sub-group 2",
                           ifelse(baps1$baps1 == 3, "BAPS1 sub-group 3",
                                  ifelse(baps1$baps1 == 4, "BAPS1 sub-group 4"
,
                                         ifelse(baps1$baps1 == 5, "BAPS1 sub-g
roup 5",
                                                ifelse(baps1$baps1 == 6, "BAPS
1 sub-group 6",
                                                       ifelse(baps1$baps1 ==
7, "BAPS1 sub-group 7",
                                                              ifelse(baps1$b
aps1 == 8, "BAPS1 sub-group 8", "BAPS1 sub-group 9")))))))
##############################################-------#############################
#################
##############################################-------#############################
#################
# enter MLST results
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')
```

```
##
## ── Column specification ────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# check the first six rows of the dataset
head(mlst)

## # A tibble: 6 x 10
##   FILE                  SCHEME   ST    aroC  dnaN  hemD  hisD  purE  sucA
thrA
##   <chr>                 <chr>    <chr> <chr> <chr> <chr> <chr> <chr> <chr>
<chr>
## 1 SRR1425284_contigs.f… senteri… 45    10    7     21    14    15    12
12
## 2 SRR5760393_contigs.f… senteri… 2930  16    2     45    744   36    39
42
## 3 SRR5851178_contigs.f… senteri… 118   16    2     45    43    36    39
42
## 4 SRR5935662_contigs.f… senteri… 118   16    2     45    43    36    39
42
## 5 SRR6881504_contigs.f… senteri… 118   16    2     45    43    36    39
42
## 6 SRR5908147_contigs.f… senteri… 132   2     57    15    14    15    20
12

# generate the id column by deriving it from the FILE column
mlst$id <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# check the first six rows of the dataset
head(mlst)

## # A tibble: 6 x 11
##   FILE           SCHEME  ST    aroC  dnaN  hemD  hisD  purE  sucA  thrA  i
d
##   <chr>          <chr>   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <
chr>
## 1 SRR1425284_co… senter… 45    10    7     21    14    15    12    12    S
RR142…
## 2 SRR5760393_co… senter… 2930  16    2     45    744   36    39    42    S
RR576…
```

```
## 3 SRR5851178_co… senter… 118   16    2     45    43    36    39    42    S
RR585…
## 4 SRR5935662_co… senter… 118   16    2     45    43    36    39    42    S
RR593…
## 5 SRR6881504_co… senter… 118   16    2     45    43    36    39    42    S
RR688…
## 6 SRR5908147_co… senter… 132   2     57    15    14    15    20    12    S
RR590…
```

```
# select the id and ST columns
mlst1 <- mlst %>%
          select(id, ST) # select id and ST columns
# use the table() function to detect extraneous characters such as "-" or "?"
in the data - those are ST misclassifications
# check for the presence of ST misclassification in the ST column
table(mlst1$ST)
```

```
##
##    -   11  112  118   13  132 1370  138   14   15  164  166 1674   19 2129
2132
##    4    4    2  800    1  192    1    1    2    2    2    9    1    5    1
2
## 2166  223   23 2362 2370 2371   24   27 2855 2930 3045   31   32 3242 3494
350
##    1    1    2    3    4    2    1    1    1    1    1   42    1    1    1
40
##  367  371 3783 3834 3865   40  413 4153 4166 4190 4219  435 4450 4493   45
450
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1  643
1
##   46 4621 4628 4640  471    5  548  582  614  680  816   83   95
##   31    2    1    1    1  529    1    1    1    1    1    4    1
```

```
# check for missing values
skim(mlst1)
```

Data summary

Name	mlst1
Number of rows	2365
Number of columns	2
_____	
Column type frequency:	
character	2
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
ST	0	1	1	4	0	61	0

```
# mutate the "-" character to NA (missing value)
mlst1 <- mlst1 %>%
            mutate_all(na_if, "-") # fill all - with NAs
# check the first six rows of the dataset
head(mlst1)
```

```
## # A tibble: 6 x 2
##   id         ST
##   <chr>      <chr>
## 1 SRR1425284 45
## 2 SRR5760393 2930
## 3 SRR5851178 118
## 4 SRR5935662 118
## 5 SRR6881504 118
## 6 SRR5908147 132
```

```
# quality control baps1 data
skim(mlst1)
```

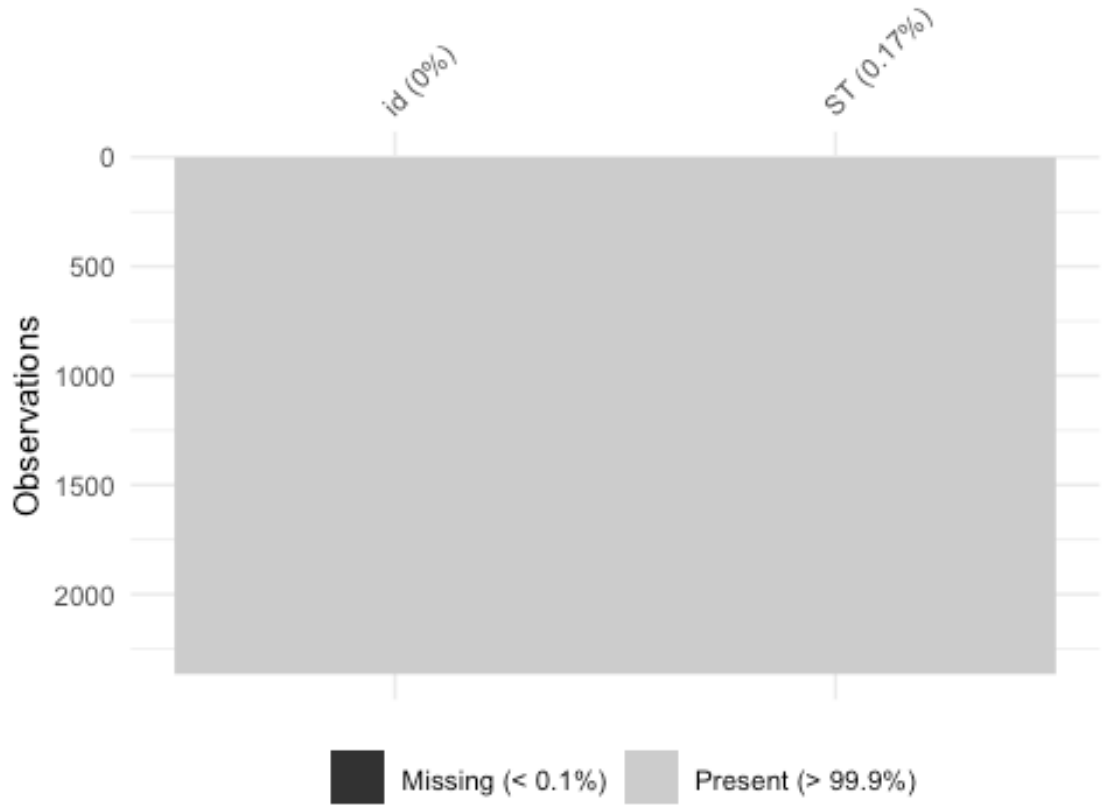
Data summary

Name	mlst1
Number of rows	2365
Number of columns	2
_____	
Column type frequency:	
character	2
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
ST	4	1	1	4	0	60	0

```
# using a plotting strategy to check for missing values
vis_miss(mlst1)
```




```
# At this stage missing values are not removed or dealt with until datasets a
re merged
##########################################--------############################
#################
##########################################-------############################
#################
# Enter SISTR results
sistr <- read_csv('~/Documents/jove_paper/data/sistr_output.csv')

##
## ── Column specification ────────────────────────────────────────────────────
──────
## cols(
##   cgmlst_ST = col_double(),
##   cgmlst_distance = col_double(),
##   cgmlst_genome_match = col_character(),
##   cgmlst_matching_alleles = col_double(),
##   cgmlst_subspecies = col_character(),
##   fasta_filepath = col_character(),
##   genome = col_character(),
##   h1 = col_character(),
##   h2 = col_character(),
##   o_antigen = col_character(),
```

```
##   qc_messages = col_character(),
##   qc_status = col_character(),
##   serogroup = col_character(),
##   serovar = col_character(),
##   serovar_antigen = col_character(),
##   serovar_cgmlst = col_character()
## )

# check the first six rows of the dataset
head(sistr)

## # A tibble: 6 x 16
##    cgmlst_ST cgmlst_distance cgmlst_genome_m… cgmlst_matching_… cgmlst_sub
speci…
##        <dbl>           <dbl> <chr>                        <dbl> <chr>
## 1         NA         0.00909 SAL_DA6272AA                   327 enterica
## 2 1297108188         0.00303 SAL_BA3042AA                   329 enterica
## 3  279966480         0.00606 SRR1122516                     328 enterica
## 4 4017665136         0.0606  SAL_EA9357AA                   310 enterica
## 5 2221566091         0.00606 SAL_EA2811AA                   328 enterica
## 6 3336043520         0.00606 SAL_FA5137AA                   328 enterica
## # … with 11 more variables: fasta_filepath <chr>, genome <chr>, h1 <chr>,
## #   h2 <chr>, o_antigen <chr>, qc_messages <chr>, qc_status <chr>,
## #   serogroup <chr>, serovar <chr>, serovar_antigen <chr>, serovar_cgmlst
<chr>

# generate the id column by deriving it from the genome column
sistr$id <- sapply(strsplit(as.character(sistr$genome),'_'), "[", 1)
# select the id and cgmlst_ST columns
sistr1 <- sistr %>%
            select(id, serovar_cgmlst, cgmlst_ST) # select id, serovar_cgmlst
, and cgmlst_ST columns
# check the first six rows of the dataset
head(sistr1)

## # A tibble: 6 x 3
##   id         serovar_cgmlst  cgmlst_ST
##   <chr>      <chr>              <dbl>
## 1 SRR1425284 Newport               NA
## 2 SRR5760393 Newport        1297108188
## 3 SRR5851178 Newport         279966480
## 4 SRR5935662 Newport        4017665136
## 5 SRR6881504 Newport        2221566091
## 6 SRR5908147 Newport        3336043520

# quality control baps1 data
skim(sistr1)
```

Data summary

Name	sistr1

Number of rows	2365
Number of columns	3

Column type frequency:

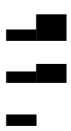
character	2
numeric	1

Group variables	None

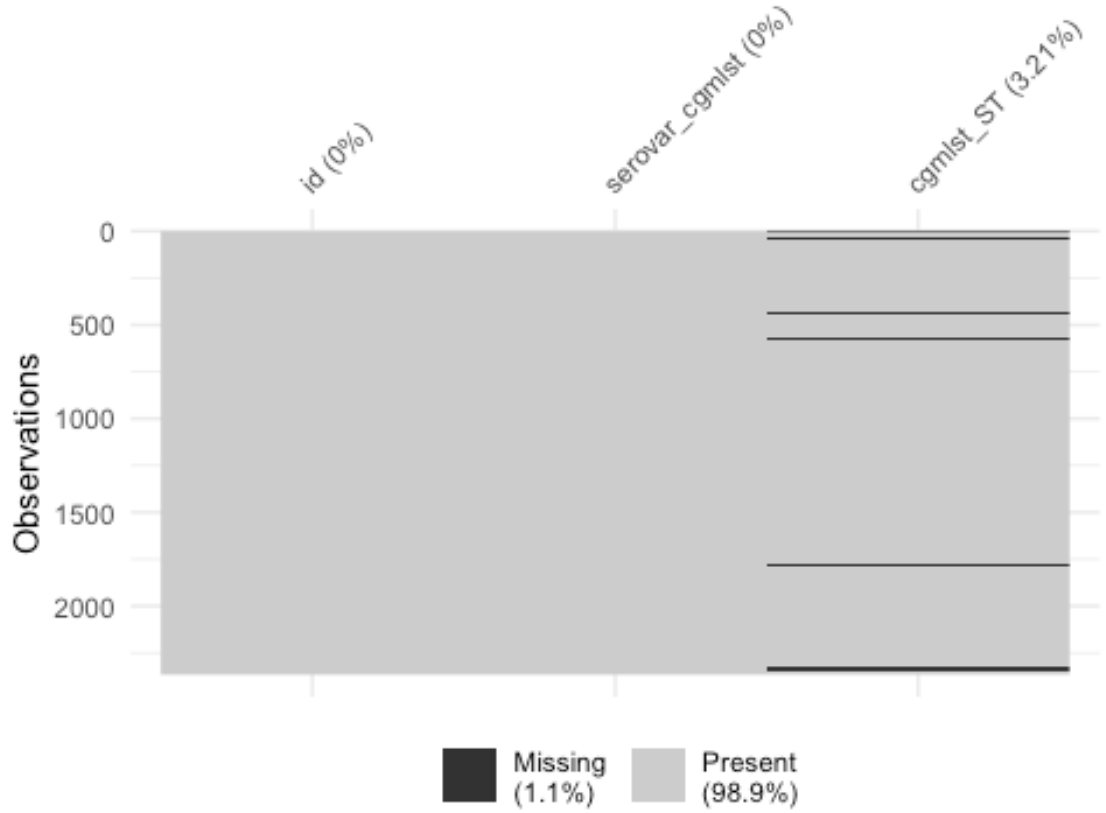
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
serovar_cgmlst	0	1	4	15	0	28	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cgmlst_ST	76	0.97	19987.99689	11508.24697	183.7685	12711.56802	16352.71886	31089.47600	42880.28711	

```
# using a plotting strategy to check for missing values
vis_miss(sistr1)
```




```
# At this stage missing values are not removed or dealt with until datasets a
re merged
##########################################--------############################
#################
##########################################-------#############################
#################
# combine baps1, mlst1, and sistr1 datasets
d1 <- left_join(baps1, mlst1, on = "id") # merge datasets on identical ids

## Joining, by = "id"

d2 <- left_join(d1, sistr1, on = "id") # merge datasets on identical ids

## Joining, by = "id"

# check data dimensionality
dim(d2)

## [1] 2365    6

# check the first six rows of the dataset
head(d2)

## # A tibble: 6 x 6
##   id         baps1 baps_1           ST    serovar_cgmlst  cgmlst_ST
```

```
##   <chr>      <dbl> <chr>              <chr> <chr>              <dbl>
## 1 SRR1002805     8 BAPS1 sub-group 8 118   Newport        1089389973
## 2 SRR1002816     8 BAPS1 sub-group 8 118   Newport           1837685
## 3 SRR1002817     1 BAPS1 sub-group 1 45    Newport        1468400426
## 4 SRR1002827     1 BAPS1 sub-group 1 45    Newport        1468400426
## 5 SRR1002828     1 BAPS1 sub-group 1 45    Newport        1468400426
## 6 SRR1002830     8 BAPS1 sub-group 8 118   Newport           1837685
```

```
# quality control baps1 data
skim(d2)
```

Data summary

Name	d2
Number of rows	2365
Number of columns	6

_____

Column type frequency:

character	4
numeric	2

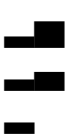
_____

Group variables	None

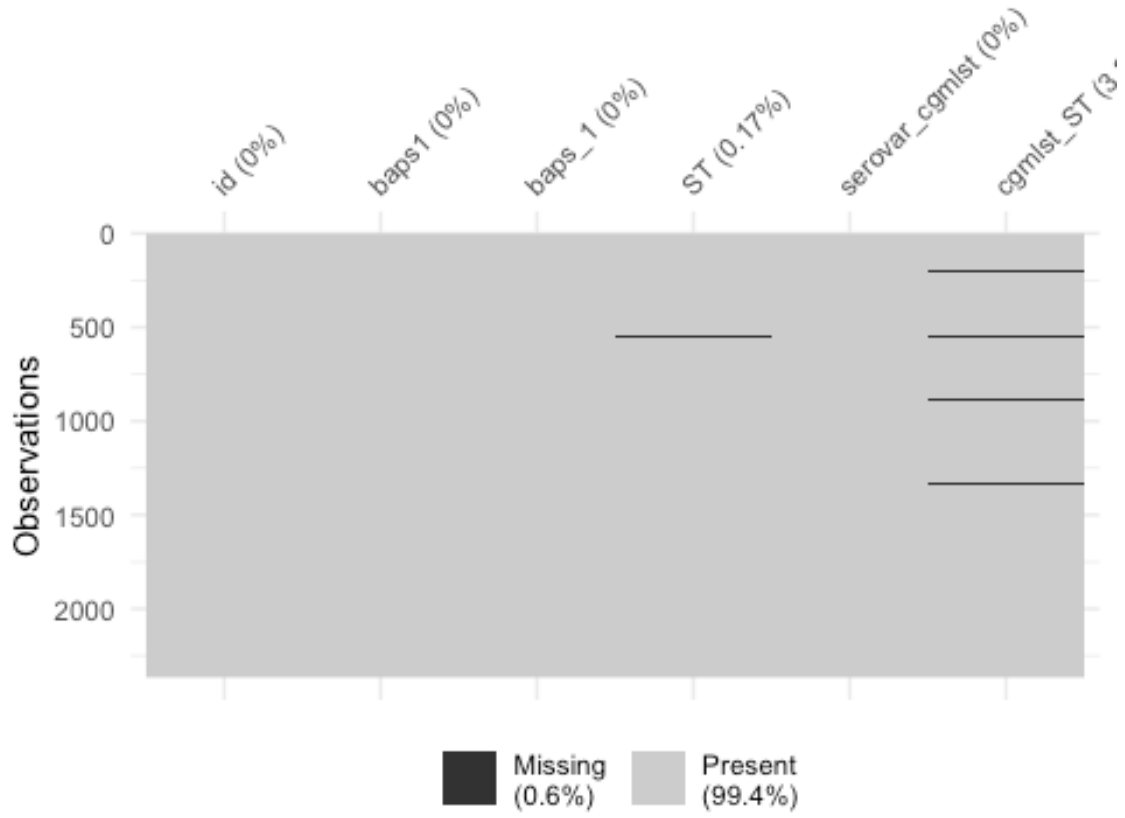
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
baps_1	0	1	17	17	0	9	0
ST	4	1	1	4	0	60	0
serovar_cgmlst	0	1	4	15	0	28	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
baps1	0	1.00	5.640000e+00	3.110000e+00	1	1	8	8	9	
cgmlst_ST	76	0.97	1.998e+09	1.1508e+09	183768	1271568	1635 2718 02	3108 9476 00	4288 0287 11	

```
# using a plotting strategy to check for missing values
vis_miss(d2)
```




```
# At this stage missing values are not removed or dealt with until datasets a
re merged
##############################################-------#############################
#################
##############################################-------#############################
#################
# group all genomes not classified as Newport as "Other serovars"
# create a new column called serovar that contains the binary classification
to group all SISTR-based misclassified genomes
d2$serovar <- ifelse(d2$serovar_cgmlst == "Newport", "Newport",
                            "Other serovars") # classify serovar_cgmlst into
Newport or Others
# check the first six rows of the dataset
head(d2)

## # A tibble: 6 x 7
##   id         baps1 baps_1            ST    serovar_cgmlst  cgmlst_ST serov
ar
##   <chr>      <dbl> <chr>             <chr> <chr>                <dbl> <chr>
## 1 SRR1002805     8 BAPS1 sub-group 8 118   Newport         1089389973 Newpo
```

```
rt
## 2 SRR1002816     8 BAPS1 sub-group 8 118   Newport           1837685 Newpo
rt
## 3 SRR1002817     1 BAPS1 sub-group 1 45    Newport        1468400426 Newpo
rt
## 4 SRR1002827     1 BAPS1 sub-group 1 45    Newport        1468400426 Newpo
rt
## 5 SRR1002828     1 BAPS1 sub-group 1 45    Newport        1468400426 Newpo
rt
## 6 SRR1002830     8 BAPS1 sub-group 8 118   Newport           1837685 Newpo
rt

# check the composition of the serovar column
table(d2$serovar)

##
##        Newport Other serovars
##           2317             48

##############################################--------############################
#################
##############################################--------############################
#################
# no transformation is needed for the baps1 dataset - no groupings or aggrega
tions are needed
##############################################--------############################
#################
##############################################--------############################
#################
# check ST distribution to focus on major STs for all subsequent analyses by
grouping by ST and counting
st_dist <- d2 %>%
  group_by(ST) %>% # group by the ST column
  count() %>% # count the number of observations
  arrange(desc(n)) # arrange the counts in decreasing order
st_dist

## # A tibble: 61 x 2
## # Groups:   ST [61]
##    ST        n
##    <chr> <int>
##  1 118     800
##  2 45      643
##  3 5       529
##  4 132     192
##  5 31       42
##  6 350      40
##  7 46       31
##  8 166       9
##  9 19        5
```

```
## 10 11        4
## # … with 51 more rows

# based on the frequency analysis, the following STs were not aggregrated: ST
118, ST45, ST5, ST132, ST31, ST350, and ST46
# create a new st column
d2$st <- ifelse(d2$ST == 5, "ST5", # create a new ST column for which minor S
Ts are aggregated as Others
                  ifelse(d2$ST == 31, "ST31",
                         ifelse(d2$ST == 45, "ST45",
                                ifelse(d2$ST == 46, "ST46",
                                       ifelse(d2$ST == 118, "ST118",
                                              ifelse(d2$ST == 132, "ST132",
                         ifelse(d2$ST == 350, "ST350", "Other STs")))))))
# check the first six rows of the dataset
head(d2)

## # A tibble: 6 x 8
##   id         baps1 baps_1           ST    serovar_cgmlst cgmlst_ST serovar
st
##   <chr>      <dbl> <chr>            <chr> <chr>               <dbl> <chr>
<chr>
## 1 SRR1002805     8 BAPS1 sub-group… 118   Newport           1.09e9 Newport
ST118
## 2 SRR1002816     8 BAPS1 sub-group… 118   Newport           1.84e6 Newport
ST118
## 3 SRR1002817     1 BAPS1 sub-group… 45    Newport           1.47e9 Newport
ST45
## 4 SRR1002827     1 BAPS1 sub-group… 45    Newport           1.47e9 Newport
ST45
## 5 SRR1002828     1 BAPS1 sub-group… 45    Newport           1.47e9 Newport
ST45
## 6 SRR1002830     8 BAPS1 sub-group… 118   Newport           1.84e6 Newport
ST118

##########################################-------############################
#################
##########################################-------############################
#################
# check cgMLST distribution by ST to focus on major cgMLSTs for all subsequen
t analyses by grouping by STs and cgMLSTs and counting
cgmlst_dist <- d2 %>%
  group_by(st, cgmlst_ST) %>% # group by st and cgmlst_ST
  count() %>% # count the number of observations
  arrange(desc(n)) # arrange in descending order
cgmlst_dist

## # A tibble: 777 x 3
## # Groups:   st, cgmlst_ST [777]
##    st     cgmlst_ST     n
##    <chr>      <dbl> <int>
```

```
##  1 ST45  1468400426   319
##  2 ST118 1271156802    96
##  3 ST132 3336043520    86
##  4 ST5     88443731    84
##  5 ST118 1492716119    64
##  6 ST5   3491314984    50
##  7 ST45  2245200879    39
##  8 ST118 2103519905    34
##  9 ST132 3109396922    34
## 10 ST5   3108947600    34
## # … with 767 more rows

# for the purposes of this analysis only the top four most frequent cgMLSTs w
ere selected and respectively renamed
d2 <- mutate(d2, cgmlst = ifelse(cgmlst_ST %in% 1468400426, "cgMLST 146840042
6", # create a new cgMLST column while aggregating minor cgMLST variants
                                 ifelse(cgmlst_ST %in% 88443731, "cgMLST
88443731",
                                        ifelse(cgmlst_ST %in% 1271156802,
"cgMLST 1271156802",
                                               ifelse(cgmlst_ST %in% 333604
3520, "cgMLST 3336043520",
                                        "Other cgMLSTs")))))
# check the first six rows of the dataset
head(d2)

## # A tibble: 6 x 9
##   id      baps1 baps_1    ST    serovar_cgmlst cgmlst_ST serovar st    cgm
lst
##   <chr>   <dbl> <chr>     <chr> <chr>              <dbl> <chr>   <chr> <ch
r>
## 1 SRR100…     8 BAPS1 su… 118   Newport           1.09e9 Newport ST118 Oth
er cg…
## 2 SRR100…     8 BAPS1 su… 118   Newport           1.84e6 Newport ST118 Oth
er cg…
## 3 SRR100…     1 BAPS1 su… 45    Newport           1.47e9 Newport ST45  cgM
LST 1…
## 4 SRR100…     1 BAPS1 su… 45    Newport           1.47e9 Newport ST45  cgM
LST 1…
## 5 SRR100…     1 BAPS1 su… 45    Newport           1.47e9 Newport ST45  cgM
LST 1…
## 6 SRR100…     8 BAPS1 su… 118   Newport           1.84e6 Newport ST118 Oth
er cg…

##########################################-------############################
#################
##########################################-------############################
#################
# select only needed columns for all subsequent analyses
d3 <- d2 %>%
```

```
          select(id, serovar, baps_1, st, cgmlst) # select columns of interes
t which are described within parenthesis
# check data dimensionality to make sure it matches that of d2
dim(d3)

## [1] 2365    5

# check the first six rows of the dataset
head(d3)

## # A tibble: 6 x 5
##   id          serovar baps_1             st    cgmlst
##   <chr>       <chr>   <chr>              <chr> <chr>
## 1 SRR1002805 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs
## 2 SRR1002816 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs
## 3 SRR1002817 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 4 SRR1002827 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 5 SRR1002828 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 6 SRR1002830 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs

# quality control baps1 data
skim(d3)
```

Data summary

Name	d3
Number of rows	2365
Number of columns	5

_____

Column type frequency:	
character	5

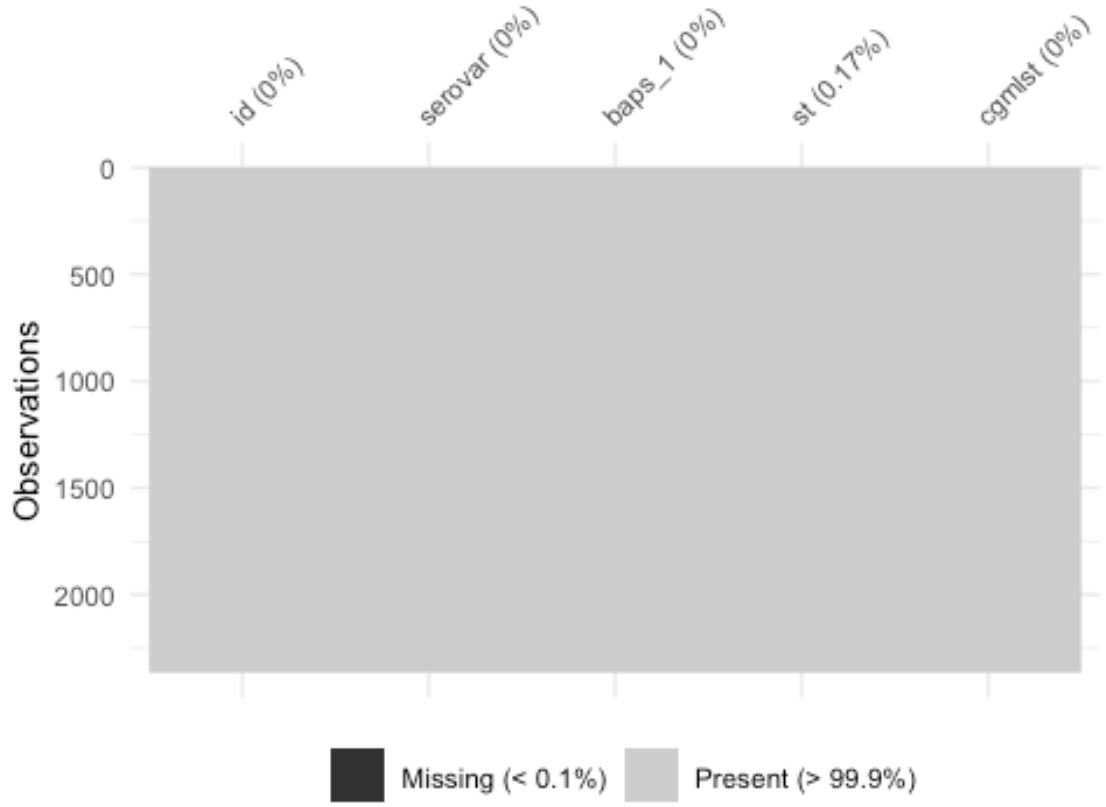
_____

Group variables	None

Variable type: character

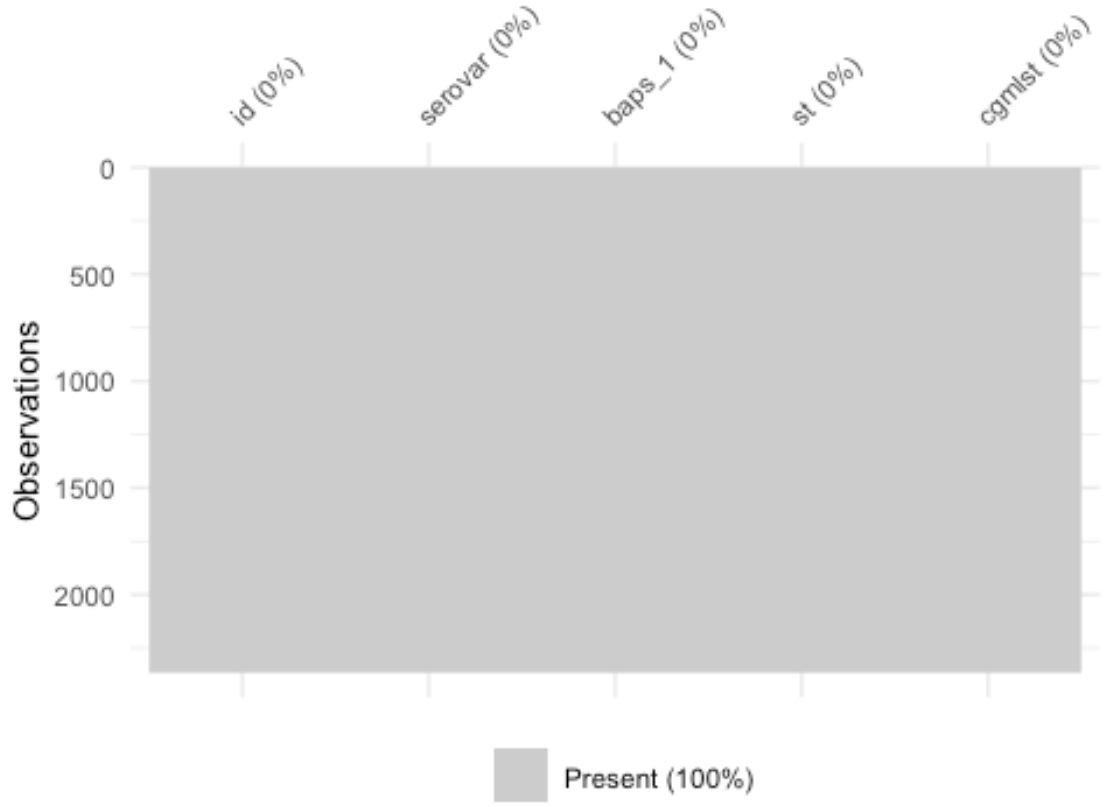
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
serovar	0	1	7	14	0	2	0
baps_1	0	1	17	17	0	9	0
st	4	1	3	9	0	8	0
cgmlst	0	1	13	17	0	5	0

```
# using a plotting strategy to check for missing values
vis_miss(d3)
```




```
# make sure there is no NA in the dataset
sum(is.na(d3))

## [1] 4

# replace NA in the ST column
d3 <- d3 %>% mutate(st = replace_na(st, "Other STs"))
# make sure there is no NA in the dataset
sum(is.na(d3))

## [1] 0

# check the distribution of serovars, baps1, st, and cgmlst classifications
table(d3$serovar)

##
##        Newport Other serovars
##           2317             48

table(d3$baps_1)

##
## BAPS1 sub-group 1 BAPS1 sub-group 2 BAPS1 sub-group 3 BAPS1 sub-group 4
##               648                 2                 7               235
## BAPS1 sub-group 5 BAPS1 sub-group 6 BAPS1 sub-group 7 BAPS1 sub-group 8
```

```
##                 9                 9                32              1394
## BAPS1 sub-group 9
##               29

table(d3$st)

##
## Other STs     ST118     ST132      ST31     ST350      ST45      ST46
ST5
##        88       800       192        42        40       643        31
529

table(d3$cgmlst)

##
## cgMLST 1271156802 cgMLST 1468400426 cgMLST 3336043520   cgMLST 88443731
##                97               321                86                 85
##     Other cgMLSTs
##              1776

# check the first six rows in the dataset
head(d3)

## # A tibble: 6 x 5
##   id         serovar baps_1            st    cgmlst
##   <chr>      <chr>   <chr>             <chr> <chr>
## 1 SRR1002805 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs
## 2 SRR1002816 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs
## 3 SRR1002817 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 4 SRR1002827 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 5 SRR1002828 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 6 SRR1002830 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs

# check the last six rows in the dataset
tail(d3)

## # A tibble: 6 x 5
##   id        serovar baps_1            st    cgmlst
##   <chr>     <chr>   <chr>             <chr> <chr>
## 1 SRR952683 Newport BAPS1 sub-group 8 ST118 Other cgMLSTs
## 2 SRR953548 Newport BAPS1 sub-group 1 ST45  Other cgMLSTs
## 3 SRR953551 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 4 SRR980337 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 5 SRR980338 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426
## 6 SRR980354 Newport BAPS1 sub-group 1 ST45  cgMLST 1468400426

# check for missing values
vis_miss(d3)
```




```
# check the data characteristics
str(d3)

## tibble [2,365 × 5] (S3: tbl_df/tbl/data.frame)
##  $ id     : chr [1:2365] "SRR1002805" "SRR1002816" "SRR1002817" "SRR100282
7" ...
##  $ serovar: chr [1:2365] "Newport" "Newport" "Newport" "Newport" ...
##  $ baps_1 : chr [1:2365] "BAPS1 sub-group 8" "BAPS1 sub-group 8" "BAPS1 su
b-group 1" "BAPS1 sub-group 1" ...
##  $ st     : chr [1:2365] "ST118" "ST118" "ST45" "ST45" ...
##  $ cgmlst : chr [1:2365] "Other cgMLSTs" "Other cgMLSTs" "cgMLST 146840042
6" "cgMLST 1468400426" ...
```

Figure 1. Visualize the phylogeny-based hierarchical distribution of genotypes.

At the center of the figure is the core-genome phylogeny. Genotypic information is plotted onto it in the following order: Serovar (innermost) BAPS1 ST cgMLST (outermost)

```
# enter the phylogeny data
tree <- read.tree("~/Documents/jove_paper/data/newport_phylogeny.tree")
# recall the metadata combining all hierarchical genotypes
# here I assign d3 to d4 just to keep the formatted metadata read for other s
teps, in case I need to do
# any transformation with the data to plot with the phylogenetic tree
```

```
d4 <- d3
# to plot with the phylogeny using ggtree we need to make the id column into
index first
d4 <- column_to_rownames(d4, var = "id")
# create the tree
# adjusting the parameters to make the tree visible or however you wish requi
res some trial and error
tree_plot <- ggtree(tree, layout = "circular") + xlim(-250, NA)
# plot figure 1 - color scheme for each layer of the plot should be chosen ba
sed on the user preferences
figure_1 <- gheatmap(tree_plot, d4, offset=.0, width=20, colnames = FALSE) +
# visualize the tree with metadata
  scale_fill_manual(values = c("coral", "darkblue",
                               "cornflowerblue", "coral", "purple", "red", "b
rown", "darkseagreen3", "darkblue", "darkgreen", "yellow",
                               "orange", "darkblue", "purple", "darkgreen", "
darkred", "steelblue", "black", "coral",
                               "black", "darkgreen", "purple", "darkblue", "g
ray"),
                    breaks = c("Newport", "Other serovars",
                               "BAPS1 sub-group 1", "BAPS1 sub-group 2", "BAP
S1 sub-group 3", "BAPS1 sub-group 4",
                               "BAPS1 sub-group 5", "BAPS1 sub-group 6", "BAP
S1 sub-group 7", "BAPS1 sub-group 8",
                               "BAPS1 sub-group 9",
                               "ST5", "ST31", "ST45", "ST46", "ST118", "ST132
", "ST350", "Other STs",
                               "cgMLST 1468400426", "cgMLST 1271156802", "cgM
LST 3336043520", "cgMLST 88443731", "Other cgMLSTs"),
                    name="Serovar (innermost) -> BAPS1 -> ST -> cgMLST (outer
most)") +  # add colors and labels for the scale
    ggtitle(expression(bold(paste("Hierarchical population structure - ", bol
ditalic(S),". Newport")))) +  # add title for the figure
  theme(plot.title = element_text(size = 40, face = "bold"),
        legend.title=element_text(size=24, face = "bold"),
    legend.text=element_text(size=22))  # customize figure's title, legend, f
ont

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.

## Scale for 'fill' is already present. Adding another scale for 'fill', whic
h
## will replace the existing scale.

figure_1
```

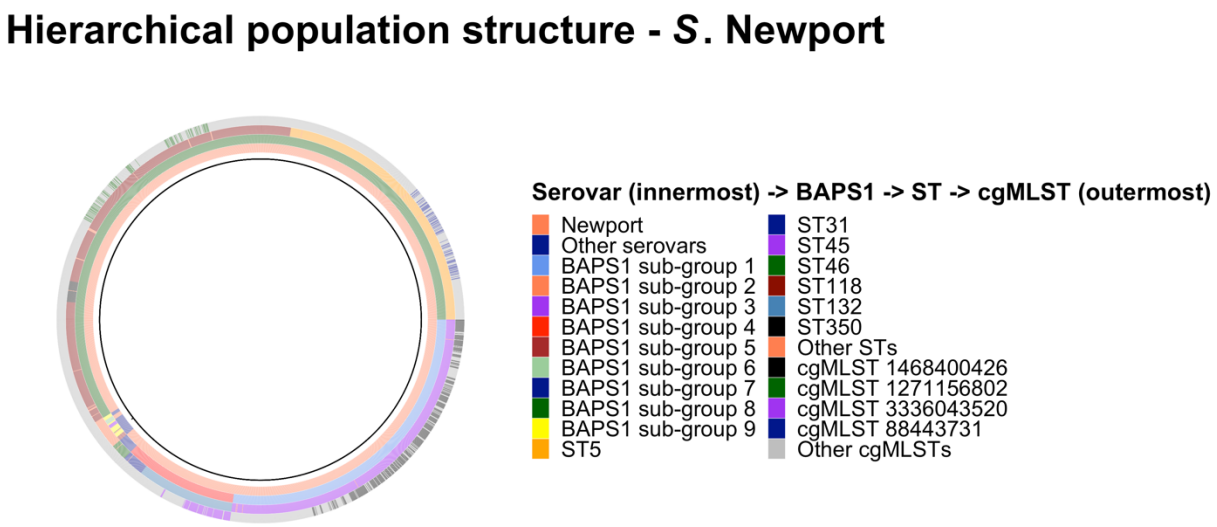


Figure 2. Visualize the frequency-based distribution of hierarchical genotypes. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

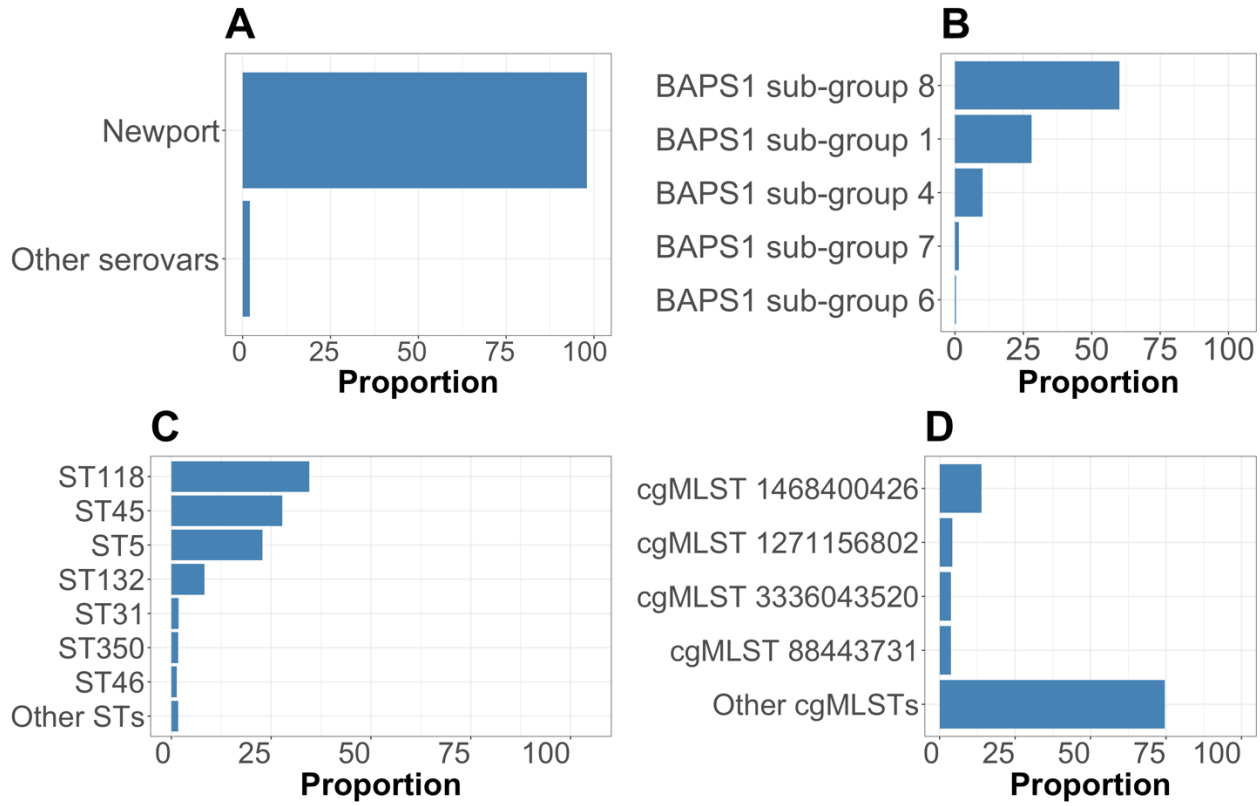
```
# bring back the metadata file
d5 <- d3
##########################################-------#############################
#################
##########################################-------#############################
#################
# plot Serovar distribution
# calculate the relative frequencies for serovar
# drop the serovars with NAs, group by serovar and then calculate the frequen
cies
serovar_data <- d5 %>%
              drop_na(serovar) %>% # remove NA based on serovar
              select(serovar) %>% # select the serovar column
              group_by(serovar) %>% # group_by serovar
              summarise(n = n()) %>% # count the number of genomes per serova
r classification
              mutate(prop_serovar = n/sum(n)*100) # calculate the proportion
of each serovar
# order serovar groups
serovar_data$serovar <- factor(serovar_data$serovar, levels=c("Other serovars
", "Newport"))
# plot the data
sero_plot <- ggplot(serovar_data, aes(x = prop_serovar, y = serovar)) +  # sh
ow serovars on y-axis and proportion on x-axis
```

```
  ylab("") + xlab("Proportion") + xlim(0, 100) +  # set labels for axis and l
imit for x-axis
  ggtitle("A") +  # add title for the plot
  theme_bw() + # choose the plot background
  theme(legend.position = "none") + # remove legend
  theme(axis.text.y = element_text(size = 30)) + # set the font size for y-ax
is text
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # set the fo
nt size for y-axis title
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # set the fo
nt size for x-axis title
  theme(axis.text.x = element_text(angle = 0, hjust = 0.7, size = 26)) + # se
t the font size for x-axis text
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize p
lot's title, legend, font
  geom_col(fill = "steelblue")  # fill the bars that represent the values wit
h blue
##########################################-------#############################
#################
##########################################-------#############################
#################
# plot BAPS1 sub-group distribution for Newport data only
# calculate the relative frequencies for serovar and BAPS1 sub-group
# drop the serovars and BAPS1 sub-groups records with NAs, group by serovar a
nd BAPS1 sub-group and then calculate the frequencies
baps_data <- d5 %>%
             drop_na(serovar, baps_1) %>% # drop NAs for serovar and baps_1
columns
             select(serovar, baps_1) %>% # select serovar and baps_1 columns
             group_by(serovar, baps_1) %>% # group data based on serovar and
baps_1
             summarise(n = n()) %>% # count the number of observations per g
roup
             mutate(prop = n/sum(n)*100) %>% # calculate the propotion for e
ach group
             filter(serovar != "Other serovars") # filter out genomes classi
fied as Other serovars

## `summarise()` has grouped output by 'serovar'. You can override using the
`.groups` argument.

# re-order the baps_1 column
baps_data$baps_1 <- factor(baps_data$baps_1, levels=c("BAPS1 sub-group 6", "B
APS1 sub-group 7", "BAPS1 sub-group 4", "BAPS1 sub-group 1", "BAPS1 sub-group
8"))
# plot the data
baps_plot <- ggplot(baps_data, aes(x = baps_1, y = prop)) +  # show BAPS sub-
groups on x-axis and proportion on y-axis
  xlab("") + ylab("Proportion") + ylim(0, 105) +  # set labels for axis and l
imit for y-axis
```

```
  ggtitle("B") +  # add title for the plot
  theme_bw() + # setting plot background
  theme(legend.position = "none") + # remove legend
  theme(axis.text.y = element_text(size = 30)) + # change font size for y-axis text
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change font size for y-axis title
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change font size for x-axis title
  theme(axis.text.x = element_text(angle = 0, size = 30)) + # change font size for x-axis text
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize plot's title, legend, font
  geom_col(fill = "steelblue") +  # fill the bars that represent the values with blue
  coord_flip()  # flip x and y axis
##############################################-------#############################################
##############################################-------#############################################
# plot ST distribution
# calculate the relative frequencies for serovar and ST
# drop the serovars and STs with NAs, group by serovar and ST and then calculate the frequencies
st_data <- d5 %>%
              drop_na(serovar, st) %>% # drop NAs for serovar and st columns
              select(serovar, st) %>% # select serovar and st columns
              group_by(serovar, st) %>% # group data based on serovar and st columns
              summarise(n = n()) %>% # count the number of observations
              mutate(prop_st = n/sum(n)*100) %>% # calculate the proportion by groups
              filter(serovar != "Other serovars") # filter out data for Other serovars

## `summarise()` has grouped output by 'serovar'. You can override using the `.groups` argument.

# re-order the st column
st_data$st <- factor(st_data$st, levels=c("Other STs", "ST46", "ST350", "ST31", "ST132", "ST5", "ST45",
                                          "ST118"))
# plot the data
st_plot <- ggplot(st_data, aes(x = st, y = prop_st)) +  # show STs on x-axis and proportion on y-axis
  xlab("") + ylab("Proportion") + ylim(0, 105) +  # set labels for axis and limit for y-axis
  ggtitle("C") +  # add title for the plot
  theme_bw() + # set the plot background
  theme(legend.position = "none") + # remove legends
```

```
  theme(axis.text.y = element_text(size = 28)) + # change font size for y-axis text
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change font size for y-axis title
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change font size for x-axis title
  theme(axis.text.x = element_text(angle = 0, hjust = 1, size = 28)) + # change font size for x-axis text
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize plot's title, legend, font
  geom_col(fill = "steelblue") +  # fill the bars that represent the values with blue
  coord_flip()  # flip x and y axis
##########################################--------#############################################
##########################################--------#############################################
# plot cgMLST distribution
# calculate the relative frequencies for serovar and cgMLST
# drop the serovars and cgMLSTs with NAs, group by serovar and cgMLST and then calculate the frequencies
cgmlst_data <- d5 %>%
               drop_na(serovar, cgmlst) %>% # drop NAs for serovar and cgmlst columns
               group_by(serovar, cgmlst) %>% # group data based on serovar and cgmlst columns
               summarise(n = n()) %>% # count the number of observations
               mutate(prop_cgmlst = n/sum(n)*100) %>% # calculate the proportions
               filter(serovar != "Other serovars") # filter out data for Other serovars

## `summarise()` has grouped output by 'serovar'. You can override using the
`.groups` argument.

# re-order cgmlst column
cgmlst_data$cgmlst <- factor(cgmlst_data$cgmlst, levels=c("Other cgMLSTs", "cgMLST 88443731", "cgMLST 3336043520", "cgMLST 1271156802", "cgMLST 1468400426"))
# plot the data
cgmlst_plot <- ggplot(cgmlst_data, aes(x = cgmlst, y = prop_cgmlst)) +  # show cgMLSTs on x-axis and proportion on y-axis
  xlab("") + ylab("Proportion") + ylim(0, 100) +  # set labels for axis and limit for y-axis
  ggtitle("D") +  # add title for the plot
  theme_bw() + # set plot background
  theme(legend.position = "none") + # remove legend
  theme(axis.text.y = element_text(size = 28)) + # set the font size for y-axis text
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # set the fo
```

```
nt size for x-axis title
  theme(axis.text.x = element_text(angle = 0, hjust = 1, size = 28)) + # set
the font size for x-axis text, orientations, and angle
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize p
lot's title, legend, font
  geom_col(fill = "steelblue") +  # fill the bars that represent the values w
ith blue
  coord_flip()  # flip x and y axis
##########################################--------############################
#################
##########################################-------############################
#################
# combine all plots in one to make Figure 2 based on 2 columns x 2 rows
figure_2  <- ggarrange(sero_plot, baps_plot, st_plot, cgmlst_plot,
                  nrow = 2, ncol = 2, widths = c(10, 10, 10, 10),
                heights = c(1, 1, 1, 1))
figure_2
```



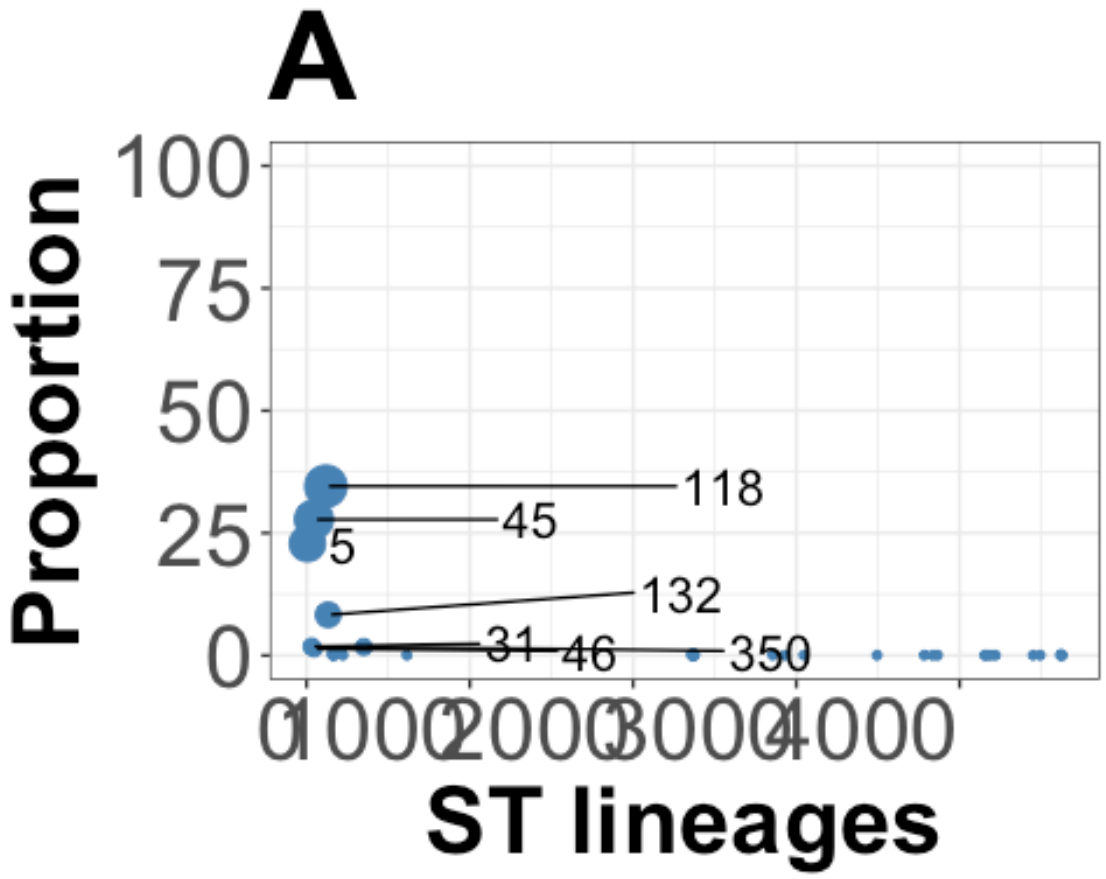
Supplementary Figure S1. Scatter plot showing the distribution of STs and cgMLSTs. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# bring the data containing the ST and cgmlst_ST information as numeric value
s
# assigning d2 to d2b
d2b <- d2
```

```r
# plot the ST distribution using a scatter plot
# select Newport serovars only, drop the STs with NAs, group by ST and then c
alculate the distribution
st_scatter <- d2b %>%
              filter(serovar == "Newport") %>% # filter for Newport data only
              select(ST) %>% # select the ST column
              mutate(ST = as.numeric(ST)) %>% # transform the column to numer
ic
              drop_na(ST) %>% # remove NA
              group_by(ST) %>% # group by ST
              summarise(n = n()) %>% # count observations
              mutate(prop = n/sum(n)*100) %>% # create a column with proporti
ons
              arrange(desc(prop))  # arrange data in descending order
# check the first six observation of st_scatter
head(st_scatter)
```

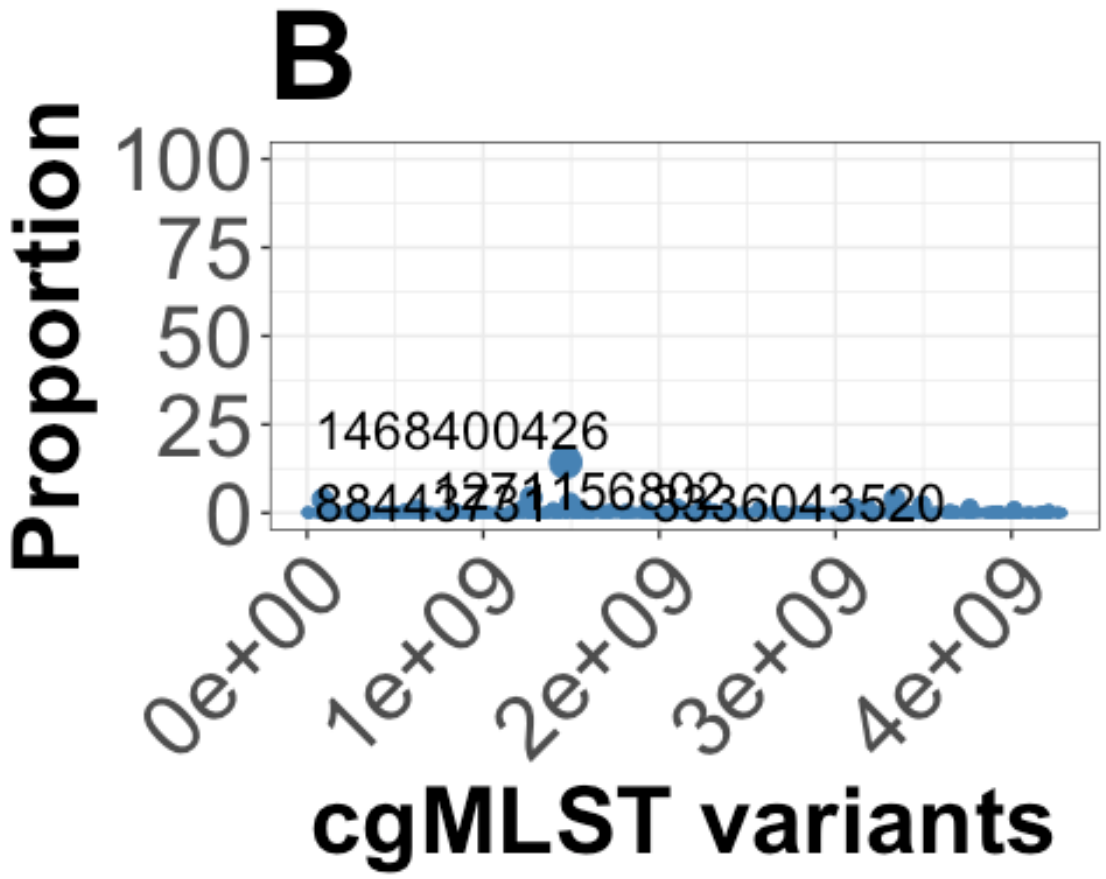
```
## # A tibble: 6 x 3
##      ST     n  prop
##   <dbl> <int> <dbl>
## 1   118   800 34.6
## 2    45   643 27.8
## 3     5   529 22.9
## 4   132   192  8.29
## 5    31    42  1.81
## 6   350    40  1.73
```

```r
# plot
st_plot <- ggplot(st_scatter, aes(x = ST, y = prop, labels = ST)) +  # show S
Ts on x-axis and proportion on y-axis
  xlab("ST lineages") + ylab("Proportion") + ylim(0, 100) +  # set labels for
axis and limit for y-axis
  ggtitle("A") +  # add title for the plot
  theme_bw() + # set plot background
  theme(legend.position = "none") + # remove legends
  theme(axis.text.y = element_text(size = 28)) + # change y-axis text font si
ze
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change y-a
xis title font size and face
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.text.x = element_text(angle = 0, hjust = 1, size = 28)) + # chan
ge x-axis text font size, angle and orientation
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize f
igure's title, legend, font
  geom_point(aes(size =  prop), color = "steelblue") +  # the points that rep
resent the values are blue with size based on the proportion
  geom_text_repel(data=subset(st_scatter, prop > 1), aes(label = ST, size = 3
0), hjust = -6)  # add text/proportion to the plot
st_plot
```




```
##########################################-------#############################
#################
##########################################-------#############################
#################
# plot the cgMLST distribution using a scatter plot
# select Newport serovars only, drop the cgMLSTs with NAs, group by cgMLST an
d then calculate the distribution
cgmlst_scatter <- d2b %>%
              filter(serovar == "Newport") %>% # select Newport genomes only
              select(cgmlst_ST) %>% # select the cgmlst_ST column
              mutate(cgmlst_ST = as.numeric(cgmlst_ST)) %>% # change column t
o numeric
              drop_na(cgmlst_ST) %>% # drop NAs
              group_by(cgmlst_ST) %>% # group by cgmlst_ST
              summarise(n = n()) %>% # count observations
              mutate(prop = n/sum(n)*100) %>% # calculate proportions
              arrange(desc(prop)) # arrange in descending order
# check the first six observation of st_scatter
head(cgmlst_scatter)

## # A tibble: 6 x 3
##    cgmlst_ST     n  prop
##        <dbl> <int> <dbl>
```

```
## 1 1468400426   321 14.3
## 2 1271156802    97  4.32
## 3 3336043520    86  3.83
## 4   88443731    85  3.78
## 5 1492716119    64  2.85
## 6 3491314984    50  2.23

# plot
cgmlst_plot <- ggplot(cgmlst_scatter, aes(x = cgmlst_ST, y = prop, labels = c
gmlst_ST)) +  # show cgMLSTs on x-axis and proportion on y-axis
  xlab("cgMLST variants") + ylab("Proportion") + ylim(0, 100) +  # set labels
for axis and limit for y-axis
  ggtitle("B") +  # add title for the plot
  theme_bw() + # set plot background
  theme(legend.position = "none") + # remove legends
  theme(axis.text.y = element_text(size = 28)) + # change y-axis text font si
ze
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change y-a
xis title font size and face
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 28)) + # cha
nge x-axis text font size, angle, and orientation
  theme(plot.title = element_text(size = 40, face = "bold")) +  # customize f
igure's title, legend, font
  geom_point(aes(size =  prop), color = "steelblue") +  # the points that rep
resent the values are blue with size based on the proportion
  geom_text_repel(data=subset(cgmlst_scatter, prop > 3), aes(label = cgmlst_S
T, size = 30), hjust = 2)  # add text/proportion to the plot
cgmlst_plot
```




```
##########################################--------############################
#################
##########################################--------############################
#################
# combine all plots in one to make Figure 3
sup_fig_s1  <- ggarrange(st_plot, cgmlst_plot,
                  nrow = 1, ncol = 2, widths = c(10, 10),
                heights = c(1, 1)) # combine plots together in one row and tw
o columns
sup_fig_s1
```

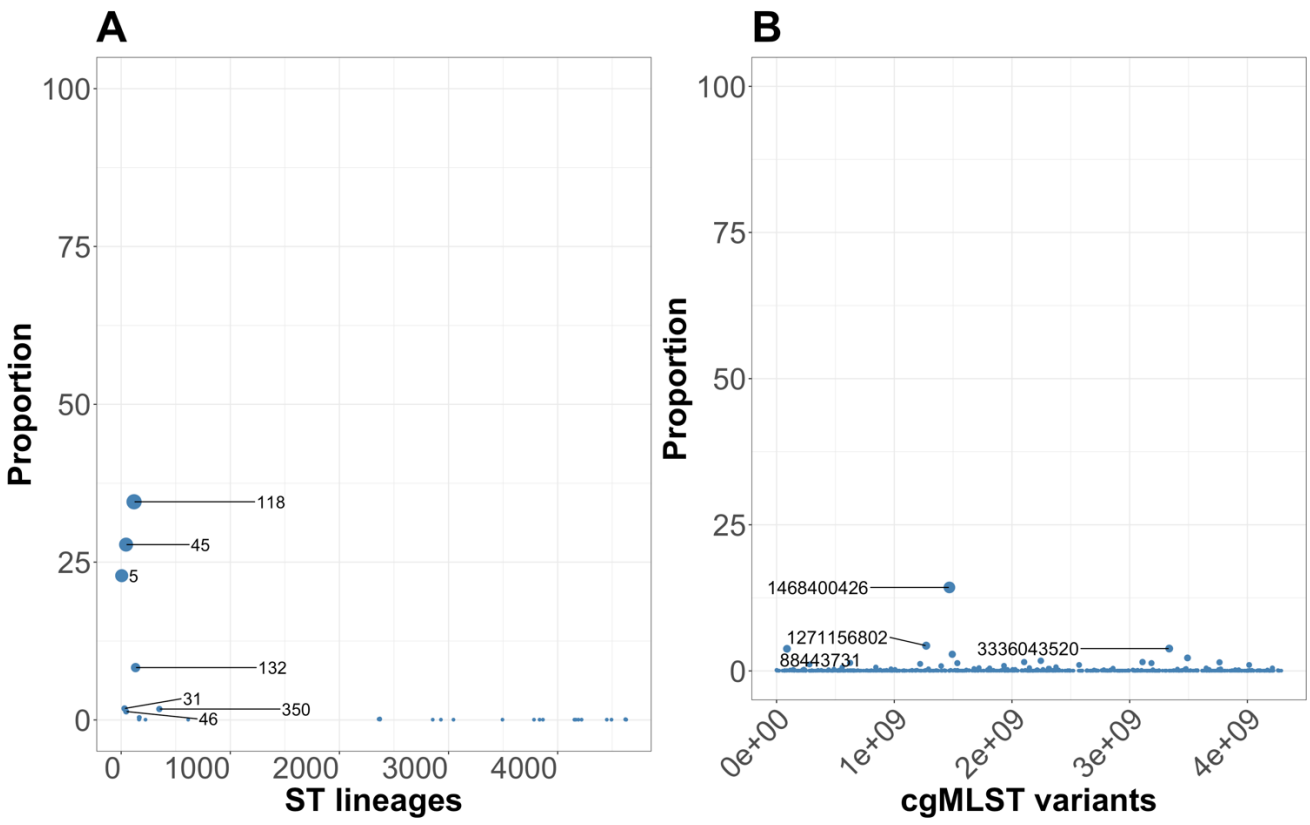
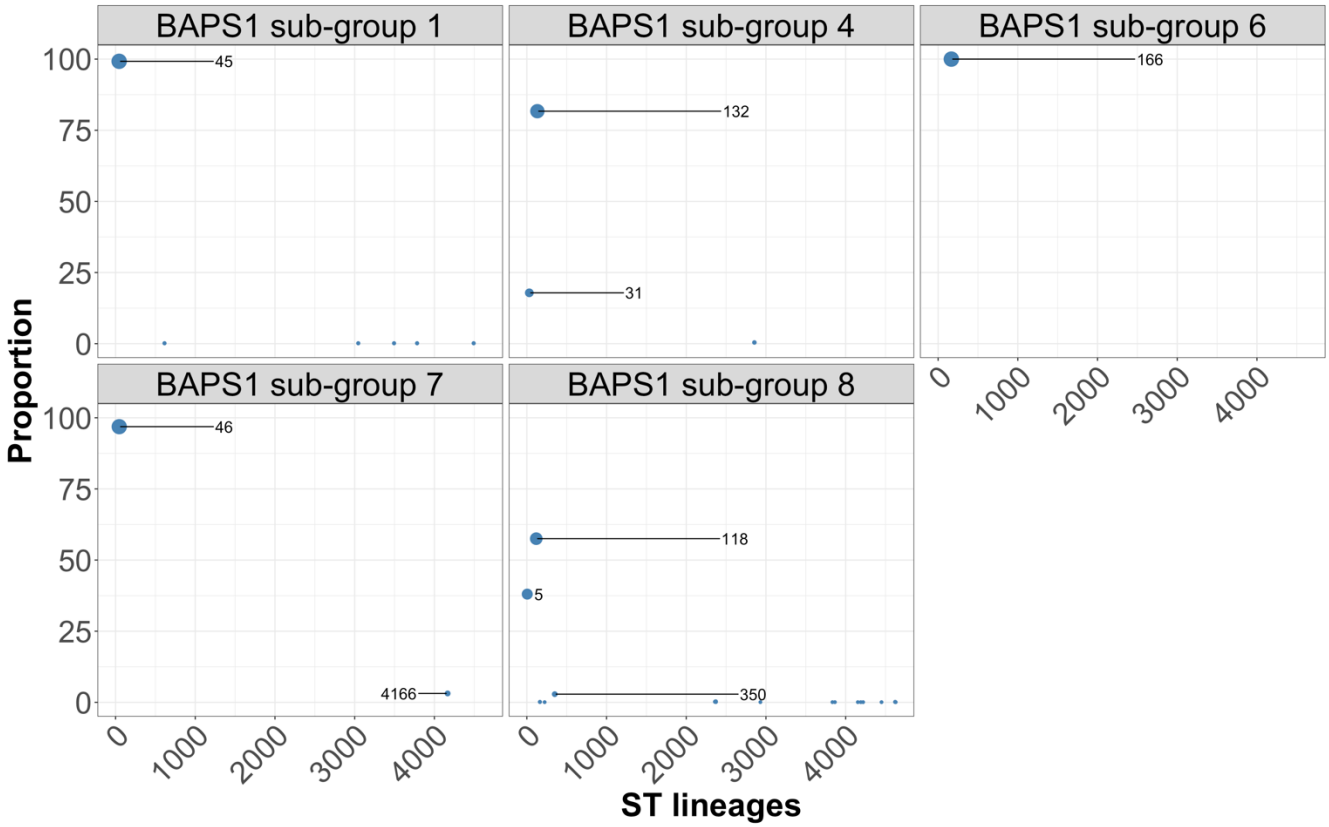


Figure 3. Distribution of STs based on BAPS level 1 sub-groups. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# bring the data containing the ST information as numeric values
# assigning d2 to d2b
d2b <- d2
# plot the ST distribution using a scatter plot
# select Newport serovars only, drop the STs and BAPS1 sub-groups with NAs, g
roup by ST and BAPS1 sub-groups and then calculate the distribution
baps <- d2b %>%
              filter(serovar == "Newport") %>% # filter Newport serovars
              select(baps_1, ST) %>% # select baps_1 and ST columns
              mutate(ST = as.numeric(ST)) %>% # change ST column to numeric
              drop_na(baps_1, ST) %>% # drop NAs
              group_by(baps_1, ST) %>% # group by baps_1 and ST
              summarise(n = n()) %>% # count observations
              mutate(prop = n/sum(n)*100) # calculate proportions

## `summarise()` has grouped output by 'baps_1'. You can override using the `
.groups` argument.

# check the first six observation of baps
head(baps)

## # A tibble: 6 x 4
## # Groups:   baps_1 [1]
```

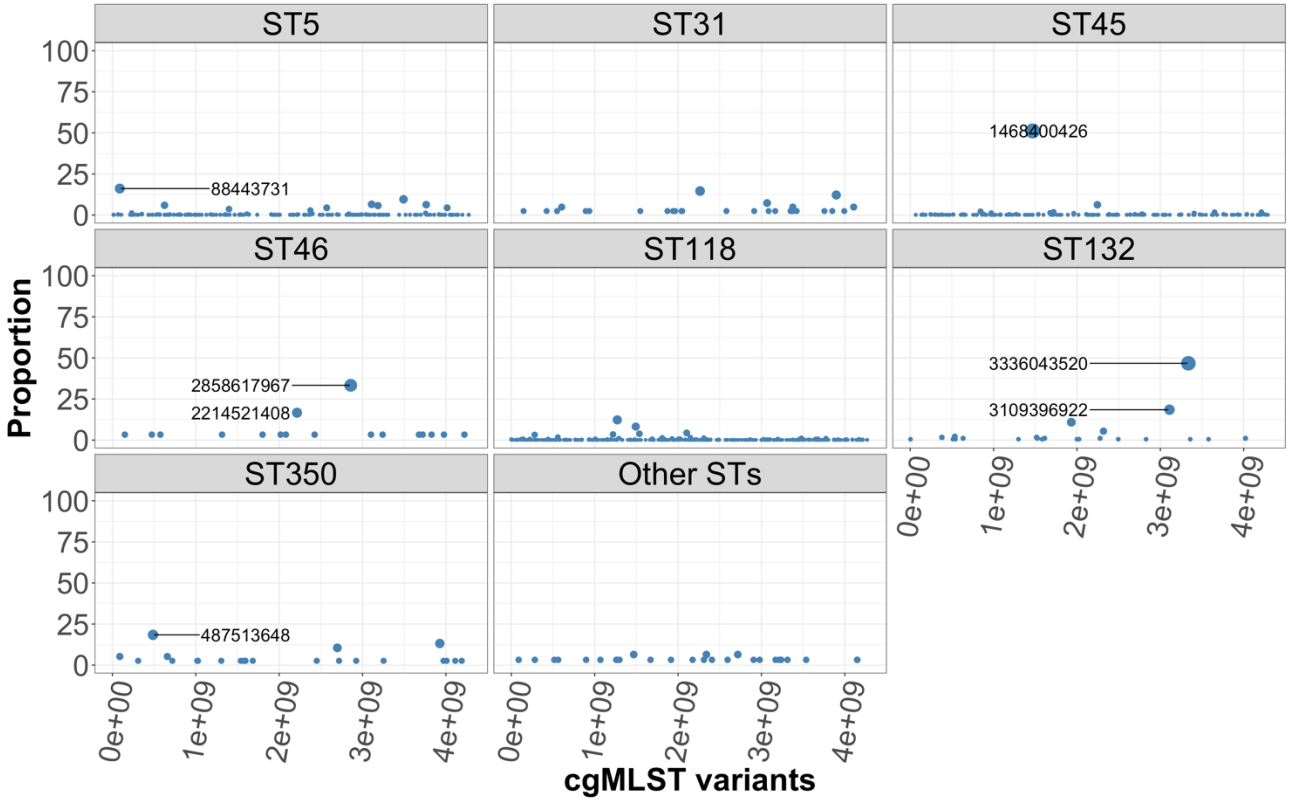
```
##   baps_1                ST     n   prop
##   <chr>              <dbl> <int>  <dbl>
## 1 BAPS1 sub-group 1    45   643 99.2
## 2 BAPS1 sub-group 1   614     1  0.154
## 3 BAPS1 sub-group 1  3045     1  0.154
## 4 BAPS1 sub-group 1  3494     1  0.154
## 5 BAPS1 sub-group 1  3783     1  0.154
## 6 BAPS1 sub-group 1  4493     1  0.154

# plot
figure_3 <- ggplot(baps, aes(x = ST, y = prop, labels = ST)) +  # show STs on
x-axis and proportion on y-axis
  xlab("ST lineages") + ylab("Proportion") + ylim(0, 100) +  # set labels for
axis and limit for y-axis
  theme_bw() + # set plot background
  theme(legend.position = "none") + # remove legends
  theme(axis.text.y = element_text(size = 28)) + # change y-axis text font si
ze
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change y-a
xis title font size and face
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 28)) + # cha
nge x-axis text font size, angle, and orientation
  theme(strip.text.x = element_text(size = 30, colour = "black", angle = 0))
+  # customize figure's title, legend, font
  geom_point(aes(size =  prop), color = "steelblue") +  # the points that rep
resent the values are blue with size based on the proportion
  geom_text_repel(data=subset(baps, prop > 1), aes(label = ST, size = 70), hj
ust = -10) +  # add text/proportion to the plot
  facet_wrap(~ baps_1)  # generate multi-plots for BAPS1 sub-groups
figure_3
```



Supplementary Figure S2. Distribution of cgMLSTs based on the major ST lineages presented in Figures 1 and 2. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# bring the data containing the cgMLST information as numeric values
# assigning d2 to d2b
d2b <- d2
# plot the ST distribution using a scatter plot
# select Newport serovars only, drop the STs and cgMLSTs with NAs, group by S
T and cgMLST and then calculate the distribution
cgmlst <- d2b %>%
              filter(serovar == "Newport") %>% # filter for Newport genomes o
nly
              select(st, cgmlst_ST) %>% # select st and cgmlst_ST columns
              drop_na(st, cgmlst_ST) %>% # drop NAs
              group_by(st, cgmlst_ST) %>% # group by st and cgmlst_ST
              summarise(n = n()) %>% # count observations
              mutate(prop = n/sum(n)*100) # calculate proportions

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# check the first six observation of cgmlst
head(cgmlst)
```

```
## # A tibble: 6 x 4
## # Groups:   st [1]
##   st          cgmlst_ST     n  prop
##   <chr>           <dbl> <int> <dbl>
## 1 Other STs   88443731     1  3.23
## 2 Other STs  281971874     1  3.23
## 3 Other STs  516287174     1  3.23
## 4 Other STs  561885424     1  3.23
## 5 Other STs  897193995     1  3.23
## 6 Other STs 1069299601     1  3.23

# re-order ST
cgmlst$st <- factor(cgmlst$st, levels=c("ST5", "ST31", "ST45", "ST46", "ST118
", "ST132", "ST350", "Other STs"))
# plot
sup_fig_2 <- ggplot(cgmlst, aes(x = cgmlst_ST, y = prop, labels = cgmlst_ST))
+  # show cgMLSTs on x-axis and proportion on y-axis
  xlab("cgMLST variants") + ylab("Proportion") + ylim(0, 100) +  # set labels
for axis and limit for y-axis
  theme_bw() + # set plot background
  theme(legend.position = "none") + # remove legend
  theme(axis.text.y = element_text(size = 28)) + # change y-axis text font si
ze
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change y-a
xis title font size and face
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.text.x = element_text(angle = 80, hjust = 1, size = 28)) + # cha
nge x-axis text font size, angle, and orientation
  theme(strip.text.x = element_text(size = 30, colour = "black", angle = 0))
+  # customize figure's title, legend, font
  geom_point(aes(size =  prop), color = "steelblue") +  # the points that rep
resent the values are blue with size based on the proportion
  geom_text_repel(data=subset(cgmlst, prop > 15), aes(label = cgmlst_ST, size
= 50), hjust = -25) +  # add text/proportion to the plot
  facet_wrap(~ st)  # generate multi-plots for STs
sup_fig_2
```



Supplementary Figure S3. Simpson's D diversity analysis of ST lineages using cgMLST and BAPS levels 1-6. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# enter BAPS1-6 data
baps <- read_csv('~/Documents/jove_paper/data/fastbaps_partition_baps_prior_l
6.csv')

##
## ── Column specification ────────────────────────────────────────────────────
──────
## cols(
##   Isolates = col_character(),
##   `Level 1` = col_double(),
##   `Level 2` = col_double(),
##   `Level 3` = col_double(),
##   `Level 4` = col_double(),
##   `Level 5` = col_double(),
##   `Level 6` = col_double()
## )

# changing column names
colnames(baps)[1:7] <- c("id", "BAPS1", "BAPS2", "BAPS3", "BAPS4", "BAPS5", "
BAPS6")
# assign baps data to d1
data1 <- baps
```

```r
# transform BAPS sub-groups to factor (categorical data)
data1 <- data1 %>% mutate(BAPS1 = as.factor(BAPS1))
data1 <- data1 %>% mutate(BAPS2 = as.factor(BAPS2))
data1 <- data1 %>% mutate(BAPS3 = as.factor(BAPS3))
data1 <- data1 %>% mutate(BAPS4 = as.factor(BAPS4))
data1 <- data1 %>% mutate(BAPS5 = as.factor(BAPS5))
data1 <- data1 %>% mutate(BAPS6 = as.factor(BAPS6))
##########################################-------############################
#################
##########################################-------############################
#################
# enter the ST data
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')

##
## ── Column specification ────────────────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# generate the id column
mlst$id2 <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# select columns of interest
data2 <- mlst %>%
          select(id2, ST) %>% # select columns of interest
            mutate_all(na_if, "-") %>% # transform any - to NA
              mutate_all(na_if, "?") %>% # transform any ? to NA
                  rename(id = id2) # rename id2 column to id
# categorize ST data to aggregate minor STs and keep only major ST lineages s
eparately
data2$st <- ifelse(data2$ST == 5, "ST5", # group major STs separately, and mi
nor STs as Others
                   ifelse(data2$ST == 31, "ST31",
                          ifelse(data2$ST == 45, "ST45",
                                 ifelse(data2$ST == 46, "ST46",
                                        ifelse(data2$ST == 118, "ST118",
                                               ifelse(d2$ST == 132, "ST132",
                          ifelse(data2$ST == 350, "ST350", "Other STs")))))))
# filter out the numerical ST column
data2 <- data2 %>% select(-ST)
```

```
##########################################--------############################
#################
##########################################--------############################
#################
# enter SISTR results
sistr <- read_csv('~/Documents/jove_paper/data/sistr_output.csv')

##
## ── Column specification ────────────────────────────────────────────────────
──────
## cols(
##   cgmlst_ST = col_double(),
##   cgmlst_distance = col_double(),
##   cgmlst_genome_match = col_character(),
##   cgmlst_matching_alleles = col_double(),
##   cgmlst_subspecies = col_character(),
##   fasta_filepath = col_character(),
##   genome = col_character(),
##   h1 = col_character(),
##   h2 = col_character(),
##   o_antigen = col_character(),
##   qc_messages = col_character(),
##   qc_status = col_character(),
##   serogroup = col_character(),
##   serovar = col_character(),
##   serovar_antigen = col_character(),
##   serovar_cgmlst = col_character()
## )

# generate the id column
sistr$id <- sapply(strsplit(as.character(sistr$genome),'_'), "[", 1)
# select id and cgmlst_ST columns
data3 <- sistr %>%
            select(id, serovar, cgmlst_ST) %>% # select columns of interest
            mutate_all(na_if, "-") %>% # transform any - to NA
              mutate_all(na_if, "?") # transform any ? to NA
# group serovars as Newport or others
data3$serovar <- ifelse(data3$serovar == "Newport", "Newport",
                            "Other serovars") # group genomes to either Newpo
rt or other serovars
##########################################--------############################
#################
##########################################--------############################
#################
# merge datasets
data4 <- left_join(data1, data2, on = "id") # join datasets based on id

## Joining, by = "id"

data5 <- left_join(data3, data4, on = "id") # join datasets based on id
```

```
## Joining, by = "id"

# filter data for Newport only
data6 <- data5 %>% filter(serovar == "Newport")
# check for missing values
skim(data6)
```

Data summary

Name	data6
Number of rows	2317
Number of columns	10

Column type frequency:

character	3
factor	6
numeric	1

Group variables	None

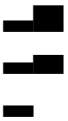
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2317	0
serovar	0	1	7	7	0	1	0
st	3	1	3	9	0	8	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
BAPS1	0	1	FALSE	5	8: 1393, 1: 648, 4: 235, 7: 32
BAPS2	0	1	FALSE	19	24: 532, 1: 486, 21: 300, 7: 189
BAPS3	0	1	FALSE	62	1: 483, 57: 263, 56: 187, 45: 155
BAPS4	0	1	FALSE	121	1: 483, 98: 149, 69: 143, 97: 114
BAPS5	0	1	FALSE	212	1: 483, 103: 143, 155: 131, 104: 93
BAPS6	0	1	FALSE	312	1: 483, 141: 143, 216: 131, 142: 93

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cgmlst_ST	70	0.97	19976 02103	11492 04978	183 768 5	12711 56802	16127 07568	31089 47600	42880 28711	

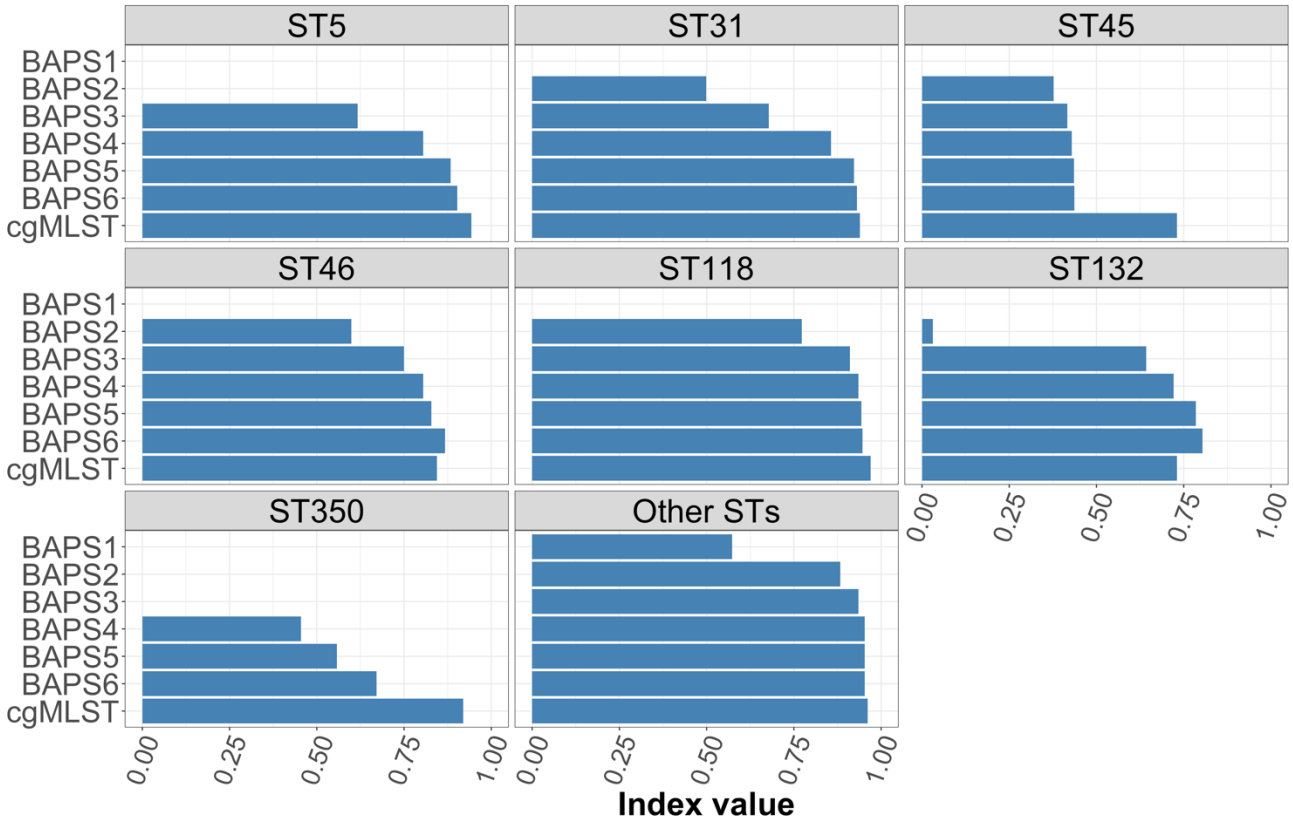
```
# group ST missing values as "Other STs"
data6 <- data6 %>% mutate(st = replace_na(st, "Other STs"))
# check the distribution of serovars to make sure there is only Newport
table(data6$serovar)

##
## Newport
##    2317

##########################################--------############################
#################
##########################################--------############################
#################
# cgMLST diversity
# drop the STs and cgMLSTs with NAs, group by ST and cgMLST and then calculat
e Simpson's index
cgmlst_div <- data6 %>%
              drop_na(st, cgmlst_ST) %>% # drop NAs
              mutate(cgmlst_ST = as.factor(cgmlst_ST)) %>% # transform column
to categorical
              select(st, cgmlst_ST) %>% # select columns of interest
              group_by(st, cgmlst_ST) %>% # group by st and cgmlst_ST
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate the i
ndex of diversity
              group_by(st) %>% # group by st
              summarise(simpson = mean(simpson)) %>% # get the mean of the si
mpson index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # convert
into long format
              mutate(strat = "cgMLST") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

##########################################--------############################
#################
##########################################--------############################
#################
# BAPS based diversity (calculated across BAPS levels 1 through 6)
```

```r
# BAPS level 1
# drop the STs and BAPS1 with NAs, group by ST and BAPS1 and then calculate S
impson's index
baps1 <- data6 %>%
              select(st, BAPS1) %>% # select columns
              drop_na(st, BAPS1) %>% # drop NAs
              group_by(st, BAPS1) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS1") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 2
# drop the STs and BAPS2 with NAs, group by ST and BAPS2 and then calculate S
impson's index
baps2 <- data6 %>%
              select(st, BAPS2) %>%  # select columns
              drop_na(st, BAPS2) %>% # drop NAs
              group_by(st, BAPS2) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS2") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 3
# drop the STs and BAPS3 with NAs, group by ST and BAPS3 and then calculate S
impson's index
baps3 <- data6 %>%
              select(st, BAPS3) %>% # select columns
              drop_na(st, BAPS3) %>% # drop NAs
              group_by(st, BAPS3) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
```

```
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS3") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 4
# drop the STs and BAPS4 with NAs, group by ST and BAPS4 and then calculate S
impson's index
baps4 <- data6 %>%
              select(st, BAPS4) %>% # select columns
              drop_na(st, BAPS4) %>% # drop NAs
              group_by(st, BAPS4) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS4") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 5
# drop the STs and BAPS5 with NAs, group by ST and BAPS5 and then calculate S
impson's index
baps5 <- data6 %>%
              select(st, BAPS5) %>% # select columns
              drop_na(st, BAPS5) %>% # drop NAs
              group_by(st, BAPS5) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS5") # create a strat column
```

```
## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 6
# drop the STs and BAPS6 with NAs, group by ST and BAPS6 and then calculate S
impson's index
baps6 <- data6 %>%
              select(st, BAPS6) %>%  # select columns
              drop_na(st, BAPS6) %>% # drop NAs
              group_by(st, BAPS6) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(simpson = diversity(n, "simpson")) %>% # calculate diver
sity
              group_by(st) %>% # group by column
              summarise(simpson = mean(simpson)) %>% # calculate the mean of
the index
              melt(id.vars=c("st"), measure.vars="simpson",
                   variable.name="index", value.name="value") %>% # covert i
nto long format
              mutate(strat = "BAPS6") # create a strat column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

##########################################-------#############################
#################
##########################################-------#############################
#################
# Concatenate all datasets
data7 <- rbind(cgmlst_div, baps1, baps2, baps3, baps4, baps5, baps6)
# order the ST column
data7$st <- factor(data7$st, levels=c("ST5", "ST31", "ST45", "ST46", "ST118",
"ST132", "ST350", "Other STs"))
##########################################-------#############################
#################
##########################################-------#############################
#################
# plot Simpson's diversity analysis
# order the strat colunmn
data7$strat <- factor(data7$strat, levels=c("cgMLST", "BAPS6", "BAPS5", "BAPS
4", "BAPS3", "BAPS2", "BAPS1"))
sup_fig_3 <- ggplot(data7, aes(x = strat, y = value)) +  # show strata on x-a
xis and index values on y-axis
  xlab("") + ylab("Index value") + ylim(0,1) +  # set labels for axis and lim
it for y-axis
  theme_bw() + # set plot background
  theme(axis.text.y = element_text(size = 28)) + # change y-axis text font si
ze
  theme(axis.title.y = element_text(size = 30, face = "bold")) + # change y-a
xis title font size and face
```

```
  theme(axis.title.x = element_text(size = 30, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.text.x = element_text(angle = 70, hjust = 1, size = 24)) + # cha
nge x-axis text font size, angle, and orientation
  theme(strip.text.x = element_text(size = 30, colour = "black", angle = 0))
+  # customize figure's title, legend, font
  geom_col(fill = "steelblue") +  # fill the bars that represent the values w
ith blue
  facet_wrap(~st) +  # generate multi-plots for STs
  coord_flip()  # flip x and y axis
sup_fig_3
```



Supplementary Figure S4. BAPS levels 1-6 distribution of sub-groups across ST lineages. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# enter BAPS1-6 data
baps <- read_csv('~/Documents/jove_paper/data/fastbaps_partition_baps_prior_l
6.csv')

##
## ── Column specification ────────────────────────────────────────────────
──────
## cols(
##   Isolates = col_character(),
```

```
##   `Level 1` = col_double(),
##   `Level 2` = col_double(),
##   `Level 3` = col_double(),
##   `Level 4` = col_double(),
##   `Level 5` = col_double(),
##   `Level 6` = col_double()
## )

# changing column names
colnames(baps)[1:7] <- c("id", "BAPS1", "BAPS2", "BAPS3", "BAPS4", "BAPS5", "
BAPS6")
# assign baps data to d1
data1 <- baps
# transform BAPS sub-groups to factor (categorical data)
data1 <- data1 %>% mutate(BAPS1 = as.factor(BAPS1))
data1 <- data1 %>% mutate(BAPS2 = as.factor(BAPS2))
data1 <- data1 %>% mutate(BAPS3 = as.factor(BAPS3))
data1 <- data1 %>% mutate(BAPS4 = as.factor(BAPS4))
data1 <- data1 %>% mutate(BAPS5 = as.factor(BAPS5))
data1 <- data1 %>% mutate(BAPS6 = as.factor(BAPS6))
##############################################--------############################
##################
##########################################-------############################
##################
# enter the ST data
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')

##
## ── Column specification ────────────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# generate the id column
mlst$id2 <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# select columns of interest and rename -/? to NAs
data2 <- mlst %>%
          select(id2, ST) %>% # select columns
            mutate_all(na_if, "-") %>% # transform any - to NAs
              mutate_all(na_if, "?") %>% # transform any ? to NAs
```

```
                  rename(id = id2) # rename column
# categorize ST data to aggregate minor STs and keep only major ST lineages s
eparately
data2$st <- ifelse(data2$ST == 5, "ST5",  # aggregate minor STs as others
                   ifelse(data2$ST == 31, "ST31",
                          ifelse(data2$ST == 45, "ST45",
                                 ifelse(data2$ST == 46, "ST46",
                                        ifelse(data2$ST == 118, "ST118",
                                               ifelse(data2$ST == 132, "ST132
",
                          ifelse(data2$ST == 350, "ST350", "Other STs")))))))
# filter out the numerical ST column
data2 <- data2 %>% select(-ST)
##########################################--------############################
#################
##########################################--------############################
################
# enter SISTR results
sistr <- read_csv('~/Documents/jove_paper/data/sistr_output.csv')

##
## ── Column specification ────────────────────────────────────────────────────
─────
## cols(
##   cgmlst_ST = col_double(),
##   cgmlst_distance = col_double(),
##   cgmlst_genome_match = col_character(),
##   cgmlst_matching_alleles = col_double(),
##   cgmlst_subspecies = col_character(),
##   fasta_filepath = col_character(),
##   genome = col_character(),
##   h1 = col_character(),
##   h2 = col_character(),
##   o_antigen = col_character(),
##   qc_messages = col_character(),
##   qc_status = col_character(),
##   serogroup = col_character(),
##   serovar = col_character(),
##   serovar_antigen = col_character(),
##   serovar_cgmlst = col_character()
## )

# generate the id column
sistr$id <- sapply(strsplit(as.character(sistr$genome),'_'), "[", 1)
# select id and cgmlst_ST columns and rename -/? to NAs
data3 <- sistr %>%
            select(id, serovar, cgmlst_ST) %>% # select columns
            mutate_all(na_if, "-") %>% # transform any - to NA
              mutate_all(na_if, "?") # transform any ? to NA
# group serovars as Newport or others
```

```
data3$serovar <- ifelse(data3$serovar == "Newport", "Newport",
                        "Other serovars")
##########################################--------#############################
#################
##########################################--------#############################
#################
# merge datasets
data4 <- left_join(data1, data2, on = "id") # join datasets on id

## Joining, by = "id"

data5 <- left_join(data3, data4, on = "id") # join datasets on id

## Joining, by = "id"

# filter data for Newport only
data6 <- data5 %>% filter(serovar == "Newport")
# check for missing values
skim(data6)
```

Data summary

Name	data6
Number of rows	2317
Number of columns	10

_____

Column type frequency:

character	3
factor	6
numeric	1

_____

Group variables	None

Variable type: character

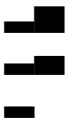
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2317	0
serovar	0	1	7	7	0	1	0
st	2	1	3	9	0	8	0

Variable type: factor

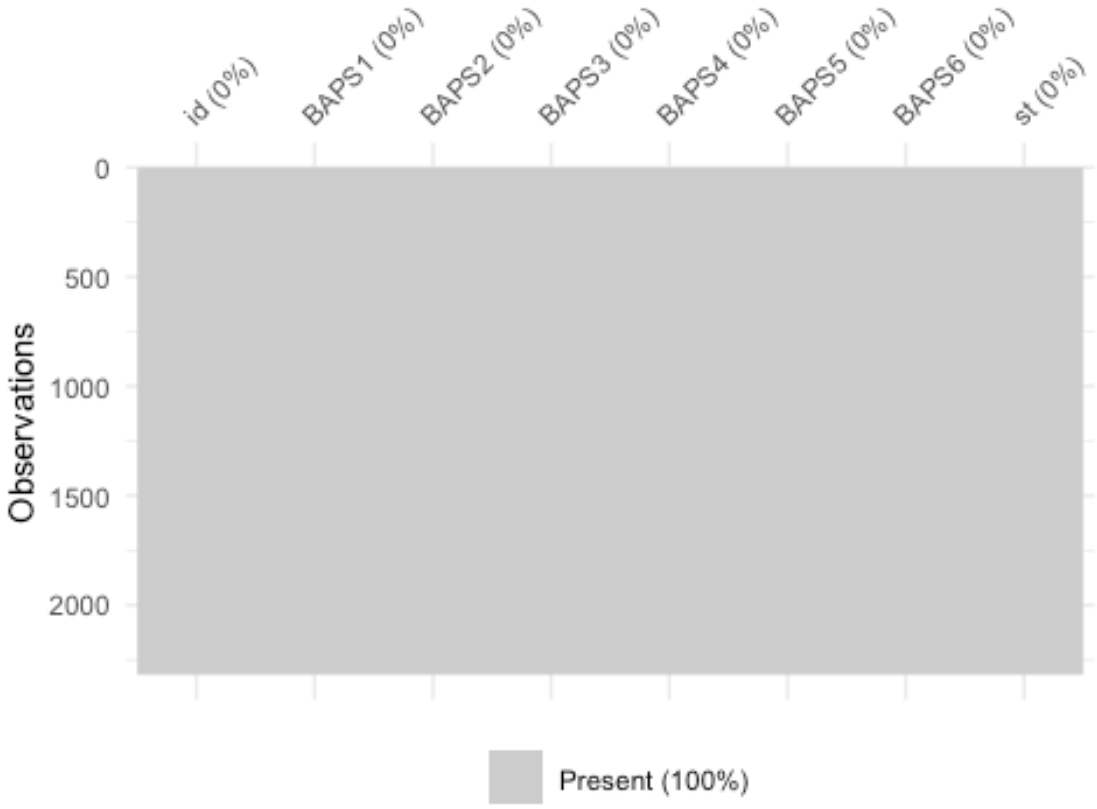
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
BAPS1	0	1	FALSE	5	8: 1393, 1: 648, 4: 235, 7: 32

BAPS2	0	1	FALSE	19	24: 532, 1: 486, 21: 300, 7: 189
BAPS3	0	1	FALSE	62	1: 483, 57: 263, 56: 187, 45: 155
BAPS4	0	1	FALSE	121	1: 483, 98: 149, 69: 143, 97: 114
BAPS5	0	1	FALSE	212	1: 483, 103: 143, 155: 131, 104: 93
BAPS6	0	1	FALSE	312	1: 483, 141: 143, 216: 131, 142: 93

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cgmlst_ST	70	0.97	1997602103	1149204978	1837685	1271156802	1612707568	3108947600	4288028711	

```
# group ST missing values as "Other STs"
data6 <- data6 %>% mutate(st = replace_na(st, "Other STs"))
# check the distribution of serovars to make sure there is only Newport
table(data6$serovar)

##
## Newport
##    2317

# select only needed columns (exclude columns labeled as serovar and cgmlst_ST)
data7 <- data6 %>% select(-c(serovar, cgmlst_ST))
# check for missing values again
vis_miss(data7)
```




```
# another way to check for missing values
skim(data7)
```

Data summary

Name	data7
Number of rows	2317
Number of columns	8

_____

Column type frequency:

character	2
factor	6

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2317	0

st	0	1	3	9	0	8	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
BAPS1	0	1	FALSE	5	8: 1393, 1: 648, 4: 235, 7: 32
BAPS2	0	1	FALSE	19	24: 532, 1: 486, 21: 300, 7: 189
BAPS3	0	1	FALSE	62	1: 483, 57: 263, 56: 187, 45: 155
BAPS4	0	1	FALSE	121	1: 483, 98: 149, 69: 143, 97: 114
BAPS5	0	1	FALSE	212	1: 483, 103: 143, 155: 131, 104: 93
BAPS6	0	1	FALSE	312	1: 483, 141: 143, 216: 131, 142: 93

```
# yet another way to check for missing values
sum(is.na(data7))

## [1] 0

##########################################-------#############################
#################
##########################################-------#############################
#################
# plot data
# BAPS levels 1-6 distributions
# BAPS level 1
# drop the STs and BAPS1 with NAs, group by ST and BAPS1 and then calculate d
istribution
baps1 <- data7 %>%
              select(st, BAPS1) %>% # select columns
              drop_na(st, BAPS1) %>% # drop NAs for selected columns
              group_by(st, BAPS1) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>%  # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS1") %>% # create a group column
              rename(baps = BAPS1) # rename a column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 2
# drop the STs and BAPS2 with NAs, group by ST and BAPS2 and then calculate d
istribution
baps2 <- data7 %>%
```

```
              select(st, BAPS2) %>% # select columns
              drop_na(st, BAPS2) %>% # drop NAs for selected columns
              group_by(st, BAPS2) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>% # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS2") %>% # create a group column
              rename(baps = BAPS2) # rename a column
```

```
## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.
```

```
# BAPS level 3
# drop the STs and BAPS3 with NAs, group by ST and BAPS3 and then calculate d
istribution
baps3 <- data7 %>%
              select(st, BAPS3) %>% # select columns
              drop_na(st, BAPS3) %>% # drop NAs for selected columns
              group_by(st, BAPS3) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>% # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS3") %>% # create a group column
              rename(baps = BAPS3) # rename a column
```

```
## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.
```

```
# BAPS level 4
# drop the STs and BAPS4 with NAs, group by ST and BAPS4 and then calculate d
istribution
baps4 <- data7 %>%
              select(st, BAPS4) %>% # select columns
              drop_na(st, BAPS4) %>% # drop NAs for selected columns
              group_by(st, BAPS4) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>% # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS4") %>% # create a group column
              rename(baps = BAPS4) # rename a column
```

```
## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.
```

```
# BAPS level 5
# drop the STs and BAPS5 with NAs, group by ST and BAPS5 and then calculate d
istribution
baps5 <- data7 %>%
              select(st, BAPS5) %>% # select columns
              drop_na(st, BAPS5) %>% # drop NAs for selected columns
              group_by(st, BAPS5) %>% # group by columns
```

```
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>% # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS5") %>% # create a group column
              rename(baps = BAPS5) # rename a column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

# BAPS level 6
# drop the STs and BAPS6 with NAs, group by ST and BAPS6 and then calculate d
istribution
baps6 <- data7 %>%
              select(st, BAPS6) %>% # select columns
              drop_na(st, BAPS6) %>% # drop NAs for selected columns
              group_by(st, BAPS6) %>% # group by columns
              summarise(n = n()) %>% # count observations
              mutate(total = sum(n)) %>% # sum up values to create a column
              mutate(prop = n/total*100) %>% # calculate proportions
              mutate(group = "BAPS6") %>% # create a group column
              rename(baps = BAPS6) # rename a column

## `summarise()` has grouped output by 'st'. You can override using the `.gro
ups` argument.

##########################################-------#############################
#################
##########################################-------#############################
#################
# Concatenate all datasets
data8 <- rbind(baps1, baps2, baps3, baps4, baps5, baps6)
# order BAPS levels
data8$group <- factor(data8$group, levels=c("BAPS1", "BAPS2", "BAPS3",
                                            "BAPS4", "BAPS5", "BAPS6"))
##########################################-------#############################
#################
##########################################-------#############################
#################
# plot data

# transform baps column to numeric
data8 <- data8 %>% mutate(baps = as.numeric(baps))
# order ST data
data8$st <- factor(data8$st, levels=c("ST5", "ST31", "ST45", "ST46", "ST118",
"ST132", "ST350", "Other STs"))
sup_fig_4 <- ggplot(data8, aes(x = baps, y = prop, fill = st)) +  # show BAPS
levels on x-axis and proportion on y-axis
  xlab("BAPS sub-groups (haplotypes)") + ylab("Proportion") + ylim(0, 100) +
# set labels for axis and limit for y-axis
  theme_bw() + # set plot background
  theme(axis.text.x = element_text(angle = 0, size = 25)) + # change x-axis t
```

```
ext font size and angle
  theme(axis.title.x = element_text(size = 35, face = "bold")) + # change x-a
xis title font size and face
  theme(axis.title.y = element_text(size = 35, face = "bold")) + # change y-a
xis title font size and face
  theme(axis.text.y = element_text(angle = 0, hjust = 1, size = 25)) + # chan
ge y-axis text font size, angle, and orientation
  theme(legend.title=element_text(size=35, face = "bold")) + # customize lege
nd title
  theme(legend.text=element_text(size=25)) + # customize legend text
  theme(strip.text.x = element_text(size = 35)) +  # customize figure's title
, legend, font
  scale_fill_manual(name = "ST", values=c("orange", "darkblue", "purple", "da
rkgreen", "darkred", "steelblue",
                                          "black", "coral")) +  # add colors
and labels for the scale
  geom_col(position = "dodge") +  # position columns side by side
  facet_wrap(~group, ncol = 2)  # generate multi-plots for BAPS with 2 column
s
sup_fig_4
```

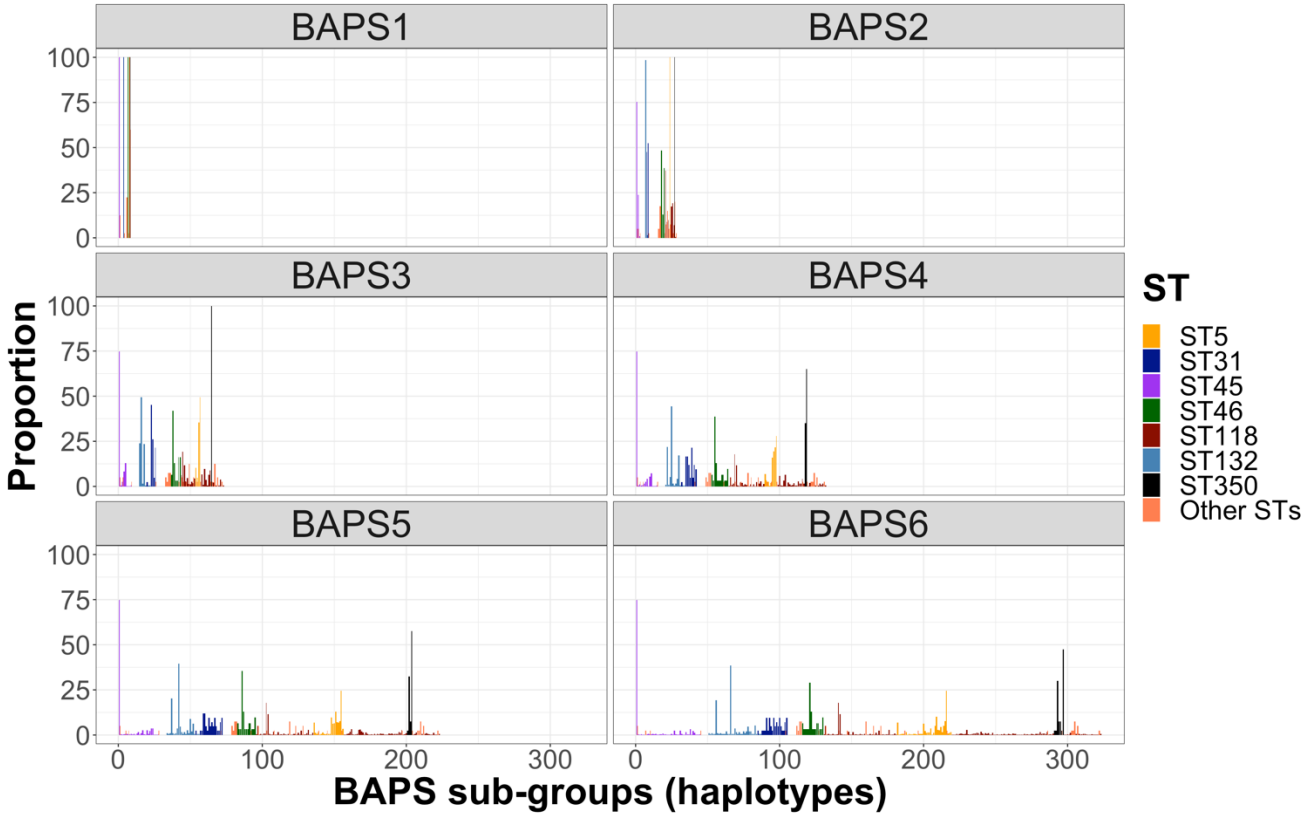


Figure 4. Distribution of Resfinder-annotated AMR loci across ST lineages. Note - only genomes classified as S. Newport by SISTR within ProkEvo are included in this analysis.

```
# enter the ST data
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')
```

```
##
## ── Column specification ────────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# generate the id column
mlst$id2 <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# select columns of interest and rename -/? to NAs
data1 <- mlst %>%
          select(id2, ST) %>% # select columns
            mutate_all(na_if, "-") %>% # transform - to NA
              mutate_all(na_if, "?") %>% # transform ? to NA
                  rename(id = id2) # rename column
# categorize ST data to aggregate minor STs and keep only major ST lineages s
eparately
data1$st <- ifelse(data1$ST == 5, "ST5", # group minor STs as others
                   ifelse(data1$ST == 31, "ST31",
                          ifelse(data1$ST == 45, "ST45",
                                 ifelse(data1$ST == 46, "ST46",
                                        ifelse(data1$ST == 118, "ST118",
                                               ifelse(data1$ST == 132, "ST132
",
                          ifelse(data1$ST == 350, "ST350", "Other STs")))))))
# filter out the numerical ST column
data1 <- data1 %>% select(-ST)
##########################################--------############################
#################
##########################################--------############################
#################
# enter the Resfinder loci databases (AMR loci)
abx <- read_csv('~/Documents/jove_paper/data/sabricate_resfinder_output.csv')

##
## ── Column specification ────────────────────────────────────────────
──────
## cols(
##   `#FILE` = col_character(),
##   SEQUENCE = col_character(),
##   START = col_double(),
```

```
##   END = col_double(),
##   GENE = col_character(),
##   COVERAGE = col_character(),
##   COVERAGE_MAP = col_character(),
##   GAPS = col_character(),
##   `%COVERAGE` = col_double(),
##   `%IDENTITY` = col_double(),
##   DATABASE = col_character(),
##   ACCESSION = col_character(),
##   PRODUCT = col_character()
## )

# create the id column
abx$id <- sapply(strsplit(as.character(abx$'#FILE'),'_'), "[", 1)
# change the name of the GENE column
abx <- abx %>% mutate(gene = GENE)
# select columns of interest
abx1 <- abx %>% select(id, gene)
##############################################--------############################
#################
##############################################-------############################
#################
# enter SISTR results
sistr <- read_csv('~/Documents/jove_paper/data/sistr_output.csv')

##
## ── Column specification ────────────────────────────────────────────────────
─────
## cols(
##   cgmlst_ST = col_double(),
##   cgmlst_distance = col_double(),
##   cgmlst_genome_match = col_character(),
##   cgmlst_matching_alleles = col_double(),
##   cgmlst_subspecies = col_character(),
##   fasta_filepath = col_character(),
##   genome = col_character(),
##   h1 = col_character(),
##   h2 = col_character(),
##   o_antigen = col_character(),
##   qc_messages = col_character(),
##   qc_status = col_character(),
##   serogroup = col_character(),
##   serovar = col_character(),
##   serovar_antigen = col_character(),
##   serovar_cgmlst = col_character()
## )

# generate the id column
sistr$id <- sapply(strsplit(as.character(sistr$genome),'_'), "[", 1)
# select id and cgmlst_ST columns and rename -/? to NAs
```

```
data3 <- sistr %>%
            select(id, serovar) %>% # select columns
            mutate_all(na_if, "-") %>% # transform - to NA
              mutate_all(na_if, "?") # transform ? to NA
# group serovars as Newport or others
data3$serovar <- ifelse(data3$serovar == "Newport", "Newport",
                        "Other serovars")
##########################################-------############################
#################
##########################################-------############################
#################
# merge datasets
data4 <- left_join(data1, abx1, on = "id") # join datasets based on id

## Joining, by = "id"

data5 <- left_join(data4, data3, on = "id") # join datasets based on id

## Joining, by = "id"

# filter data for Newport only
data6 <- data5 %>% filter(serovar == "Newport")
# check for NAs
skim(data6)
```

Data summary

Name	data6
Number of rows	9027
Number of columns	4
_____	
Column type frequency:	
character	4
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2317	0
st	4	1	3	9	0	8	0
gene	0	1	6	15	0	119	0
serovar	0	1	7	7	0	1	0

```
# group ST missing values as "Other STs"
data6 <- data6 %>% mutate(st = replace_na(st, "Other STs"))
```

```r
# check for NAs again
skim(data6)
```

Data summary

Name	data6
Number of rows	9027
Number of columns	4

_____

Column type frequency:	
character	4

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2317	0
st	0	1	3	9	0	8	0
gene	0	1	6	15	0	119	0
serovar	0	1	7	7	0	1	0

```r
##########################################-------#############################
#################
##########################################-------#############################
#################
# calculate the proportion of AMR loci for each ST lineage
# calculations for ST5
d2a <- data6 %>% filter(st == "ST5")
# for ST5, calculate the proportion of AMR loci and only keep proportion grea
ter than 10%
d3a <- d2a %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4a <- d3a %>%
         group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[8]) %>% # get the total counts of st
```

```
         mutate(prop = (value/total)*100) # calculate proportions
d5a <- d4a %>% mutate(st = "ST5")
##########################################--------############################
#################
##########################################--------############################
#################
# calculations for ST31
d2b <- data6 %>% filter(st == "ST31")
# for ST31, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3b <- d2b %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4b <- d3b %>% group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[4]) %>% # get the total counts of st
         mutate(prop = (value/total)*100) # calculate proportions
d5b <- d4b %>% mutate(st = "ST31")
##########################################--------############################
#################
##########################################--------############################
#################
# calculations for ST45
d2c <- data6 %>% filter(st == "ST45")
# for ST45, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3c <- d2c %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4c <- d3c %>% group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[6]) %>% # get the total counts of st
         mutate(prop = (value/total)*100) # calculate proportions
d5c <- d4c %>% mutate(st = "ST45")
##########################################--------############################
```

```
#################
##########################################-------############################
#################
# calculations for ST46
d2d <- data6 %>% filter(st == "ST46")
# for ST46, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3d <- d2d %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4d <- d3d %>% group_by(gene) %>%  # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[7]) %>% # get the total counts of st
         mutate(prop = (value/total)*100) # calculate proportions
d5d <- d4d %>% mutate(st = "ST46")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST118
d2e <- data6 %>% filter(st == "ST118")
# for ST118, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3e <- d2e %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4e <- d3e %>% group_by(gene) %>%  # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[2]) %>% # get the total counts of st
         mutate(prop = (value/total)*100) # calculate proportions
d5e <- d4e %>% mutate(st = "ST118")
##########################################-------############################
#################
##########################################-------############################
#################
```

```r
# calculations for ST132
d2f <- data6 %>% filter(st == "ST132")
# for ST132, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3f <- d2f %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4f <- d3f %>% group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[3]) %>% # get the total counts of st
         mutate(prop = (value/total)*100)  # calculate proportions
d5f <- d4f %>% mutate(st = "ST132")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST350
d2g <- data6 %>% filter(st == "ST350")
# for ST350, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3g <- d2g %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4g <- d3g %>% group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[5]) %>% # get the total counts of st
         mutate(prop = (value/total)*100)  # calculate proportions
d5g <- d4g %>% mutate(st = "ST350")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for Other ST lineages
d2h <- data6 %>% filter(st == "Other STs")
# for other STs, calculate the proportion of AMR loci and only keep proportio
```

```r
n greater than 10%
d3h <- d2h %>%
    select(id, gene) %>%  # select columns
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace counts equal
to 2 by 1
    filter(count <= 1) # filter counts below or equal to 1

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4h <- d3h %>% group_by(gene) %>% # group by gene
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[1]) %>% # get the total counts of st
         mutate(prop = (value/total)*100) # calculate proportions
d5h <- d4h %>% mutate(st = "Other STs")
##############################################--------############################
##################
##########################################-------############################
##################
# combined datasets
d6 <- rbind(d5a, d5b, d5c, d5d, d5e, d5f, d5g, d5h)
##############################################--------############################
##################
##########################################-------############################
##################
# export data table containing ST and AMR loci information
abx_newport_st <- d6
write.csv(abx_newport_st,"~/Documents/jove_paper/data/abx_newport_st.csv", ro
w.names = FALSE)
#########################################################
# filter AMR loci with proportion higher than or equal to %
d7 <- d6 %>% filter(prop >= 10)
# order STs
d7$st <- factor(d7$st, levels=c("ST5", "ST31", "ST45", "ST46", "ST118", "ST13
2", "ST350", "Other STs"))
# plot Figure 8
# create a vector object with the number of unique genes or loci in the data
colourCount = length(unique(d7$gene))
figure_4 <- ggplot(data = d7, mapping = aes(x = prop, y=gene, fill = gene)) +
# show genes on y-axis and proportion on x-axis
      xlab("Proportion") + ylab("AMR loci") + xlim(0, 105) +  # set labels fo
r axis and limit for x-axis
      theme_bw() + # set the plot background
      theme(legend.position = "none") + # remove legend
      theme(axis.text.y = element_text(size = 12)) + # change font size for y
-axis text
      theme(axis.title.y = element_text(size = 35, face = "bold")) + # custom
ize y-axis title font size and face
```

```
      theme(axis.title.x = element_text(size = 30, face = "bold")) + # custom
ize x-axis title font size and face
      theme(axis.text.x = element_text(angle = 0, hjust = 1, size = 20)) + #
customize x-axis text font size, angle, and orientation
      theme(strip.text.x = element_text(size = 30, colour = "black", angle =
0)) +  # customize figure's title, legend, font
      scale_fill_manual(values = colorRampPalette(brewer.pal(8, "Accent"))(co
lourCount)) +  # add colors for the scale
      geom_bar(stat = "identity") +  # show the bars with y values
      facet_wrap(~ st)  # generate multi-plots for STs
figure_4
```

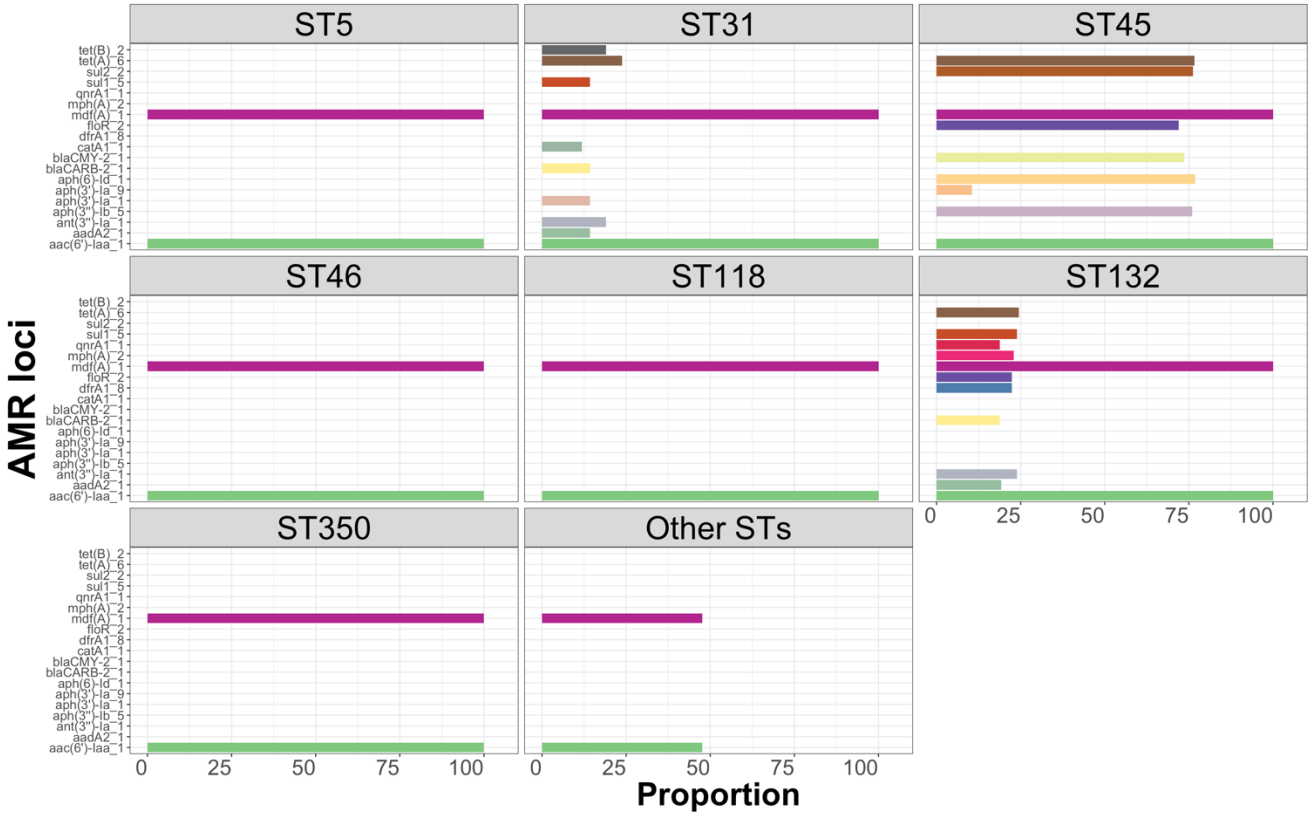


Figure 5 without legend. Phylogeny-guided mapping of hierarchical genotypes along with ST-differentiating AMR loci.

```
# enter the ST data
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')

##
## ── Column specification ────────────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
```

```
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# generate the id column
mlst$id2 <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# select columns of interest and rename -/? to NAs
data1 <- mlst %>%
          select(id2, ST) %>% # select columns
            mutate_all(na_if, "-") %>% # change - to NA
              mutate_all(na_if, "?") %>% # change ? to NA
                  rename(id = id2) # rename column
# categorize ST data to aggregate minor STs and keep only major ST lineages s
eparately
data1$st <- ifelse(data1$ST == 5, "ST5", # aggregate minor STs as Others
                   ifelse(data1$ST == 31, "ST31",
                          ifelse(data1$ST == 45, "ST45",
                                 ifelse(data1$ST == 46, "ST46",
                                        ifelse(data1$ST == 118, "ST118",
                                               ifelse(data1$ST == 132, "ST132
",
                          ifelse(data1$ST == 350, "ST350", "Other STs")))))))
# filter out the numerical ST column
data1 <- data1 %>% select(-ST)
##########################################-------############################
#################
##########################################-------############################
#################
# enter the Resfinder loci databases (AMR loci)
abx <- read_csv('~/Documents/jove_paper/data/sabricate_resfinder_output.csv')

##
## ── Column specification ────────────────────────────────────────────────
──────
## cols(
##   `#FILE` = col_character(),
##   SEQUENCE = col_character(),
##   START = col_double(),
##   END = col_double(),
##   GENE = col_character(),
##   COVERAGE = col_character(),
##   COVERAGE_MAP = col_character(),
##   GAPS = col_character(),
##   `%COVERAGE` = col_double(),
##   `%IDENTITY` = col_double(),
##   DATABASE = col_character(),
```

```
##   ACCESSION = col_character(),
##   PRODUCT = col_character()
## )

# create the id column
abx$id <- sapply(strsplit(as.character(abx$'#FILE'),'_'), "[", 1)
# change the name of the GENE column
abx <- abx %>% mutate(gene = GENE)
# select columns of interest
abx1 <- abx %>% select(id, gene)
##########################################-------############################
#################
##########################################-------############################
#################
# join ST and AMR loci data
d1 <- left_join(data1, abx1, on = "id")

## Joining, by = "id"

# check for NAs or missing values
skim(d1)
```

Data summary

Name	d1
Number of rows	9169
Number of columns	3

_____

Column type frequency:

character	3

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
st	11	1	3	9	0	8	0
gene	0	1	6	15	0	127	0

```
# group ST missing values as "Other STs"
d1 <- d1 %>% mutate(st = replace_na(st, "Other STs"))
# check for NAs again
skim(d1)
```

Data summary

Name	d1
Number of rows	9169
Number of columns	3

_____

Column type frequency:

character	3

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
st	0	1	3	9	0	8	0
gene	0	1	6	15	0	127	0

```
##########################################--------############################
#################
##########################################--------############################
#################
# calculate the proportion of AMR loci for each ST lineage
# calculations for ST5
d2a <- d1 %>% filter(st == "ST5")
# for ST5, calculate the proportion of AMR loci and only keep proportion grea
ter than 10%
d3a <- d2a %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4a <- d3a %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[8]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5a <- d4a %>% mutate(st = "ST5")
##########################################--------############################
```

```
##################
##########################################-------############################
##################
# calculations for ST31
d2b <- d1 %>% filter(st == "ST31")
# for ST31, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3b <- d2b %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4b <- d3b %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[4]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5b <- d4b %>% mutate(st = "ST31")
##########################################-------############################
##################
##########################################-------############################
##################
# calculations for ST45
d2c <- d1 %>% filter(st == "ST45")
# for ST45, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3c <- d2c %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4c <- d3c %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[6]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5c <- d4c %>% mutate(st = "ST45")
```

```
##########################################-------#############################
#################
##########################################-------#############################
#################
# calculations for ST46
d2d <- d1 %>% filter(st == "ST46")
# for ST46, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3d <- d2d %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4d <- d3d %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[7]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5d <- d4d %>% mutate(st = "ST46")
##########################################-------#############################
#################
##########################################-------#############################
#################
# calculations for ST118
d2e <- d1 %>% filter(st == "ST118")
# for ST118, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3e <- d2e %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4e <- d3e %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[2]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
```

```r
d5e <- d4e %>% mutate(st = "ST118")
##############################################--------#############################
##################
##############################################-------#############################
##################
# calculations for ST132
d2f <- d1 %>% filter(st == "ST132")
# for ST132, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3f <- d2f %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4f <- d3f %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[3]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5f <- d4f %>% mutate(st = "ST132")
##############################################--------#############################
##################
##############################################-------#############################
##################
# calculations for ST350
d2g <- d1 %>% filter(st == "ST350")
# for ST350, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3g <- d2g %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4g <- d3g %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[5]) %>% # get the total number of obs
ervations per ST
```

```
         mutate(prop = (value/total)*100) # calculate proportions
d5g <- d4g %>% mutate(st = "ST350")
##############################################--------############################
#################
##############################################--------############################
#################
# calculations for Other ST lineages
d2h <- d1 %>% filter(st == "Other STs")
# for other STs, calculate the proportion of AMR loci and only keep proportio
n greater than 10%
d3h <- d2h %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4h <- d3h %>% group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[1]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5h <- d4h %>% mutate(st = "Other STs")
##############################################--------############################
#################
##############################################--------############################
#################
# combined datasets
d6 <- rbind(d5a, d5b, d5c, d5d, d5e, d5f, d5g, d5h)
##############################################--------############################
#################
##############################################--------############################
#################
# filter AMR loci with proportion higher than or equal to %
d7 <- d6 %>% filter(prop >= 10)
# create a list of genes with proportion >= 10%
d8 <- d7 %>% select(gene) %>% unique()
# transform dataframe to list of characters or vector
d9 <- pull(d8, gene)
##############################################--------############################
#################
##############################################--------############################
#################
# spread the abx1 table - from long to wide format
# count gene occurrences
abx2 <- abx1 %>%
```

```
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count the number of observations
    filter(count <= 1) # filter genes with 0 or 1 counts

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

# spread from long to wide format
abx3 <- spread(abx2, key = gene, value = count)
# select columns of interest
abx4 <- abx3 %>% select(d9)

## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(d9)` instead of `d9` to silence this message.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Adding missing grouping variables: `id`

# replace all NAs with zeros
abx4[is.na(abx4)] <- 0
# merge with original data containing the hierarchical genotypes
d4 <- left_join(d3, abx4, on = "id")

## Joining, by = "id"

##############################################--------#############################
#################
##############################################-------#############################
################
# change binary values from AMR data locus by locus
d4 <- d4 %>% mutate(`aac(6')-Iaa_1` = factor(ifelse(`aac(6')-Iaa_1` == 1, "aa
c(6')-Iaa_1 (present)", "aac(6')-Iaa_1 (absent)")))
d4 <- d4 %>% mutate(`mdf(A)_1` = factor(ifelse(`mdf(A)_1` == 1, "mdf(A)_1 (pr
esent)", "mdf(A)_1 (absent)")))
d4 <- d4 %>% mutate(aadA2_1 = factor(ifelse(aadA2_1 == 1, "aadA2_1 (present)"
, "aadA2_1 (absent)")))
d4 <- d4 %>% mutate(`ant(3'')-Ia_1` = factor(ifelse(`ant(3'')-Ia_1` == 1, "an
t(3'')-Ia_1 (present)", "ant(3'')-Ia_1 (absent)")))
d4 <- d4 %>% mutate(`aph(3')-Ia_1` = factor(ifelse(`aph(3')-Ia_1` == 1, "aph(
3')-Ia_1 (present)", "aph(3')-Ia_1 (absent)")))
d4 <- d4 %>% mutate(`blaCARB-2_1` = factor(ifelse(`blaCARB-2_1` == 1, "blaCAR
B-2_1 (present)", "blaCARB-2_1 (absent)")))
d4 <- d4 %>% mutate(catA1_1 = factor(ifelse(catA1_1 == 1, "catA1_1 (present)"
, "catA1_1 (absent)")))
d4 <- d4 %>% mutate(sul1_5 = factor(ifelse(sul1_5 == 1, "sul1_5 (present)", "
sul1_5 (absent)")))
d4 <- d4 %>% mutate(`tet(A)_6` = factor(ifelse(`tet(A)_6` == 1, "tet(A)_6 (pr
esent)", "tet(A)_6 (absent)")))
d4 <- d4 %>% mutate(`tet(B)_2` = factor(ifelse(`tet(B)_2` == 1, "tet(B)_2 (pr
esent)", "tet(B)_2 (absent)")))
```

```
d4 <- d4 %>% mutate(`aph(3'')-Ib_5` = factor(ifelse(`aph(3'')-Ib_5` == 1, "aph(3'')-Ib_5 (present)", "aph(3'')-Ib_5 (absent)")))
d4 <- d4 %>% mutate(`aph(3')-Ia_9` = factor(ifelse(`aph(3')-Ia_9` == 1, "aph(3')-Ia_9 (present)", "aph(3')-Ia_9 (absent)")))
d4 <- d4 %>% mutate(`aph(6)-Id_1` = factor(ifelse(`aph(6)-Id_1` == 1, "aph(6)-Id_1 (present)", "aph(6)-Id_1 (absent)")))
d4 <- d4 %>% mutate(`blaCMY-2_1` = factor(ifelse(`blaCMY-2_1` == 1, "blaCMY-2_1 (present)", "blaCMY-2_1 (absent)")))
d4 <- d4 %>% mutate(floR_2 = factor(ifelse(floR_2 == 1, "floR_2  (present)", "floR_2 (absent)")))
d4 <- d4 %>% mutate(sul2_2 = factor(ifelse(sul2_2 == 1, "sul2_2 (present)", "sul2_2 (absent)")))
d4 <- d4 %>% mutate(dfrA1_8 = factor(ifelse(dfrA1_8 == 1, "dfrA1_8 (present)", "dfrA1_8 (absent)")))
d4 <- d4 %>% mutate(`mph(A)_2` = factor(ifelse(`mph(A)_2` == 1, "mph(A)_2 (present)", "mph(A)_2 (absent)")))
d4 <- d4 %>% mutate(qnrA1_1 = factor(ifelse(qnrA1_1 == 1, "qnrA1_1 (present)", "qnrA1_1 (absent)")))
##########################################-------#############################################
##########################################-------#############################################
# enter the phylogeny data
tree <- read.tree("~/Documents/jove_paper/data/newport_phylogeny.tree")
# bring data containing hierarchical genotypes and AMR loci
d5 <- d4
# to plot with the phylogeny using ggtree we need to make the id column into index first
d6 <- column_to_rownames(d5, var = "id")
# create the tree
# adjusting the parameters to make the tree visible or however you wish requires some trial and error
tree_plot <- ggtree(tree, layout = "circular") + xlim(-50, NA)
# plot figure 1 - color scheme for each layer of the plot should be chosen based on the user preferences
figure_5 <- gheatmap(tree_plot, d6, offset=.0, width=50, colnames = FALSE) +
# visualize the tree with metadata
  scale_fill_manual(values = c("coral", "darkblue",
                               "cornflowerblue", "coral", "purple", "red", "brown", "darkseagreen3", "darkblue", "darkgreen", "yellow",
                               "orange", "darkblue", "purple", "darkgreen", "darkred", "steelblue", "black", "coral",
                               "black", "darkgreen", "purple", "darkblue", "gray",
                               "darkblue", "gray",
                               "darkblue", "gray",
                               "darkblue", "gray",
                               "darkblue", "gray",
                               "darkblue", "gray",
                               "darkblue", "gray",
```

```
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray"),
                    breaks = c("Newport", "Other serovars",
                                "BAPS1 sub-group 1", "BAPS1 sub-group 2", "BAP
S1 sub-group 3", "BAPS1 sub-group 4",
                                "BAPS1 sub-group 5", "BAPS1 sub-group 6", "BAP
S1 sub-group 7", "BAPS1 sub-group 8",
                                "BAPS1 sub-group 9",
                                "ST5", "ST31", "ST45", "ST46", "ST118", "ST132
", "ST350", "Other STs",
                                "cgMLST 1468400426", "cgMLST 1271156802", "cgM
LST 3336043520", "cgMLST 88443731", "Other cgMLSTs",
                                "aac(6')-Iaa_1 (present)", "aac(6')-Iaa_1 (abs
ent)",
                                "mdf(A)_1 (present)", "mdf(A)_1 (absent)",
                                "aadA2_1 (present)", "aadA2_1 (absent)",
                                "ant(3'')-Ia_1 (present)", "ant(3'')-Ia_1 (abs
ent)",
                                "aph(3')-Ia_1 (present)", "aph(3')-Ia_1 (absen
t)",
                                "blaCARB-2_1 (present)", "blaCARB-2_1 (absent)
",
                                "catA1_1 (present)", "catA1_1 (absent)",
                                "sul1_5 (present)", "sul1_5 (absent)",
                                "tet(A)_6 (present)", "tet(A)_6 (absent)",
                                "tet(B)_2 (present)", "tet(B)_2 (absent)",
                                "aph(3'')-Ib_5 (present)", "aph(3'')-Ib_5 (abs
ent)",
                                "aph(3')-Ia_9 (present)", "aph(3')-Ia_9 (absen
t)",
                                "aph(6)-Id_1 (present)", "aph(6)-Id_1 (absent)
",
                                "blaCMY-2_1 (present)", "blaCMY-2_1 (absent)",
                                "floR_2  (present)", "floR_2 (absent)",
                                "sul2_2 (present)", "sul2_2 (absent)",
                                "dfrA1_8 (present)", "dfrA1_8 (absent)",
                                "mph(A)_2 (present)", "mph(A)_2 (absent)",
                                "qnrA1_1 (present)", "qnrA1_1 (absent)"),
                    name="Serovar -> BAPS1 -> ST -> cgMLST -> AMR loci") +  #
```

```
add colors and labels for the scale
  theme(legend.title=element_text(size=24, face = "bold"),
    legend.text=element_text(size=22)) +
        theme(legend.position = "none")  # customize figure's title, legend,
font

## Warning: attributes are not identical across measure variables;
## they will be dropped

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.

## Scale for 'fill' is already present. Adding another scale for 'fill', whic
h
## will replace the existing scale.

figure_5
```

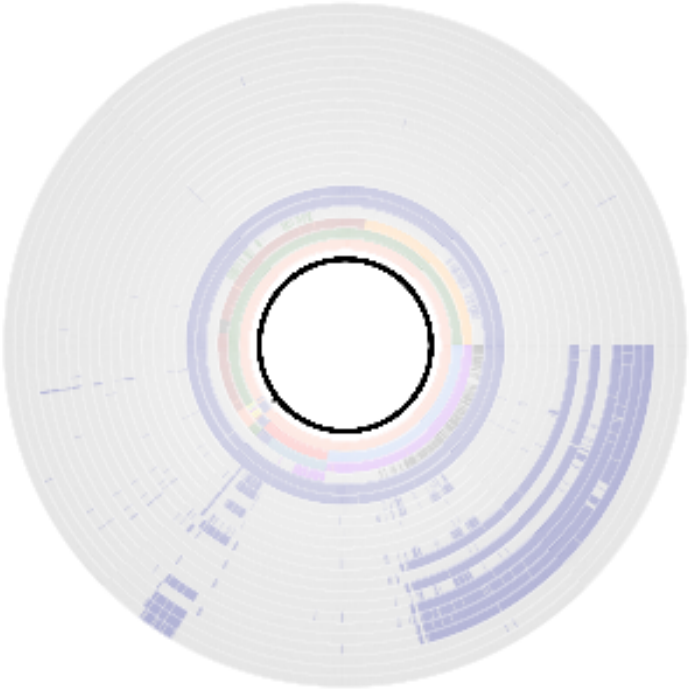


Figure 5 with legend. Phylogeny-guided mapping of hierarchical genotypes along with ST-differentiating AMR loci.

```
# enter the ST data
mlst <- read_csv('~/Documents/jove_paper/data/salmonellast_output.csv')
```

```
##
## ── Column specification ────────────────────────────────────────────
──────
## cols(
##   FILE = col_character(),
##   SCHEME = col_character(),
##   ST = col_character(),
##   aroC = col_character(),
##   dnaN = col_character(),
##   hemD = col_character(),
##   hisD = col_character(),
##   purE = col_character(),
##   sucA = col_character(),
##   thrA = col_character()
## )

# generate the id column
mlst$id2 <- sapply(strsplit(as.character(mlst$FILE),'_'), "[", 1)
# select columns of interest and rename -/? to NAs
data1 <- mlst %>%
          select(id2, ST) %>% # select columns
            mutate_all(na_if, "-") %>% # change - to NA
              mutate_all(na_if, "?") %>% # change ? to NA
                  rename(id = id2) # rename column
# categorize ST data to aggregate minor STs and keep only major ST lineages s
eparately
data1$st <- ifelse(data1$ST == 5, "ST5", # aggregate minor STs as Others
                   ifelse(data1$ST == 31, "ST31",
                          ifelse(data1$ST == 45, "ST45",
                                 ifelse(data1$ST == 46, "ST46",
                                        ifelse(data1$ST == 118, "ST118",
                                               ifelse(data1$ST == 132, "ST132
",
                          ifelse(data1$ST == 350, "ST350", "Other STs")))))))
# filter out the numerical ST column
data1 <- data1 %>% select(-ST)
##########################################--------############################
#################
##########################################--------############################
#################
# enter the Resfinder loci databases (AMR loci)
abx <- read_csv('~/Documents/jove_paper/data/sabricate_resfinder_output.csv')

##
## ── Column specification ────────────────────────────────────────────
──────
## cols(
##   `#FILE` = col_character(),
##   SEQUENCE = col_character(),
##   START = col_double(),
```

```
##   END = col_double(),
##   GENE = col_character(),
##   COVERAGE = col_character(),
##   COVERAGE_MAP = col_character(),
##   GAPS = col_character(),
##   `%COVERAGE` = col_double(),
##   `%IDENTITY` = col_double(),
##   DATABASE = col_character(),
##   ACCESSION = col_character(),
##   PRODUCT = col_character()
## )

# create the id column
abx$id <- sapply(strsplit(as.character(abx$'#FILE'),'_'), "[", 1)
# change the name of the GENE column
abx <- abx %>% mutate(gene = GENE)
# select columns of interest
abx1 <- abx %>% select(id, gene)
##########################################-------############################
#################
##########################################-------############################
#################
# join ST and AMR loci data
d1 <- left_join(data1, abx1, on = "id")

## Joining, by = "id"

# check for NAs or missing values
skim(d1)
```

Data summary

Name	d1
Number of rows	9169
Number of columns	3

_____

Column type frequency:

character	3

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
st	11	1	3	9	0	8	0

gene	0	1	6	15	0	127	0

```
# group ST missing values as "Other STs"
d1 <- d1 %>% mutate(st = replace_na(st, "Other STs"))
# check for NAs again
skim(d1)
```

Data summary

Name	d1
Number of rows	9169
Number of columns	3

_____

Column type frequency:	
character	3

_____

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	9	11	0	2365	0
st	0	1	3	9	0	8	0
gene	0	1	6	15	0	127	0

```
##########################################-------#############################
#################
##########################################-------#############################
#################
# calculate the proportion of AMR loci for each ST lineage
# calculations for ST5
d2a <- d1 %>% filter(st == "ST5")
# for ST5, calculate the proportion of AMR loci and only keep proportion grea
ter than 10%
d3a <- d2a %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4a <- d3a %>%
         group_by(gene) %>% # group by column
```

```
        summarize(value = n()) %>% # count observations
        mutate(total = table(data1$st)[8]) %>% # get the total number of obs
ervations per ST
        mutate(prop = (value/total)*100) # calculate proportions
d5a <- d4a %>% mutate(st = "ST5")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST31
d2b <- d1 %>% filter(st == "ST31")
# for ST31, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3b <- d2b %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold
```

```
## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.
```

```
d4b <- d3b %>%
        group_by(gene) %>% # group by column
        summarize(value = n()) %>% # count observations
        mutate(total = table(data1$st)[4]) %>% # get the total number of obs
ervations per ST
        mutate(prop = (value/total)*100) # calculate proportions
d5b <- d4b %>% mutate(st = "ST31")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST45
d2c <- d1 %>% filter(st == "ST45")
# for ST45, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3c <- d2c %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold
```

```
## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.
```

```r
d4c <- d3c %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[6]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5c <- d4c %>% mutate(st = "ST45")
##############################################--------############################
#################
##############################################--------############################
#################
# calculations for ST46
d2d <- d1 %>% filter(st == "ST46")
# for ST46, calculate the proportion of AMR loci and only keep proportion gre
ater than 10%
d3d <- d2d %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold
```

```
## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.
```

```r
d4d <- d3d %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[7]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5d <- d4d %>% mutate(st = "ST46")
##############################################--------############################
#################
##############################################--------############################
#################
# calculations for ST118
d2e <- d1 %>% filter(st == "ST118")
# for ST118, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3e <- d2e %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold
```

```
## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4e <- d3e %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[2]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5e <- d4e %>% mutate(st = "ST118")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST132
d2f <- d1 %>% filter(st == "ST132")
# for ST132, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3f <- d2f %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4f <- d3f %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[3]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5f <- d4f %>% mutate(st = "ST132")
##########################################-------############################
#################
##########################################-------############################
#################
# calculations for ST350
d2g <- d1 %>% filter(st == "ST350")
# for ST350, calculate the proportion of AMR loci and only keep proportion gr
eater than 10%
d3g <- d2g %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
```

```
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4g <- d3g %>%
         group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[5]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5g <- d4g %>% mutate(st = "ST350")
##############################################--------############################
#################
##############################################--------############################
#################
# calculations for Other ST lineages
d2h <- d1 %>% filter(st == "Other STs")
# for other STs, calculate the proportion of AMR loci and only keep proportio
n greater than 10%
d3h <- d2h %>%
    select(id, gene) %>% # select columns
    group_by(id, gene) %>% # group by columns
    summarize(count = n()) %>% # count observations
    mutate(count = replace(count, count == 2, 1)) %>% # replace values in a c
olumn
    filter(count <= 1) # filter based on a threshold

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

d4h <- d3h %>% group_by(gene) %>% # group by column
         summarize(value = n()) %>% # count observations
         mutate(total = table(data1$st)[1]) %>% # get the total number of obs
ervations per ST
         mutate(prop = (value/total)*100) # calculate proportions
d5h <- d4h %>% mutate(st = "Other STs")
##############################################--------############################
#################
##############################################--------############################
#################
# combined datasets
d6 <- rbind(d5a, d5b, d5c, d5d, d5e, d5f, d5g, d5h)
##############################################--------############################
#################
##############################################--------############################
#################
# filter AMR loci with proportion higher than or equal to %
d7 <- d6 %>% filter(prop >= 10)
# create a list of genes with proportion >= 10%
```

```r
d8 <- d7 %>% select(gene) %>% unique()
# transform dataframe to list of characters or vector
d9 <- pull(d8, gene)
##########################################-------#############################
#################
##########################################-------#############################
#################
# spread the abx1 table - from long to wide format
# count gene occurrences
abx2 <- abx1 %>%
    group_by(id, gene) %>% # group by id and gene
    summarize(count = n()) %>% # count the number of observations
    filter(count <= 1) # filter genes with 0 or 1 counts

## `summarise()` has grouped output by 'id'. You can override using the `.gro
ups` argument.

# spread from long to wide format
abx3 <- spread(abx2, key = gene, value = count)
# select columns of interest
abx4 <- abx3 %>% select(d9)

## Adding missing grouping variables: `id`

# replace all NAs with zeros
abx4[is.na(abx4)] <- 0
# merge with original data containing the hierarchical genotypes
d4 <- left_join(d3, abx4, on = "id")

## Joining, by = "id"

##########################################-------#############################
#################
##########################################-------#############################
#################
# change binary values from AMR data locus by locus
d4 <- d4 %>% mutate(`aac(6')-Iaa_1` = factor(ifelse(`aac(6')-Iaa_1` == 1, "aa
c(6')-Iaa_1 (present)", "aac(6')-Iaa_1 (absent)")))
d4 <- d4 %>% mutate(`mdf(A)_1` = factor(ifelse(`mdf(A)_1` == 1, "mdf(A)_1 (pr
esent)", "mdf(A)_1 (absent)")))
d4 <- d4 %>% mutate(aadA2_1 = factor(ifelse(aadA2_1 == 1, "aadA2_1 (present)"
, "aadA2_1 (absent)")))
d4 <- d4 %>% mutate(`ant(3'')-Ia_1` = factor(ifelse(`ant(3'')-Ia_1` == 1, "an
t(3'')-Ia_1 (present)", "ant(3'')-Ia_1 (absent)")))
d4 <- d4 %>% mutate(`aph(3')-Ia_1` = factor(ifelse(`aph(3')-Ia_1` == 1, "aph(
3')-Ia_1 (present)", "aph(3')-Ia_1 (absent)")))
d4 <- d4 %>% mutate(`blaCARB-2_1` = factor(ifelse(`blaCARB-2_1` == 1, "blaCAR
B-2_1 (present)", "blaCARB-2_1 (absent)")))
d4 <- d4 %>% mutate(catA1_1 = factor(ifelse(catA1_1 == 1, "catA1_1 (present)"
, "catA1_1 (absent)")))
d4 <- d4 %>% mutate(sul1_5 = factor(ifelse(sul1_5 == 1, "sul1_5 (present)", "
```

```
sul1_5 (absent)")))
d4 <- d4 %>% mutate(`tet(A)_6` = factor(ifelse(`tet(A)_6` == 1, "tet(A)_6 (pr
esent)", "tet(A)_6 (absent)")))
d4 <- d4 %>% mutate(`tet(B)_2` = factor(ifelse(`tet(B)_2` == 1, "tet(B)_2 (pr
esent)", "tet(B)_2 (absent)")))
d4 <- d4 %>% mutate(`aph(3'')-Ib_5` = factor(ifelse(`aph(3'')-Ib_5` == 1, "ap
h(3'')-Ib_5 (present)", "aph(3'')-Ib_5 (absent)")))
d4 <- d4 %>% mutate(`aph(3')-Ia_9` = factor(ifelse(`aph(3')-Ia_9` == 1, "aph(
3')-Ia_9 (present)", "aph(3')-Ia_9 (absent)")))
d4 <- d4 %>% mutate(`aph(6)-Id_1` = factor(ifelse(`aph(6)-Id_1` == 1, "aph(6)
-Id_1 (present)", "aph(6)-Id_1 (absent)")))
d4 <- d4 %>% mutate(`blaCMY-2_1` = factor(ifelse(`blaCMY-2_1` == 1, "blaCMY-2
_1 (present)", "blaCMY-2_1 (absent)")))
d4 <- d4 %>% mutate(floR_2 = factor(ifelse(floR_2 == 1, "floR_2  (present)",
"floR_2 (absent)")))
d4 <- d4 %>% mutate(sul2_2 = factor(ifelse(sul2_2 == 1, "sul2_2 (present)", "
sul2_2 (absent)")))
d4 <- d4 %>% mutate(dfrA1_8 = factor(ifelse(dfrA1_8 == 1, "dfrA1_8 (present)"
, "dfrA1_8 (absent)")))
d4 <- d4 %>% mutate(`mph(A)_2` = factor(ifelse(`mph(A)_2` == 1, "mph(A)_2 (pr
esent)", "mph(A)_2 (absent)")))
d4 <- d4 %>% mutate(qnrA1_1 = factor(ifelse(qnrA1_1 == 1, "qnrA1_1 (present)"
, "qnrA1_1 (absent)")))
##########################################--------############################
#################
##########################################-------#############################
#################
# enter the phylogeny data
tree <- read.tree("~/Documents/jove_paper/data/newport_phylogeny.tree")
# bring data containing hierarchical genotypes and AMR loci
d5 <- d4
# to plot with the phylogeny using ggtree we need to make the id column into
index first
d6 <- column_to_rownames(d5, var = "id")
# create the tree
# adjusting the parameters to make the tree visible or however you wish requi
res some trial and error
tree_plot <- ggtree(tree, layout = "circular") + xlim(-50, NA)
# plot figure 1 - color scheme for each layer of the plot should be chosen ba
sed on the user preferences
figure_5b <- gheatmap(tree_plot, d6, offset=.0, width=50, colnames = FALSE) +
# visualize the tree with metadata
  scale_fill_manual(values = c("coral", "darkblue",
                               "cornflowerblue", "coral", "purple", "red", "b
rown", "darkseagreen3", "darkblue", "darkgreen", "yellow",
                               "orange", "darkblue", "purple", "darkgreen", "
darkred", "steelblue", "black", "coral",
                               "black", "darkgreen", "purple", "darkblue", "g
ray",
                               "darkblue", "gray",
```

```
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray",
                                "darkblue", "gray"),
                    breaks = c("Newport", "Other serovars",
                                "BAPS1 sub-group 1", "BAPS1 sub-group 2", "BAP
S1 sub-group 3", "BAPS1 sub-group 4",
                                "BAPS1 sub-group 5", "BAPS1 sub-group 6", "BAP
S1 sub-group 7", "BAPS1 sub-group 8",
                                "BAPS1 sub-group 9",
                                "ST5", "ST31", "ST45", "ST46", "ST118", "ST132
", "ST350", "Other STs",
                                "cgMLST 1468400426", "cgMLST 1271156802", "cgM
LST 3336043520", "cgMLST 88443731", "Other cgMLSTs",
                                "aac(6')-Iaa_1 (present)", "aac(6')-Iaa_1 (abs
ent)",
                                "mdf(A)_1 (present)", "mdf(A)_1 (absent)",
                                "aadA2_1 (present)", "aadA2_1 (absent)",
                                "ant(3'')-Ia_1 (present)", "ant(3'')-Ia_1 (abs
ent)",
                                "aph(3')-Ia_1 (present)", "aph(3')-Ia_1 (absen
t)",
                                "blaCARB-2_1 (present)", "blaCARB-2_1 (absent)
",
                                "catA1_1 (present)", "catA1_1 (absent)",
                                "sul1_5 (present)", "sul1_5 (absent)",
                                "tet(A)_6 (present)", "tet(A)_6 (absent)",
                                "tet(B)_2 (present)", "tet(B)_2 (absent)",
                                "aph(3'')-Ib_5 (present)", "aph(3'')-Ib_5 (abs
ent)",
                                "aph(3')-Ia_9 (present)", "aph(3')-Ia_9 (absen
t)",
                                "aph(6)-Id_1 (present)", "aph(6)-Id_1 (absent)
",
                                "blaCMY-2_1 (present)", "blaCMY-2_1 (absent)",
                                "floR_2  (present)", "floR_2 (absent)",
```

```
                                "sul2_2 (present)", "sul2_2 (absent)",
                                "dfrA1_8 (present)", "dfrA1_8 (absent)",
                                "mph(A)_2 (present)", "mph(A)_2 (absent)",
                                "qnrA1_1 (present)", "qnrA1_1 (absent)"),
                     name="Serovar -> BAPS1 -> ST -> cgMLST -> AMR loci") +  #
add colors and labels for the scale
  theme(legend.title=element_text(size=24, face = "bold"),
    legend.text=element_text(size=22)) # customize figure's title, legend, fo
nt

## Warning: attributes are not identical across measure variables;
## they will be dropped

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.

## Scale for 'fill' is already present. Adding another scale for 'fill', whic
h
## will replace the existing scale.

figure_5b
```