Tidyamplicons workshop

Stijn Wittouck

Overview

1. R, RStudio and the tidyverse
2. Introduction to data visualization
3. Introduction to data manipulation
4. Introduction to tidy data
5. Introduction to tidyamplicons

1. R, RStudio and the tidyverse

About this workshop

Originally constructed by

- Stijn Wittouck
- Sander Wuyts

Originally given at the BBC 2017 symposium in Leuven

Only knowledge required: variables and assignment

What is R

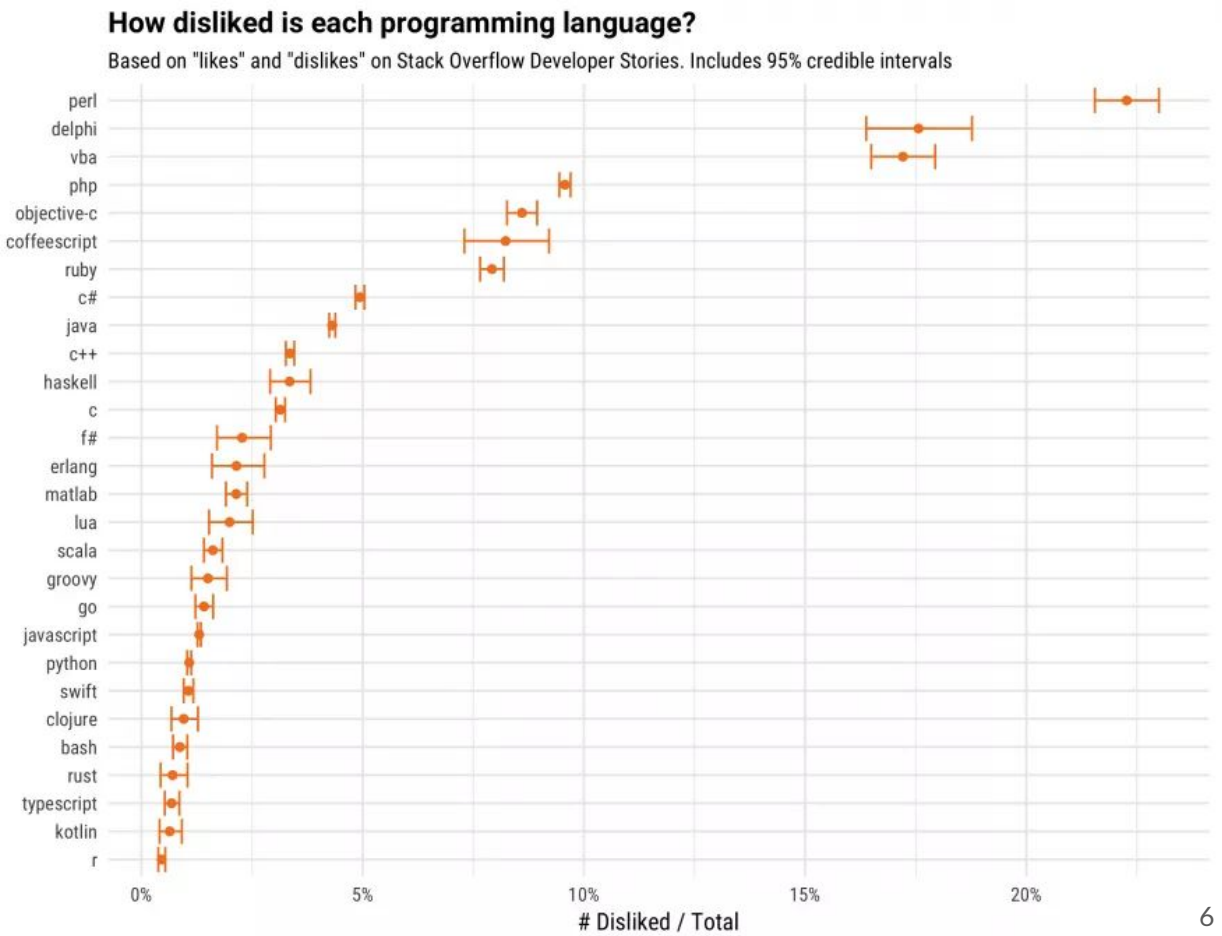
Two meanings:

- **Programming language** for statistical computing
- **Program** that processes scripts written in this language

The R program is **open source**!

The R language can be extended with **packages**

- Comprehensive R Archive Network (CRAN)
- > 11,000 packages available through CRAN, Bioconductor, Github, …



https://boingboing.net/2017/10/31/perl-is-the-most-hated-program.html

What is RStudio

Integrated Development Environment (IDE)

- **Editor** to write scripts
- **Console** to send commands to R and show result
- **Plot window** for figures
- **Environment overview** to inspect current R session
- **HTML viewer** to browse help files

What is the tidyverse

Set of R packages for data processing and visualization

- Common design philosophy
- Developed by Hadley Wickham

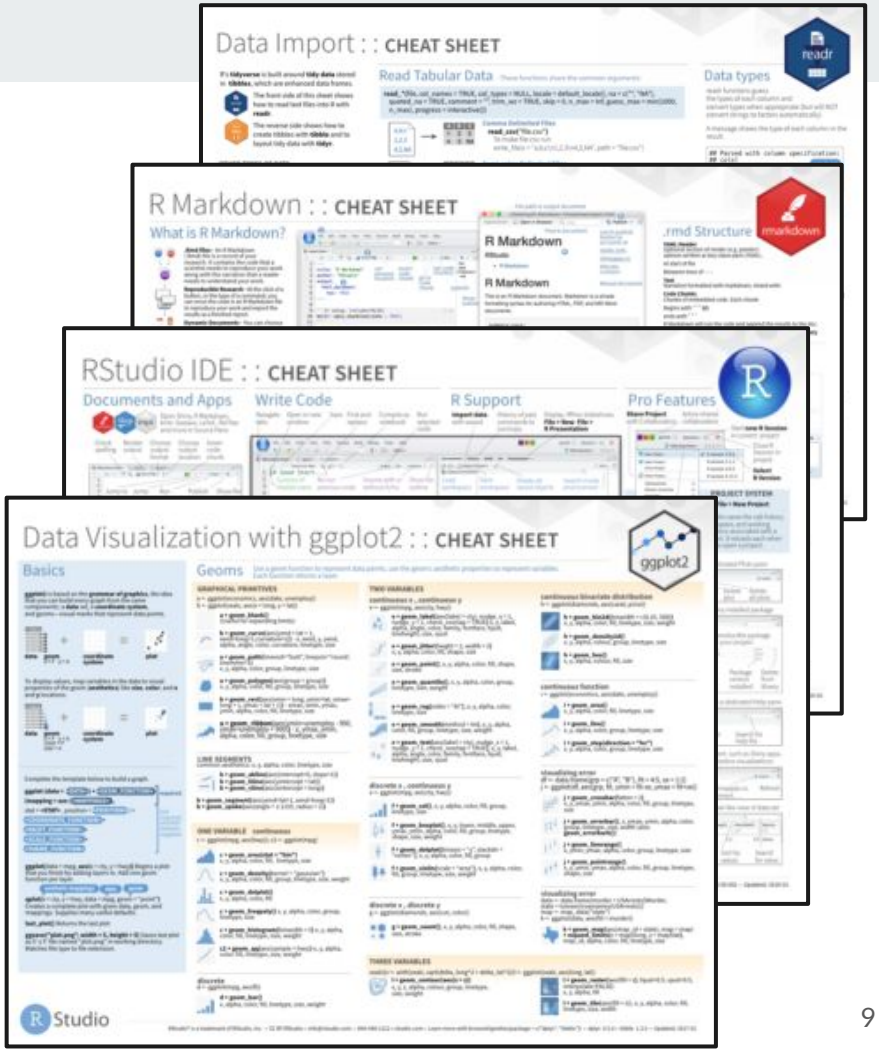
Most important packages:

- readr: import tables
- tidyr: make tables tidy
- dplyr: processing tidy tables
- ggplot: visualizing tidy tables

RStudio cheat sheets

Very good reference if you can't remember the right syntax

https://www.rstudio.com/resources/cheatsheets/



Practical

Required software: R, RStudio, tidyverse

Datasets

- NCBI's eukaryote genome data → demonstrations
- enterotype dataset (Arumugam *et al.*, 2011) → exercises

Download our slides and these datasets from:
https://github.com/SWittouck/tidyverse_workshop

2. Introduction to data visualization

Grammar of Graphics

“An abstraction which makes thinking, reasoning and communicating graphics easier”

- First described by Leland Wilkinson (**G**rammar of **G**raphics, 1999)
- Implemented in **gg**plot2 (Hadley Wickham)
- Divide your graphics in different layers based on grammar

=> Use building blocks to create your visualisation

Two basic components

geoms

- visual objects in your graph
- e.g. points, lines, boxplots, bars
- represent your data points

aesthetics

- visual properties of geoms
- e.g. x-coordinate, y-coordinate, color, size, shape
- represent your variables

Building your plot: layers

Data layer

- dataset (tibble)
- aesthetic mappings: variables ⟶ aesthetics

Geom layers

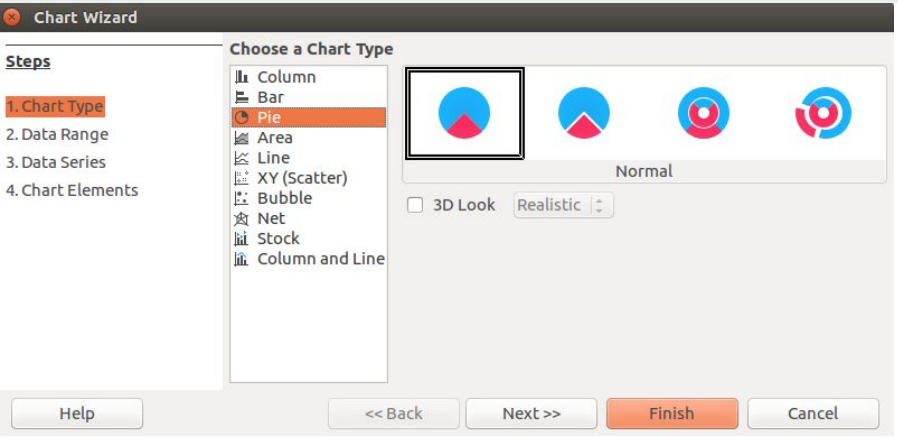
Optional layers

- scale layers: customize aesthetics
- theme layers: customize other visual aspects
- special layers: facet layers, coordinate flip, ...

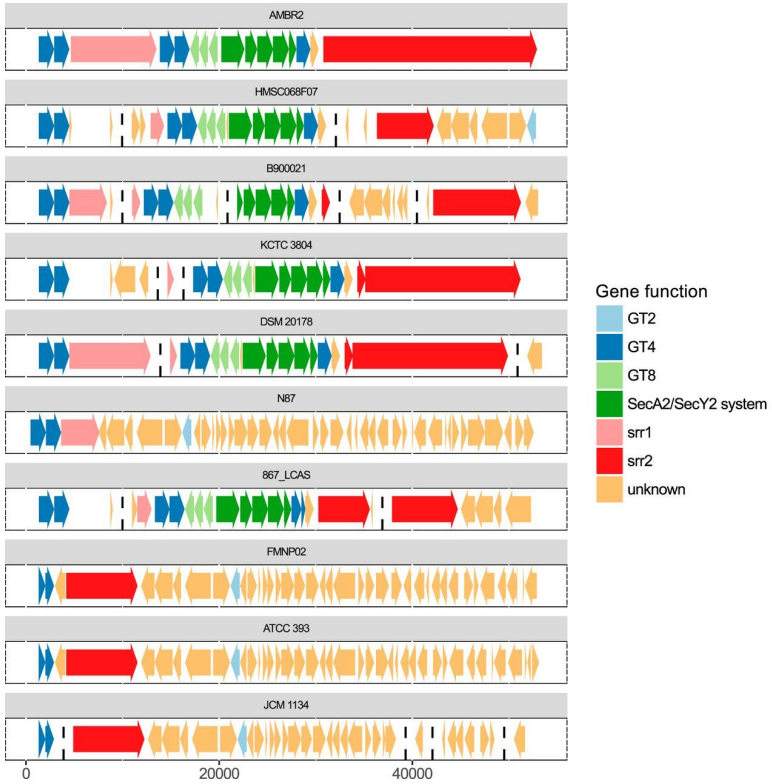
ggplot vs excel

Why ggplot?

- automatic documentation
- reproducible
- flexible
- connects with data transformation



“Click button” visualizations



Grammar of graphics visualisation
ggplot2

Geoms and aesthetics

Most interesting geoms:

- geom_point: x, y, col, shape, size
- geom_line: x, y, group, col, lty, size
- geom_boxplot: x, y, group, col, fill
- geom_bar: x, col, fill
- geom_density: x, col, fill

You can combine geoms!

Reading data

```
read_tsv()

read_csv()

read_table()
```

Exercises chapter 2

1. Read in “sampledata.tsv”
2. Explore the dataset
3. Plot the amount of males and females in this study using a barchart
4. Do the same but for the nationality of the participants instead
5. Create a boxplot showing the age distribution of each nationality. Use the fill aesthetic to make it a little bit more colorful
6. Add an extra layer to 5. with plotting points over the boxplot. Remove that layer again and explore the difference with geom_jitter()
7. Advanced: Make a density plot of the age variable of all participants coloured by gender, faceted per nationality.

3. Introduction to data manipulation

Grammar of data manipulation

ggplot: grammar of graphics → similar “grammar of data manipulation”?

data structure: “**tibble**”

verbs:

- functions that perform one task
- tibble as input (first argument)
- tibble as output

complex tasks can be expressed as sequences of simple verbs

Grammar of data manipulation

package **dplyr**

5 essential verbs:

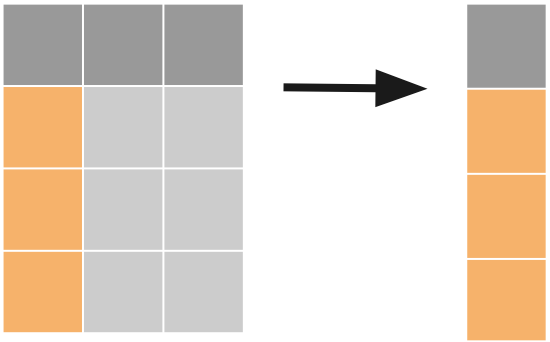
- `select()`: select columns
- `mutate()`: make new columns
- `filter()`: select rows
- `arrange()`: order rows
- `summarize()`: summarize rows

Select columns: `select()`

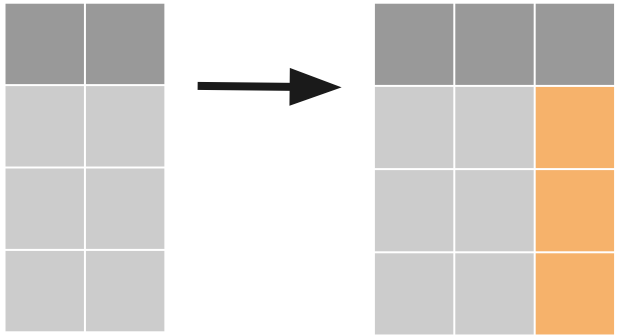
```
tibble <- select(tibble, var_1, var_2, …)
```

helper verbs for variable selection:

- `contains()`
- `starts_with()`
- `ends_with()`
- `` `-` ``



Make columns: `mutate()`



```
tibble <- mutate(tibble,
  new_var_1 = expression_1,
  new_var_2 = expression_2, …
)
```

use “=”, not “<–”

expressions:

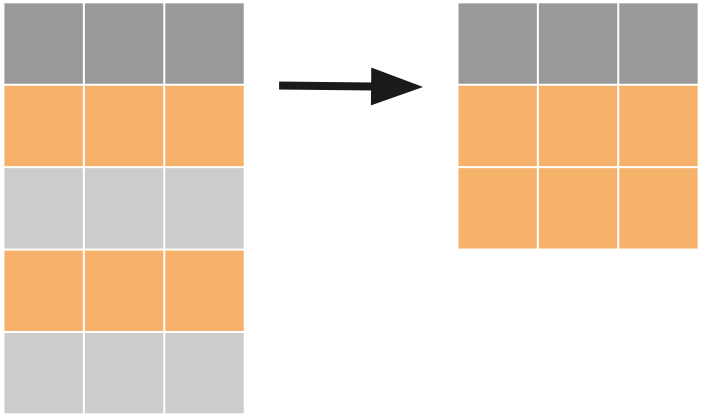
- should result in a vector: e.g. +, –, *, /, ^
- or in one value: `mean()`, `median()`, ...

Select rows: `filter()`

```
tibble <- filter(tibble, logical_variable)
```

logical variable:

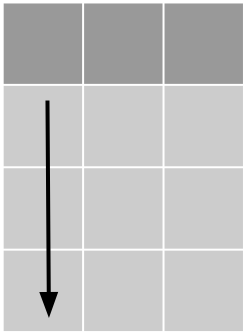
- type directly, e.g. `c(T, T, F, T, ...)`
- create from variables:
  - `==, !=, >, <, >=, <=, %in%,`
  - `is.na()`



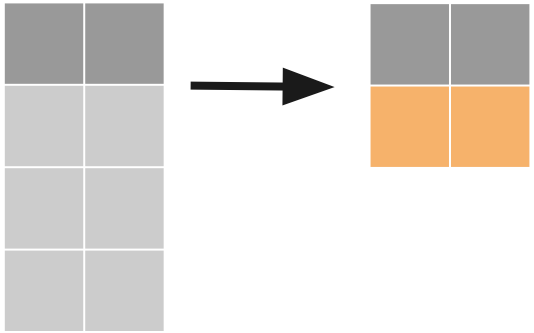
Order rows: `arrange()`

```
tibble <- arrange(tibble, var_1, - var_2)
```

use “-” to sort in inverse order



Summarize rows: `summarize()`



```
tibble <- summarize(tibble,
aggregated_var = expression)
```

expressions should result in one value, e.g.:

- `mean()`
- `median()`
- `sd()`
- `sum()`
- `n()`

Group-wise analysis

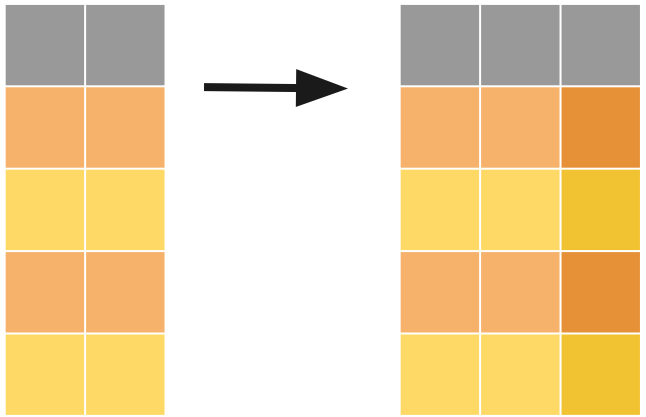
You can add grouping structure to a tibble

- computations within other verbs (e.g. `mutate()`, `summarize()`) will happen per group
- verbs:
  - `group_by()` : add grouping
  - `ungroup()` : remove grouping
- you can group by multiple variables simultaneously
  - groups will be combinations of variable values

Group-wise analysis

Workflow for group-wise **mutate**:

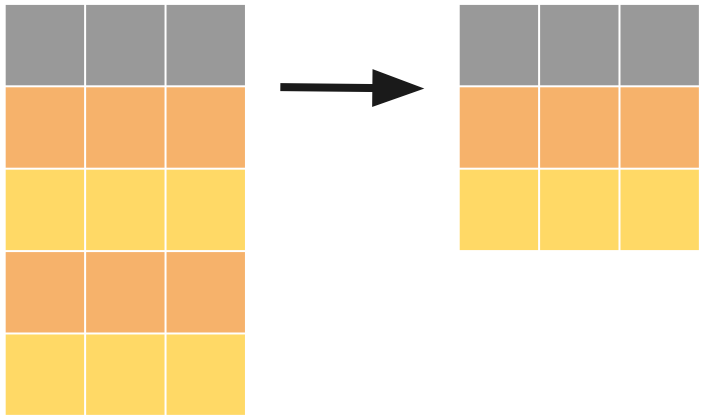
1. `group_by()`
2. `mutate()`
3. `ungroup()`



Group-wise analysis

Workflow for group-wise **summarize**:

1. group_by()
2. summarize()



The pipe operator (%>%)

pass output of LHS as first argument to RHS

`f(x, y)` can be written as `x %>% f(y)`

advantages:

- less typing
- less redundancy (easier to change object names)
- more readable code

Exercises chapter 3

1. **Import** the file "sampledata.tsv" as a tibble.
2. **Filter** out all rows where the variables nationality or bmi_group are NA. Also, **drop all variables except nationality and bmi_group.** Store the resulting tibble as “sample_data_filtered”.
3. Start from “sample_data_filtered”. Make a tibble "sample_data_summary" with a **count of participants per combination of nationality and bmi_group.** Sort the table by nationality and inversely by count within each nationality.
4. Make a **bar plot** to inspect whether some nationalities have more obese participants than others.

4. Introduction to tidy data

Untidy data

How would you make the following figure using ggplot2:

- day on the x-axis
- count on the y-axis (numbers of turnips)

name	day_1	day_2	day_3
edmund	10	11	11
baldrick	19	21	17
percy	3	5	6

Untidy data

Plotting impossible!

Why?

- Turnip count should be one variable, but it is spread over multiple columns
- Day should be a variable, but this information is now in the column headers

name	day_1	day_2	day_3
edmund	10	11	11
baldrick	19	21	17
percy	3	5	6

Tidy data

What changed?

1. The variable “turnip count” is now in one column
2. The variable “day” is now a separate column
3. Values in all other columns are duplicated

name	day	turnips
edmund	day_1	10
edmund	day_2	11
edmund	day_3	11
baldrick	day_1	19
baldrick	day_2	21
baldrick	day_3	17
percy	day_1	3
percy	day_2	5
percy	day_3	6

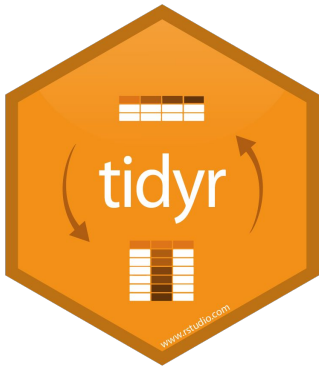
name	day_1	day_2	day_3
edmund	10	11	11
baldrick	19	21	17
percy	3	5	6

Tidy data:

1. Each row corresponds to one observation
2. Each column corresponds to one variable

name	day	turnips
edmund	day_1	10
edmund	day_2	11
edmund	day_3	11
baldrick	day_1	19
baldrick	day_2	21
baldrick	day_3	17
percy	day_1	3
percy	day_2	5
percy	day_3	6

Tidying verbs



Package “tidyr”

- `gather()`: make table tidy (wide to long)
- `spread()`: make table untidy (long to wide)

Tidying: `gather()`

```
tibble <- gather(tibble, value = “variable_values”,
key = “variable_headers”, var_1, var_2, …)
```

Input:

1. the variables you want to gather into one variable
2. the name of that new variable (value)
3. the name of the new variable with the info from the headers (key)

Why tidy data?

Easier to create ggplot visualizations

Easier to manipulate (e.g. aggregating levels)

More scalable format (adding more “value” variables possible)

Exercises chapter 4

1. **Import** the file otutable.tsv as a tibble.
2. **Tidy the tibble**. The result should be a tibble with three columns: sample, taxon, abundance.
3. Add a fourth column with **relative abundances within a sample**. Call it “rel_abundance”. (Hint: if this is difficult, first try to make a new column “total_sample_abundance” with the summed abundances per sample.)
4. Filter the tibble so that only **taxa are retained with a mean relative abundance of at least 1%**. Important: this is not the same as just filtering out rows with rel_abundance < 1%. You should throw out all rows belonging to taxa that have a mean relative abundance < 1%. (Hint: if this is difficult, first try to make a column “mean_taxon_relative_abundance”.)
5. **Make a tile plot** to visualize the relative abundances. Put the samples on the x-axis and taxa on the y-axis.

5. Introduction to tidyamplicons

Tidy philosophy

TA objects

- list of three tidy tibbles: samples, taxa, abundances

TA verbs

- TA object as input (first argument)
- TA object as output

TA verbs

Basic verbs: select/mutate /filter + samples/taxa/abundances

Adder verbs:

- samples: lib_size, diversity_measures, sample_clustered, pcoa, spike _ratio
- taxa: max_rel_abundance, total_rel_abundance, rel_occurrence, taxon_name, taxon_name_color, jervis_bardy, presence_counts
- abundances: rel_abundance

Aggregation verbs: aggregate_samples, aggregate_taxa

Visualization

Built-in: `get_bar_plot()`

- clusters your samples on the x-axis
- gives taxa with identical classification a number
- gives the n most abundant taxa a color
- automatically calculates relative abundances

Visualization

Make your own visualizations!

- easy because of tidiness
- use samples(), taxa(), abundances(), everything() to get a tibble

Exercises chapter 5

Hint: try to make one single “pipe chain” for each exercise

1. **Load the data** by running `load("data/carrots.rda")`
2. Add **library sizes** (numbers of reads per sample) to the tidyamplicons object and visualize them using a density geom.
3. Filter the data to only retain samples of type “FP” (ferme pekes samples). Add **diversity measures**. Now make a plot to compare inverse simpson diversities between the days of fermentation.
4. Make a **PCoA plot**. Color the samples according to the type of fermentation.
5. Filter the data to retain only read counts of the phylum “Firmicutes”. Make a bar plot using `get_bar_plot()`.

Additional table processing functions

Count verbs

`add_count(tibble, vars)`

- add a column with redundant counts

`count(tibble, vars)`

- summarize and add a column with counts

Joining tables

Verbs:

- `left_join()`
- `right_join()`
- `inner_join()`
- `full_join()`

Joins by columns with same name

Splitting and merging columns

```
separate(tible, col, into, sep, remove)

unite(data, col, vars, sep)
```

Other tidyverse packages

Other tidyverse packages

- *lubridate* => Work with different time formats
- *rvest* => Scrape the web for data
  http://www.maartenlambrechts.com/2016/10/03/how-i-built-a-scraper-to-measure-activity-of-mps.html
  http://t-k.blue/blog/where-to-live-in-poland-or-where-to-move/
- *purr* => Functional programming
- *stringr* => Work with strings