Taxonomic compositions

Calculate abundances. Group by treatment for use in plotting.

```
VFLA_rescript_parsed$data$taxon_counts <- calc_taxon_abund(VFLA_rescript_parsed,
    data = "tax_data",
    cols = nogrometadata$SampleID)
VFLA_rescript_parsed$data$tax_prop_data <- calc_obs_props(VFLA_rescript_parsed,
    "tax_data", cols = nogrometadata$SampleID)
VFLA_rescript_parsed$data$tax_abund <- calc_taxon_abund(VFLA_rescript_parsed,
    "tax_prop_data",
    groups = nogrometadata$treatments,
    cols = nogrometadata$SampleID)
```

Start by filtering the data. Note that the proportions will stop summing to 1 as this work progresses, but are still based on the total reads in a sample so that samples with much greater total reads don't disproportionately affect the data. Here, Chloroplasts, Mitochondria, uncultured bacteria, and any that are not assigned a name at genus level are filtered out.

```
filtered_abundance <- VFLA_rescript_parsed$data$tax_abund %>%
    left_join(VFLA_rescript_parsed$data$class_data,
        by = c(taxon_id = "taxon_id")) %>%
    filter(tax_name !=
        "Chloroplast") %>%
    filter(tax_name !=
        "uncultured") %>%
    filter(tax_name !=
        "Mitochondria") %>%
    filter(tax_rank ==
        "g") %>%
    distinct(taxon_id,
        .keep_all = TRUE)
```

Subsample counts that can be used for bar charts are created.

```
treatments_for_counts <- c("Plasticskin",
    "Plasticgills", "Seaweedskin",
    "Seaweedgills", "Plastic",
    "Seaweed")

for (treatment in treatments_for_counts) {

    df_name <- paste0(treatment,
        "counts")

    assign(df_name, filtered_abundance %>%
        select(ends_with(treatment),
            tax_name) %>%
        gather(variables,
            Frequency,
```

```
            1:ncol(.) -
                1) %>%
        mutate(variables = factor(variables)) %>%
        group_by(variables) %>%
        filter(sum(Frequency) >
            0) %>%
        droplevels())
}
```

Pool the smallest taxa so decrease the number of taxa in the plot. We set threshold to 17 to suit the number of colours in our chosen palette, which has 18.

```
for (treatment in treatments_for_counts) {

    df_counts <- get(paste0(treatment,
        "counts"))

    df_pool <- df_counts %>%
        group_by(tax_name) %>%
        summarise(Frequency = max(Frequency),
            mean = mean(Frequency),
            .groups = "drop") %>%
        mutate(pool = rank(-Frequency) >
            17) %>%
        select(tax_name,
            pool)

    assign(paste0(treatment,
        "pool"), df_pool)
}
```

Define colour palette

```
nb.cols <- 18
mycolors <- colorRampPalette(brewer.pal(9,
    "Set1"))(nb.cols)
```

Generate bar charts from the data subsets.These are used later to create combined bar charts.

```
generate_barchart <- function(treatment) {

    df <- inner_join(get(paste0(treatment,
        "counts")), get(paste0(treatment,
        "pool")), by = "tax_name") %>%
        group_by(tax_name) %>%
        mutate(mean = mean(Frequency)) %>%
        ungroup() %>%
        mutate(tax_name = if_else(pool,
            "Other",
            tax_name),
            cycle = case_when(grepl("A_",
                variables) ~
                "A",
                grepl("B_",
                  variables) ~
                  "B",
```

```
            grepl("C_",
              variables) ~
              "C"),
        ring = case_when(grepl("1_",
            variables) ~
            "Pen 1",
            grepl("3_",
              variables) ~
              "Pen 3",
            grepl("8_",
              variables) ~
              "Pen 8",
            grepl("4_",
              variables) ~
              "Pen 4",
            grepl("7_",
              variables) ~
              "Pen 7",
            grepl("12_",
              variables) ~
              "Pen 12"),
        ring = factor(ring,
            levels = c("Pen 1",
              "Pen 3",
              "Pen 8",
              "Pen 4",
              "Pen 7",
              "Pen 12")),
        tax_name = if_else(tax_name ==
            "Methylobacterium-Methylorubrum",
            "Methylorubrum",
            tax_name)) %>%
    group_by(ring,
        tax_name,
        cycle) %>%
    summarise(Frequency = sum(Frequency),
        mean = min(mean),
        .groups = "drop") %>%
    mutate(tax_name = factor(tax_name),
        tax_name = fct_reorder(tax_name,
            mean,
            .desc = FALSE))

ggplot(df, aes(x = factor(ring),
    y = Frequency,
    fill = tax_name)) +
    geom_bar(position = "fill",
        stat = "identity",
        width = 0.8) +
    scale_fill_manual(values = rev(mycolors)) +
    scale_y_continuous(expand = c(0,
        0)) + theme_classic() +
    labs(x = "Samples",
```

```
            y = "Proportion",
            fill = "Genus",
            title = ifelse(grepl("Plastic",
                treatment),
                "Plastic",
                "Seaweed")) +
        guides(x = guide_axis(angle = 30)) +
        facet_wrap(. ~
            cycle) +
        theme(text = element_text(size = 8),
            legend.key.size = unit(0.3,
                "cm"))
}

barcharts <- map(treatments_for_counts,
    generate_barchart) %>%
    set_names(treatments_for_counts)
```

The taxonomy is modified in such a way that it can be used for filtering the feature table. This is easier for the purposes of using corncob than using metacoder's functions.

```
nutaxnofill <- VFLA_taxonomy %>%
    separate(Taxon, into = c("Kingdom",
        "Phylum", "Class",
        "Order", "Family",
        "Genus", "Species"),
        sep = ";", fill = "right") %>%
    mutate(across(Kingdom:Species,
        ~str_remove(.,
            "^[a-z]__"))) %>%
    mutate(across(Kingdom:Species,
        ~ifelse(is.na(.),
            "", .))) %>%
    ungroup() %>%
    select(-Confidence)
```

Genus level data frame created that can be used with corncob.In addition to removing other ranks, the select function removes two samples, that are not used in the analysis (contains "esc").

```
genusdata <- as.data.frame(VFLA_table) %>%
    rownames_to_column(var = "Feature.ID") %>%
    left_join(nutaxnofill,
        by = "Feature.ID") %>%
    filter(Kingdom !=
        "Unassigned") %>%
    filter(Kingdom !=
        "Eukaryota") %>%
    filter(Order != "Chloroplast") %>%
    filter(Family !=
        "Mitochondria") %>%
    filter(Genus != "uncultured") %>%
    filter(Genus != "") %>%
    select(-c(Feature.ID,
        Kingdom, Phylum,
        Class, Order,
```

```
        Family, Species,
        contains("esc"))) %>%
    group_by(Genus) %>%
    summarise(across(everything(),
        \(x)
            sum(x, na.rm = TRUE))) %>%
    arrange(Genus) %>%
    column_to_rownames(var = "Genus")
```

The data frame needs to be reordered so that it matches the metadata.

```
matching_order <- match(nogrometadata$SampleID,
    colnames(genusdata))


genusdata_reordered <- genusdata[,
    matching_order]
```

Below are subsets that will be used to analyse the data for abundance differences.

```
gillsdata <- genusdata_reordered %>%
    select(ends_with("g"))

topgills <- gillsdata %>%
    mutate(Total = rowSums(across(everything()))) %>%
    arrange(desc(Total)) %>%
    slice_head(n = 30) %>%
    arrange(row.names(.)) %>%
    select(-Total)

gillsmetadata <- subset(metadata,
    Type == "Gills")
```

```
skindata <- genusdata_reordered %>%
    select(ends_with("sk"))

topskin <- skindata %>%
    mutate(Total = rowSums(across(everything()))) %>%
    arrange(desc(Total)) %>%
    slice_head(n = 30) %>%
    arrange(row.names(.)) %>%
    select(-Total)

skinmetadata <- subset(metadata,
    Type == "Skin")
```

```
shelterdata <- genusdata_reordered %>%
    select(matches("(p$|sw$)"))

topshelter <- shelterdata %>%
    mutate(Total = rowSums(across(everything()))) %>%
    arrange(desc(Total)) %>%
    slice_head(n = 30) %>%
    arrange(row.names(.)) %>%
    select(-Total)
```

```
sheltermetadata <- subset(nogrometadata,
    Type1 == "Shelter")
```

Differential abundance analysis. In addition to the significant models and p values, the differential abundance values are saved for plotting purposes.

```
topgills_analysis <- differentialTest(formula = ~Shelter1,
    phi.formula = ~Shelter1,
    formula_null = ~1,
    phi.formula_null = ~Shelter1,
    test = "Wald", boot = FALSE,
    data = topgills,
    sample_data = gillsmetadata,
    taxa_are_rows = TRUE,
    fdr_cutoff = 0.05)

fortopgillsplot <- plot(topgills_analysis,
    data_only = TRUE)
topgills_analysis$significant_taxa
topgills_p_values <- stack(topgills_analysis$p_fdr[!is.na(topgills_analysis$p_fdr)])
```

```
topskin_analysis <- differentialTest(formula = ~Shelter1,
    phi.formula = ~Shelter1,
    formula_null = ~1,
    phi.formula_null = ~Shelter1,
    test = "Wald", boot = FALSE,
    data = topskin, sample_data = skinmetadata,
    taxa_are_rows = TRUE,
    fdr_cutoff = 0.05)

fortopskinplot <- plot(topskin_analysis,
    data_only = TRUE)
topskin_analysis$significant_taxa
topskin_p_values <- stack(topskin_analysis$p_fdr[!is.na(topskin_analysis$p_fdr)])
```

```
topshelter_analysis <- differentialTest(formula = ~Shelter1,
    phi.formula = ~Shelter1,
    formula_null = ~1,
    phi.formula_null = ~Shelter1,
    test = "Wald", boot = FALSE,
    data = topshelter,
    sample_data = sheltermetadata,
    taxa_are_rows = TRUE,
    fdr_cutoff = 0.05)

fortopshelterplot <- plot(topshelter_analysis,
    data_only = TRUE)
topshelter_analysis$significant_taxa
topshelter_p_values <- stack(topshelter_analysis$p_fdr[!is.na(topshelter_analysis$p_fdr)])
```

Ggplots created based on the p values from the analysis above as well as the plotting data.

```
gillssigpvalues <- subset(topgills_p_values,
    values < 0.05) %>%
    mutate(rounded = case_when(values >
```

```
        0.009 ~ "*",
        values > 9e-04 ~
            "**", values <
            0.001 ~ "***"))


fortopgillsplot <- fortopgillsplot %>%
    mutate(color = ifelse(x >
        0, "#0047AB",
        "#e73121"), relabund = ifelse(x >
        0, "Seaweed",
        "Plastic"))

colors <- fortopgillsplot$color[order(fortopgillsplot$taxa)]

sigtopgillsplot <- fortopgillsplot %>%
    ggplot(aes(x = x,
        y = taxa)) +
    geom_point(aes(color = relabund)) +
    geom_errorbarh(aes(xmax = xmax,
        xmin = xmin,
        height = 0.2,
        color = relabund)) +
    geom_vline(xintercept = 0,
        linetype = "dotted") +
    scale_color_manual(values = c("#e73121",
        "#0047ab")) +
    scale_y_discrete(labels = ~paste0(.x,
        " (", gillssigpvalues$rounded[match(.x,
            fortopgillsplot$taxa)],
        ")")) + labs(x = "Relative abundance",
    y = "Genus", color = "More abundant in") +
    theme_classic() +
    theme(axis.text.y = element_text(colour = colors),
        axis.ticks.y = element_blank(),
        text = element_text(size = 9))

sigtopgillsplot
```

```
skinsigpvalues <- subset(topskin_p_values,
    values < 0.05) %>%
    mutate(rounded = case_when(values >
        0.009 ~ "*",
        values > 9e-04 ~
            "**", values <
            0.001 ~ "***"))


fortopskinplot <- fortopskinplot %>%
    mutate(color = ifelse(x >
        0, "#0047AB",
        "#e73121"), relabund = ifelse(x >
        0, "Seaweed",
        "Plastic"), taxa = if_else(taxa ==
```

```
        "Methylobacterium-Methylorubrum",
        "Methylorubrum",
        taxa))


colors <- fortopskinplot$color[order(fortopskinplot$taxa)]

sigtopskinplot <- fortopskinplot %>%
    ggplot(aes(x = x,
        y = taxa)) +
    geom_point(aes(color = relabund)) +
    geom_errorbarh(aes(xmax = xmax,
        xmin = xmin,
        height = 0.2,
        color = relabund)) +
    geom_vline(xintercept = 0,
        linetype = "dotted") +
    scale_color_manual(values = c("#e73121",
        "#0047ab")) +
    scale_y_discrete(labels = ~paste0(.x,
        " (", skinsigpvalues$rounded[match(.x,
            fortopskinplot$taxa)],
        ")")) + labs(x = "Relative abundance",
    y = "Genus", color = "More abundant in") +
    theme_classic() +
    theme(axis.text.y = element_text(colour = colors),
        axis.ticks.y = element_blank(),
        text = element_text(size = 9))

sigtopskinplot
```

```
sheltersigpvalues <- subset(topshelter_p_values,
    values < 0.05) %>%
    mutate(rounded = case_when(values >
        0.009 ~ "*",
        values > 9e-04 ~
            "**", values <
            0.001 ~ "***"))


fortopshelterplot <- fortopshelterplot %>%
    mutate(color = ifelse(x >
        0, "#0047AB",
        "#e73121"), relabund = ifelse(x >
        0, "Seaweed",
        "Plastic"))

colors <- fortopshelterplot$color[order(fortopshelterplot$taxa)]

sigtopshelterplot <- fortopshelterplot %>%
    ggplot(aes(x = x,
        y = taxa)) +
    geom_point(aes(color = relabund)) +
    geom_errorbarh(aes(xmax = xmax,
```

```
        xmin = xmin,
        height = 0.2,
        color = relabund)) +
    geom_vline(xintercept = 0,
        linetype = "dotted") +
    scale_color_manual(values = c("#e73121",
        "#0047ab")) +
    scale_y_discrete(labels = ~paste0(.x,
        " (", sheltersigpvalues$rounded[match(.x,
            fortopshelterplot$taxa)],
        ")")) + labs(x = "Relative abundance",
    y = "Genus", color = "More abundant in") +
    theme_classic() +
    theme(axis.text.y = element_text(colour = colors),
        axis.ticks.y = element_blank(),
        text = element_text(size = 9))

sigtopshelterplot
```

Combination plots with bar charts for plastic and seaweed as well as a differential abundance plot. The three figures are for gills, skin, and the shelters themselves.

```
ggarrange(ggarrange(barcharts[["Plasticskin"]],
    barcharts[["Seaweedskin"]],
    nrow = 2, labels = c("a)",
        "b)")), sigtopskinplot,
    ncol = 2, labels = c("",
        "c)"))
ggsave("skinplots.tiff",
    dpi = 300)
```

```
ggarrange(ggarrange(barcharts[["Plasticgills"]],
    barcharts[["Seaweedgills"]],
    nrow = 2, labels = c("a)",
        "b)")), sigtopgillsplot,
    ncol = 2, labels = c("",
        "c)"))
ggsave("gillplots.tiff",
    dpi = 300)
```

```
ggarrange(ggarrange(barcharts[["Plastic"]],
    barcharts[["Seaweed"]],
    nrow = 2, labels = c("a)",
        "b)")), sigtopshelterplot,
    ncol = 2, labels = c("",
        "c)"))
ggsave("shelterlots.tiff",
    dpi = 300)
```