# SeqAfrica_data_webinar

2025-11-04

```
# Load the tidyverse and readxl packages
library(readxl)
library(readr)
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
#Set my working directory (folder) to where my excel file is saved
setwd("/Users/nlac/Documents/2025/SeqAfrica/Webinars/Understanding_data")
getwd()
```

```
## [1] "/Users/nlac/Documents/2025/SeqAfrica/Webinars/Understanding_data"
```

```
# Read Excel file
qc_amr_data <- read_excel("Example_sequencing_results.xlsx", sheet = 1)

# Save as TSV
write_tsv(qc_amr_data, "qc_results.tsv")

qc <- read_tsv("qc_results.tsv")
```

```
## Rows: 11 Columns: 130
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (120): SampleID, Species_Identification, refseq_closest_match, Genera_pr...
## dbl  (10): Total.reads, Total.bases, Clean.reads, Clean.bases, CoverageDepth...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
qc
```

```
## # A tibble: 11 x 130
##    SampleID Species_Identification     refseq_closest_match       Genera_present
##    <chr>    <chr>                      <chr>                      <chr>
##  1 Control  Escherichia coli (100%)    Escherichia coli O157:H7 ~ Escherichia
```

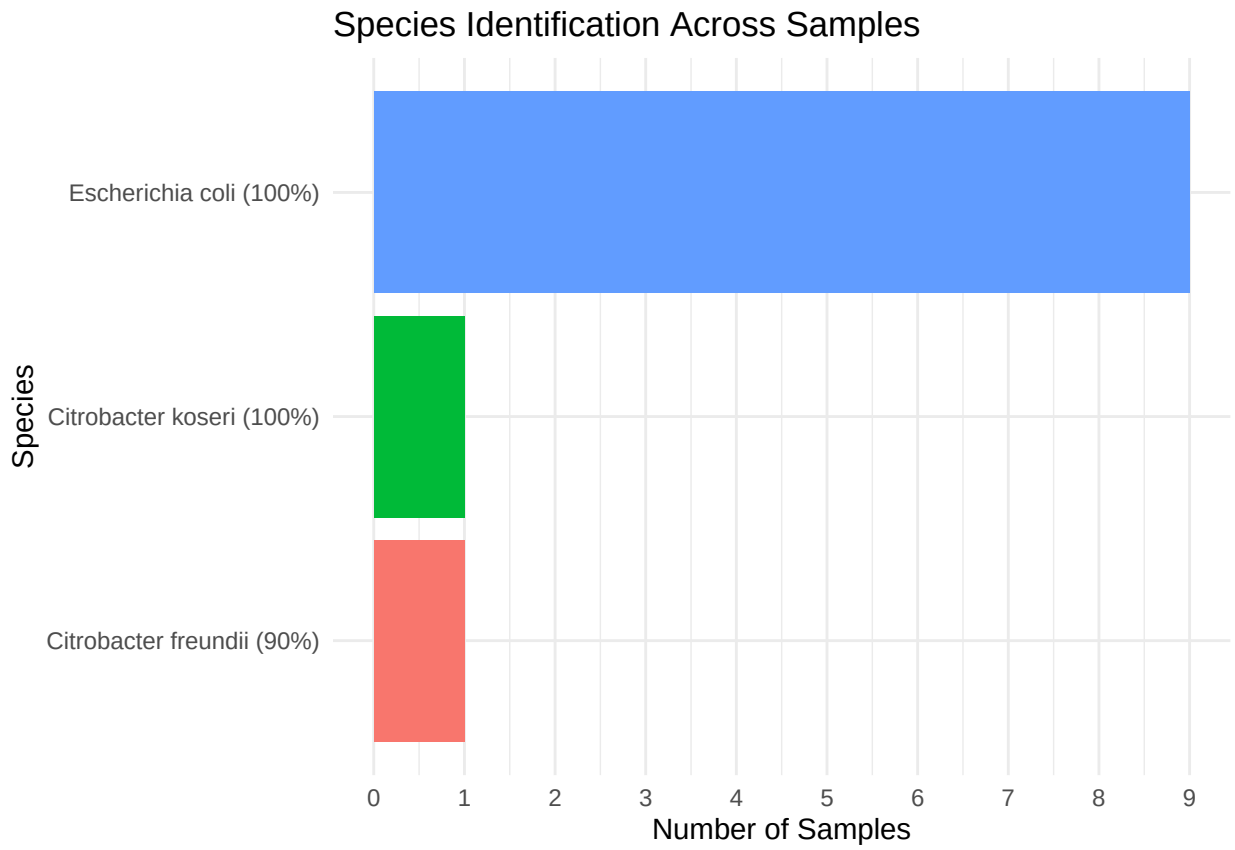
```
##  2 Sample1  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia:C~
##  3 Sample2  Citrobacter freundii (90%) Citrobacter freundii CFNI~ Citrobacter
##  4 Sample3  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia
##  5 Sample4  Escherichia coli (100%)    Escherichia coli IAI39     Escherichia
##  6 Sample5  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia
##  7 Sample6  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia
##  8 Sample7  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia
##  9 Sample8  Citrobacter koseri (100%)  Citrobacter freundii CFNI~ Citrobacter
## 10 Sample9  Escherichia coli (100%)    Escherichia coli O104:H4 ~ Escherichia
## 11 Sample10 Escherichia coli (100%)    Escherichia coli str. K-1~ Escherichia
## # i 126 more variables: ContaminationPresent <chr>, `kraken2_match_#1` <chr>,
## #   `kraken2_match_#2` <chr>, `kraken2_match_#3` <chr>,
## #   `kraken2_match_#4` <chr>, Total.reads <dbl>, Total.bases <dbl>,
## #   Clean.reads <dbl>, Clean.bases <dbl>, CoverageDepth <dbl>,
## #   Contig.num <dbl>, Contigs.GC.content <dbl>, N50.value <dbl>,
## #   Longest.contig <dbl>, Total.bases.assembly <dbl>, Scheme.MLST <chr>,
## #   ST <chr>, adk <chr>, fumC <chr>, gyrB <chr>, icd <chr>, mdh <chr>, ...
```

```r
# Load the tidyverse package (includes ggplot2, dplyr, and readr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

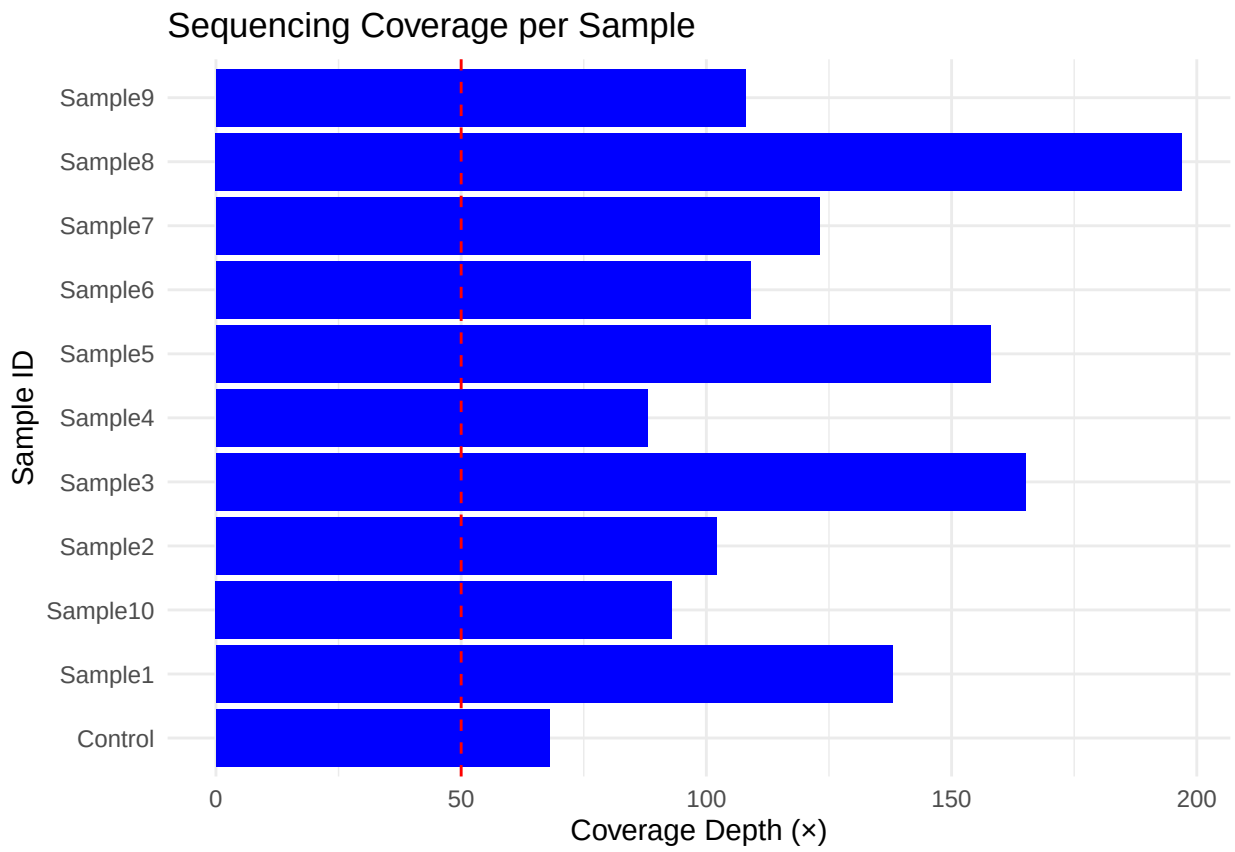
```r
# Create a bar plot showing how many samples belong to each species
qc %>%
  # Count the number of occurrences for each species in the dataset
  count(Species_Identification) %>%

  # Start a ggplot using the counted data
  ggplot(aes(
    x = reorder(Species_Identification, n),  # reorder species by count
    y = n,                                   # number of samples per species
    fill = Species_Identification             # fill color by species
  )) +

  # Draw bars (columns)
  geom_col(show.legend = FALSE) +             # hide legend since species are on the axis

  # Flip coordinates so species names appear on the y-axis (easier to read)
  coord_flip() +

  # Use a simple clean theme
  theme_minimal() +

  # Customize the x-axis to show integer breaks (1, 2, 3, ...)
```

```
  scale_y_continuous(breaks = seq(0, max(qc %>% count(Species_Identification) %>% pull(n)), by = 1)) +

  # Add informative labels and a title
  labs(
    title = "Species Identification Across Samples",
    x = "Species",
    y = "Number of Samples"
  )
```




```
# Use ggplot2 (part of the tidyverse) to visualize sequencing coverage for each sample
ggplot(qc, aes(x = SampleID, y = CoverageDepth)) +

  # Create vertical bars for each sample showing its average coverage depth
  geom_col(fill = "blue") +

  # Add a horizontal dashed red line at 50× coverage
  # This helps visualize the recommended minimum coverage for bacterial WGS
  geom_hline(yintercept = 50, linetype = "dashed", color = "red") +

  # Flip the plot so sample names are easier to read along the y-axis
  coord_flip() +

  # Apply a clean, minimalistic theme (no grid clutter)
  theme_minimal() +
```

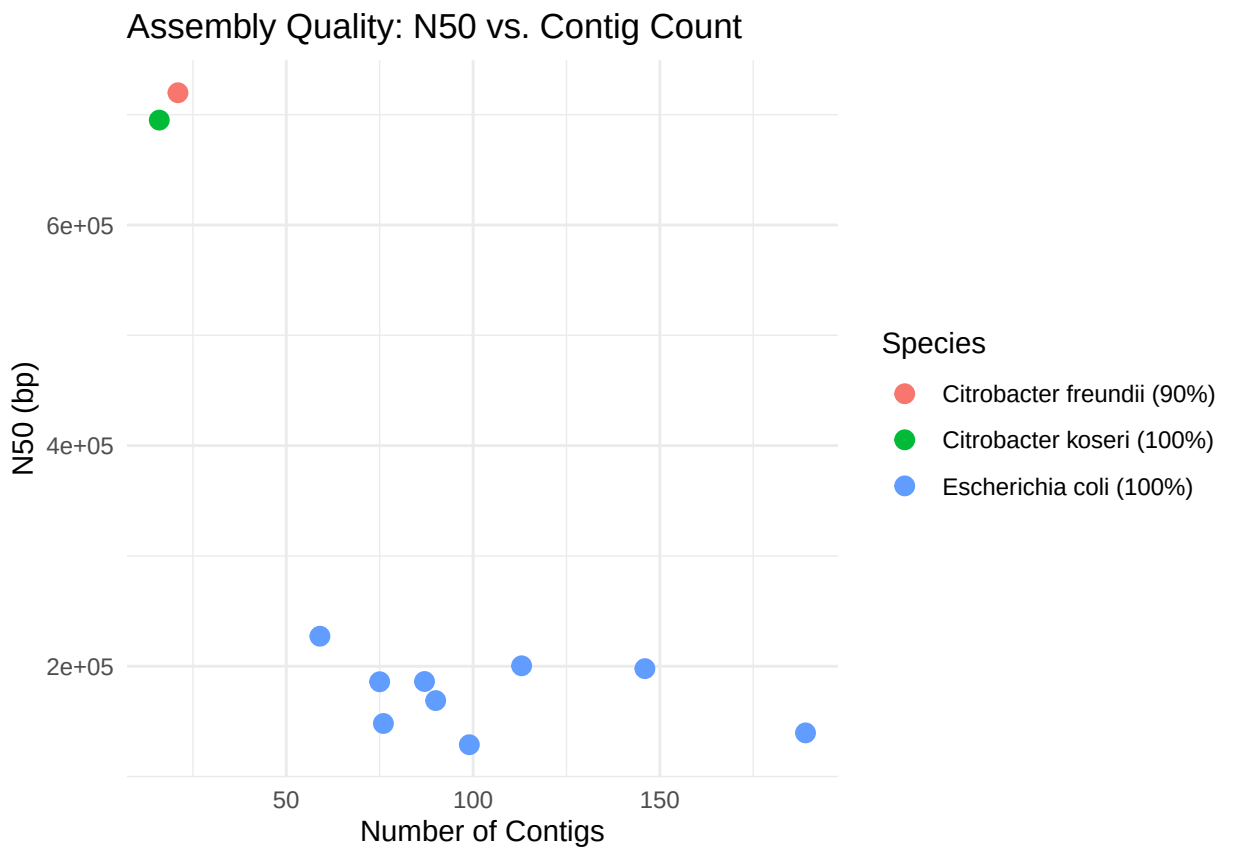
```
  # Add a descriptive title and axis labels
  labs(
    title = "Sequencing Coverage per Sample",
    x = "Sample ID",              # Label for the sample axis
    y = "Coverage Depth (×)"      # Label for the coverage axis
  )
```




The dashed red line at 50× shows the minimum recommended coverage for bacterial genomes.

```
# Create a scatter plot to assess assembly quality
# comparing the number of contigs (assembly fragmentation)
# against the N50 value (assembly contiguity)
ggplot(qc, aes(x = Contig.num, y = N50.value, color = Species_Identification)) +

  # Plot each sample as a point
  # The color of each point corresponds to its identified species
  geom_point(size = 3) +

  # Apply a clean, simple theme suitable for presentations
  theme_minimal() +

  # Add an informative title, axis labels, and legend title
  labs(
    title = "Assembly Quality: N50 vs. Contig Count",  # Overall plot title
    x = "Number of Contigs",                           # X-axis label
    y = "N50 (bp)",                                    # Y-axis label
```

```
    color = "Species"                                    # Legend label
)
```




Good-quality genomes cluster with low contig counts (<150) and high N50 (>100k). If you see outliers (many contigs + low N50), they might indicate poor DNA quality or contamination.

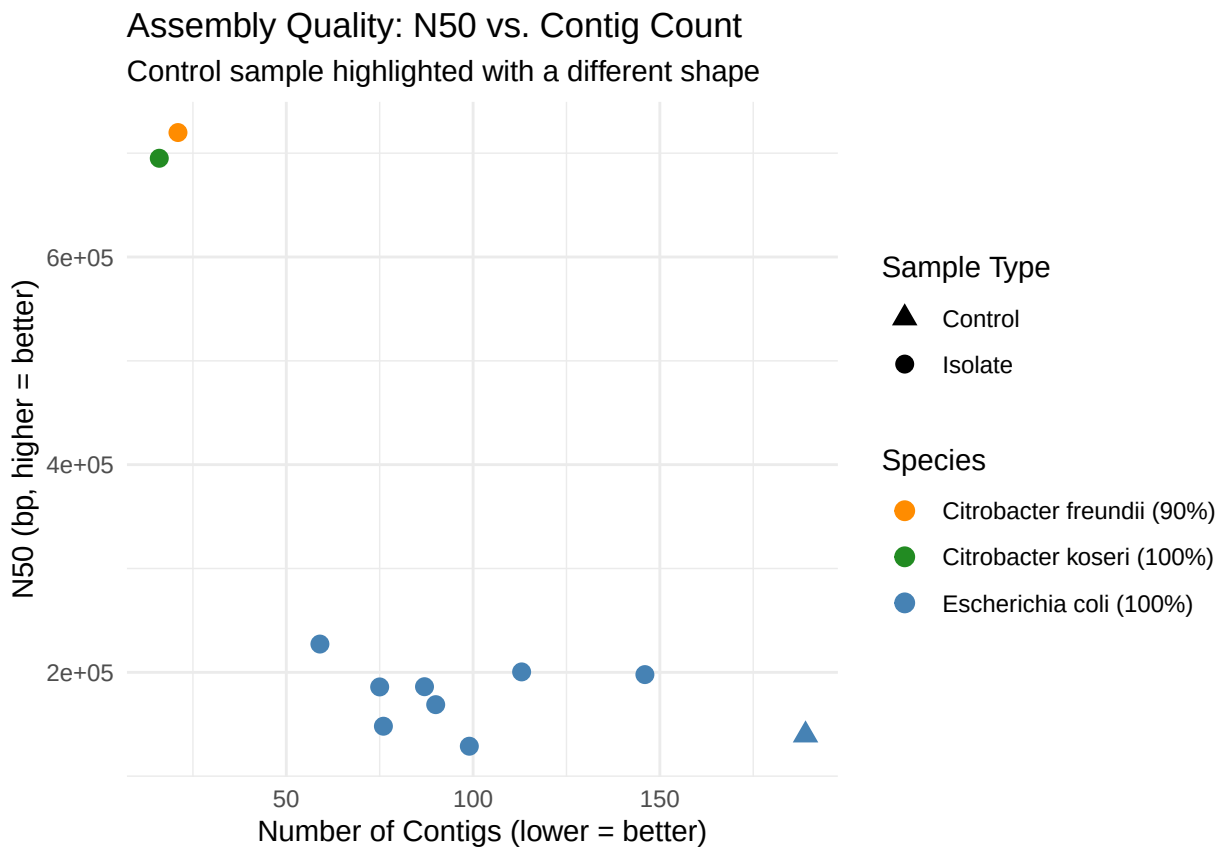
Each dot represents one sample. The x-axis shows how fragmented the genome assembly is (more contigs = more fragmented). The y-axis (N50) shows how contiguous the assembly is (higher = better). The color indicates species identity - so E. coli and Citrobacter appear in different colors. Ideal assemblies cluster in the top-left corner: → few contigs (low fragmentation) → high N50 (good contiguity). Lower right points would indicate poor assemblies (many small contigs, low N50).- But they are not poor quality, they're just fragmented, which is normal because Illumina reads (150–250 bp) can't span long repetitive regions.

Assume Citrobacter easier to resolve (simpler genome structure, higher coverage), not because the E. coli runs are bad.

When we look at this plot, we see that the E. coli assemblies cluster in the middle range, about 60–150 contigs and N50 values around 150–200 kb. That's actually normal for Illumina short-read assemblies and indicates good-quality data, even if the genomes are fragmented. In contrast, the Citrobacter isolates appear at the high-quality end, with very few contigs and much higher N50 values, which reflects very contiguous assemblies. So, the E. coli data are not low quality - they're typical of what we expect for this sequencing technology

```
# Create a new column to highlight the control sample
# This creates a new variable called "SampleType"
# It labels the control as "Control" and all others as "Sample"
qc <- qc %>%
```

```r
  mutate(SampleType = if_else(SampleID == "Control", "Control", "Isolate"))

# Plot N50 vs Contig number, colored by species but with a distinct shape/color for the control
ggplot(qc, aes(x = Contig.num, y = N50.value)) +

  # Plot all regular isolates (non-control) in grey with species-based fill color
  geom_point(
    aes(color = Species_Identification, shape = SampleType),
    size = 3
  ) +

  # Apply a simple clean theme
  theme_minimal() +

  # Customize colors and shapes to make the control stand out
  scale_color_manual(
    values = c(
      "Escherichia coli (100%)" = "steelblue",
      "Citrobacter freundii (90%)" = "darkorange",
      "Citrobacter koseri (100%)" = "forestgreen"
    )
  ) +
  scale_shape_manual(
    values = c("Control" = 17, "Isolate" = 16) # triangle for control, circle for others
  ) +

  # Add descriptive labels and title
  labs(
    title = "Assembly Quality: N50 vs. Contig Count",
    subtitle = "Control sample highlighted with a different shape",
    x = "Number of Contigs (lower = better)",
    y = "N50 (bp, higher = better)",
    color = "Species",
    shape = "Sample Type"
  )
```




Added symbols in to show control.

```
library(tidyverse)
# Read Excel file
amr <- read_excel("Example_sequencing_results.xlsx", sheet = 4)

# Save as TSV
write_tsv(amr, "amrfinderplus_results.tsv")

# Load your AMRFinderPlus subset
amr <- read_tsv("amrfinderplus_results.tsv", show_col_types = FALSE)

#Look at the data
amr
```

```
## # A tibble: 265 x 23
##    SampleID  `Protein identifier` `Contig id`  Start   Stop Strand `Gene symbol`
##    <chr>     <chr>                <chr>         <dbl>  <dbl> <chr>  <chr>
##  1 Control-~ No hit found         contig00001 202108 203526 +      espX1
##  2 Control-~ No hit found         contig00001 293676 294905 +      mdtM
##  3 Control-~ No hit found         contig00002 258695 262936 -      fdeC
##  4 Control-~ No hit found         contig00003 245537 246526 -      nleC
##  5 Control-~ No hit found         contig00003 246590 247567 -      nleB2
##  6 Control-~ No hit found         contig00011 137716 139071 -      glpT_E448K
##  7 Control-~ No hit found         contig00012  30484  31083 +      lpfA2
##  8 Control-~ No hit found         contig00012  89490  90677 -      emrD
##  9 Control-~ No hit found         contig00012 133461 135134 +      tir
```

```
## 10 Control-~ No hit found         contig00012 135805 138606 +      eae
## # i 255 more rows
## # i 16 more variables: `Sequence name` <chr>, Scope <chr>,
## #   `Element type` <chr>, `Element subtype` <chr>, Class <chr>, Subclass <chr>,
## #   Method <chr>, `Target length` <dbl>, `Reference sequence length` <dbl>,
## #   `% Coverage of reference sequence` <dbl>,
## #   `% Identity to reference sequence` <dbl>, `Alignment length` <dbl>,
## #   `Accession of closest sequence` <chr>, ...
```

```
colnames(amr)
```

```
##  [1] "SampleID"                          "Protein identifier"
##  [3] "Contig id"                         "Start"
##  [5] "Stop"                              "Strand"
##  [7] "Gene symbol"                       "Sequence name"
##  [9] "Scope"                             "Element type"
## [11] "Element subtype"                   "Class"
## [13] "Subclass"                          "Method"
## [15] "Target length"                     "Reference sequence length"
## [17] "% Coverage of reference sequence" "% Identity to reference sequence"
## [19] "Alignment length"                  "Accession of closest sequence"
## [21] "Name of closest sequence"          "HMM id"
## [23] "HMM description"
```

```
#Clean up column names (makes them lowercase, replaces spaces with underscores)
amr <- amr %>%
  clean_names()

amr
```

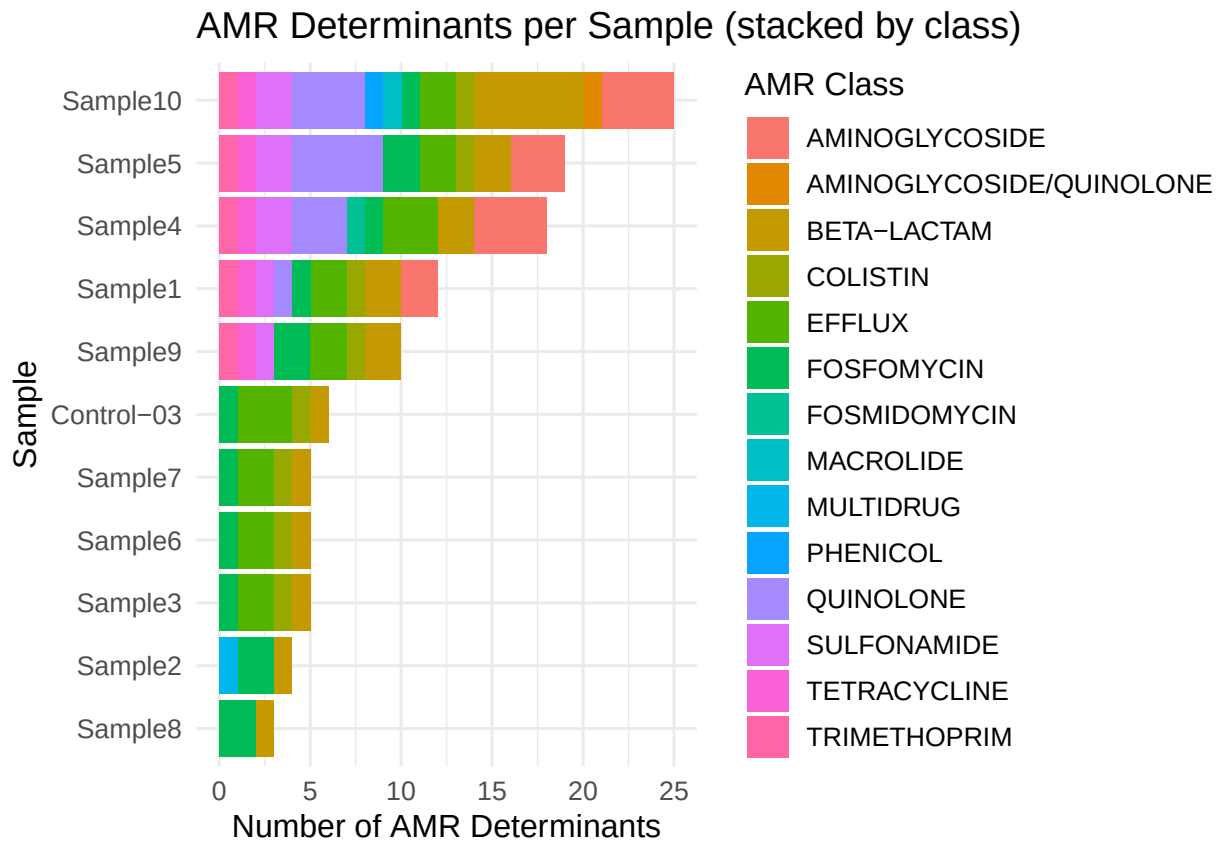
```
## # A tibble: 265 x 23
##    sample_id  protein_identifier contig_id    start   stop strand gene_symbol
##    <chr>      <chr>              <chr>        <dbl>  <dbl> <chr>  <chr>
##  1 Control-03 No hit found       contig00001 202108 203526 +      espX1
##  2 Control-03 No hit found       contig00001 293676 294905 +      mdtM
##  3 Control-03 No hit found       contig00002 258695 262936 -      fdeC
##  4 Control-03 No hit found       contig00003 245537 246526 -      nleC
##  5 Control-03 No hit found       contig00003 246590 247567 -      nleB2
##  6 Control-03 No hit found       contig00011 137716 139071 -      glpT_E448K
##  7 Control-03 No hit found       contig00012  30484  31083 +      lpfA2
##  8 Control-03 No hit found       contig00012  89490  90677 -      emrD
##  9 Control-03 No hit found       contig00012 133461 135134 +      tir
## 10 Control-03 No hit found       contig00012 135805 138606 +      eae
## # i 255 more rows
## # i 16 more variables: sequence_name <chr>, scope <chr>, element_type <chr>,
## #   element_subtype <chr>, class <chr>, subclass <chr>, method <chr>,
## #   target_length <dbl>, reference_sequence_length <dbl>,
## #   percent_coverage_of_reference_sequence <dbl>,
## #   percent_identity_to_reference_sequence <dbl>, alignment_length <dbl>,
## #   accession_of_closest_sequence <chr>, name_of_closest_sequence <chr>, ...
```

```r
#Clean up column names (makes them lowercase, replaces spaces with underscores)
amr <- amr %>%
  clean_names()

# Filter to keep only antimicrobial resistance (AMR) hits
#    and remove rows with "No hit found"
amr <- amr %>%
  filter(element_type == "AMR", gene_symbol != "No hit found")

# Display the cleaned and filtered results
amr
```

```
## # A tibble: 112 x 23
##    sample_id  protein_identifier contig_id    start   stop strand gene_symbol
##    <chr>      <chr>              <chr>         <dbl>  <dbl> <chr>  <chr>
##  1 Control-03 No hit found       contig00001 293676 294905 +      mdtM
##  2 Control-03 No hit found       contig00011 137716 139071 -      glpT_E448K
##  3 Control-03 No hit found       contig00012  89490  90677 -      emrD
##  4 Control-03 No hit found       contig00015  48587  49717 +      blaEC
##  5 Control-03 No hit found       contig00015  94099  95187 +      pmrB_Y358N
##  6 Control-03 No hit found       contig00016 115548 118694 +      acrF
##  7 Sample1    No hit found       contig00001 348698 350053 +      glpT_E448K
##  8 Sample1    No hit found       contig00009  75713  76942 +      mdtM
##  9 Sample1    No hit found       contig00015  49172  50302 +      blaEC
## 10 Sample1    No hit found       contig00015  94840  95928 +      pmrB_Y358N
## # i 102 more rows
## # i 16 more variables: sequence_name <chr>, scope <chr>, element_type <chr>,
## #   element_subtype <chr>, class <chr>, subclass <chr>, method <chr>,
## #   target_length <dbl>, reference_sequence_length <dbl>,
## #   percent_coverage_of_reference_sequence <dbl>,
## #   percent_identity_to_reference_sequence <dbl>, alignment_length <dbl>,
## #   accession_of_closest_sequence <chr>, name_of_closest_sequence <chr>, ...
```

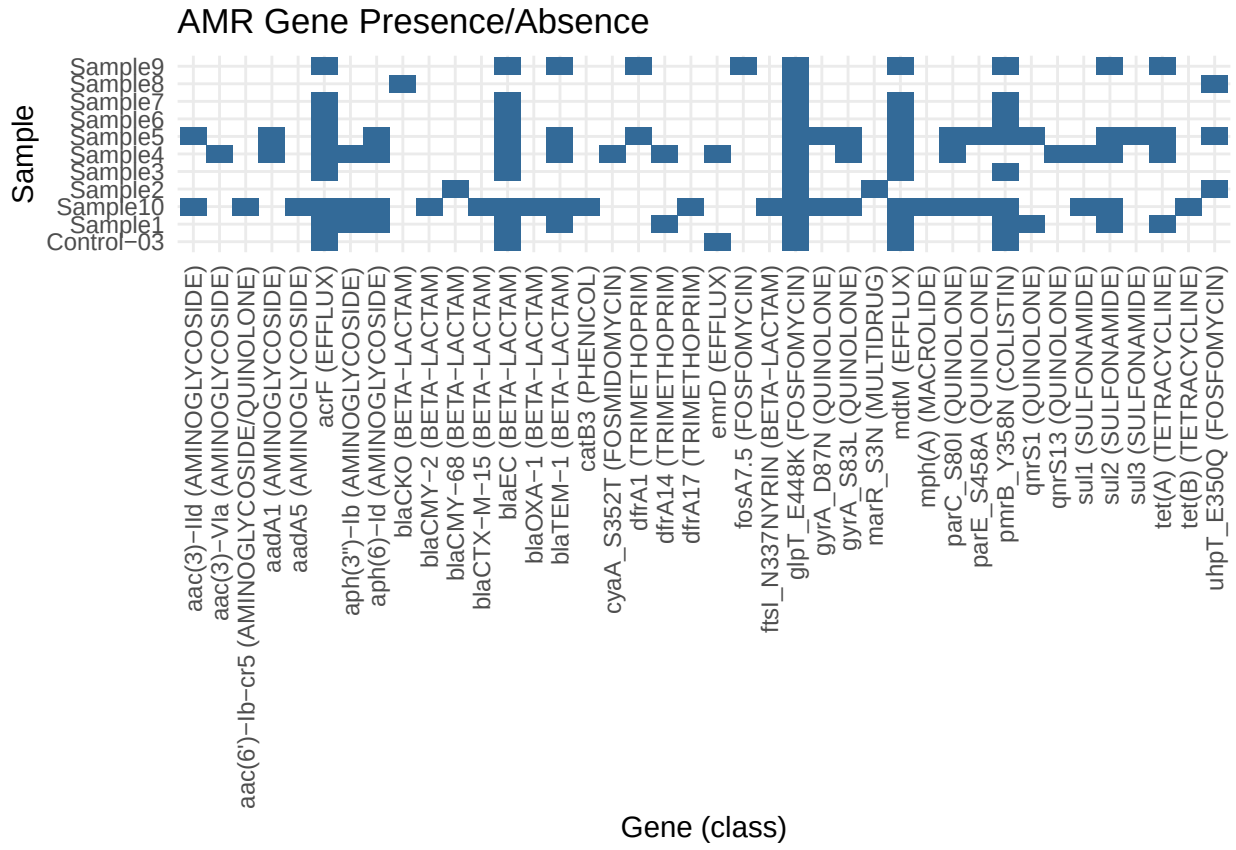
```r
# ---- AMR determinants per sample (stacked by class) ----
# This plot summarizes how many AMR genes each sample carries,
# grouped and color-coded by antibiotic resistance class.

library(tidyverse)
library(forcats)

amr %>%
  # Count the number of AMR hits per sample and per class
  # 'count()' groups by sample_id and class, and creates a column "hits" showing how many genes were fo
  count(sample_id, class, name = "hits") %>%

  # Start the ggplot visualization
  ggplot(aes(
    # Reorder samples on the x-axis by total number of hits
    # fct_reorder() ensures samples with more hits appear at the top when flipped
    x = fct_reorder(sample_id, hits, .fun = sum),

    # The height of each bar corresponds to the number of AMR hits
    y = hits,
```

```r
  # Color fill represents the AMR class (e.g., BETA-LACTAM, TETRACYCLINE, etc.)
  fill = class
)) +

# Draw stacked bars - one bar per sample, subdivided by class
geom_col() +

# Flip the coordinates so samples are on the y-axis
# This makes it easier to read long sample IDs and compare totals visually
coord_flip() +

# Add axis labels, legend title, and plot title
labs(
  x = "Sample",                    # Y-axis label after flipping
  y = "Number of AMR Determinants",  # Total number of detected AMR genes
  fill = "AMR Class",              # Legend label
  title = "AMR Determinants per Sample (stacked by class)"
) +

# Apply a clean and readable theme
theme_minimal(base_size = 12)
```




```r
# ---- Basic AMR gene presence/absence heatmap with comments ----

library(tidyverse)
```

```r
amr %>%
  #  Create a simple indicator showing that a gene is present
  # Since all entries in this dataset represent a gene hit, we assign "1" to all rows
  mutate(
    present = 1,

    # Combine the gene name and class into one label
    # This helps show both the gene identity and its resistance class on the x-axis
    gene_lab = paste0(gene_symbol, " (", class, ")")
  ) %>%

  # Keep only unique sample-gene combinations
  # This removes any duplicate rows if a gene was listed multiple times for the same sample
  distinct(sample_id, gene_lab, .keep_all = TRUE) %>%

  # Start building the plot
  ggplot(aes(
    x = gene_lab,         # Each column represents a different AMR gene (with class)
    y = sample_id,        # Each row is a different bacterial isolate or sample
    fill = present        # Fill color shows if the gene is present (1)
  )) +

  # Draw tiles for each sample-gene combination
  # geom_tile() creates a heatmap-like grid where each cell represents one pair
  geom_tile() +

  # Remove the legend since all tiles are "present" and legend isn't informative
  scale_fill_continuous(guide = "none") +

  # Add axis labels and a clear title
  labs(
    x = "Gene (class)",                   # The x-axis shows the AMR genes and their classes
    y = "Sample",                         # The y-axis shows the isolate or sample ID
    title = "AMR Gene Presence/Absence"   # Title summarizing what the plot shows
  ) +

  # Apply a clean, minimal theme for presentation clarity
  theme_minimal(base_size = 11) +

  # Rotate x-axis text labels 90° to prevent overlap (gene names are long!)
  theme(
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)
  )
```

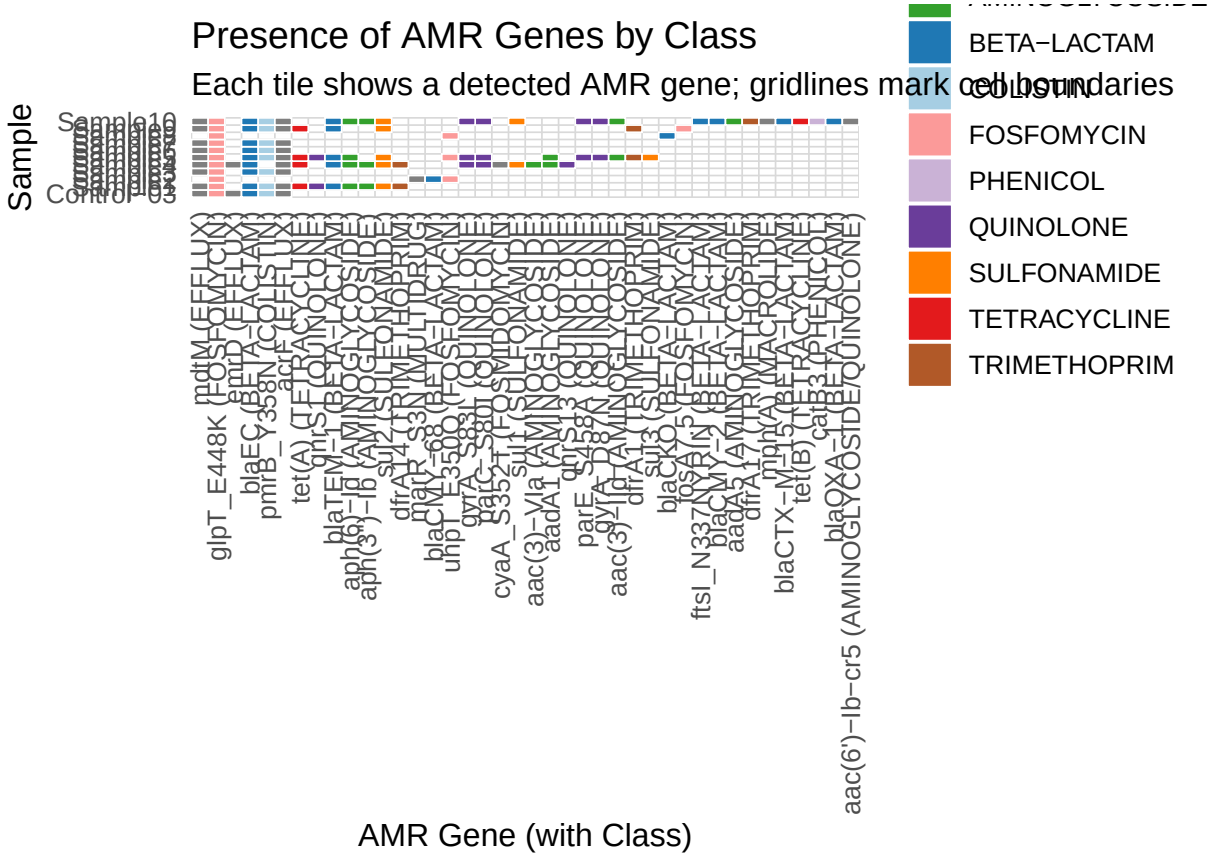



```r
# ---- AMR gene presence/absence heatmap with aligned grid ----
# This version makes the tile borders line up perfectly with a custom grid
# and includes detailed comments for teaching or webinar demonstrations.

# STEP 1: Prepare and clean the AMR data for plotting
pdat <- amr %>%
  mutate(
    # Create a binary variable for presence (since all entries here indicate presence)
    present = 1,

    # Combine the gene symbol and AMR class for clearer labels on the x-axis
    gene_lab = paste0(gene_symbol, " (", class, ")"),

    # Ensure sample IDs are factors (helps control order on y-axis)
    sample_id = factor(sample_id)
  ) %>%

  # Keep only unique combinations of sample and gene (avoid duplicates)
  distinct(sample_id, gene_lab, .keep_all = TRUE) %>%

  # Convert gene labels and sample IDs to factors to preserve order and spacing
  mutate(
    gene_lab  = factor(gene_lab),
    sample_id = factor(sample_id)
  )
```

```r
# STEP 2: Count the number of x and y categories to know where to draw grid lines
nx <- nlevels(pdat$gene_lab)
ny <- nlevels(pdat$sample_id)

# STEP 3: Build the heatmap
ggplot(pdat, aes(x = gene_lab, y = sample_id, fill = present)) +

  # ---- Draw custom grid lines that align perfectly with tiles ----
  # These vertical and horizontal lines are drawn at half-integer positions
  # (0.5, 1.5, 2.5, ...) so they sit *between* the discrete tile positions.
  geom_vline(xintercept = seq(0.5, nx + 0.5, by = 1),
             color = "grey85", linewidth = 0.3) +
  geom_hline(yintercept = seq(0.5, ny + 0.5, by = 1),
             color = "grey85", linewidth = 0.3) +

  # ---- Draw the tiles for presence/absence ----
  # Each tile represents one gene-sample pair.
  # White tile borders keep the plot readable even when colors are dense.
  geom_tile(color = "white", linewidth = 0.4) +

  # ---- Color scale ----
  # We only have "presence" (1), so we use a simple gradient to highlight detected genes.
  scale_fill_gradient(low = "white", high = "steelblue", guide = "none") +

  # ---- Labels ----
  labs(
    x = "AMR Gene (Class)",
    y = "Sample ID",
    title = "Presence of AMR Genes Across Samples"
  ) +

  # ---- Formatting for clean layout ----
  # Remove padding around the tiles so they fill the plot area.
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0)) +

  # Use equal aspect ratio so tiles are square
  coord_equal() +

  # ---- Theme adjustments ----
  theme_minimal(base_size = 11) +
  theme(
    # Rotate x-axis labels for readability (especially with long gene names)
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),

    # Remove default panel grid (since we've drawn our own aligned grid)
    panel.grid = element_blank()
  )
```


```
library(tidyverse)
library(forcats)

# ---- 1) Filter + prepare data ----
amr_heatmap <- amr %>%
  filter(element_type == "AMR", gene_symbol != "No hit found") %>%
  mutate(
    present  = 1,
    gene_lab = paste0(gene_symbol, " (", class, ")")
  ) %>%
  distinct(sample_id, gene_lab, class, .keep_all = TRUE) %>%
  # make axes explicitly discrete factors (so we can count levels for gridlines)
  mutate(
    sample_id = fct_inorder(sample_id),
    gene_lab  = fct_inorder(gene_lab)
  )

# Precompute gridline positions: at every cell boundary
x_grid <- seq(0.5, nlevels(amr_heatmap$gene_lab)  + 0.5, by = 1)
y_grid <- seq(0.5, nlevels(amr_heatmap$sample_id) + 0.5, by = 1)

# ---- 2) Plot with perfectly aligned grid ----
ggplot(amr_heatmap, aes(x = gene_lab, y = sample_id, fill = class)) +
  # draw the grid FIRST so it sits behind the tiles
  geom_vline(xintercept = x_grid, color = "grey85", linewidth = 0.3) +
  geom_hline(yintercept = y_grid, color = "grey85", linewidth = 0.3) +
```

```
# draw the tiles (white borders for crisp separation)
geom_tile(color = "white", linewidth = 0.3) +

# remove expansion so tiles touch the panel edge (and the grid matches)
scale_x_discrete(expand = c(0, 0)) +
scale_y_discrete(expand = c(0, 0)) +

labs(
  title    = "Presence of AMR Genes by Class",
  subtitle = "Each tile shows a detected AMR gene; gridlines mark cell boundaries",
  x = "AMR Gene (with Class)",
  y = "Sample",
  fill = "AMR Class"
) +
theme_minimal(base_size = 12) +
theme(
  axis.text.x  = element_text(angle = 90, vjust = 0.5, hjust = 1),
  panel.grid   = element_blank(),   # we draw our own grid
  legend.position = "right"
) +
# optional class palette (edit/remove to taste)
scale_fill_manual(values = c(
  "BETA-LACTAM"     = "#1f78b4",
  "AMINOGLYCOSIDE"  = "#33a02c",
  "TETRACYCLINE"    = "#e31a1c",
  "SULFONAMIDE"     = "#ff7f00",
  "QUINOLONE"       = "#6a3d9a",
  "TRIMETHOPRIM"    = "#b15928",
  "COLISTIN"        = "#a6cee3",
  "PHENICOL"        = "#cab2d6",
  "FOSFOMYCIN"      = "#fb9a99"
))
```


Presence of AMR Genes by Class
Each tile shows a detected AMR gene; gridlines mark cell boundaries
Sample
Control_03
AMR Gene (with Class)
BETA−LACTAM
COLISTIN
FOSFOMYCIN
PHENICOL
QUINOLONE
SULFONAMIDE
TETRACYCLINE
TRIMETHOPRIM