Babraham Bioinformatics

Exercises:
Introduction to ggplot2

Version 2021-09

Licence

Exercise 1: Simple point and line plots

Load the data from the `weight_chart.txt` file. This is a tab delimited text file. You'll need to use `library(tidyverse)` to load the tidyverse functions, then set the working directory with `Session > Set Working Directory > Choose Directory` in RStudio then use `read_delim()` to load the file and save it to a variable.

This file contains the details of the growth of a baby over the first few months of its life.

- Draw a scatterplot (using `geom_point`) of the `Age` vs `Weight`. When defining your aesthetics the `Age` will be the `x` and `Weight` will be the `y`.

- Make all of the points filled with `blue2` by putting a fixed aesthetic into `geom_point()` and give them a size of 3

- You will see that an obvious relationship exists between the two variables. Change the geometry to `geom_line` to see another way to represent this plot.

- Combine the two plots by adding both a `geom_line` and a `geom_point` geometry to show both the individual points and the overall trend.

Load the data for the `chromosome_position_data.txt` file

- Use `pivot_longer` to put the data into tidy format, by combining the three data columns together. The options to `pivot_longer` will be:
  - The columns to restructure: `cols=Mut1:WT`
  - The name of the new names column: `names_to="Sample"`
  - The name of the values column: `values_to="Value"`

- Draw a line (`geom_line`) graph to plot the position (`x=Position`) against the value (`y=Value`) and splitting the Samples by colour (`colour=Sample`). Use the `size` attribute in `geom_line` to make the lines slightly thicker than their default width.

If you have time

- Load in the `genomes.csv` file and use the `separate` function to turn the `Groups` column into `Domain`, `Kingdom` and `Class` based on a semicolon delimiter.

- Plot a point graph of `log10(Size)` vs `Chromosomes` and colour it by `Domain`

Exercise 2: Barplots and Distributions

Load the data from `small_file.txt` using `read_delim`

- Plot out a barplot of the lengths of each sample from category A
  - Start by filtering the data to keep only Sample A samples
    - `small %>% filter(Category == "A")`
  - Pass this filtered tibble to `ggplot`
  - Your x aesthetic will be Sample and your y will be length
  - Since the value in the data is the bar height you need to use `geom_col`

- Plot out a barplot (using `geom_bar`) of the mean length for each category in `small.file`
  - You will need to set `stat="summary", fun="mean"` in `geom_bar` so it plots the mean value

- Add a call to `geom_jitter()` to the last plot so you can also see the individual points
  - Colour the points by `Category` and decrease the `width` of the jitter columns to get better separation. Make sure height is set to 0
  - If you don't want to see the legend then you can set `show.legend=FALSE` in `geom_jitter`.

Load the data from `expression.txt` using `read_delim.`

- Plot out the distribution of `Expression` values in this data. You can try both `geom_histogram` and `geom_density`. Try changing the color and fill parameters to make the plot look prettier. In `geom_histogram` try changing the `binwidth` parameter to alter the resolution of the distribution.

Load the data from `cancer_stats.csv` using `read_delim`.

- Plot a barplot (`geom_col`) of the number of Male deaths for all Sites. (`x=Site, y=`Male Deaths``) make sure you let the RStudio auto-complete help you to fill in the `Male Deaths` column name so you get the correct backtick quotes around it.

- You won't be able to show all of the categories so just show the first 5 (`cancer %>% slice(1:5) %>% ggplot…`)

If you have time

Create a new variable in `child.variants` loaded from `Child_Variants.csv` called `Good` using `mutate` and `if_else`. The value should be "`GOOD`" if `QUAL == 200` otherwise it should be "`BAD`"

Plot out a violin plot, using `geom_violin()` of the `MutantReads` for the two `Good` categories.

Exercise 3: Annotation, Scaling and Colours

Use `theme_set` to set your ggplot theme to be `theme_bw` with a `base_size` of 12. Replot one of your earlier plots to see how its appearance changed.

In the cancer barplot you did in exercise 2 you had to exclude sites because you couldn't show them on the x axis. Use the `coord_flip` transformation to switch the x and y axes so you can remove the `slice` function which restricted you to 5 sites, and show all of the sites again.

Load the data from `brain_bodyweight.tsv`

- Plot a scatterplot of the brain against the body
- Change the axis labels (`xlab` and `ylab`) to say Brainweight (g) and Bodyweight (kg) and add a suitable title (`ggtitle`).
- Both brainweight and bodyweight are better displayed on a log scale – try implementing this in one of the ways below
    - Turn the axes into log scale axes (`scale_x_log10` and `scale_y_log10`)
    - Modify the data to be log transformed when creating the aesthetic mapping (pass the column name into `log10()` when defining the aesthetic mapping in `aes()`)
    - Use `mutate` to modify the original data before passing it to `ggplot`
- Color the plot by Category, and change the colours to use the ColorBrewer "Set1" palette (`scale_colour_brewer`)
- Change the ordering of the categories to be "Domesticated", "Wild", "Extinct"

If you have time

Create a barplot of the brainweight of all species, coloured by their bodyweight. Use a custom colour scheme for the colouring of the bars. You will again need to use a log scale for the brain and bodyweight.

Exercise 4: Summary Overlays

Load the data in `treatments.csv` with `read_delim`.

- Plot a stripchart of the four conditions using `geom_jitter()`
- Overlay a boxplot of the same data along with the raw points
    - Adjust the `size` and `width` of spread of the points in `geom_point` to something sensible
    - Adjust the `size` of the lines in the boxplot
    - Make sure `geom_boxplot` is drawn first so you can see everything
    - Try colouring the points by the condition to see if it's any clearer.
- Plot the same data as a barplot with errorbars for the SEM
    - Use a `geom_bar` for the barplot with `stat="summary"` and then use `stat_summary` with a geometry of `errorbar` with the default `mean_se` values.

If you have time

Take the same `treatment` data and pre-calculate a `mean` and `sd` from it using `group_by` and `summarise`. Use these pre-calculated values to plot out the same barplot as before.

Replot the stripchart, but instead of overlaying a boxplot, use `stat_summary` to just add a bar to indicate the mean.

Exercise 5: Faceting and Highlighting

Load the data in `up_down_expression.txt`

- Plot out a scatterplot of `Condition1` vs `Conditon2` coloured by `State`
- Change the coloring using `scale_colour_manual` so that up is red, unchanging is grey and down is blue
- Add text labels to the following genes
    - `Col1A2, TCL1B, SPTSSB, SULF2`
    - Filter the full dataset using `up_down %>% filter( Gene %in% c("Col1A2", "etc" ))` and save the result
    - Pass the filtered dataset to the `data` option of `geom_text`. Make sure you added `label=Gene` to your aesthetic mappings
    - Colour the labels black and use `hjust=1.2` to position them to be readable away from the actual points, or use `geom_text_repel` to adjust the positions automatically.
- Use `geom_abline` to put a null line across the diagonal (`slope=1, intercept=0`)

Load the data in `DownloadFestival.csv`

- Draw a stripchart (`geom_jitter`) of the cleanliness values for males and females separately
- Use `facet_grid(cols=vars(day))` to split the plot based on the day of the festival to see the effect this had on the data
- Make some additions to the plot
    - Colour the male samples red and the female samples blue
    - Add a line to show the mean by using a stat summary

If you have time

Add a new column called `attendance` to the data to say how many days people attended the festival. To do this you will need to:

1. Group by the `person`
2. Use `count` to get a count of how many times each `person` occurred
3. Use `right_join` to merge the counts back into the original data
4. Rename the `n` column to `attendance`

Now redraw the plot but faceting by both `attendance` and `day`