Data Carpentry Tutorial

Ella Crotty

Contents

Filtering **1**

Dataframes **2**

Dates **3**

Tidyverse **4**

Plotting **5**

Faceting 11
Saving a theme 12
Patchwork 13

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- 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
```

```
library(lubridate)
```

Tutorial used

- Object names can't start with a number
- Styles can include "lower_snake", "UPPER_SNAKE", "lowerCamelCase", "UpperCamelCase"
- Tidyverse Style Guide
- R indices start at 1

Filtering

```
round(digits = 2, x = 3.14159)
```

```
## [1] 3.14
```

```
round(3.14159, 2)
```

```
## [1] 3.14
```

```
weight_g <- c(21, 34, 39, 54, 54, 55)
weight_g <- c(weight_g, 90) # add to the end of the vector
weight_g > 50    # will return logicals with TRUE for the indices that meet the condition
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

```
#> [1] FALSE FALSE FALSE  TRUE  TRUE
## so we can use this to select only the values above 50
weight_g[weight_g > 50]
```

```
## [1] 54 54 55 90
```

```
weight_g[weight_g == 39 | weight_g == 54]
```

```
## [1] 39 54 54
```

```
weight_g[weight_g %in% c(21,39, 54)]
```

```
## [1] 21 39 54 54
```

```
heights <- c(63, 69, 60, 65, NA, 68, 61, 70, 61, 59, 64, 69, 63, 63, NA, 72, 65, 64, 70,
↪  63, 65)
heights_no_na <- na.omit(heights)
heights_no_na
```

```
##  [1] 63 69 60 65 68 61 70 61 59 64 69 63 63 72 65 64 70 63 65
## attr(,"na.action")
## [1]  5 15
## attr(,"class")
## [1] "omit"
```

Dataframes

```
# download.file(url = "https://ndownloader.figshare.com/files/2292169",
              # destfile =
              ↪  "~/Dropbox/Coding/Summer2024/RTutorials/DataCarpentry/RawData/portal_data_joined.csv"
surveys <-
↪  read_csv("~/Dropbox/Coding/Summer2024/RTutorials/DataCarpentry/RawData/portal_data_joined.csv")
```

```
## Rows: 34786 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): species_id, sex, genus, species, taxa, plot_type
## dbl (7): record_id, month, day, year, plot_id, hindfoot_length, weight
##
## 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.
```

```
summary(surveys)
```

```
##    record_id         month             day             year         plot_id
##  Min.   :    1   Min.   : 1.000   Min.   : 1.0   Min.   :1977   Min.   : 1.00
##  1st Qu.: 8964   1st Qu.: 4.000   1st Qu.: 9.0   1st Qu.:1984   1st Qu.: 5.00
##  Median :17762   Median : 6.000   Median :16.0   Median :1990   Median :11.00
```

```
##  Mean   :17804   Mean   : 6.474   Mean   :16.1   Mean   :1990   Mean   :11.34
##  3rd Qu.:26655   3rd Qu.:10.000   3rd Qu.:23.0   3rd Qu.:1997   3rd Qu.:17.00
##  Max.   :35548   Max.   :12.000   Max.   :31.0   Max.   :2002   Max.   :24.00
##
##   species_id            sex            hindfoot_length     weight
##  Length:34786       Length:34786       Min.   : 2.00   Min.   :  4.00
##  Class :character   Class :character   1st Qu.:21.00   1st Qu.: 20.00
##  Mode  :character   Mode  :character   Median :32.00   Median : 37.00
##                                        Mean   :29.29   Mean   : 42.67
##                                        3rd Qu.:36.00   3rd Qu.: 48.00
##                                        Max.   :70.00   Max.   :280.00
##                                        NA's   :3348    NA's   :2503
##     genus              species              taxa            plot_type
##  Length:34786       Length:34786       Length:34786       Length:34786
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

```
names(surveys)
```

```
##  [1] "record_id"       "month"           "day"             "year"
##  [5] "plot_id"         "species_id"      "sex"             "hindfoot_length"
##  [9] "weight"          "genus"           "species"         "taxa"
## [13] "plot_type"
```

```
rownames(surveys)[1:5]
```

```
## [1] "1" "2" "3" "4" "5"
```

```
year_fct <- factor(c(1990, 1983, 1977, 1998, 1990))
as.numeric(year_fct)               # Wrong! And there is no warning, it just returns the
↪  levels
```

```
## [1] 3 2 1 4 3
```

```
as.numeric(as.character(year_fct)) # Works...
```

```
## [1] 1990 1983 1977 1998 1990
```

```
as.numeric(levels(year_fct))[year_fct]    # The recommended way.
```

```
## [1] 1990 1983 1977 1998 1990
```

Dates

```
surveys$date <- ymd(paste(surveys$year, surveys$month, surveys$day, sep = "-"))
```

```
## Warning: 129 failed to parse.
```

```
summary(surveys$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "1977-07-16" "1984-03-12" "1990-07-22" "1990-12-15" "1997-07-29" "2002-12-31"
```

```
##          NA's
##         "129"
```

```
missing_dates <- surveys[is.na(surveys$date), c("year", "month", "day")]
```

Tidyverse

Tidy data:

1. Each variable has its own column
2. Each observation has its own row
3. Each value must have its own cell
4. Each type of observational unit forms a table

`pivot_wider()` takes three principal arguments:

1. the data
2. the `names_from` column variable whose values will become new column names.
3. the `values_from` column variable whose values will fill the new column variables.

Further arguments include `values_fill` which, if set, fills in missing values with the value provided.

```
surveys_gw <- surveys %>%
  filter(!is.na(weight)) %>%
  group_by(plot_id, genus) %>%
  summarize(mean_weight = mean(weight))
```

```
## `summarise()` has grouped output by 'plot_id'. You can override using the
## `.groups` argument.
```

```
surveys_wide <- surveys_gw %>%
  pivot_wider(names_from = genus, values_from = mean_weight)
```

`pivot_longer()` takes four principal arguments:

1. the data
2. the `names_to` column variable we wish to create from column names.
3. the `values_to` column variable we wish to create and fill with values.
4. cols are the name of the columns we use to make this pivot (or to drop).

To recreate `surveys_gw` from `surveys_wide` we would create a names variable called genus and value variable called `mean_weight`.

In pivoting longer, we also need to specify what columns to reshape. If the columns are directly adjacent as they are here, we don't even need to list the all out: we can just use the : operator!

```
surveys_long <- surveys_wide %>%
  pivot_longer(names_to = "genus", values_to = "mean_weight", cols = -plot_id) # plot_id
  ↪  stays the same
```

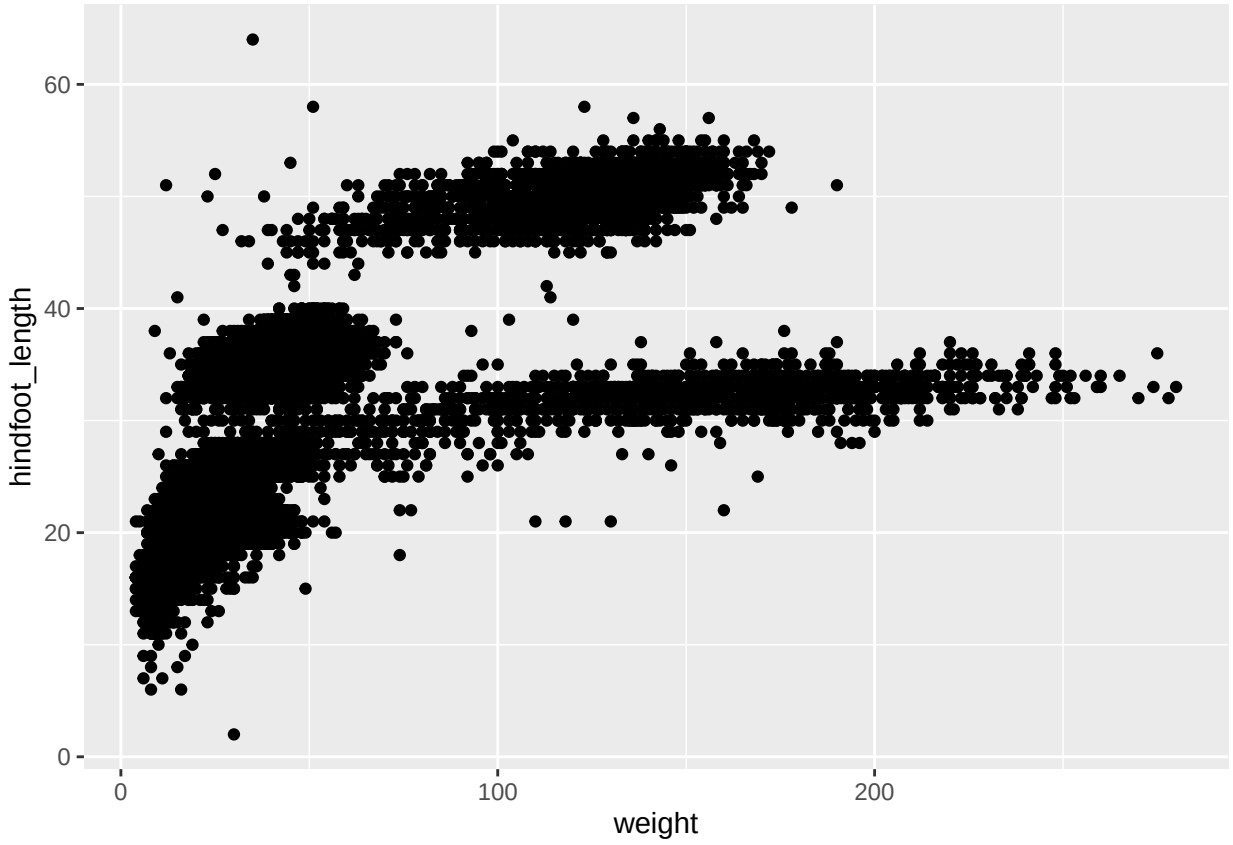
Cleaning tip:

```
surveys_complete <- surveys %>%
  filter(!is.na(weight),           # remove missing weight
         !is.na(hindfoot_length),  # remove missing hindfoot_length
         !is.na(sex))                # remove missing sex
```

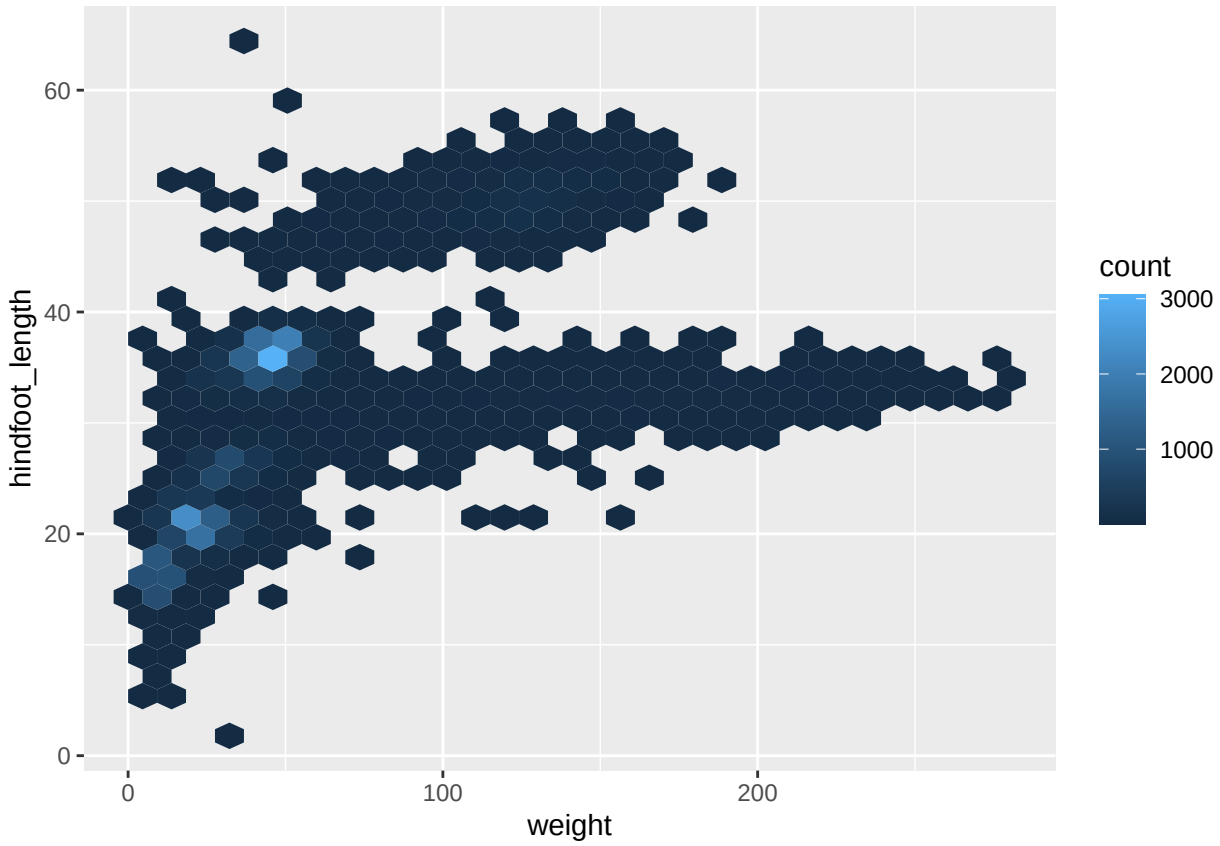
Plotting

```
library(hexbin)
```

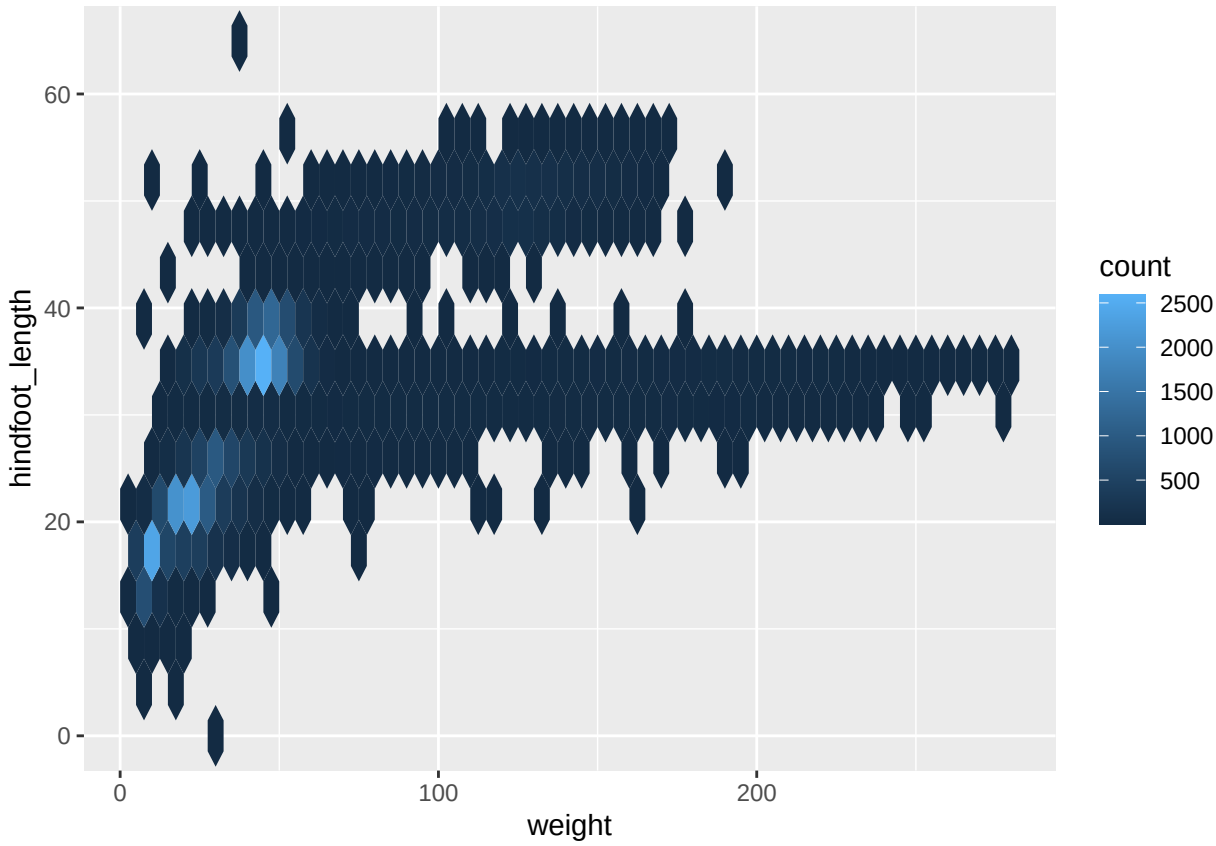
```
surveys_plot <- ggplot(data = surveys_complete, mapping = aes(x = weight, y =
↪  hindfoot_length))

surveys_plot +
  geom_point()
```




```
surveys_plot +
  geom_hex()
```

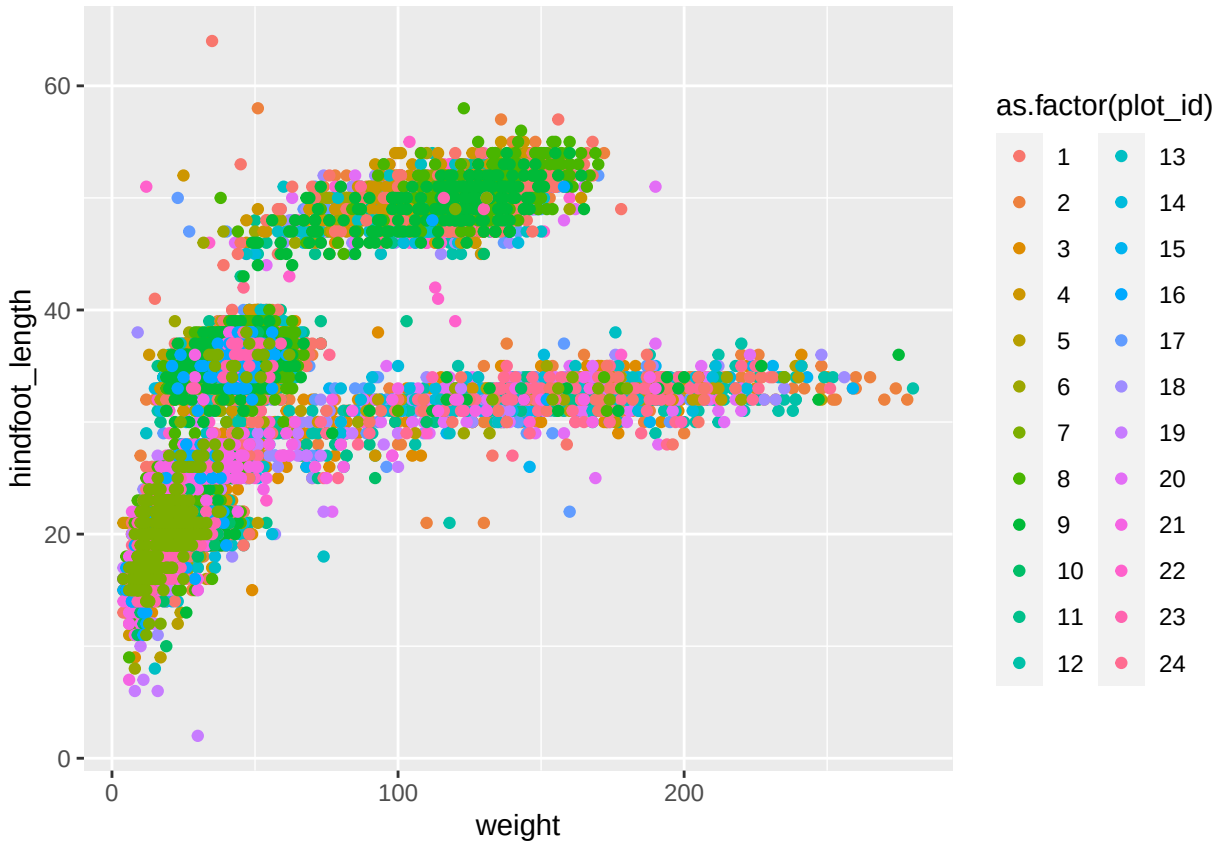



```
surveys_plot +
  geom_hex(binwidth = 5) # oogh, that messes with the proportions
```

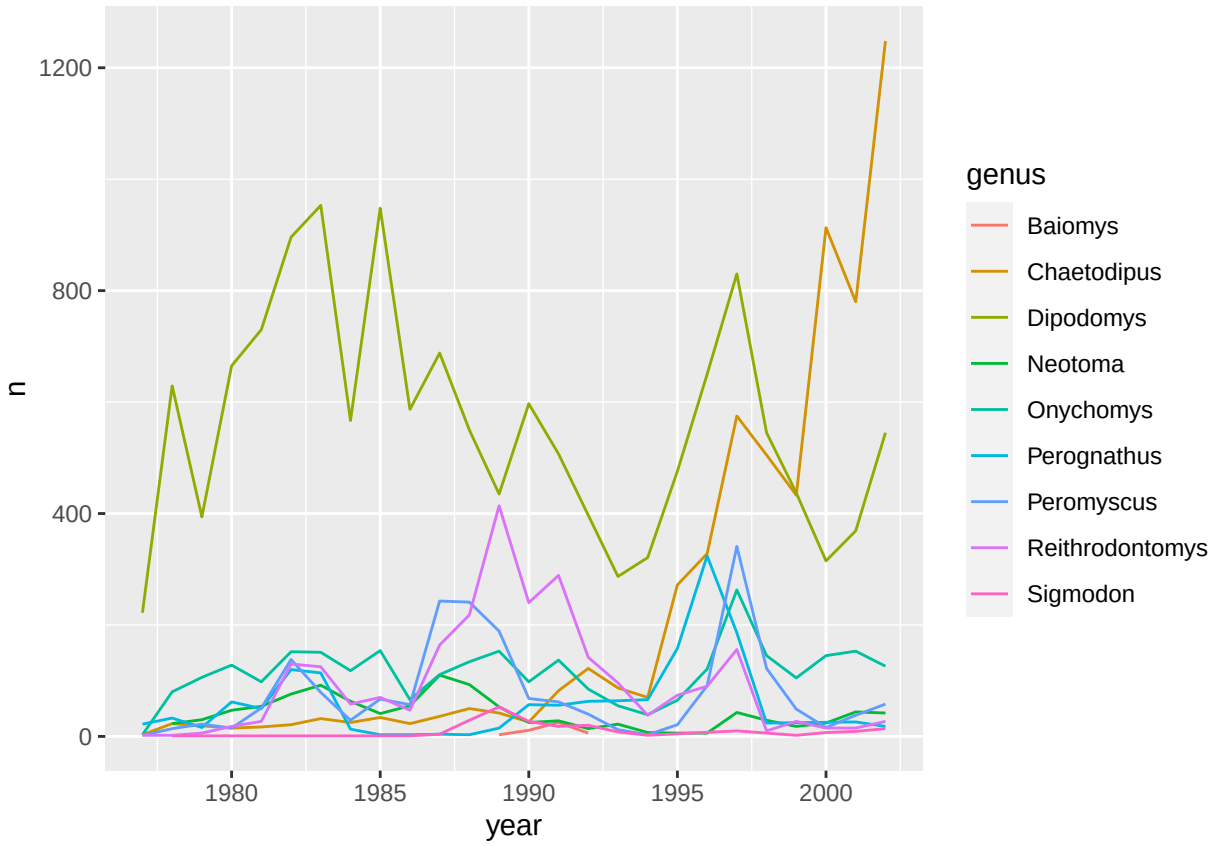



```
surveys_plot +
  geom_hex(bins = 100) + # Maybe too much but tbh strikes a nice balance between points
  ↪  and hexes
  ggtitle("100 Bins Hexbin Test") +
  theme_bw()
```

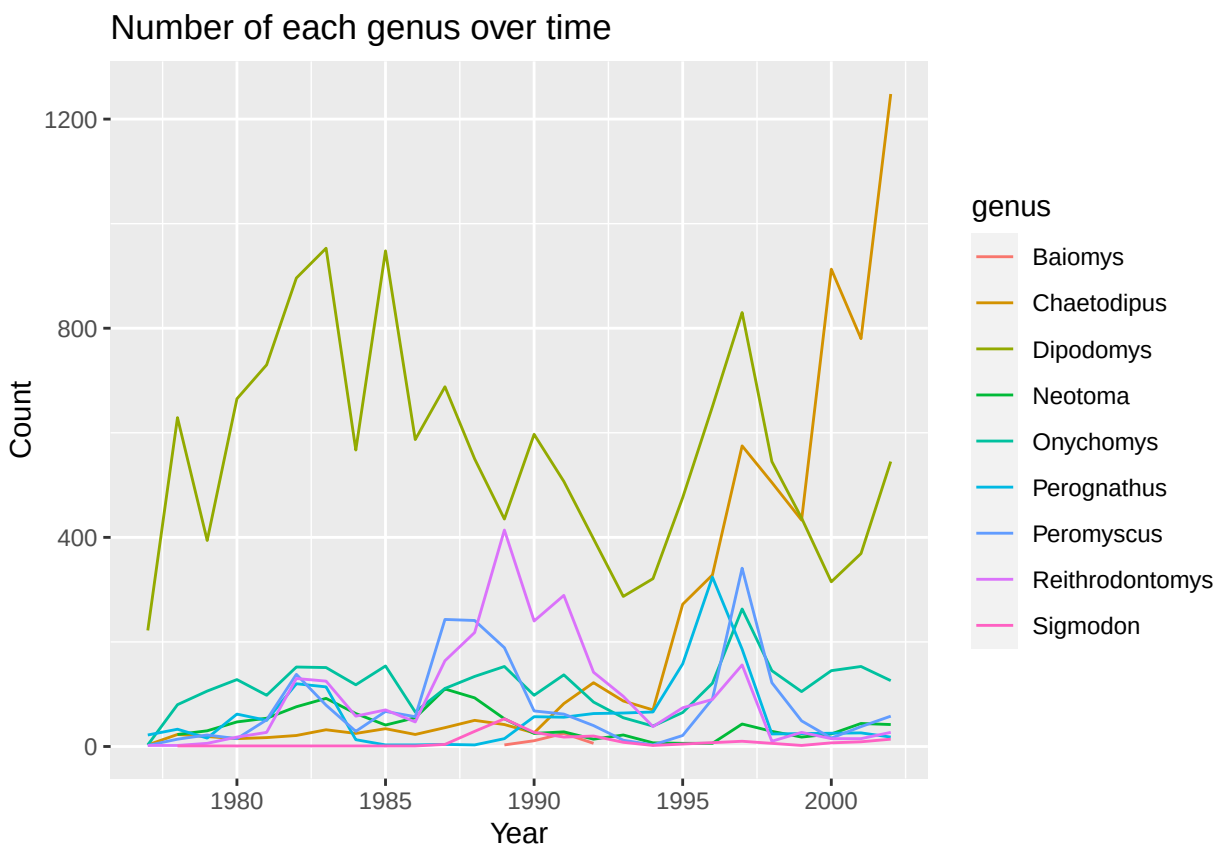

```
ggplot(data = surveys_complete,
                      mapping = aes(x = weight, y = hindfoot_length, color =
                       ↪  as.factor(plot_id))) +
  geom_point()
```




```
# Plot multiple groups on a time series
yearly_counts <- surveys_complete %>%
  count(year, genus) # speedy group_by/summarize for counting
ggplot(data = yearly_counts, aes(x = year, y = n, group = genus, color = genus)) +
    geom_line() # Just adding color = genus would have the same result generally
```

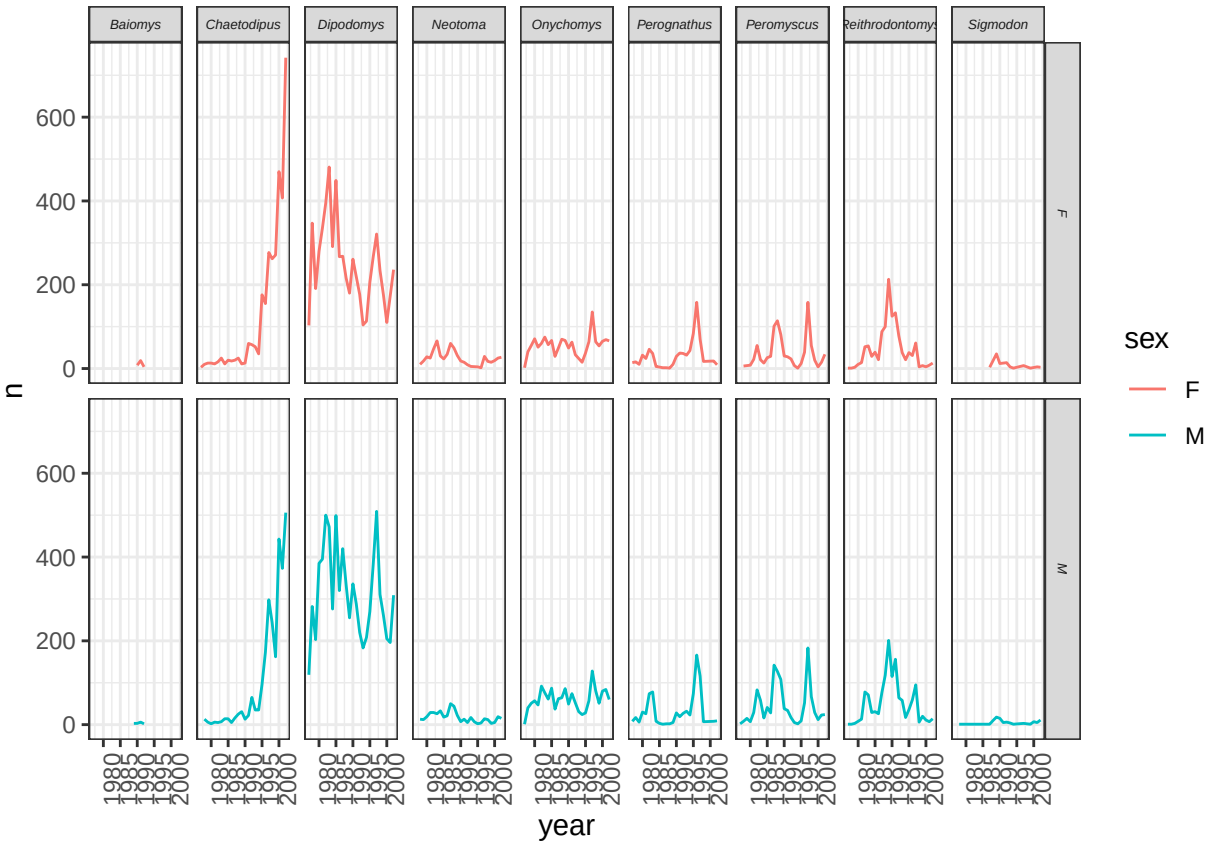



```
# Deranged pipe series
surveys_complete %>%
    count(year, genus) %>%
    ggplot(mapping = aes(x = year, y = n, color = genus)) +
    geom_line() +
    labs(title = "Number of each genus over time", x = "Year", y = "Count")
```



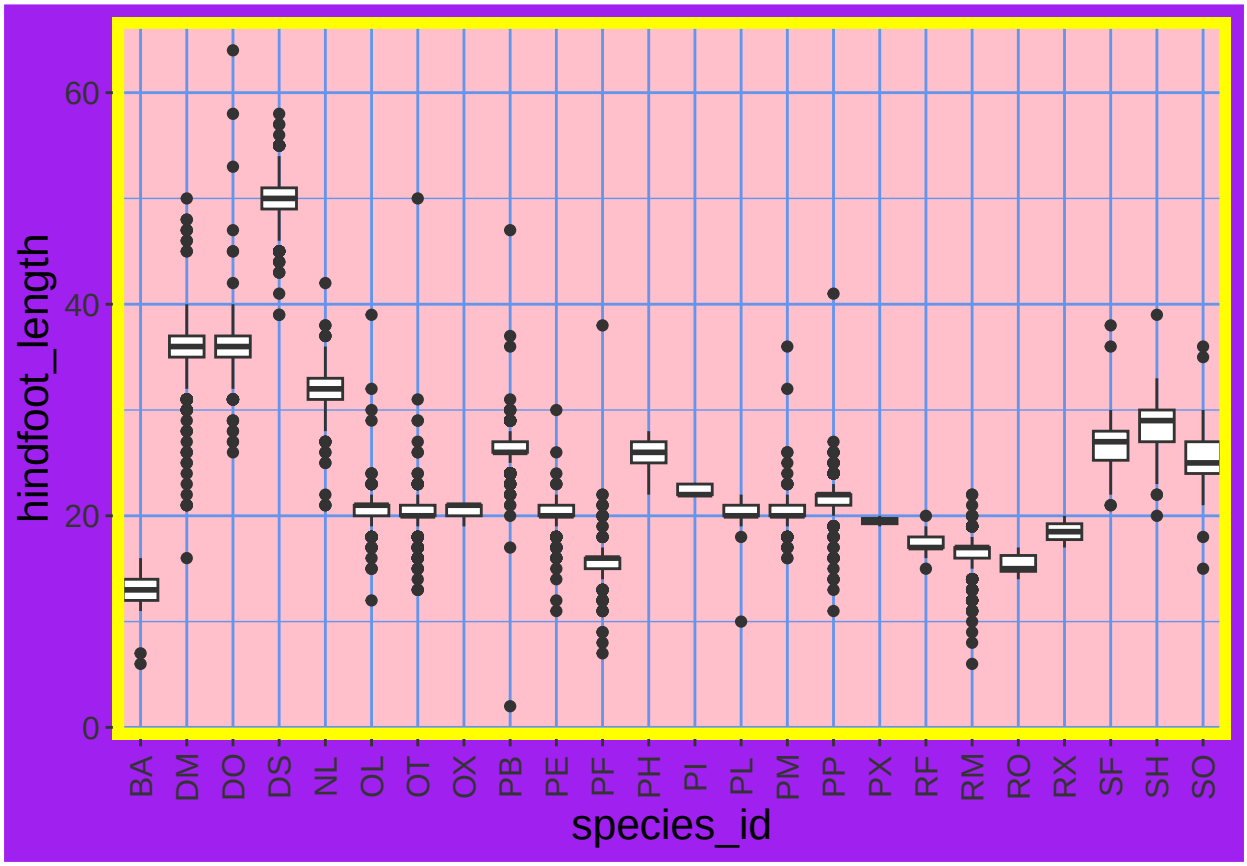
Faceting

```
yearly_sex_counts <- surveys_complete %>%
                     count(year, genus, sex)
ggplot(data = yearly_sex_counts,
       mapping = aes(x = year, y = n, color = sex)) +
  geom_line() +
  facet_grid(rows = vars(sex), cols =  vars(genus)) +
  theme_bw() + # This has to go before additional theme() calls or it'll override them
  theme(axis.text.x = element_text(angle = 90),
        strip.text = element_text(face = "italic", size = 5)) # Make genus names (strip
        ↪  is the facet title) italicized
```



Saving a theme

```
grey_theme <- theme(axis.text.x = element_text(colour="grey20", size = 12,
                                               angle = 90, hjust = 0.5,
                                               vjust = 0.5),
                    axis.text.y = element_text(colour = "grey20", size = 12),
                    text=element_text(size = 16),
                    panel.background = element_rect(fill = "pink"),
                    plot.background = element_rect(fill = "purple"),
                    panel.border = element_rect(fill = NA, color = "yellow", linewidth =
                    ↪  4),
                    panel.grid = element_line(color = "cornflowerblue"))

ggplot(surveys_complete, aes(x = species_id, y = hindfoot_length)) +
    geom_boxplot() +
    grey_theme # This is aesthetically pretty bad, but a useful guide.
```



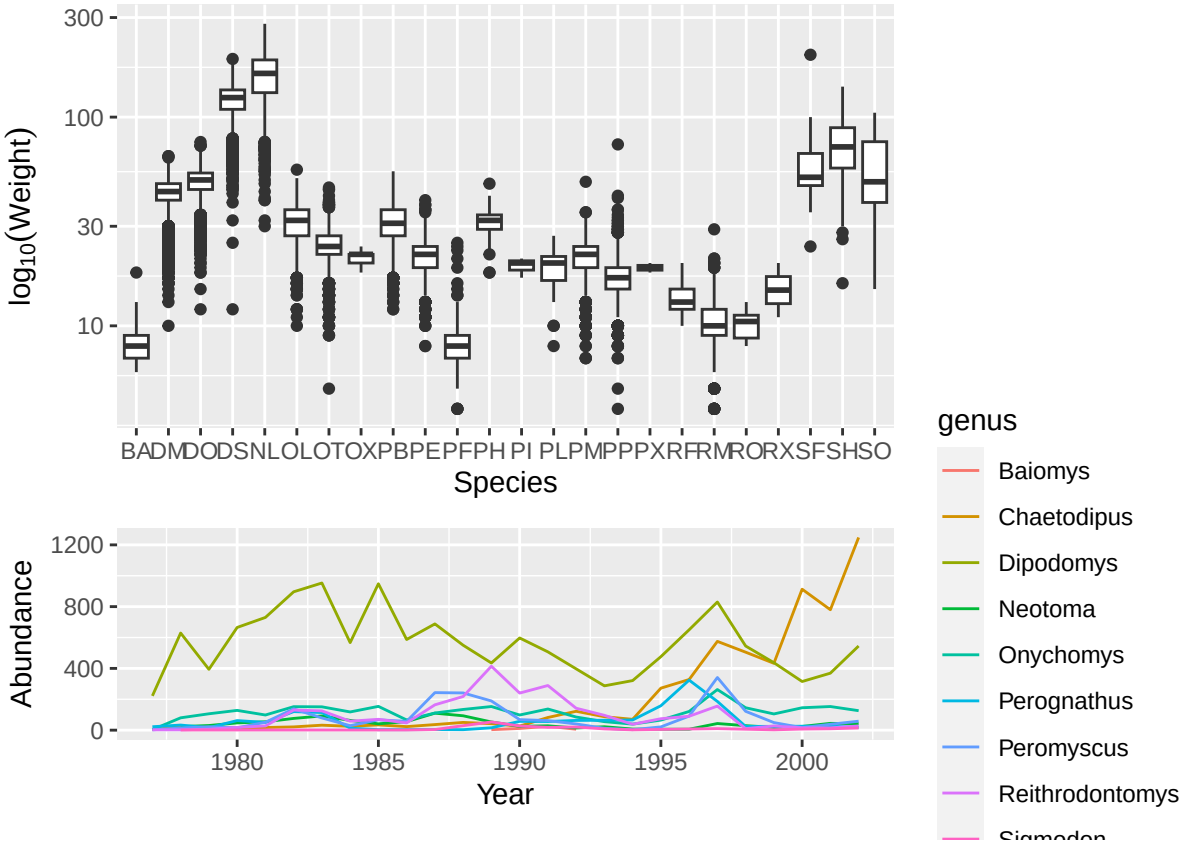
Patchwork

This package lets us put several plots in one figure. Here are some more examples

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.1
```

```
plot_weight <- ggplot(data = surveys_complete, aes(x = species_id, y = weight)) +
  geom_boxplot() +
  labs(x = "Species", y = expression(log[10](Weight))) +
  scale_y_log10()

plot_count <- ggplot(data = yearly_counts, aes(x = year, y = n, color = genus)) +
  geom_line() +
  labs(x = "Year", y = "Abundance")

plot_weight / plot_count + plot_layout(heights = c(4, 2))
```

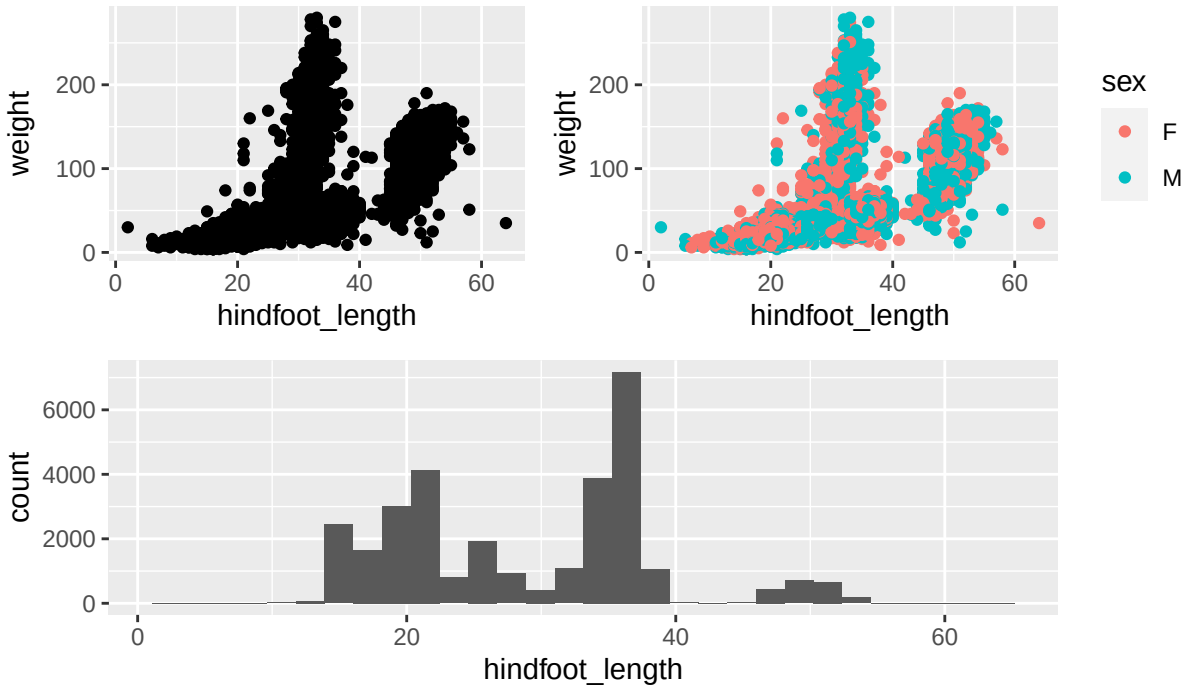



```
scatter1 <- ggplot(data = surveys_complete, aes(x = hindfoot_length, y = weight)) +
  geom_point()
scatter2 <- ggplot(data = surveys_complete, aes(x = hindfoot_length, y = weight, color =
↪  sex)) +
  geom_point()
histogram3 <- ggplot(data = surveys_complete, aes(x = hindfoot_length)) +
  geom_histogram()

plot_combined <- (scatter1 | scatter2) / histogram3 # patchwork times

plot_combined2 <- plot_combined + plot_annotation(
  title = 'The surprising truth about Surveys',
  subtitle = 'These 3 plots will reveal yet-untold secrets about our beloved data-set',
  caption = 'Disclaimer: None of these plots are insightful'
)

plot_combined2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

The surprising truth about Surveys

These 3 plots will reveal yet−untold secrets about our beloved data−set



Disclaimer: None of these plots are insightful

```
ggsave("~/Dropbox/Coding/Summer2024/RTutorials/DataCarpentry/Figures/plot_combined.png",
↪  plot_combined2, width = 10, dpi = 300)
```

```
## Saving 10 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

How to query larger online databases