Lunch and Learn and Quarto

Paul Villanueva

8/13/2021

Contents

1. Lunch and Learn and Quarto

This is an example Quarto project made for RStudio's Lunch and Learn on 8/3/2021. To learn more about Quarto visit https://quarto.org.

Part I.

Section title

2. First Quarto

This is an example Quarto document. Note the `qmd` extension - this tells Quarto that this is a Markdown files that contains computations.

Since Quarto based on Markdown, we can **bold** and *italicize* text. We can also make headers.

3. Let's make a table

Meal	Food
Breakfast	Coffee
Lunch	Leftovers
Dinner	Spam Musubi

4. Example images

Figure 4.1.: My support system

Figure 4.2.: Lunch: Leftovers

Figure 4.3.: Dinner: Spam musubi

5. Code and chunk options

Quarto is based on **R**Markdown, so you can do all the R stuff you're used to as well.

```
library(tidyverse)

standard_curves <- readxl::read_xlsx('data/std_curve.xlsx', sheet = "everything") %>%
  janitor::clean_names() %>%
  filter(amoa < 40)

lm_eqn = function(df){
    m = lm(log_qty ~ ct, df);
    data.frame(
      a = format(as.numeric(coef(m)[1]), digits = 2),
      b = format(as.numeric(coef(m)[2]), digits = 2),
      r2 = format(summary(m)$r.squared, digits = 3)
    )
}

st_splits <-  standard_curves %>%
  group_by(amoa, run) %>%
  group_split()

eqs <- st_splits %>%
  lapply(., lm_eqn) %>%
  bind_rows()

labels <- lapply(st_splits, slice_head, n = 1) %>%
  bind_rows() %>%
  select(amoa, run) %>%
  bind_cols(eqs) %>%
  mutate(amoa = paste0("amoA_AOB_p", amoa)) %>%
  mutate(eq_label = paste0("y = ", a, " - ", abs(as.numeric(b)), "x<br>r^2 = ", r2))


standard_curves %>%
  mutate(amoa = paste0("amoA_AOB_p", amoa)) %>%
```

```
  ggplot(aes(log_qty, ct)) +
  geom_point() +
  facet_grid(run ~ amoa, scales = "free") +
  theme(
    panel.border = element_rect(color = "black", size = 1, fill = NA),
    panel.grid.minor.x = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.major.x = element_line(color = "gray", size = 0.5, linetype = "dashed"),
    panel.grid.major.y = element_line(color = "gray", size = 0.5, linetype = "dashed"),
    panel.spacing = unit(0.5, "lines"),
    panel.background = element_blank(),
    strip.background = element_rect(color = "black", size = 1, fill = NA),
  ) +
  labs(
    x = "Log(gene copies per reaction)",
    y = "Ct"
  ) +
  scale_x_continuous(limits = c(0, 7), breaks = seq(0, 7, 1), expand = c(0, 0)) +
  scale_y_continuous(limits = c(0, 25)) +
  geom_smooth(aes(group=1), method="lm", se=FALSE) +
  ggtext::geom_richtext(data = labels, aes(x = 3, y = 5, label = eq_label),
                        size = 4, fontface = "bold", inherit.aes = FALSE)
```

We can also throw some Python in here:

```
xs = [x for x in range(10)]

print(*(f'{x} squared is {x ^ 2}.' for x in xs), sep='\n')
```

```
0 squared is 2.
1 squared is 3.
2 squared is 0.
3 squared is 1.
4 squared is 6.
5 squared is 7.
6 squared is 4.
7 squared is 5.
8 squared is 10.
9 squared is 11.
```

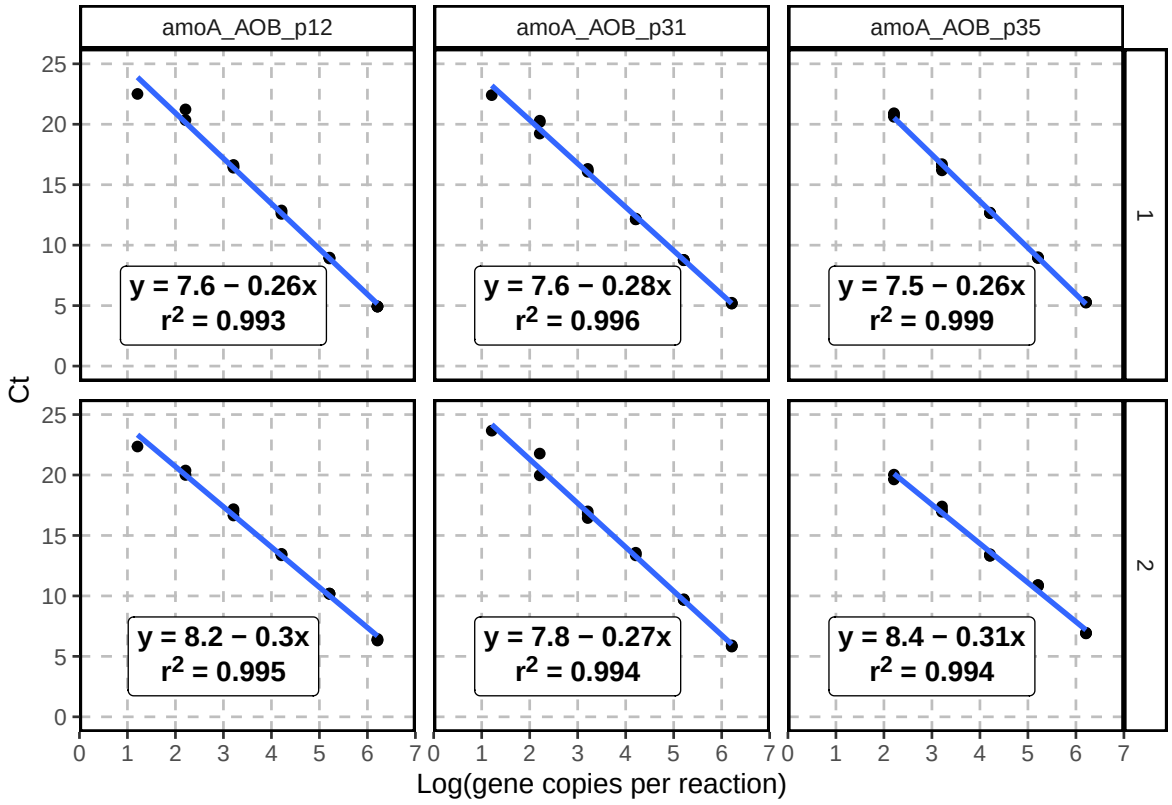


Figure 5.1.: Standard curves for the LAMPS crop priming experiment

```
import matplotlib.pyplot as plt
import numpy as np

Z = np.random.rand(6, 10)
x = [x + 0.5 for x in xs]
y = np.arange(4.5, 11, 1)

fig, ax = plt.subplots();
ax.pcolormesh(x, y, Z)
```

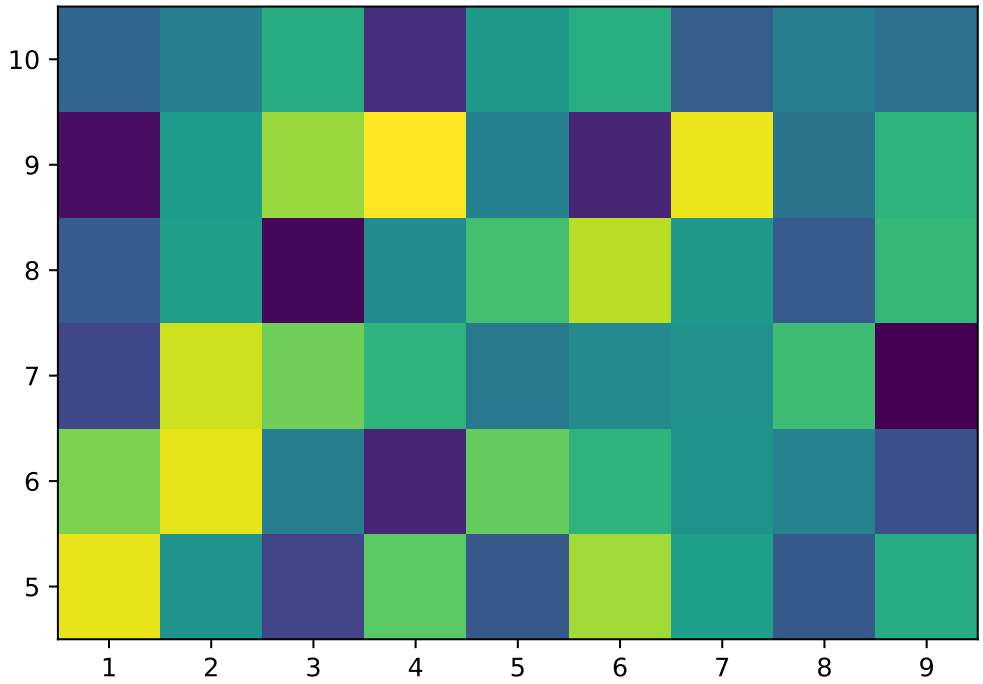


Figure 5.2.: That's a heatmap, baby!

5.1. Math stuff

We can also write math stuff! For example, here is a definition:

Definition 5.1 (Continuity)**.** The function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *continuous at a point* $x \in \mathbb{R}^n$ if for all $\varepsilon > 0$ there exists $\delta > 0$ such that if $|x - x_0| < \delta$, then $|f(x) - f(x_0)| < \varepsilon$. If this is true for all such x in the domain of f, we say that f is a *continuous function.*

5.1.0.1. Example

Define $f : \mathbb{R} \to \mathbb{R}$ by:

$$f(x) = \begin{cases} 1, x \in \mathbb{Q}, \\ 0, x \notin \mathbb{Q} \end{cases}$$

Prove that f is not a continuous function.

Proof. Let $\varepsilon = \frac{1}{2}$ and choose any $x \in \mathbb{Q}$. For any $\delta > 0$, we can find some $c \notin \mathbb{Q}$ such that $|x - c| < \delta$ since the irrationals are dense in $\mathbb{R}$. But then$|f(x) - f(c)| = |1 - 0| = 1 > \frac{1}{2}$, showing that f is not continuous at x. ■

5.2. Adding references

We can also add references. For instance, the following definition of k-partially colored comes from this paper: (Blair et al. 2020)

Definition 5.2 (k-partially colored)**.** Let D be a diagram of a link L with n crossings. We call D *k-partially colored* if we have specified a subset A of the strands of D and a function $f :\to \{1, 2, \ldots, k\}$. We refer to this partial coloring by the tuple (A, f). Given k-partial colorings (A_1, f_1) and (A_2, f_2) of D, we say (A_2, f_2) is the result of a coloring move on (A_1, f_1) if

1. $A_1 \subset A_2$ and A_2 $A_1 = \{s_j\}$ for some strand s_j in D;
2. $f_2|_{A_1} = f_1$;
3. s_j is adjacent to s_i at some crossing $c \in v(D)$, and $s_i \in A_1$;
4. the over-strand s_k at c is an element of A_1;
5. $f_1(s_i) = f_2(s_j)$.

5.3. Cross-references

Along the way, we've been giving each of the items above labels. The Visual Editor knows about these labels and we can call them up for cross referencing. For example:

- We were pretty happy about the standard curves in fig. 5.1
- I love me some heatmaps like fig. 5.2
- def. 5.2 is trivially true for the unknot.

6. Bibliography

7. Second: A Pure Python qmd

This is a pure Python qmd document. Since there are no R code chunks, it is executed via the Jupyter kernel.

7.1. Adding days per month from date range to a dataframe

Suppose you have a dataset with a column of start dates and column of end dates. For example:

```
import pandas as pd
import calendar

date_df = pd.DataFrame({
    "START_TM": ['2/15/2010', '2/15/2010', '3/16/2010'],
    "END_TM": ['4/18/2010', '2/18/2010', '5/20/2010']
})
date_df["START_TM"] = date_df["START_TM"].astype('datetime64')
date_df["END_TM"] = date_df["END_TM"].astype('datetime64')
date_df
```

	START_TM	END_TM
0	2010-02-15	2010-04-18
1	2010-02-15	2010-02-18
2	2010-03-16	2010-05-20

Our goal is to count the number of days in each month this range of dates falls over.

We start by adding columns for each month:

```
months = {calendar.month_name[i]:[0 for _ in range(date_df.shape[0])] for i in range(1, 13)}
for m in months:
    date_df[m] = [0 for _ in range(date_df.shape[0])]
date_df
```

	START_TM	END_TM	January	February	March	April	May	June	July	August	Septemb
0	2010-02-15	2010-04-18	0	0	0	0	0	0	0	0	
1	2010-02-15	2010-02-18	0	0	0	0	0	0	0	0	
2	2010-03-16	2010-05-20	0	0	0	0	0	0	0	0	

7.1.1. Helper functions

```
def insert_days_per_month(outer_row):
    dpm = days_per_month(outer_row)
    for index, inner_row in dpm.iterrows():
        outer_row[inner_row['Month']] = inner_row['NumDays']
    return(outer_row)

def days_per_month(row):
    s = pd.Series(index = pd.date_range(row[0], row[1]))[1: ]
    days_in_month = s.resample('MS').size().to_period('m').\
    rename_axis('Month').reset_index(name = 'NumDays')
    days_in_month['Month'] = days_in_month['Month'].apply(
        lambda x: calendar.month_name[x.month])
    return(days_in_month)
```

We can get the desired result with apply:

```
date_df = date_df.apply(lambda x: insert_days_per_month(x), axis = 1)
date_df
```

```
DeprecationWarning:

The default dtype for empty Series will be 'object' instead of 'float64' in a future version
```

	START_TM	END_TM	January	February	March	April	May	June	July	August	Septemb
0	2010-02-15	2010-04-18	0	13	31	18	0	0	0	0	
1	2010-02-15	2010-02-18	0	3	0	0	0	0	0	0	
2	2010-03-16	2010-05-20	0	0	15	30	20	0	0	0	

7.2. Interactive Widgets

Quarto has support for interactive documents. Supported formats include:

- JavaScript: Observable JS
- R: Shiny
- Python: Jupyter Widgets are all supported, such as IPyLeaflet and Plotly

7.3. IPyLeaflet

```
from ipyleaflet import Map, Marker

good_eats = {
  "Cham Soot Gol": (33.772819, -117.9694484),
  "The Boiling Crab": (33.6996179, -117.8905689),
  "Tan Hoang Huong": (33.7446965, -117.9629173)
}

cham_soot_gol = Map(center=good_eats["Cham Soot Gol"], scroll_wheel_zoom=True)
for place in good_eats:
  cham_soot_gol.add_layer(Marker(location=good_eats[place], title=place))
cham_soot_gol
```

```
Map(center=[33.772819, -117.9694484], controls=(ZoomControl(options=['position', 'zoom_in_te
```

Can do everything you're used to with Python but with the awesome Visual Editor stuff:

7.4. Plotly

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",
                 color="species",
                 marginal_y="violin", marginal_x="box",
                 trendline="ols", template="simple_white")
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

8. Example Jupyter Notebook

The editing experience with Jupyter + Quarto is very similar to the RStudio editing experience.

I'm not lying!

When we make changes and save here, the preview will update. Here's some code:

```
for x in range(10):
    print(f'{x} squared is {x ^2}.')
```

```
0 squared is 2.
1 squared is 3.
2 squared is 0.
3 squared is 1.
4 squared is 6.
5 squared is 7.
6 squared is 4.
7 squared is 5.
8 squared is 10.
9 squared is 11.
```

Here's a figure.

```
import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])

ax.grid(True)
plt.show()
```

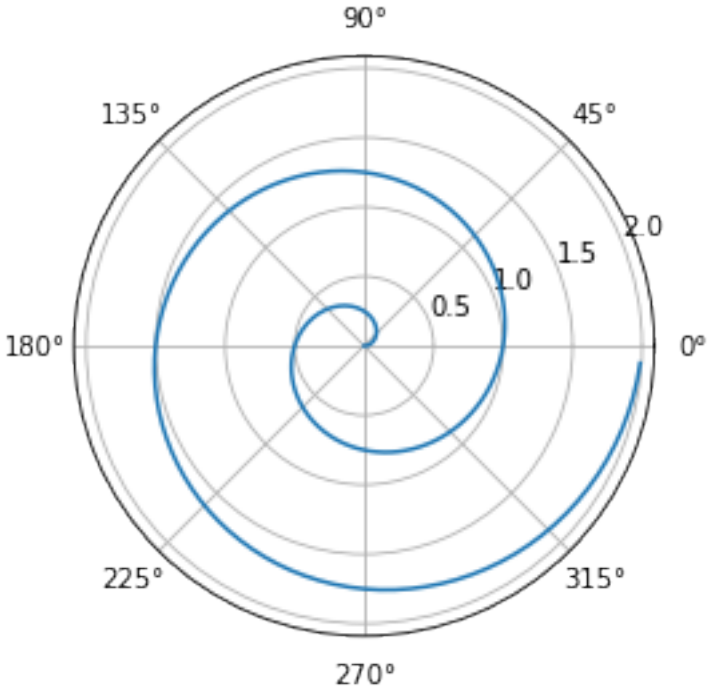


Figure 8.1.: **?(caption)**

And we can add chunk options just like we did in RStudio.

See fig. 8.1 for an example of a projection of a straight line into polar coordinates.

9. DNA String Stuff

Here are some functions to do some basic DNA string calculations.

```
import pandas as pd
def reverse_complement(nuc_sequence: str) -> str:
    """
    Returns the reverse complement of a nucleotide sequence.
    >>> reverse_complement('ACGT')
    'ACGT'
    >>> reverse_complement('ATCGTGCTGCTGTCGTCAAGAC')
    'GTCTTGACGACAGCAGCACGAT'
    >>> reverse_complement('TGCTAGCATCGAGTCGATCGATATATTTAGCATCAGCATT')
    'AATGCTGATGCTAAATATATCGATCGACTCGATGCTAGCA'
     """
    complements = {
        "A": "T",
        "C": "G",
        "G": "C",
        "T": "A"
    }
    rev_seq = "".join([complements[s] for s in nuc_sequence.upper()[::-1]])
    return rev_seq

def gc_content(nuc_sequence: str) -> float:
    """
    Calculates the GC content of a nucleotide sequence.
    >>> gc_content('ACGT')
    0.5
    """
    gc_tally = 0
    for nuc in nuc_sequence.lower():
        if nuc == 'g' or nuc == 'c':
            gc_tally += 1
    return gc_tally / len(nuc_sequence)

def random_dna_string(seq_length: int = 10) -> str:
```

```
    """
    Generates a random DNA string seq_length bp long
    >>> len(random_dna_string())
    10
    >>> len(random_dna_string(20))
    20
    """
    from random import choice

    dna_string = ""
    for _ in range(seq_length):
        dna_string += choice("ACGT")
    return dna_string

def make_strings_df(num_strings: int = 10, str_length: int = 10) -> pd.DataFrame:
    """
    Generates a pandas dataframe with num_strings DNA sequences of length str_length with
    columns "Sequence", "GC Content", "Reverse Complement"
    >>> df = make_strings_df(100, 37)
    >>> df.shape
    (100, 3)
    >>> len(df['Sequence'][0])
    37
    """
    dna_strings_list = [random_dna_string(str_length) for _ in range(num_strings)]
    strings_df = pd.DataFrame({
        "Sequence": dna_strings_list
    })
    strings_df['GC Content'] = strings_df['Sequence'].apply(gc_content)
    strings_df['Reverse Complement'] = strings_df['Sequence'].apply(reverse_complement)
    return strings_df

import doctest
doctest.testmod(verbose=0)
```

```
TestResults(failed=0, attempted=9)
```

But that's a lot of function definitions and code testing that a lot of people probably don't care about. Let's set `fold` and `summary` to hide this chunk.

Let's use the function and create a histogram of the GC contents for the simulated sequences.

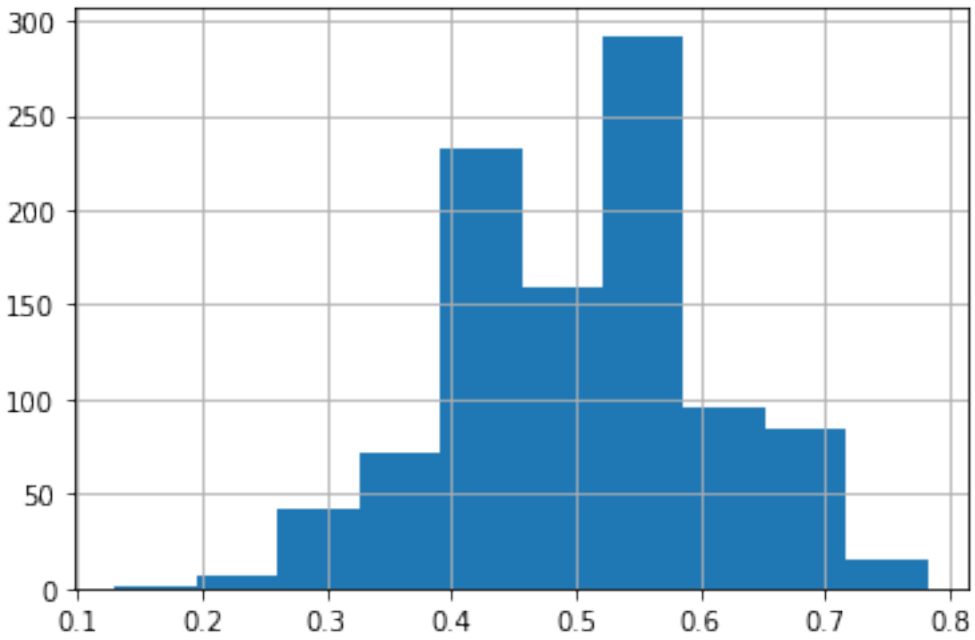
```
strings_df = make_strings_df(1000, 23)
print(f'strings_df has {strings_df.shape[0]} rows and {strings_df.shape[1]} columns.')
```

```
strings_df has 1000 rows and 3 columns.
```

```
strings_df.head(10)
```

	Sequence	GC Content	Reverse Complement
0	TAATAATGGGCTAAACTATGTTT	0.260870	AAACATAGTTTAGCCCATTATTA
1	GACCGTGACCCAAGGCAGATGGG	0.652174	CCCATCTGCCTTGGGTCACGGTC
2	TAGGGTTGTGCTTTACCTTACAT	0.391304	ATGTAAGGTAAAGCACAACCCTA
3	GCAAGGCCGGATACGCGTATAAT	0.521739	ATTATACGCGTATCCGGCCTTGC
4	ACCACTCCTCAAACGTTACTGAT	0.434783	ATCAGTAACGTTTGAGGAGTGGT
5	CCTCGTCAGTTGTCACTTCTATG	0.478261	CATAGAAGTGACAACTGACGAGG
6	ACAATGATCGCAGCCGAGGTATA	0.478261	TATACCTCGGCTGCGATCATTGT
7	GTTGGATATTCCGCAGCAGAGGA	0.521739	TCCTCTGCTGCGGAATATCCAAC
8	CGCTTAAAATCCCTGCATAGACC	0.478261	GGTCTATGCAGGGATTTTAAGCG
9	AGACCACTACTGGGTGGAGACGG	0.608696	CCGTCTCCACCCAGTAGTGGTCT

```
strings_df['GC Content'].hist();
```



10. Sample Analysis

```
library(vegan)
library(ggplot2)
library(here)
library(tidyverse)
library(microViz)
library(phyloseq)
library(ggtext)

theme_set(theme_minimal())
```

10.0.1. Data

Reading in data:

```
data.priming <- read.csv(here("data", "priming_amoA_deltaCt.csv"), header = T) %>%
  rename(sample_id = X)

data.raw <- read.csv(here("data", "priming_amoA_rawCt.csv"), header = T) %>%
  rename(sample_id = X)

data.priming.long <- data.priming %>%
  pivot_longer(cols = amoA.001:amoA.078, names_to = "amoA", values_to = "deltaCT")

data.raw.long <- data.raw %>%
  pivot_longer(cols = amoA.001:amoA.078, names_to = "amoA", values_to = "CT")

data.priming.long$sample_id <- fct_reorder(data.priming.long$sample_id, parse_number(data.pri

df <- data.priming[, -1]
rownames(df) <- data.priming[, 1]

metadata <- df %>%
  select(fert_level:field_rep) %>%
```

```
  mutate(across(everything(), as.factor))


amoa_counts <- df %>%
  select(starts_with("amoA"))
```

`data.priming` contains the data for our experiment. There are rows for samples, columns for the delta CTs of the different amoAs, and some metadata.

```
data.priming[1:5, 1:5]
```

```
  sample_id  amoA.001 amoA.002  amoA.003  amoA.004
1        2b 10.119249 27.41268  8.764504  8.992937
2       35b  9.837943 27.51089  9.300077 10.445448
3       52f 26.345485 26.34548 26.345485 26.345485
4       34f 26.914432 26.91443 26.914432 26.914432
5       16f  8.337293 25.94591  8.371314 25.945907
```

`data.raw` contains the same columns but lists the raw CT values instead of the 16s-normalized ones.

```
data.raw[1:5, 1:5]
```

```
  sample_id amoA.001 amoA.002 amoA.003 amoA.004
1        2b 22.70657       40 21.35182 21.58026
2       35b 22.32706       40 21.78919 22.93456
3       52f 40.00000       40 40.00000 40.00000
4       34f 40.00000       40 40.00000 40.00000
5       16f 22.39139       40 22.42541 40.00000
```

THe `long` versions of these dataframes contains the same info but in long format to play nicely with `ggplot`.

10.1. Removing amoAs

We'll start by removing those amoAs from our data that are not present in over 30 samples across both treatments.

We'll first start by counting the non-detects for each amoA.

```
non_detect_counts <- data.raw.long %>%
  group_by(fert_level, amoA) %>%
  count(CT == 40) %>%
  rename(non_detect = `CT == 40`) %>%
  filter(non_detect == TRUE)
```

Finding the amoAs that are not detected in > 30 across both samples

```
removes <- non_detect_counts %>%
  pivot_wider(names_from = fert_level, values_from = n, names_prefix = "fert.") %>%
  filter(fert.0 > 30 & fert.336 > 30) %>%
  pivot_longer(cols = fert.0:fert.336, names_to = "fert_level", values_to = "n")
```

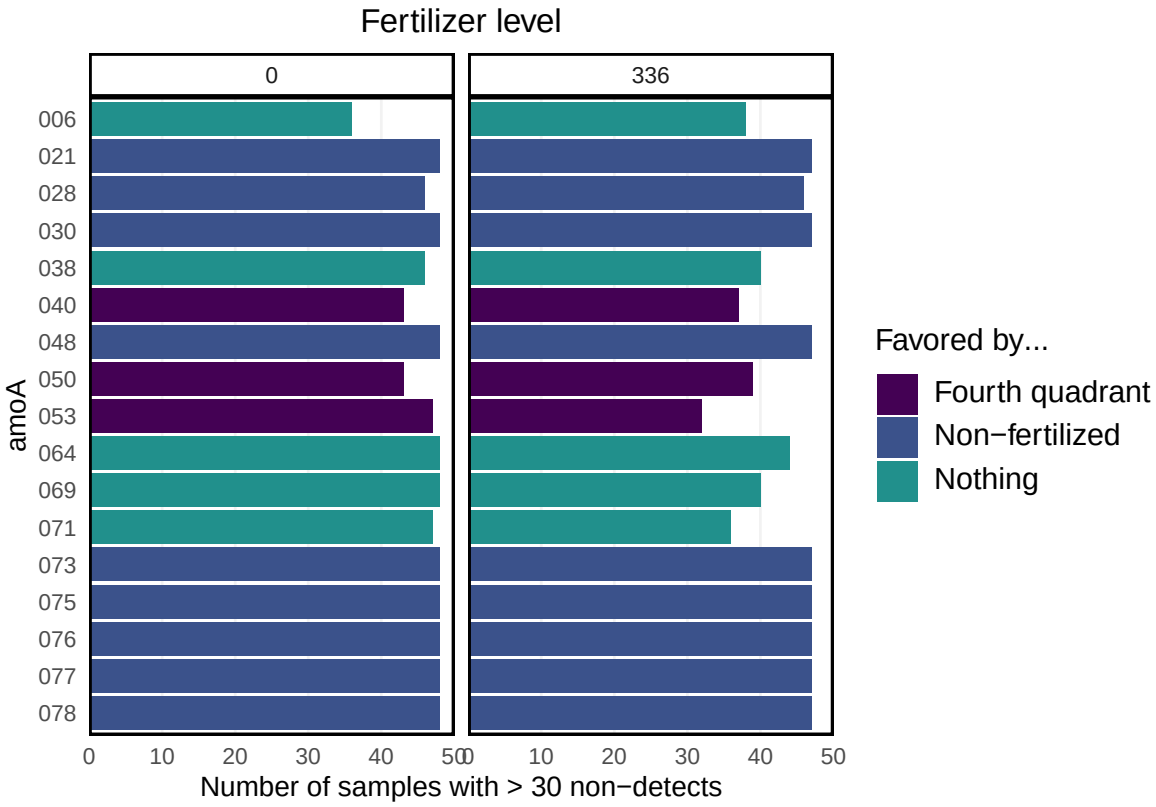
We'll now reduce `data.priming` by removing those amoAs that are largely non-detects. We'll also update the `long` version while we're at it

```
data.priming.reduced <- data.priming %>%
   select(-one_of(removes$amoA))

data.priming.reduced.long <- data.priming.reduced %>%
  select(-sample_id, field_rep) %>%
  pivot_longer(cols = contains("amoa"))
```

Here's a barchart of what we're removing:

```
removes %>%
  mutate(amoA = str_sub(amoA, -3)) %>%
  mutate(favored = case_when(
    amoA %in% c("006", "038", "064", "069", "071") ~ "Nothing",
    amoA %in% c("021", "028", "030", "048", "073", "075", "076", "077", "078") ~ "Non-fertil
    amoA %in% c("040", "050", "053") ~ "Fourth quadrant",
    TRUE ~ "First quadrant"
  )) %>%
  mutate(fert_level = str_sub(fert_level, start = 6)) %>%
  ggplot(aes(amoA, n, fill = favored )) +
  geom_col() +
  facet_wrap(~ fert_level) +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.text = element_markdown(size = 12),
    legend.title = element_markdown(size = 12, hjust = 0),
```

```
    strip.background = element_rect(size = 1, color = "black", fill = "NA"),
    panel.grid = element_line(color = "gray95"),
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.border = element_rect(color = "black", size = 1, fill = NA)
  ) +
  scale_fill_viridis_d(begin = 0, end = 0.5) +
  scale_y_continuous(limits = c(0, 50), expand = expansion(add = c(0, 0))) +
  scale_x_discrete(limits = rev) +
  coord_flip() +
  labs(
    y = "Number of samples with > 30 non-detects",
    title = "Fertilizer level",
    fill = "Favored by..."
  )
```



Note that most of the non-detects that we're removing are from the non-fertilized group.

Next, we'll convert the CT values to presence/absence for use in later analysis.

```
amoA_presence_absence <- data.raw %>%
  select(sample_id, starts_with("amoA")) %>%
  mutate(across(starts_with("amoA"), ~ ifelse(.x == 40, 0, 1)))
```

10.2. Ordination

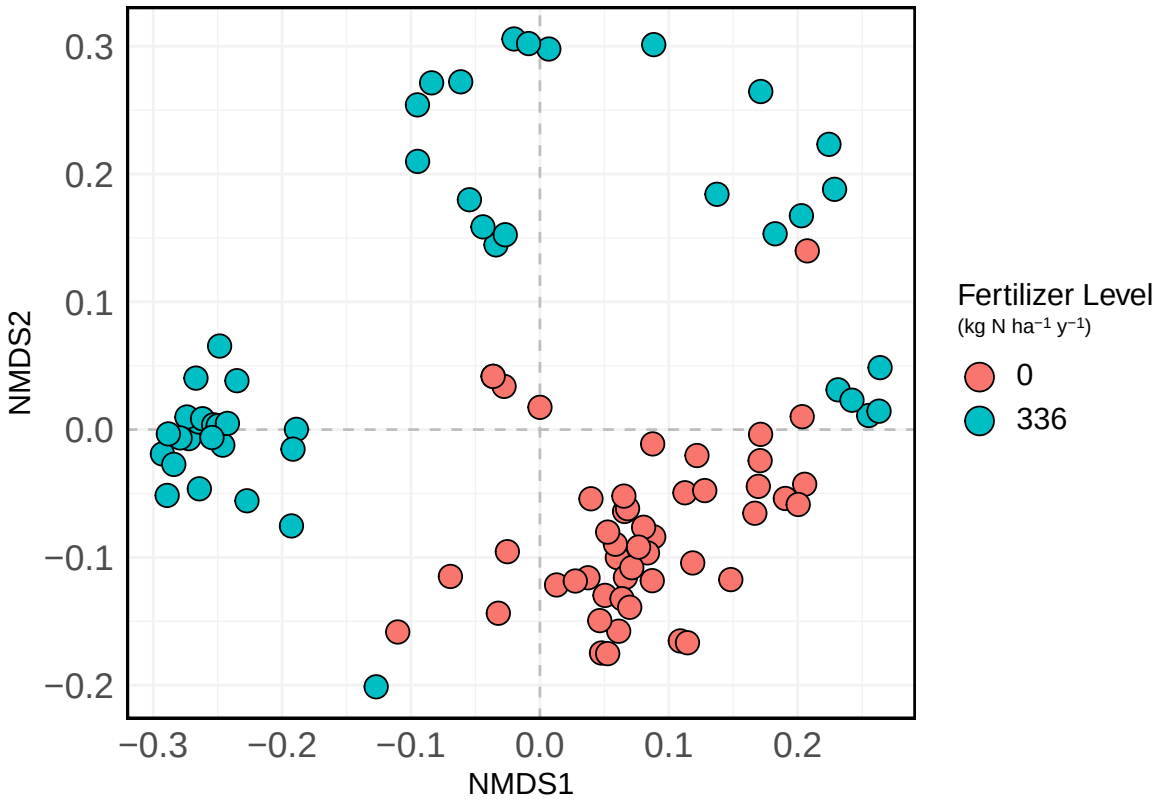
Calculating the NMDS (positioning the sites):

```
mds.priming = metaMDS(data.priming.reduced %>% select(contains("amoa")), distance = "bray", k

site.scores <- as.data.frame(scores(mds.priming, display = "sites")) %>%
  mutate(sample_id = data.priming.reduced$sample_id,
         Crop = data.priming.reduced$crop,
         Fert_Level = as.factor(data.priming.reduced$fert_level),
         Day = as.factor(data.priming.reduced$doe),
         Substrate_Addition = as.factor(data.priming.reduced$addition))
```

This is enough to plot a basic NMDS:

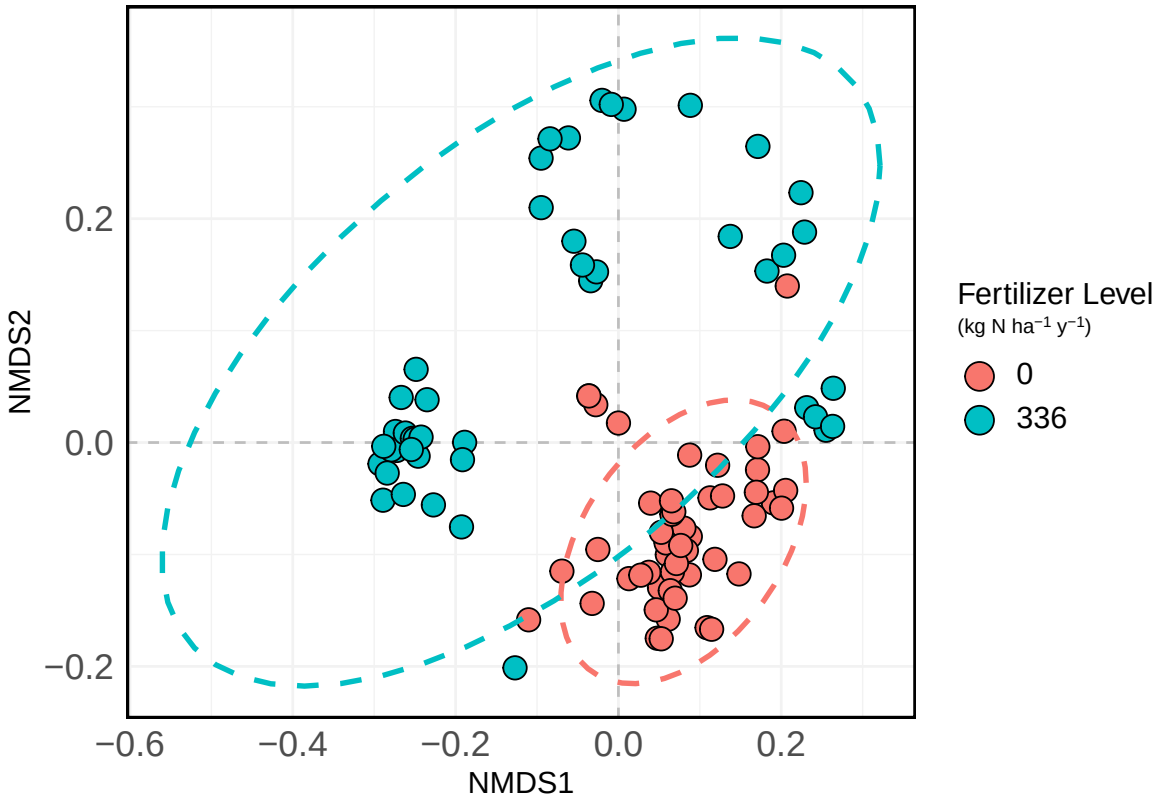
```
nmds.plot <- site.scores %>%
  ggplot(aes(NMDS1, NMDS2, fill = Fert_Level)) +
  geom_hline(yintercept = 0.0,
             colour = "grey",
             lty = 2) +
  geom_vline(xintercept = 0.0,
             colour = "grey",
             lty = 2) +
  geom_point(size = 4, shape = 21) +
    theme(
    plot.title = element_text(hjust = 0.5),
    legend.text = element_markdown(size = 12),
    legend.title = element_markdown(size = 12, hjust = 0),
    axis.text.x = element_text(size = 14),
    axis.text.y = element_text(size = 14),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    panel.grid = element_line(color = "gray95"),
    panel.border = element_rect(color = "black", size = 1, fill = NA)
```

```
  ) +
  scale_fill_discrete(name = "Fertilizer Level<br>
                              <span style = 'font-size:8pt;'>
                              (kg N ha<sup>-1</sup> y<sup>-1</sup>)
                              </span>") +
  guides(
    fill = guide_legend(override.aes = list(shape = 21, size = 5))
  )

nmds.plot
```



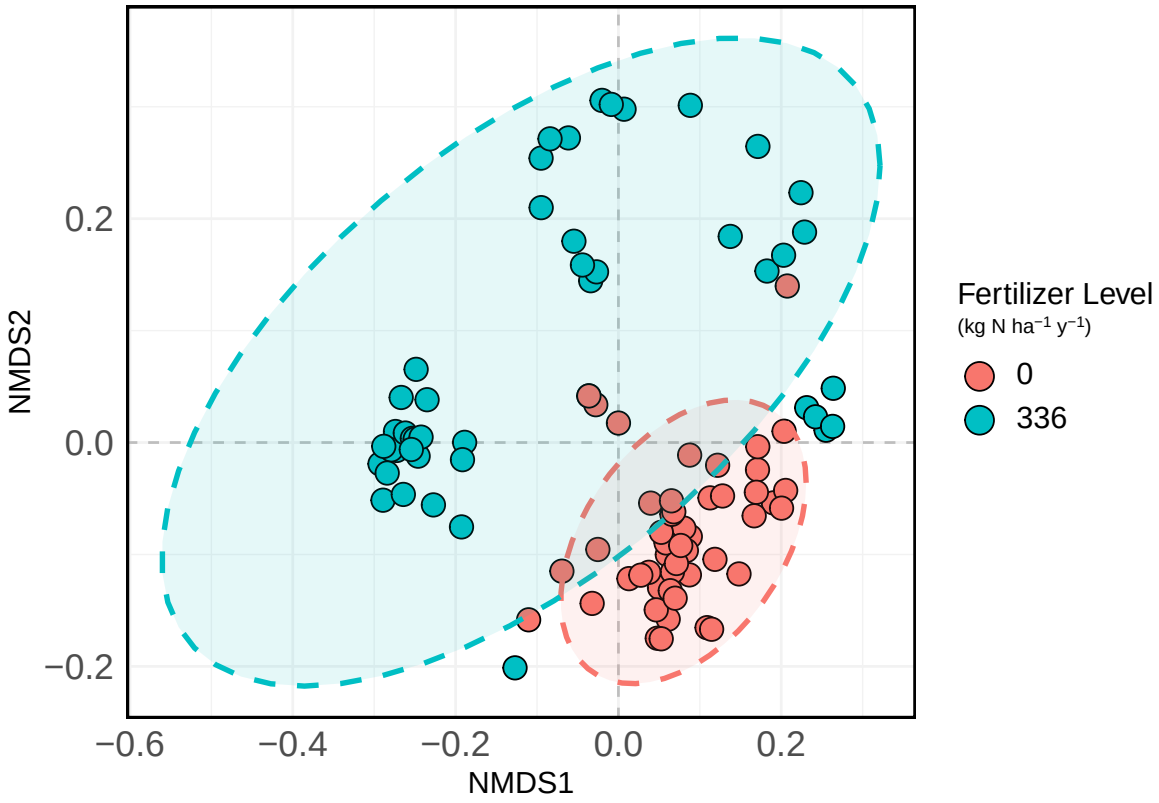
We can also add ellipses to the plot indicate confidence intervals if you're interested in that:

```
nmds.plot +
  stat_ellipse(aes(color = Fert_Level), size = 1, linetype = "dashed", show.legend = FALSE)
```



And again with shading:

```
nmds.plot +
  stat_ellipse(aes(color = Fert_Level), size = 1, linetype = "dashed", show.legend = FALSE) +
    stat_ellipse(aes(fill = Fert_Level), size = 1, linetype = "dashed", show.legend = FALSE,
```



10.3. Arrows!

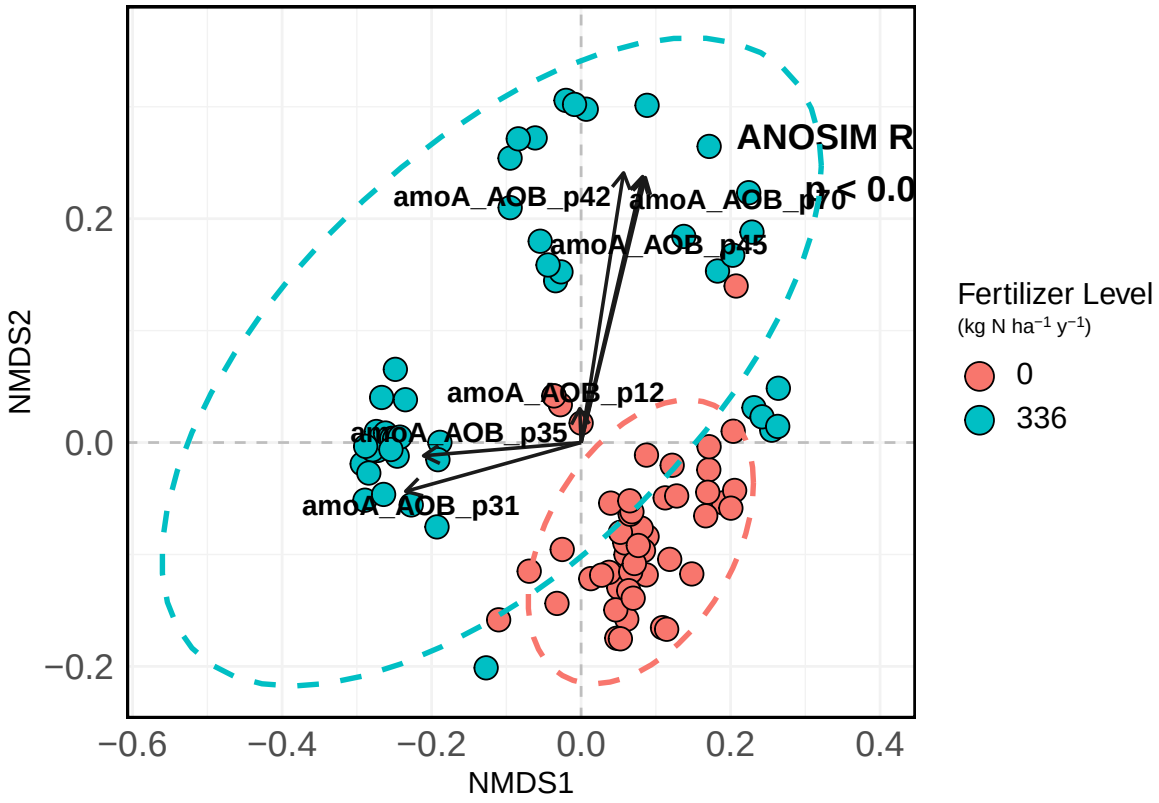
Let's calculate the loading factors of the individual amoas:

```
dune_dist <- vegdist(data.priming %>% select(starts_with('amoA')))

amoa_anosim <- anosim(dune_dist, data.priming$fert_level)

mds.spp.fit <- envfit(mds.priming, data.priming.reduced %>% select(contains("amoa")), permuta

spp.scrs <- as.data.frame(scores(mds.spp.fit, display = "vectors"))
spp.scrs <- cbind(spp.scrs, Species = rownames(spp.scrs))
spp.scrs <- cbind(spp.scrs, pval = mds.spp.fit$vectors$pvals)

spp.scores <- as.data.frame(scores(mds.spp.fit, display = "vectors")) %>%
  mutate(Species = rownames(.),
         pval = mds.spp.fit$vectors$pvals)
```

This is enough to plot arrows on the NMDS. We'll show the loadings of some amoAs of interest that we identified in a previous analysis.

```
special <- c("amoA.012", "amoA.031", "amoA.035", "amoA.042", "amoA.045", "amoA.070")

special_arrows <- spp.scores %>%
  rownames_to_column() %>%
  filter(rowname %in% special) %>%
  mutate(x = -0.25 * NMDS1,
         y = -0.25 * NMDS2,
         assay = str_sub(rowname, -2),
         assay = paste0("amoA_AOB_p", assay)
         )

nmds.plot +
    geom_segment(data = special_arrows,
                aes(x = 0, xend = -0.3 * NMDS1,
                    y = 0, yend = -0.3 * NMDS2),
                size = 0.66,
                arrow = arrow(length = unit(0.25, "cm")),
                              color = "grey10", lwd = 0.3,
                inherit.aes = FALSE) +
  ggrepel::geom_text_repel(
    data = special_arrows,
    aes(x * 1, y * 1, label = assay),
    fontface = "bold",
    size = 4,
    inherit.aes = FALSE,
    force = 1,
    nudge_x = -0.001
  ) +
  annotate(
    "text",
    label = paste0("ANOSIM R = ", round(amoa_anosim$statistic, 2),
                   "\np < 0.001"),
    x = 0.4,
    y = 0.25,
    size = 5,
    fontface = 2
  ) +
  stat_ellipse(aes(color = Fert_Level), size = 1, linetype = "dashed", show.legend = FALSE)
```

```
Warning: Duplicated aesthetics after name standardisation: size
```



10.4. Statistics

10.4.1. Which factors have an impact on overall community composition?

```
X <- data.priming.reduced %>%
  select(-c(contains("amoa")))
Y <- data.priming.reduced %>%
  select(c(contains("amoa")))

adonis(Y ~ X$fert_level + X$addition + X$crop + X$timepoint)
```

```
Call:
adonis(formula = Y ~ X$fert_level + X$addition + X$crop + X$timepoint)

Permutation: free
Number of permutations: 999
```

```
Terms added sequentially (first to last)

             Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
X$fert_level  1    1.4426 1.44255  56.092 0.37093  0.001 ***
X$addition    2    0.0327 0.01633   0.635 0.00840  0.667
X$crop        1    0.0882 0.08823   3.431 0.02269  0.022 *
X$timepoint   1    0.0367 0.03671   1.427 0.00944  0.190
Residuals    89    2.2889 0.02572         0.58855
Total        94    3.8890                 1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This tells us that fertilization level is very significant and explains ~37% of the variation in our samples. Crop is also a significant factor on community composition, though it only explains 2.3% of the variation.

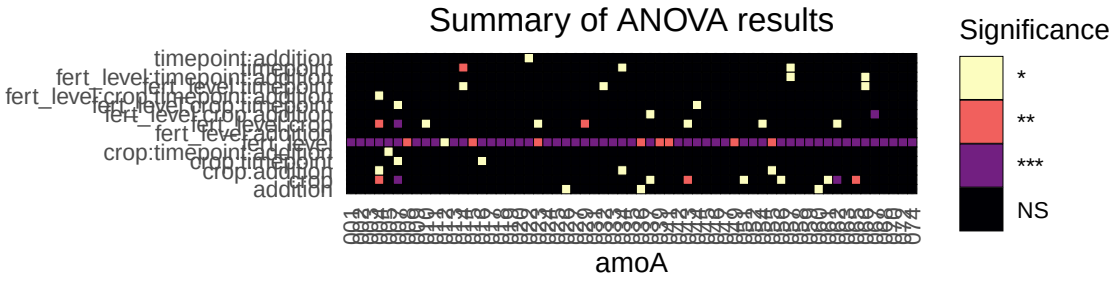
10.4.2. How do the treatment factors affect the "abundance" of genes on an individual level?

All the code below does is perform an ANOVA of the gene's abundance against all the terms and all of their interactions.

```
formulae <- lapply(colnames(data.priming.reduced %>% select(amoA.001:amoA.074)), function(x)

res <- lapply(formulae, function(x) broom::tidy(aov(x, data = data.priming.reduced)))
names(res) <- format(formulae)
names(res) <- str_sub(names(res), end = 8)

anova_results <- lapply(seq_along(res), function(i) res[[i]] %>% mutate(gene = names(res)[[i]
  bind_rows() %>%
  filter(term != "Residuals") %>%
  mutate(sig = case_when(
    p.value < 0.05 & p.value > 0.01 ~ "*",
    p.value < 0.01 & p.value > 0.001 ~ "**",
    p.value < 0.001 ~ "***",
    TRUE ~ "NS"
  ))
```

Visualization:

```r
anova_results %>%
  mutate(gene = str_sub(gene, -3)) %>%
  ggplot(aes(gene, term, fill = sig)) +
  geom_tile(color = "black") +
  coord_equal() +
  labs(y = "",
       x = "amoA",
       title = "Summary of ANOVA results",
       fill = "Significance ") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.text.x = element_text(angle = 90, hjust = 0, vjust = 0.5)
  ) +
  scale_fill_viridis_d(option = "magma", direction = -1)
```

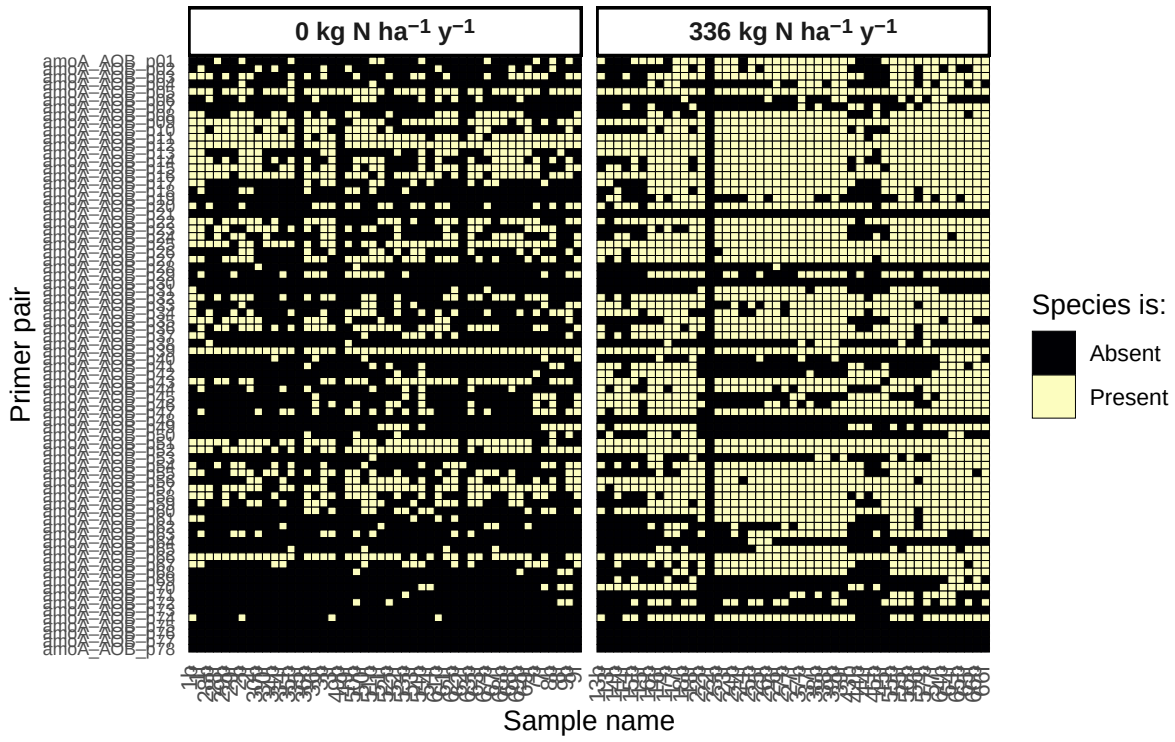


Overall, we see that, again, fertilization level has a significant impact on abundance levels of the individual genes, and it's not even really that close. There are other factors that might be worth investigating on a gene-by-gene basis, too, but that's for later.

10.5. Biodiversity

Let's start by visualizing the presence/absence table:

```
amoA_presence_absence %>%
  pivot_longer(cols = amoA.001:amoA.078, names_to = "amoA", values_to = "presence") %>%
  mutate(amoA = str_sub(amoA, -2),
         amoA = paste0("amoA_AOB_p", amoA),
         presence = as.factor(presence)) %>%
  left_join(metadata %>% rownames_to_column(var = "sample_id")) %>%
  mutate(strip_label = paste0(fert_level, " kg N ha<sup>-1</sup> y<sup>-1</sup>")) %>%
  ggplot(aes(sample_id, amoA, fill = presence)) +
  geom_tile(color = "black") +
  labs(
    x = "Sample name",
    y = "Primer pair",
    fill = "Species is:",
    title = "",
    subtitle = ""
  ) +
  scale_fill_viridis_d(labels = c("Absent", "Present"),
                       begin = 0, end = 1,
                       option = "magma") +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.25),
    axis.text.y = element_text(size = 7),
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    strip.text = element_markdown(size = 10, face = "bold"),
    strip.background = element_rect(size = 1, color = "blaCk", fill = NA),
    plot.margin = unit(c(0, 0.1, 0.1, 0.1), "cm")
  ) +
  scale_y_discrete(limits = rev) +
  facet_grid(~ strip_label, scales = "free")
```



10.6. Reading in the best BLAST hit info:

```
amoA_organism_info <- readxl::read_xlsx(here("data", "amoa_mfp_qpcr_org_accessions.xlsx"), sh
  select(-c(contains(c("forward", "reverse", "notes"))))
```

Counts of best BLAST hits:

```
amoA_organism_info %>%
  count(best_blast_hits, sort = TRUE)
```

```
# A tibble: 25 x 2
   best_blast_hits                                n
   <chr>                                      <int>
 1 Nitrosolobus multiformis AmoA (amoA) gene     10
 2 Nitrosospira sp. En13 AmoA                     9
```

```
 3 Nitrosospira multiformis ATCC 25196          7
 4 Nitrosospira sp. Wyke8 AmoA                  7
 5 Nitrosospira lacus strain APG3               6
 6 Nitrosospira sp. Np39-19                     6
 7 Nitrosospira sp. Wyke2                       4
 8 Nitrosospira sp. NpAV                        3
 9 Nitrosomonas sp. JL21                        2
10 Nitrosospira briensis                        2
# ... with 15 more rows
```

10.7. Creating a phyloseq object

```
amoa_tax_table <- amoA_organism_info %>%
  select(array_name, best_blast_hits) %>%
  column_to_rownames(var = "array_name") %>%
  tax_table()

rownames(amoa_tax_table) <- amoA_organism_info$array_name
```

```
ps <- phyloseq(
  otu_table(amoA_presence_absence %>% column_to_rownames(var = "sample_id"), taxa_are_rows =
  sample_data(metadata),
  amoa_tax_table
)
```

10.8. Richness analysis

How does observed richness and evenness change with treatment level? This is a modified diversity function that does a bunch of nice stuff that `phyloseq::estimate_richness` doesn't do.

```
estimate_richness_mod <- function(physeq, split=TRUE, measures=NULL){


  if( !split ){
    OTU <- taxa_sums(physeq)
  } else if( split ){
    OTU <- as(otu_table(physeq), "matrix")
```

```
  if( taxa_are_rows(physeq) ){ OTU <- t(OTU) }
}


renamevec = c("Observed", "Chao1", "ACE", "Shannon", "Pielou", "Simpson", "InvSimpson", "Si
names(renamevec) <- c("S.obs", "S.chao1", "S.ACE", "shannon", "pielou", "simpson", "invsimp

if( is.null(measures) ){
  measures = as.character(renamevec)
}

if( any(measures %in% names(renamevec)) ){
  measures[measures %in% names(renamevec)] <- renamevec[names(renamevec) %in% measures]
}


if( !any(measures %in% renamevec) ){
  stop("None of the `measures` you provided are supported. Try default `NULL` instead.")
}


outlist = vector("list")

estimRmeas = c("Chao1", "Observed", "ACE")
if( any(estimRmeas %in% measures) ){
  outlist <- c(outlist, list(t(data.frame(estimateR(OTU)))))
}
if( "Shannon" %in% measures ){
  outlist <- c(outlist, list(shannon = diversity(OTU, index="shannon")))
}
if( "Pielou" %in% measures){
  #print("Starting Pielou")
  outlist <- c(outlist, list(pielou = diversity(OTU, index = "shannon")/log(estimateR(OTU)
}
if( "Simpson" %in% measures ){
  outlist <- c(outlist, list(simpson = diversity(OTU, index="simpson")))
}
if( "InvSimpson" %in% measures ){
  outlist <- c(outlist, list(invsimpson = diversity(OTU, index="invsimpson")))
}
if( "SimpsonE" %in% measures ){
```

```
    outlist <- c(outlist, list(simpsone = diversity(OTU, index="invsimpson")/estimateR(OTU)['
  }
  if( "Fisher" %in% measures ){
    fisher = tryCatch(fisher.alpha(OTU, se=TRUE),
                      warning=function(w){
                        warning("phyloseq::estimate_richness: Warning in fisher.alpha(). See
                        suppressWarnings(fisher.alpha(OTU, se=TRUE)[, c("alpha", "se")])
                      }
    )
    if(!is.null(dim(fisher))){
      colnames(fisher)[1:2] <- c("Fisher", "se.fisher")
      outlist <- c(outlist, list(fisher))
    } else {
      outlist <- c(outlist, Fisher=list(fisher))
    }
  }
  out = do.call("cbind", outlist)

  namechange = intersect(colnames(out), names(renamevec))
  colnames(out)[colnames(out) %in% namechange] <- renamevec[namechange]

  colkeep = sapply(paste0("(se\\.){0,}", measures), grep, colnames(out), ignore.case=TRUE)
  out = out[, sort(unique(unlist(colkeep))), drop=FALSE]

  out <- as.data.frame(out)
  return(out)
}
```

```
metrics <- c("Observed", "Shannon")
richness <- estimate_richness_mod(ps, measures = metrics) %>%
  rownames_to_column(var = "sample_id") %>%
  mutate(sample_id = str_sub(sample_id, start = 2))

richness <- left_join(sample_data(ps) %>% data.frame() %>% rownames_to_column(var = "sample_
  pivot_longer(cols = Observed:Shannon, names_to = "Metric", values_to = "Value")
```

```
Joining, by = "sample_id"
```

10.9. Statistical tests

10.9.1. Significance test of fertilization level on richness.

```
(sig_rich_fert <- kruskal.test(Value ~ fert_level, data = richness %>% filter(Metric == "Obse
```

```
    Kruskal-Wallis rank sum test

data:  Value by fert_level
Kruskal-Wallis chi-squared = 54.212, df = 1, p-value = 1.8e-13
```

The `p-value < 0.001` gives us strong statistical evidence that richness is significantly different between fertilization treatment groups.

10.9.2. Significance test of fertilization level on richness

```
(sig_even_fert <- kruskal.test(Value ~ fert_level, data = richness %>% filter(Metric == "Shan
```

```
    Kruskal-Wallis rank sum test

data:  Value by fert_level
Kruskal-Wallis chi-squared = 54.268, df = 1, p-value = 1.75e-13
```

The `p-value < 0.001` gives us strong statistical evidence that Shannon diversity is significantly different between fertilization treatment groups.

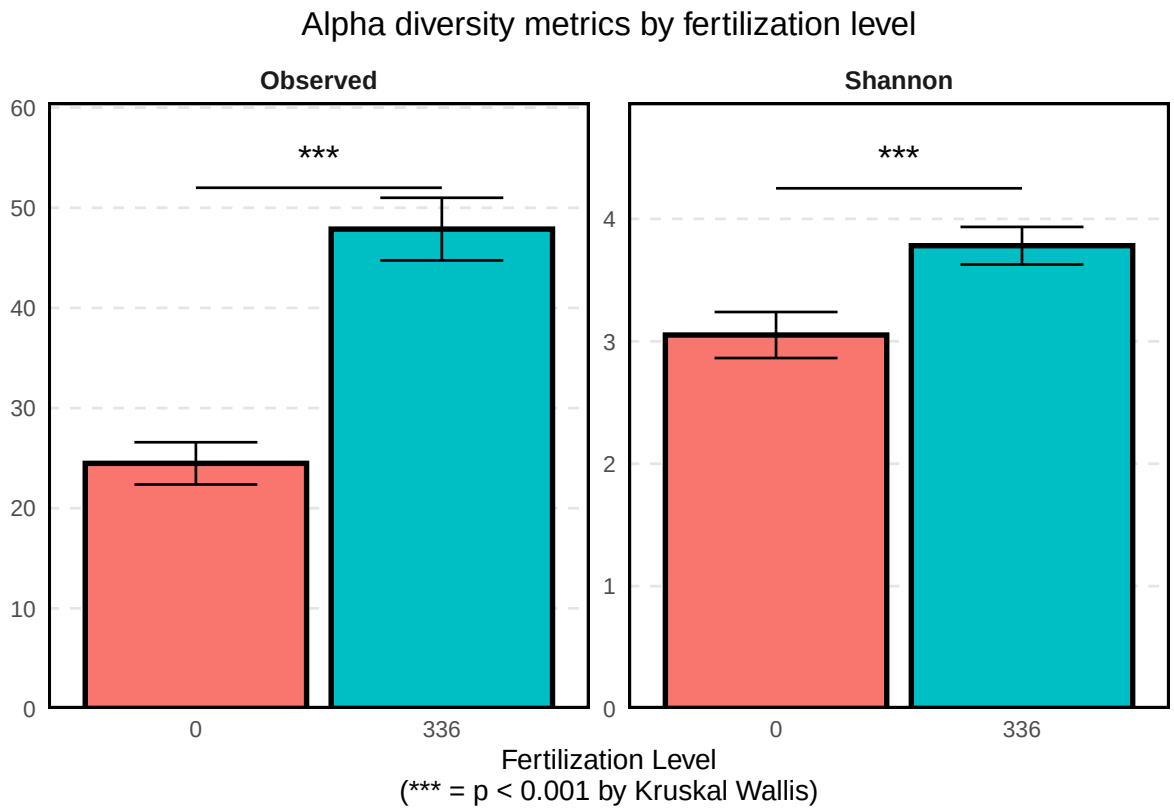
10.10. Making nice plots for stat differences

Standard deviations, mean

```
summaries <- richness %>%
  group_by(Metric, fert_level) %>%
  summarize(mean_val = mean(Value),
            sd_val = sd(Value) / 4,
            .groups = "drop")
```

```r
this_annotation <- data.frame(
  Metric = c("Observed", "Shannon"),
  lab = c("***", "***"),
  x = 1.5,
  y = c(50 + 5, 4 + 0.5),
  lineheights = c(50 + 2, 4 + 0.25)
)

summaries %>%
  ggplot(aes(fert_level, mean_val, fill = fert_level)) +
  geom_col(color = "black", size = 1) +
  facet_wrap(~ Metric, scales = "free_y") +
  theme(
    legend.position = "none",
    strip.background = element_blank(),
    axis.title.y = element_blank(),
    strip.placement = "outside",
    plot.title = element_text(hjust = 0.5),
    strip.text.y = element_text(face = "bold", size = 10),
    strip.text = element_text(face = "bold", size = 10),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.major.y = element_line(color = "gray90", linetype = "dashed"),
    axis.ticks = element_blank(),
    panel.border = element_rect(color = "black", size = 1, fill = "NA")
  ) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1))) +
  geom_errorbar(aes(ymin = mean_val - sd_val, ymax = mean_val + sd_val, width = 0.5)) +
  geom_text(
    data = this_annotation,
    aes(x = x, y = y, label = lab),
    inherit.aes = FALSE,
    size = 5
    ) +
  geom_segment(data = this_annotation,
               aes(x = 1,
                   xend = 2,
                   y = lineheights,
                   yend = lineheights),
               inherit.aes = FALSE) +
  labs(
```

```
    x = "Fertilization Level\n(*** = p < 0.001 by Kruskal Wallis)",
    title = "Alpha diversity metrics by fertilization level"

  )
```



10.11. Beta diversity

We'll start beta diversity analysis off by doing an ADONIS/PERMANOVA to determine if treatment centroids/treatment variations are different between groups.

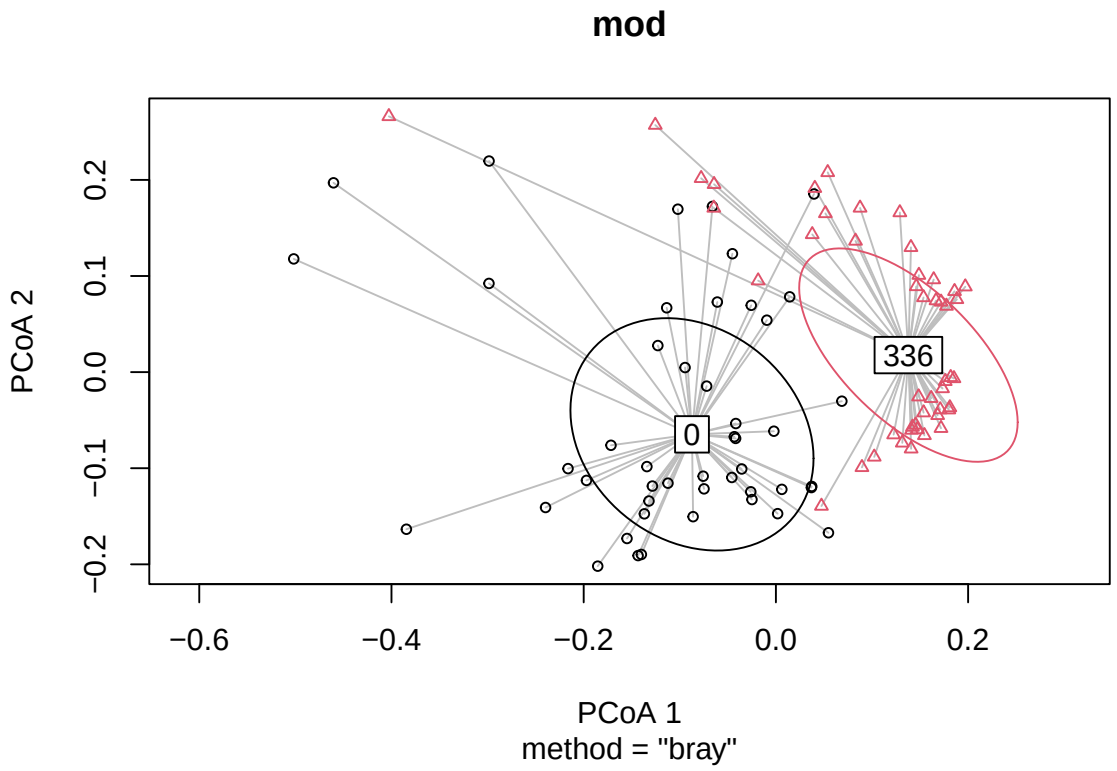
```
dis <- vegdist(otu_table(ps))
groups <- sample_data(ps)$fert_level
mod <- betadisper(dis, groups)
anova(mod)
```

```
Analysis of Variance Table
```

```
Response: Distances
          Df  Sum Sq Mean Sq F value    Pr(>F)
Groups     1 0.52595 0.52595  24.589 3.228e-06 ***
Residuals 92 1.96781 0.02139
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

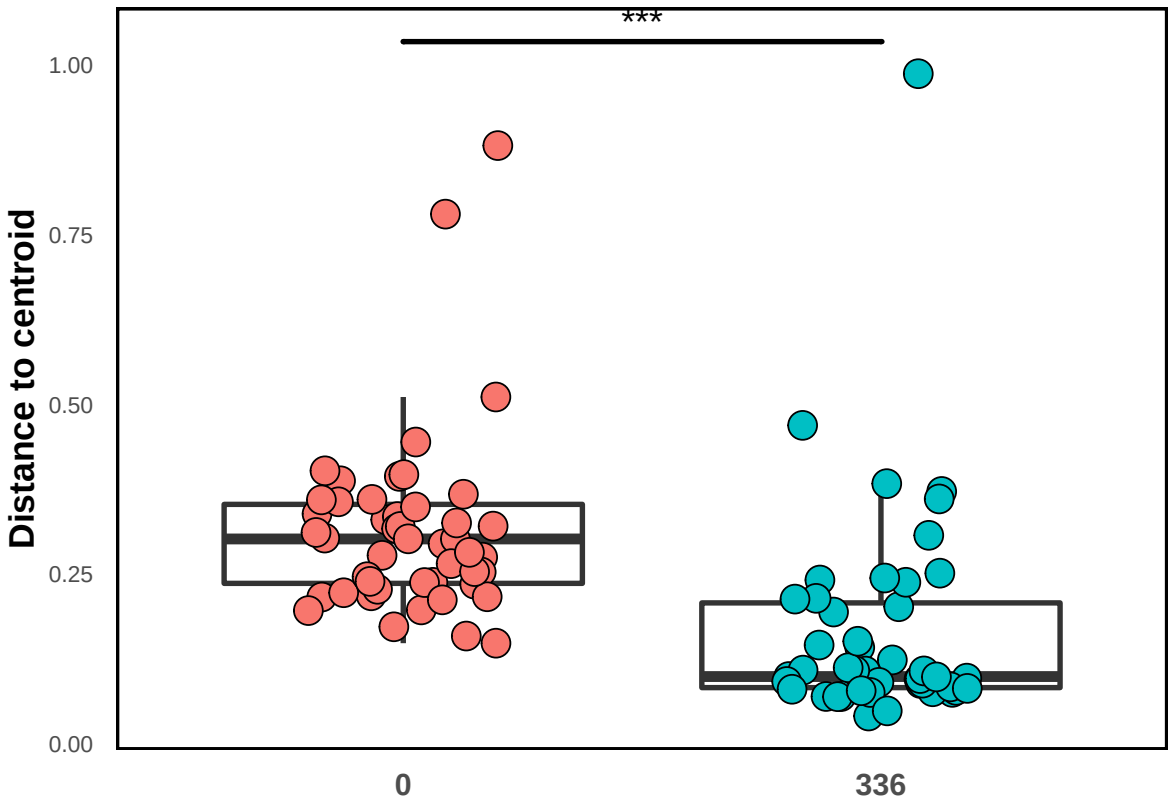
Since `p <<<< 0.0001`, there is strong evidence that the overall community compositions are significantly different (treatment centroid, distance to centroid, community variation) between the two groups. W can visualize this with a 1 SD ellipse:

```
plot(mod, ellipse = TRUE, hull = FALSE)
```



We see that there is clear separation between the two treatment centroids. Let's do some more analysis on the distance-to-centroids that we're seeing:

```r
betadistances <- data.frame(
  time_frame = mod$group,
  distance = mod$distances
)

betadistances %>%
  ggplot(aes(time_frame, distance)) +
  geom_boxplot(size = 1, outlier.shape = NA) +
  geom_jitter(aes(fill = time_frame), size = 5, shape = 21, width = 0.2) +
  theme(
    legend.position = "none",
    panel.grid.minor.x = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.y = element_blank(),
    panel.grid.major.y = element_blank(),
    plot.title = element_text(size = 17),
    plot.subtitle = element_text(size = 9),
    axis.ticks.length = unit(0.25, "cm"),
    axis.ticks.x = element_blank(),
    axis.text.x = element_text(face = "bold", angle = 0, size = 12),
    panel.border = element_rect(color = "black", size = 1, fill = NA),
    axis.title.x = element_blank(),
    axis.title.y = element_text(size = 14, face = "bold"),

  ) +
  labs(
    color = "",
    y = "Distance to centroid"
  ) +
  ggsignif::geom_signif(
    map_signif_level = TRUE,
    comparisons = list(c("0", "336")),
    test = "t.test",
    step_increase = 0.1,
    color = "black",
    size = 1,
    textsize = 5,
    tip_length = 0
  )
```



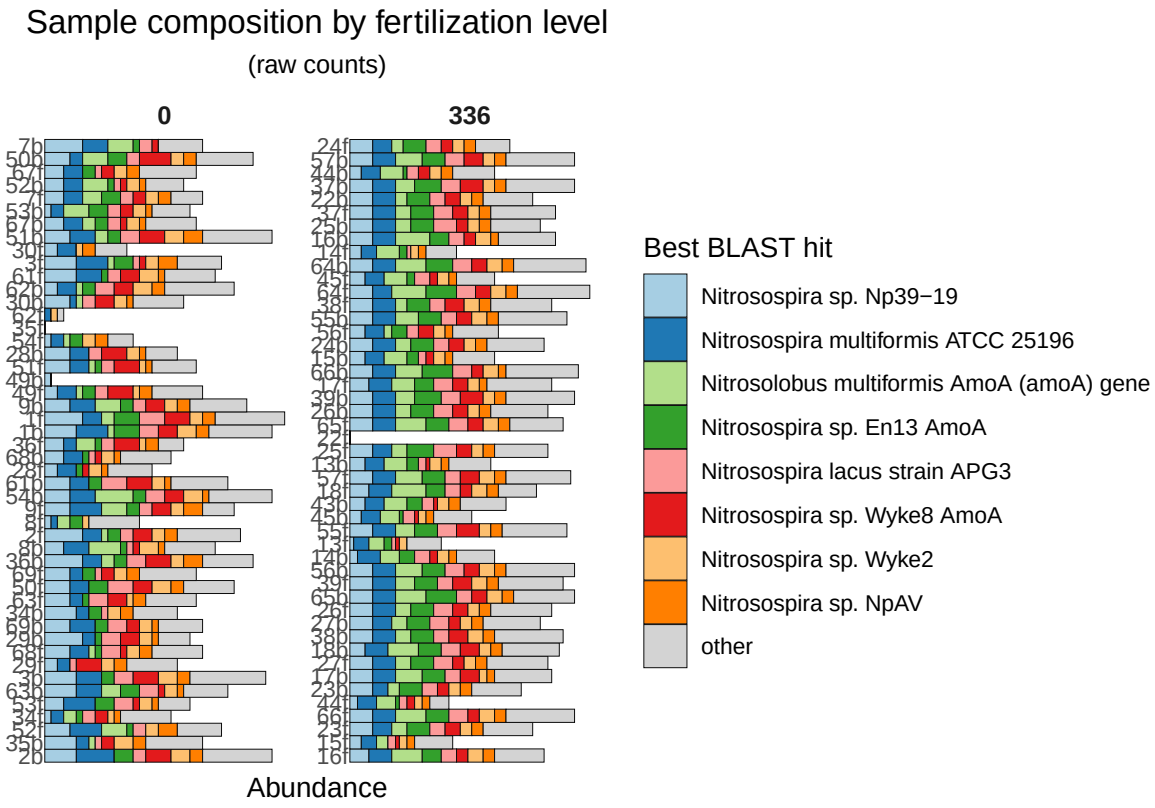
The significance bar is coming from the PERMANOVA test we did above. We see that there is actually less beta diversity (as meaasured by distance-to-centroid) in the fertilized group than in the non- fertilized group. We'll see another visualization backing this up in the next section:

10.12. Composition

Let's visualize the composition of the communities, separated by fertilization. We'll start with raw counts - how many times was that best BLAST hit seen in that sample?

```
comp_barplot(ps, "ta1",
             facet_by = "fert_level",
             sample_order =  "default",
             tax_transform_for_plot = "identity") +
  coord_flip() +
  labs(
    title = "Sample composition by fertilization level",
    subtitle = "(raw counts)"
```

```
  ) +
  theme(
    axis.text.x = element_blank(),
    axis.text.y = element_text(margin = margin(r = -7)),
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5, size = 10),
    strip.text = element_text(size = 10, face = "bold")
  ) +
  guides(
    fill = guide_legend(title = "Best BLAST hit", reverse = TRUE)
  )
```

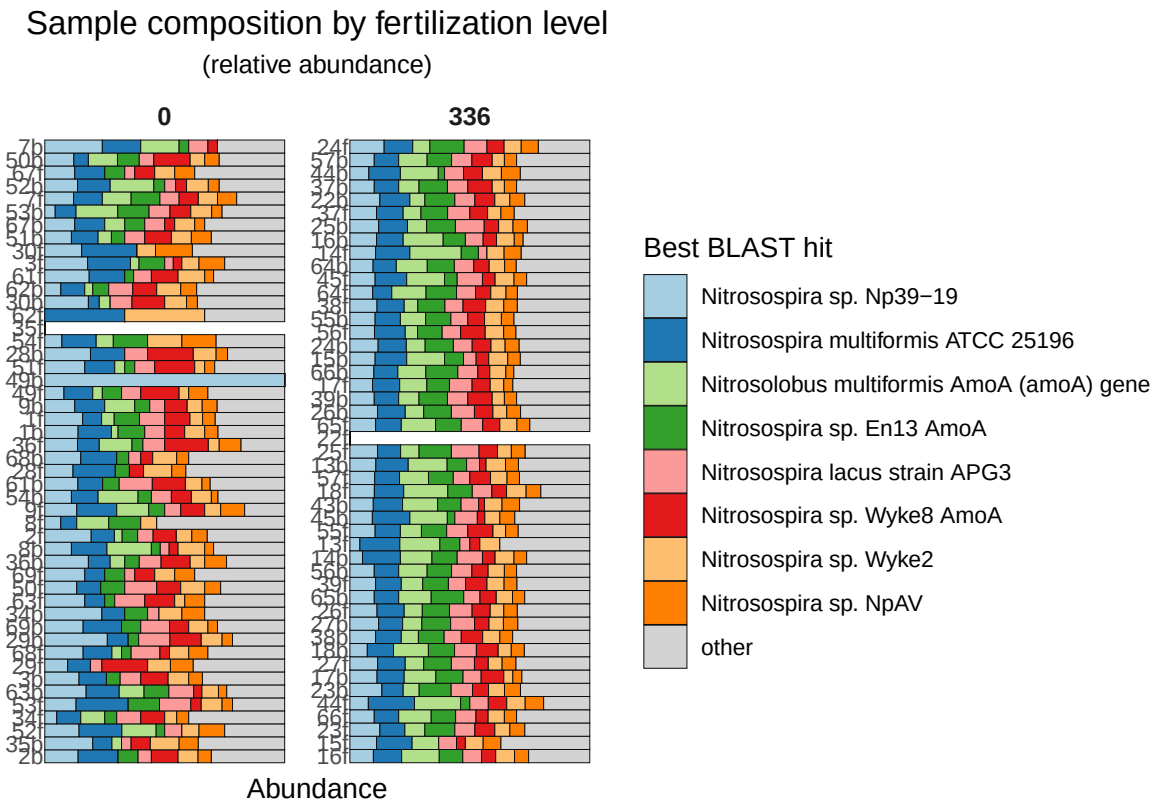


We see that overall the fertilized group appears to have more richness in it.

How about sample composition? IE, relative abundances?

```
comp_barplot(ps, "ta1",
             facet_by = "fert_level",
             sample_order =  "default") +
```

```
  coord_flip() +
  labs(
    title = "Sample composition by fertilization level",
    subtitle = "(relative abundance)"
  ) +
  theme(
    axis.text.x = element_blank(),
    axis.text.y = element_text(margin = margin(r = -7)),
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5, size = 10),
    strip.text = element_text(size = 10, face = "bold")
  ) +
  guides(
    fill = guide_legend(title = "Best BLAST hit", reverse = TRUE)
  )
```



Two big things pop out:

- species distribution is more even in the fertilized group. This makes sense given previous

results showing that Shannon entropy is higher and beta diversity is lower in the fertilized group. You can also see that the communities just look more like each other in the fertilized group, which manifests in shorter distance-to-centroids/lower community variation.
- There's more green in the fertilized group.

10.13. Statistics on a best BLAST hit level

The next chunk is just doing some data transformation stuff to count the number of times each organism was seen in each sample in preparation for the statistical analysis.

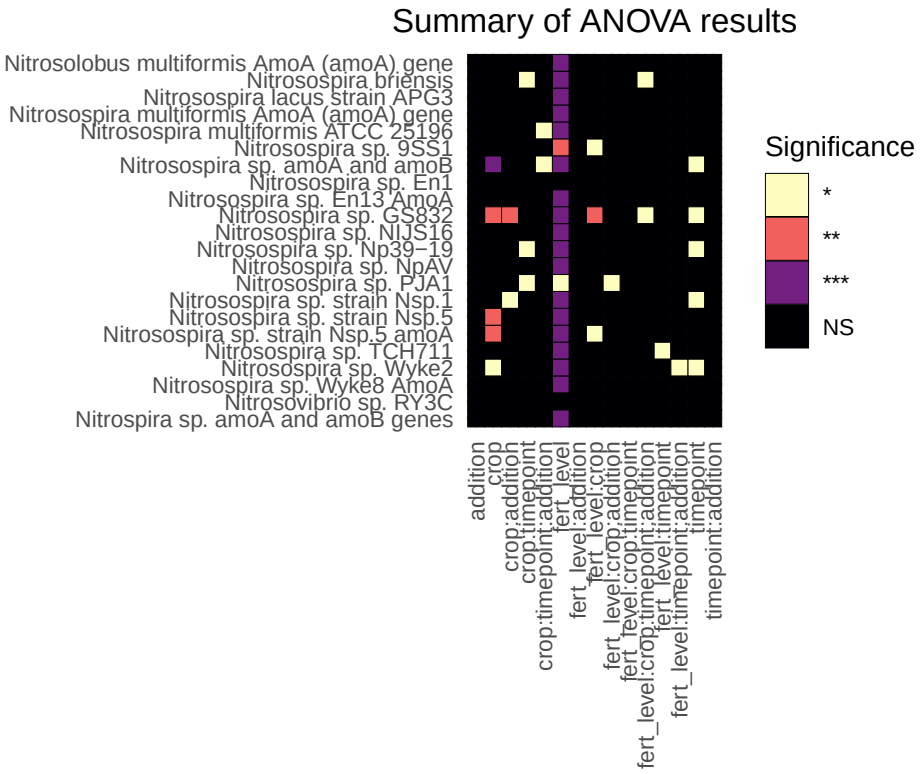
```
pa_count <- ps %>%
  otu_table() %>%
  data.frame %>%
  rownames_to_column(var = "sample_id") %>%
  pivot_longer(amoA.001:amoA.074)


org_table <- tax_table(ps) %>%
  data.frame %>%
  rownames_to_column(var = "name") %>%
  rename(bbh = ta1) %>%
  mutate(cleaned_names = janitor::make_clean_names(bbh))

bbh_sample_counts <- left_join(pa_count, org_table, by = "name") %>%
  group_by(sample_id, bbh) %>%
  summarize(value = sum(value)) %>%
  pivot_wider(names_from = "bbh", values_from = value)

bbh_level_counts <- left_join(bbh_sample_counts,
          sample_data(ps) %>%
            data.frame %>%
            rownames_to_column(var = 'sample_id') %>%
            right_join(bbh_sample_counts)
) %>%
  ungroup()
```

Here, we're preparing formulas to feed to a `lapply` function to perform a Kruskal-Wallis test on all of the organisms.

```
formulae <- lapply(colnames(bbh_sample_counts %>% select(-sample_id) %>% janitor::clean_names

formulae[[1]] <- NULL

res <- lapply(formulae, function(x) broom::tidy(aov(x, data = bbh_level_counts %>% janitor::c
names(res) <- format(formulae)
names(res) <- lapply(names(res), function(x) str_split(x, "~")[[1]][1]) %>% unlist()

anova_results.counts <- lapply(seq_along(res), function(i) res[[i]] %>% mutate(gene = names(r
  bind_rows() %>%
  filter(term != "Residuals") %>%
  mutate(gene = str_trim(gene))
```

Visualizing the results again:

```
anova_results.counts %>%
  left_join(org_table, by = c("gene" ="cleaned_names")) %>%
  mutate(sig = case_when(
    p.value < 0.05 & p.value > 0.01 ~ "*",
    p.value < 0.01 & p.value > 0.001 ~ "**",
    p.value < 0.001 ~ "***",
    TRUE ~ "NS"
  )) %>%
  ggplot(aes(term, bbh, fill = sig)) +
  geom_tile(color = "black") +
  labs(y = "",
       x = "",
       title = "Summary of ANOVA results",
       fill = "Significance ") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
    axis.text.y = element_text()
  ) +
  scale_fill_viridis_d(option = "magma", direction = -1) +
  scale_y_discrete(limits = rev) +
  coord_equal()
```



We see the same pattern at the organism level as when we did this at the gene level: fertilization level is by far the most significant factor affecting Presence/Absence of organisms.

Blair, R., A. Kjuchukova, R. Velazquez, and P. Villanueva. 2020. "Wirtinger Systems of Generators of Knot Groups." *Communications in Analysis and Geometry* 28 (2): 243–62. https://doi.org/10.4310/cag.2020.v28.n2.a2.