Tidy data and visualisation with R

Richard White

27 Sept 2022

Busch
Lab

Topics

- Data manipulation
    - Introduction to the tidyverse
    - loading data (readr)
    - tidy data (tidyr)
    - selecting, filtering and creating new columns (dplyr)
- ggplot2
    - geoms
    - aesthetics
    - shapes and colours
    - facets
    - themes

Preliminaries

- Code is shown in a different font with a grey background
  e.g. `head(data_frame)`
- lines that start with a # are comments
  e.g.

  ```
  # this shows the top 6 lines of a data frame
  head(data_frame)
  ```

- The pipe operator %>%
  - Provided by the `magrittr` package
  - Equivalent of the unix pipe |
    `cut -f1,2,4 data.txt | head`

    ```
    # select 3 columns and see the top lines
    select(data, c(1,2,4)) %>% head()
    ```

  - Allows sending the results of one function into another
  - Since v4.1.0, R has had its own version |>
    Has different functionality (see here)

Nested functions are hard to read

```
eat(slice(bake(put(pour(mix(ingredients), into = baking_mould),
    into = oven), time = 30), pieces = 6), 1)
```

Even if you format the code nicely

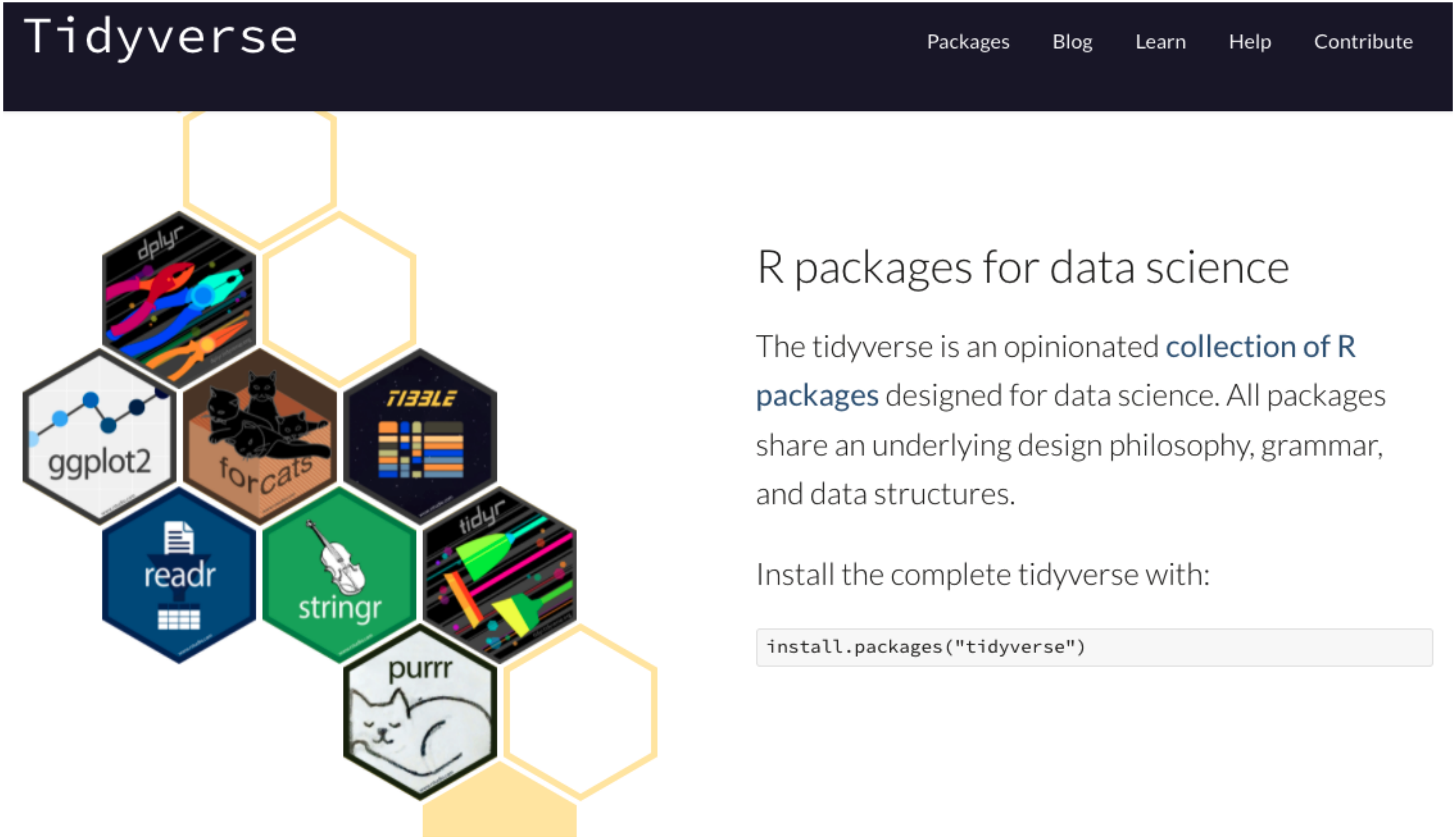
```
eat(
  slice(
    bake(
      put(
        pour(
          mix(ingredients),
          into = baking_mould),
        into = oven),
      time = 30),
    pieces = 6),
  1)
```

The pipe makes code easier to read

```
ingredients %>%
  mix() %>%
  pour(into = baking_mould) %>%
  put(into = oven) %>%
  bake(time = 30) %>%
  slice(pieces = 6) %>%
  eat(1)
```

from @dmi3k on Twitter

Tidyverse



R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

tidyverse.org

Tidyverse packages

readr

dplyr

tidyr

ggplot2

readr

- `read_tsv()`
  - reads in tab-delimited data and tries to guess the data type of each column
  - character, integer, numeric, logical
- `read_csv()`
  - same for comma-separated files

readr.tidyverse.org

read_tsv

```r
data_file <- 'assets/test-data.tsv'
mock_data <- read_tsv(data_file)
```

```
## Rows: 2000 Columns: 7
## ── Column specification ────────────────────────────────────────
## Delimiter: "\t"
## chr (2): Gene, Chr
## dbl (5): pval, adjp, Start, End, Strand
##
## 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
```

```r
head(mock_data, 3)
```

```
## # A tibble: 3 × 7
##   Gene                pval  adjp Chr    Start    End Strand
##   <chr>              <dbl> <dbl> <chr>  <dbl>  <dbl>  <dbl>
## 1 ENSDARG00000000001 0.771     1 14      5352   5542     -1
## 2 ENSDARG00000000002 0.643     1 19    261334 261907     -1
## 3 ENSDARG00000000003 0.687     1 13    604674 605540      1
```

Column types

`readr` tries to correctly guess the data types
It does this by randomly sampling rows from the dataset (1000 rows by default).
If you find that `readr` is not correctly guessing the data types there are two solutions.

1. increase `guess_max`

2. define the column types explicitly

```
read_tsv(data_file, guess_max = 5000)
```

```
read_tsv(data_file, col_types = "cddciif")
```

```
read_tsv(data_file,
  col_types = cols(Chr = col_character(),
                   Strand = col_factor(levels = c("1", "-1"))))
```

Factors

- Factors are used to work with categorical variables
- variables that have a fixed and known set of possible values

```
(genotype <- rep(c('wt', 'het', 'hom'), each = 3))
```

```
## [1] "wt"  "wt"  "wt"  "het" "het" "het" "hom" "hom" "hom"
```

```
factor(genotype)
```

```
## [1] wt  wt  wt  het het het hom hom hom
## Levels: het hom wt
```

```
factor(genotype,
       levels = c("wt", "het", "hom"))
```

```
## [1] wt  wt  wt  het het het hom hom hom
## Levels: wt het hom
```

Factors

```r
factor(c(genotype, 'hot'),
       levels = c("wt", "het", "hom"))
```

```
##  [1] wt   wt   wt   het  het  het  hom  hom  hom  <NA>
## Levels: wt het hom
```

forcats.tidyverse.org

Tidy data

1. Every column is a variable.
2. Every row is an observation.
3. Every cell is a single value.

name	gene	sample	count	normalised_count
slc35a5	ENSDARG00000000001	sample_1	35	36.28
ccdc80	ENSDARG00000000002	sample_1	75	75.51
slc35a5	ENSDARG00000000001	sample_2	30	33.43
ccdc80	ENSDARG00000000002	sample_2	115	113.15

Tidy data

Data is often not in this format. Often for very good reasons.
One common arrangement for RNA-seq data is like this:

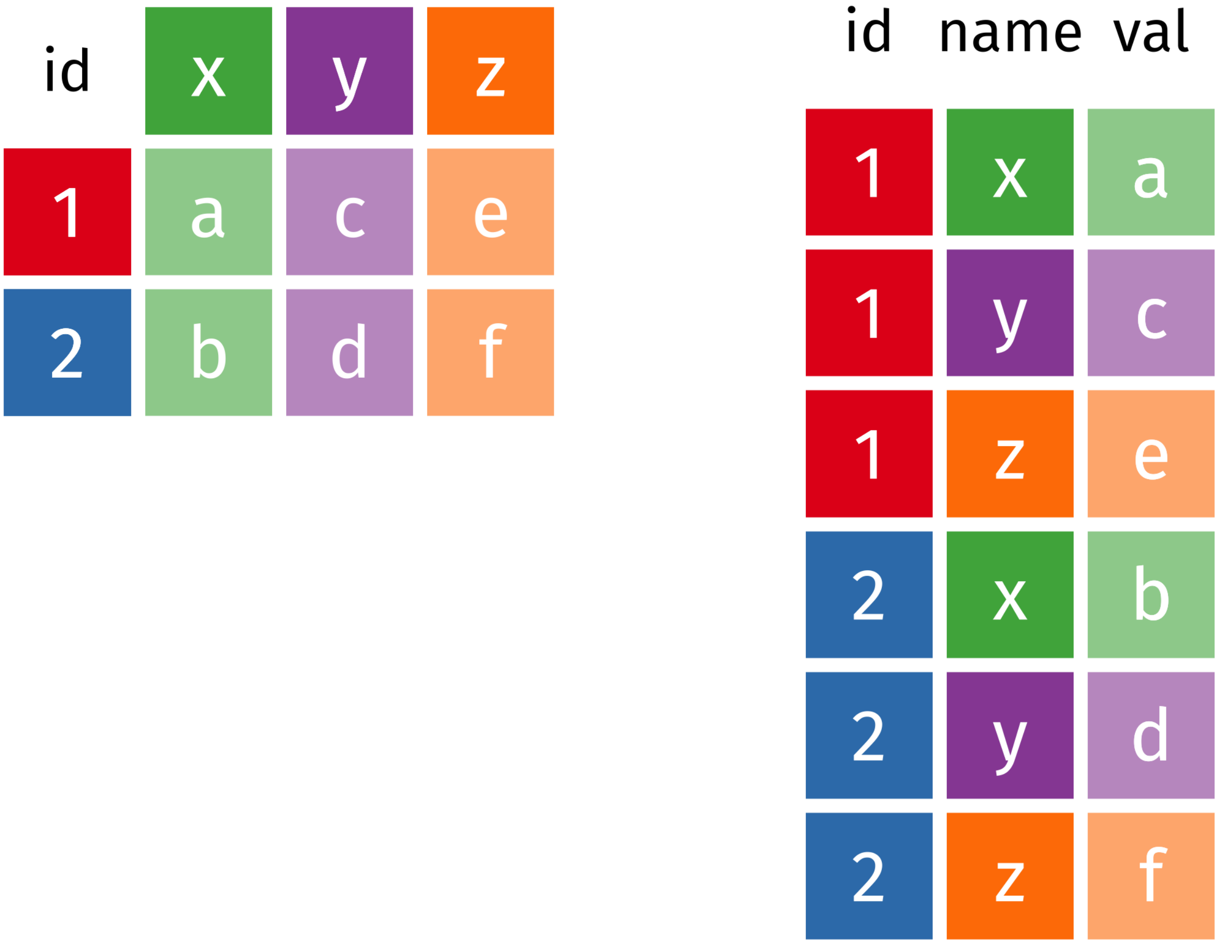
name	description	s1_count	s2_count	s1_norm_count	s2_norm_count
slc35a5	ENSDARG00000000001	35	30	36.28	33.43
ccdc80	ENSDARG00000000002	75	115	75.51	113.15
nrf1	ENSDARG00000000003	300	283	305.95	281.17

For example the Amphetamine dataset has 32520 rows (Genes) and 55 columns (18 Mb)
In tidy format it would be 715440 rows and 14 columns (130 Mb)

For plotting with `ggplot`, tidy data is required.
The `tidyr` package is designed to do this kind of rearrangement.

tidyr.tidyverse.org

Wide vs long data



id	x	y	z
1	a	c	e
2	b	d	f

id	name	val
1	x	a
1	y	c
1	z	e
2	x	b
2	y	d
2	z	f

pivot_longer()

```
df
```

```
## # A tibble: 2 × 4
##      id x     y     z    
##   <dbl> <chr> <chr> <chr>
## 1     1 a     c     e    
## 2     2 b     d     f    
```

```
pivot_longer(df, cols = c(x, y, z),
             names_to = 'sample',
             values_to = 'count')
```

```
## # A tibble: 6 × 3
##      id sample count
##   <dbl> <chr>  <chr>
## 1     1 x      a    
## 2     1 y      c    
## 3     1 z      e    
## 4     2 x      b    
## 5     2 y      d    
## 6     2 z      f    
```

pivot_wider()

```
df_long
```

```
## # A tibble: 6 × 3
##      id sample count
##   <dbl> <chr>  <chr>
## 1     1 x      a
## 2     1 y      b
## 3     1 z      c
## 4     2 x      d
## 5     2 y      e
## 6     2 z      f
```

```
pivot_wider(df_long, id_cols = id,
            names_from = 'sample',
            values_from = 'count')
```

```
## # A tibble: 2 × 4
##      id x     y     z
##   <dbl> <chr> <chr> <chr>
## 1     1 a     b     c
## 2     2 d     e     f
```

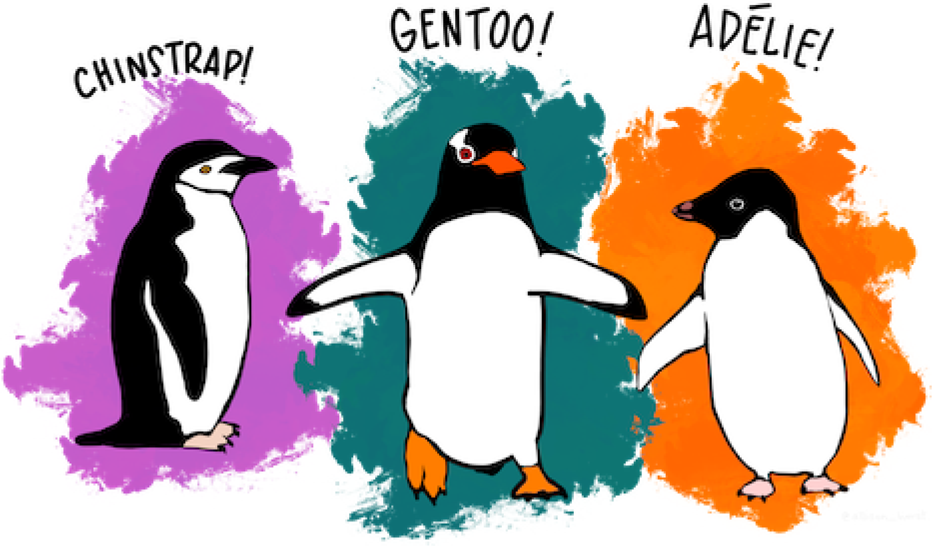
Data manipulation

- `select()` - pick variables
- `filter()` - pick rows
- `arrange()` - sort rows
- `mutate()` - create new variables
- `summarise()` - reduce multiple values down to a single summary (mean, min, max etc.)

dplyr.tidyverse.org

Palmer Penguins

This dataset is available in the `palmerpenguins` package.
It contains data for 3 different species of penguins (344 individuals), collected from 3 islands in the Palmer Archipelago, Antarctica.



Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

https://allisonhorst.github.io/palmerpenguins

Palmer Penguins

```r
library(palmerpenguins)
head(penguins)
```

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	NA	NA	NA	NA	NA	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007
Adelie	Torgersen	39.3	20.6	190	3650	male	2007

`select()`

- Choose variables from a table
    - use column names explicitly: `select(data, GeneID)`
    - or positions: `select(data, c(1,5,9))`
    - column names can be used as if they are positions
      e.g. `select(data, GeneID:Name)`
    - or search functions
      starts_with() `select(data, starts_with('Ctrl'))`
      ends_with() `select(data, ends_with('count'))`
      contains() `select(data, contains('3dpf'))`
      matches() `select(data, matches('3dpf.*count'))`

select()

```
select(penguins, species, year, body_mass_g) %>% head(4)
```

```
## # A tibble: 4 × 3
##   species  year body_mass_g
##   <fct>   <int>       <int>
## 1 Adelie   2007        3750
## 2 Adelie   2007        3800
## 3 Adelie   2007        3250
## 4 Adelie   2007          NA
```

```
select(penguins, -species, -flipper_length_mm) %>% head(4)
```

```
## # A tibble: 4 × 6
##   island    bill_length_mm bill_depth_mm body_mass_g sex     year
##   <fct>              <dbl>         <dbl>       <int> <fct>  <int>
## 1 Torgersen           39.1          18.7        3750 male    2007
## 2 Torgersen           39.5          17.4        3800 female  2007
## 3 Torgersen           40.3          18          3250 female  2007
## 4 Torgersen           NA            NA            NA <NA>    2007
```

select()

```
select(penguins, starts_with('bill')) %>% head(4)
```

```
## # A tibble: 4 × 2
##   bill_length_mm bill_depth_mm
##            <dbl>         <dbl>
## 1           39.1          18.7
## 2           39.5          17.4
## 3           40.3          18  
## 4           NA            NA  
```

```
select(penguins, ends_with('mm')) %>% head(4)
```

```
## # A tibble: 4 × 3
##   bill_length_mm bill_depth_mm flipper_length_mm
##            <dbl>         <dbl>             <int>
## 1           39.1          18.7               181
## 2           39.5          17.4               186
## 3           40.3          18                 195
## 4           NA            NA                  NA
```

select()

```
select(penguins, species, contains('length')) %>% head(4)
```

```
## # A tibble: 4 × 3
##   species bill_length_mm flipper_length_mm
##   <fct>            <dbl>             <int>
## 1 Adelie            39.1               181
## 2 Adelie            39.5               186
## 3 Adelie            40.3               195
## 4 Adelie            NA                  NA
```

```
select(penguins, matches('_[mg]')) %>% head(4)
```

```
## # A tibble: 4 × 4
##   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##            <dbl>         <dbl>             <int>       <int>
## 1           39.1          18.7               181        3750
## 2           39.5          17.4               186        3800
## 3           40.3          18                 195        3250
## 4           NA            NA                  NA          NA
```

`filter()`

- Choose rows where conditions are true
  If there are mulitple conditions, they must all be true
- check equality with == (`!=` for not equal to)
- Also <, >, <= `(≤)`, >= `(≥)`
- combine operators with `&` (AND), | (OR), `!` (NOT)

`filter()` equality

```r
filter(penguins, species == "Gentoo")
```

```
## # A tibble: 124 × 8
##    species island bill_length_mm bill_depth_mm flipper_len…¹ body_…² sex
##    <fct>   <fct>           <dbl>         <dbl>         <int>   <int> <fct>
##  1 Gentoo  Biscoe           46.1          13.2           211    4500 fema…
##  2 Gentoo  Biscoe           50            16.3           230    5700 male
##  3 Gentoo  Biscoe           48.7          14.1           210    4450 fema…
##  4 Gentoo  Biscoe           50            15.2           218    5700 male
##  5 Gentoo  Biscoe           47.6          14.5           215    5400 male
##  6 Gentoo  Biscoe           46.5          13.5           210    4550 fema…
##  7 Gentoo  Biscoe           45.4          14.6           211    4800 fema…
##  8 Gentoo  Biscoe           46.7          15.3           219    5200 male
##  9 Gentoo  Biscoe           43.3          13.4           209    4400 fema…
## 10 Gentoo  Biscoe           46.8          15.4           215    5150 male
## # … with 114 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

`filter()` greater than

```
filter(penguins, bill_length_mm  > 50)
```

```
## # A tibble: 52 × 8
##    species island bill_length_mm bill_depth_mm flipper_len…¹ body_…² sex
##    <fct>   <fct>           <dbl>         <dbl>         <int>   <int> <fct>
##  1 Gentoo  Biscoe           50.2          14.3           218    5700 male
##  2 Gentoo  Biscoe           59.6          17             230    6050 male
##  3 Gentoo  Biscoe           50.5          15.9           222    5550 male
##  4 Gentoo  Biscoe           50.5          15.9           225    5400 male
##  5 Gentoo  Biscoe           50.1          15             225    5000 male
##  6 Gentoo  Biscoe           50.4          15.3           224    5550 male
##  7 Gentoo  Biscoe           54.3          15.7           231    5650 male
##  8 Gentoo  Biscoe           50.7          15             223    5550 male
##  9 Gentoo  Biscoe           51.1          16.3           220    6000 male
## 10 Gentoo  Biscoe           52.5          15.6           221    5450 male
## # … with 42 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

`filter()` combine conditions

```
filter(penguins, bill_length_mm  > 50, flipper_length_mm >= 230)
```

```
## # A tibble: 5 × 8
##   species island bill_length_mm bill_depth_mm flipper_leng…¹ body_…² sex
##   <fct>   <fct>           <dbl>         <dbl>          <int>   <int> <fct
## 1 Gentoo  Biscoe           59.6          17              230    6050 male
## 2 Gentoo  Biscoe           54.3          15.7            231    5650 male
## 3 Gentoo  Biscoe           52.1          17              230    5550 male
## 4 Gentoo  Biscoe           51.5          16.3            230    5500 male
## 5 Gentoo  Biscoe           55.1          16              230    5850 male
## # … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

arrange()

```
arrange(penguins, bill_length_mm) %>% head(4)
```

```
## # A tibble: 4 × 8
##   species island    bill_length_mm bill_depth_mm flipper_l…¹ body_…² sex
##   <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct>
## 1 Adelie  Dream               32.1          15.5         188    3050 fema…
## 2 Adelie  Dream               33.1          16.1         178    2900 fema…
## 3 Adelie  Torgersen           33.5          19           190    3600 fema…
## 4 Adelie  Dream               34            17.1         185    3400 fema…
## # … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

```
# reverse order
arrange(penguins, desc(bill_length_mm)) %>% head(4)
```

```
## # A tibble: 4 × 8
##   species   island bill_length_mm bill_depth_mm flipper_le…¹ body_…² sex
##   <fct>     <fct>           <dbl>         <dbl>        <int>   <int> <fct>
## 1 Gentoo    Biscoe           59.6          17            230    6050 male
## 2 Chinstrap Dream            58            17.8          181    3700 fema…
## 3 Gentoo    Biscoe           55.9          17            228    5600 male
## 4 Chinstrap Dream            55.8          19.8          207    4000 male
## # … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

arrange()

```
arrange(penguins, species) %>% head(3)
```

```
## # A tibble: 3 × 8
##   species island    bill_length_mm bill_depth_mm flipper_l…¹ body_…² sex
##   <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7         181    3750 male
## 2 Adelie  Torgersen           39.5          17.4         186    3800 fema…
## 3 Adelie  Torgersen           40.3          18           195    3250 fema…
## # … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

```
# data frames arranged by factor are
# sorted according to the levels of the factor
arrange(penguins, island) %>% head(3)
```

```
## # A tibble: 3 × 8
##   species island bill_length_mm bill_depth_mm flipper_leng…¹ body_…² sex
##   <fct>   <fct>           <dbl>         <dbl>          <int>   <int> <fct>
## 1 Adelie  Dream            39.5          16.7            178    3250 fema…
## 2 Adelie  Dream            37.2          18.1            178    3900 male
## 3 Adelie  Dream            39.5          17.8            188    3300 fema…
## # … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

`mutate()`

- mutate makes new columns
- mutate keeps the original column
- transmute keeps only the new variables you create

mutate()

```
mutate(penguins,
       mass_by_bill_length = body_mass_g/bill_length_mm) %>%
  head()
```

```
## # A tibble: 6 × 5
##   species island    body_mass_g bill_length_mm mass_by_bill_length
##   <fct>   <fct>           <int>          <dbl>               <dbl>
## 1 Adelie  Torgersen        3750           39.1                95.9
## 2 Adelie  Torgersen        3800           39.5                96.2
## 3 Adelie  Torgersen        3250           40.3                80.6
## 4 Adelie  Torgersen          NA           NA                  NA  
## 5 Adelie  Torgersen        3450           36.7                94.0
## 6 Adelie  Torgersen        3650           39.3                92.9
```

mutate()

```
mutate(penguins,
       bill_depth_mm_transformed = case_when(
         species == "Adelie" ~ bill_depth_mm * 10,
         species == "Chinstrap" ~ bill_depth_mm / 10,
         TRUE ~ bill_depth_mm
      ))
```

```
## # A tibble: 6 × 4
##   species   island    bill_depth_mm bill_depth_mm_transformed
##   <fct>     <fct>             <dbl>                     <dbl>
## 1 Adelie    Torgersen          18.7                    187   
## 2 Adelie    Torgersen          17.4                    174   
## 3 Chinstrap Dream              17.9                      1.79
## 4 Chinstrap Dream              19.5                      1.95
## 5 Gentoo    Biscoe             13.2                     13.2 
## 6 Gentoo    Biscoe             16.3                     16.3 
```

summarise()

```r
summarise(penguins,
          min_weight = min(body_mass_g, na.rm = TRUE),
          mean_weight = mean(body_mass_g, na.rm = TRUE),
          max_weight = max(body_mass_g, na.rm = TRUE))
```

```
## # A tibble: 1 × 3
##   min_weight mean_weight max_weight
##        <int>       <dbl>      <int>
## 1       2700       4202.       6300
```

```r
group_by(penguins, species) %>%
summarise(min_weight = min(body_mass_g, na.rm = TRUE),
          mean_weight = mean(body_mass_g, na.rm = TRUE),
          max_weight = max(body_mass_g, na.rm = TRUE))
```

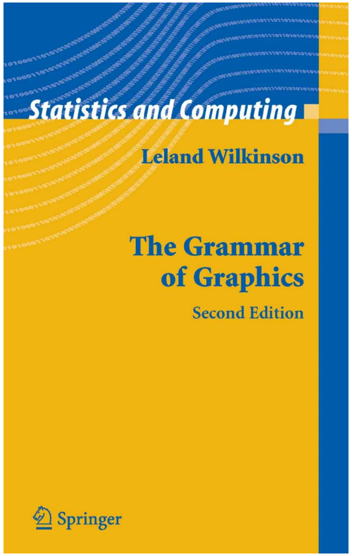
```
## # A tibble: 3 × 4
##   species   min_weight mean_weight max_weight
##   <fct>          <int>       <dbl>      <int>
## 1 Adelie          2850       3701.       4775
## 2 Chinstrap       2700       3733.       4800
## 3 Gentoo          3950       5076.       6300
```

Exercises

Open r-data-vis-exercises.pdf and do the **Tidy Data** exercises

ggplot2

- Grammar of Graphics
    - Leland Wilkinson (2005)
- Components of a plot
    1. data
    2. geom
        - How the data is represented e.g. points, lines, bars, text
    3. aesthetics
        - attributes of a plot that variables in the data are mapped to
        - x, y, colour, shape, length, size, linetype

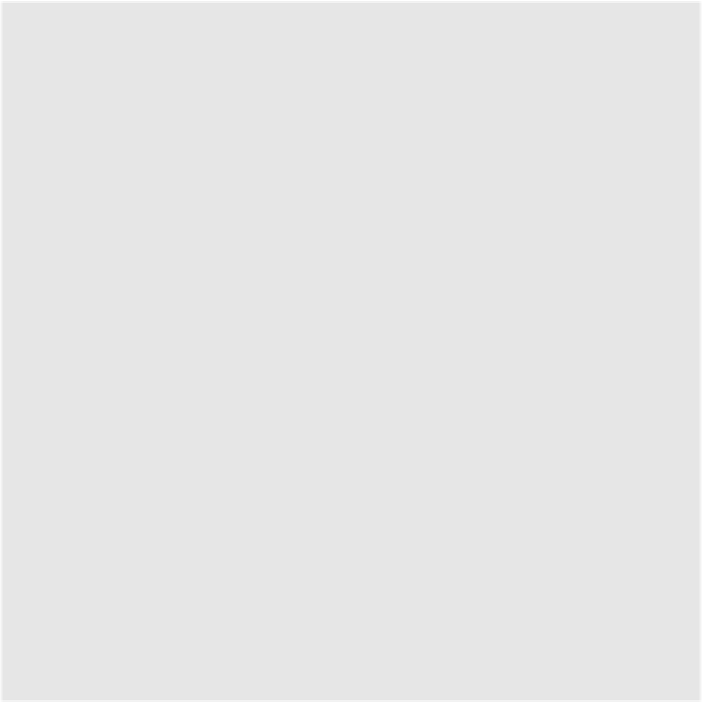


```
ggplot(data = <DATA>) + <GEOM_FUNCTION>(mapping =
aes(<MAPPINGS>))
```

ggplot2.tidyverse.org

Base plot

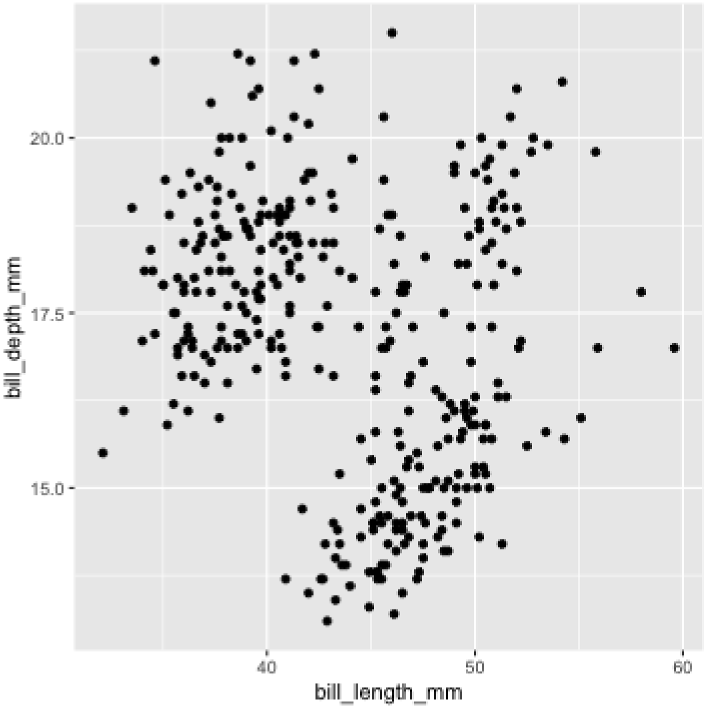
```
ggplot(data = penguins)
```



Scatterplot

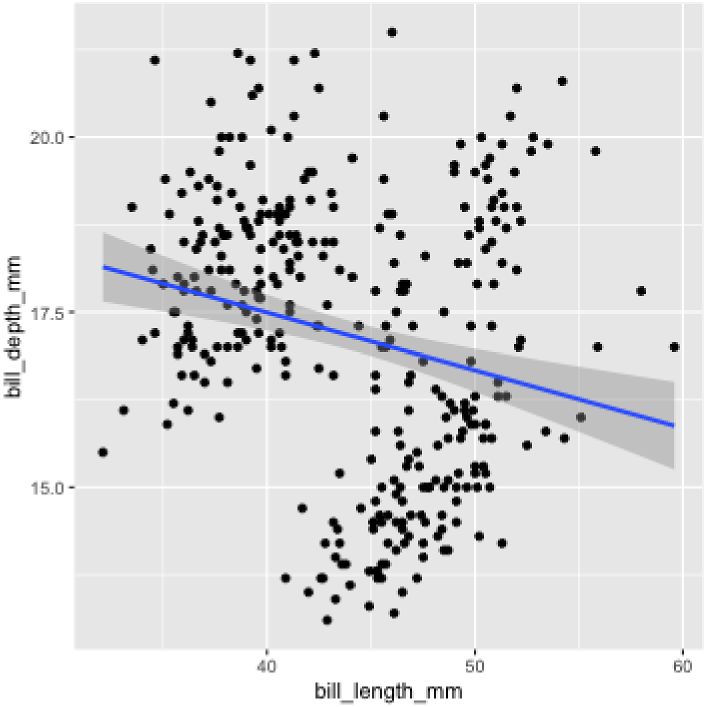
```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm))
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



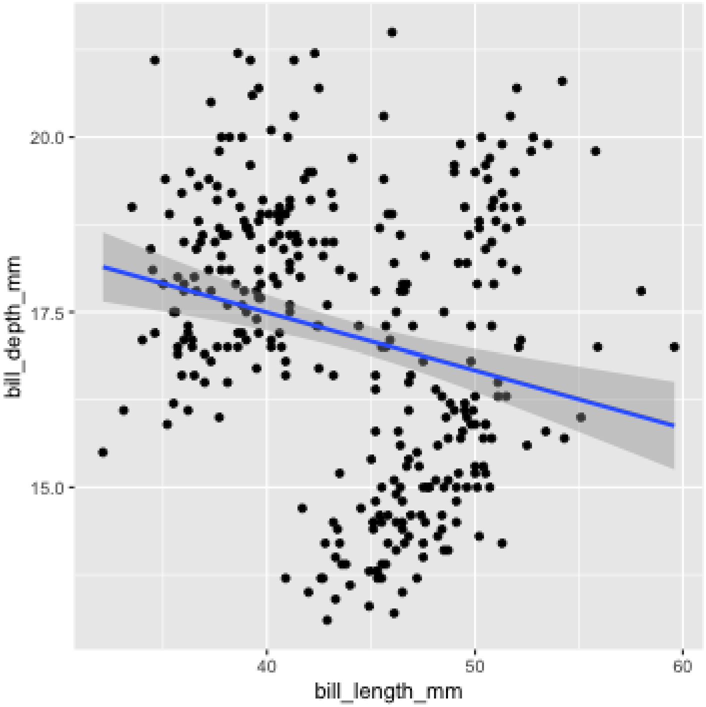
Add another geom

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm)) +
  geom_smooth(mapping = aes(x = bill_length_mm, y = bill_depth_mm),
              method = "lm",formula =  y ~ x)
```



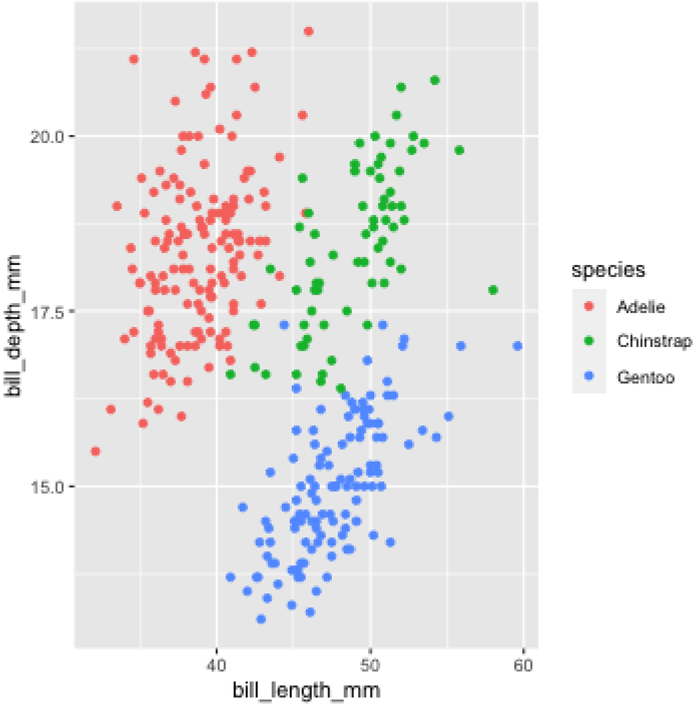
Set the aesthetics for the whole plot

```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm, y = bill_depth_mm)) +
  geom_point() +
  geom_smooth(method = "lm",formula =  y ~ x)
```



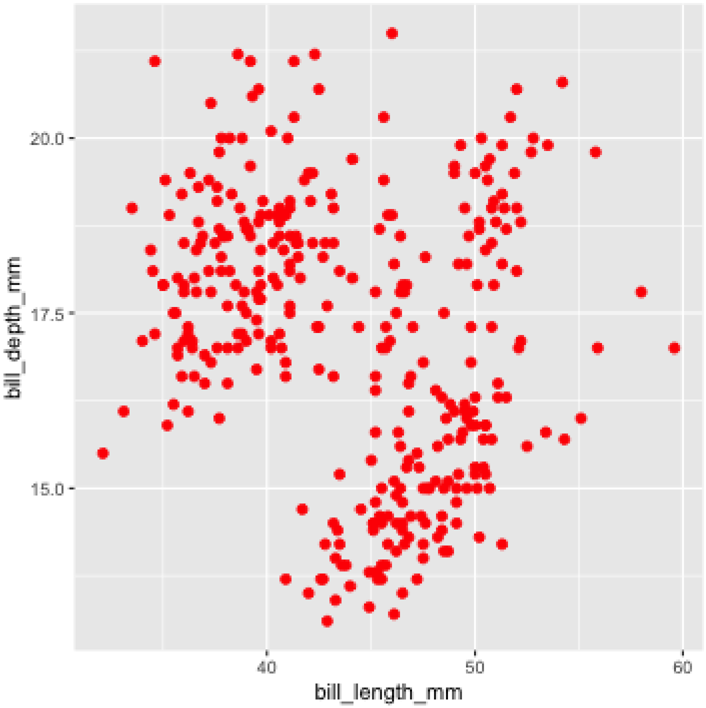
Map categorical variable to colour

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           colour = species))
```



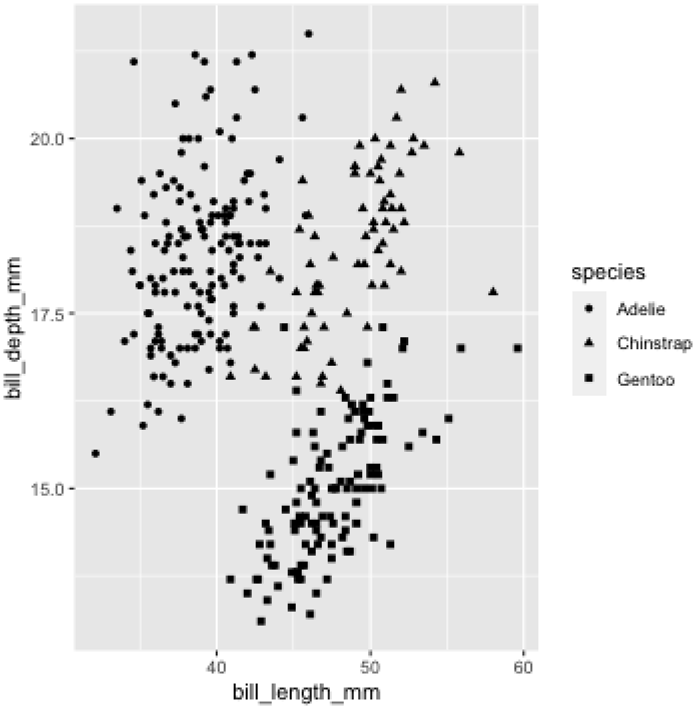
Setting arguments outside of `aes()`

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm),
             colour = 'red', size = 2)
```



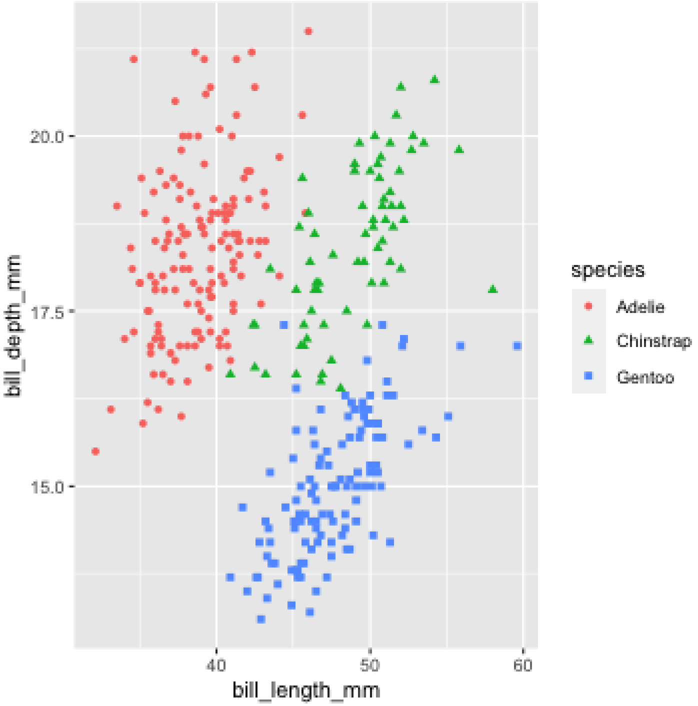
Map categorical variable to shape

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           shape = species))
```



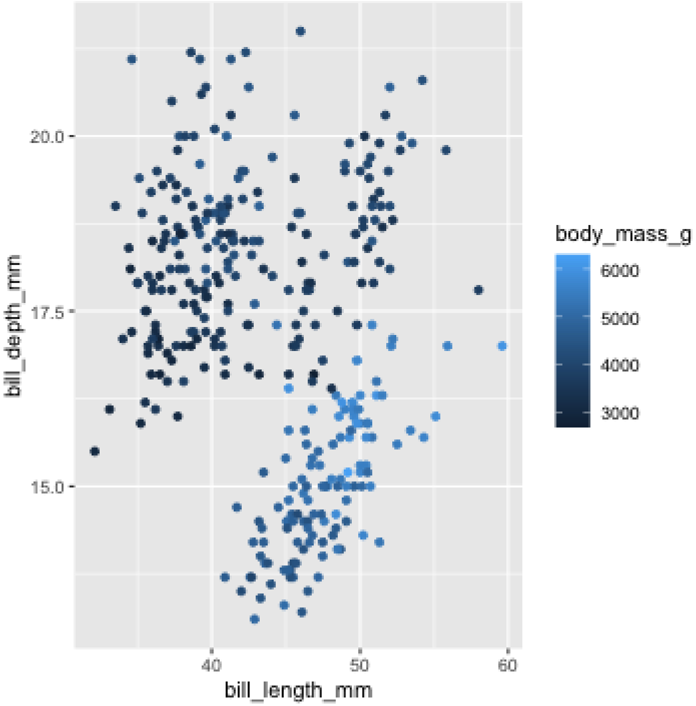
Map to colour and shape

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           colour = species, shape = species))
```



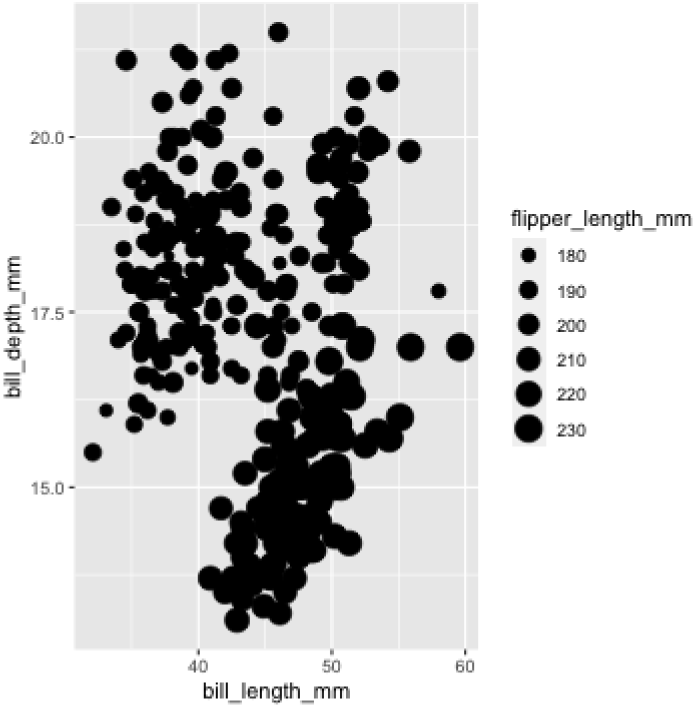
Map continuous variable to colour

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           colour = body_mass_g))
```



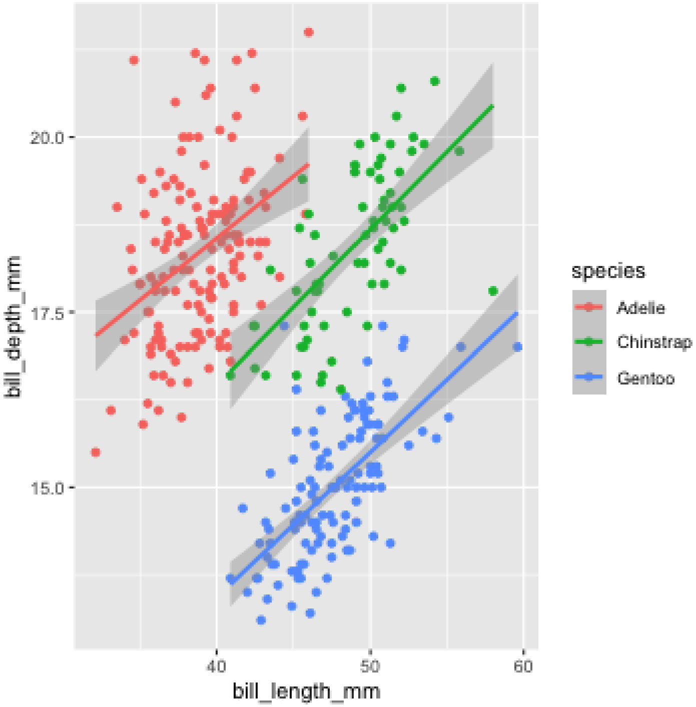
Map continuous variable to size

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                            size = flipper_length_mm))
```



Add extra geoms

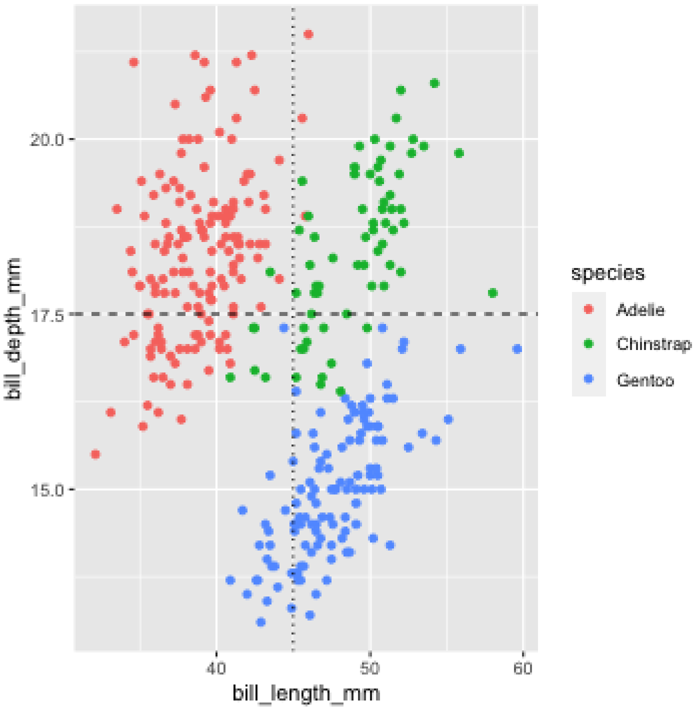
```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                     colour = species)) +
  geom_point() + geom_smooth(method = 'lm', formula = y ~ x)
```



ggplot2.tidyverse.org/reference

Add extra geoms

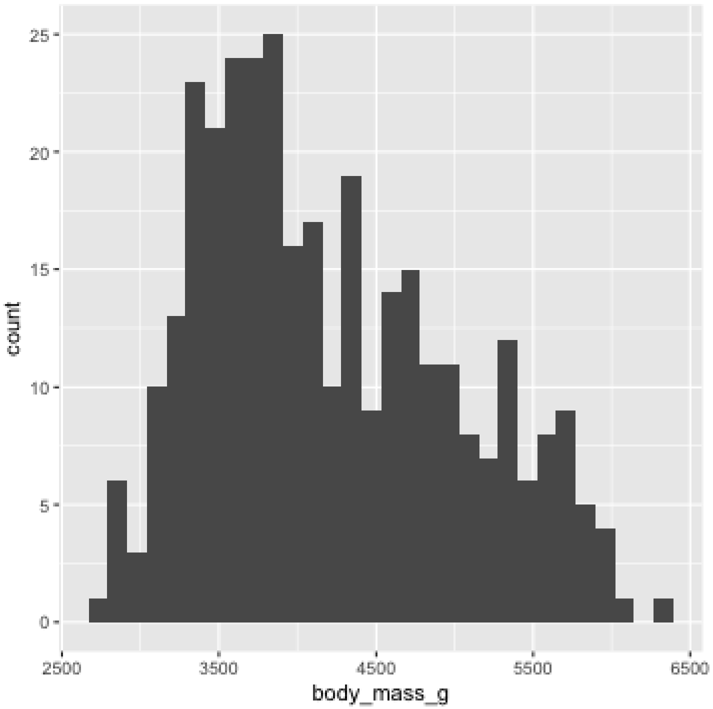
```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                     colour = species)) +
  geom_point() +
  geom_hline(yintercept = 17.5, linetype = "dashed") +
  geom_vline(xintercept = 45, linetype = "dotted")
```



geom_histogram

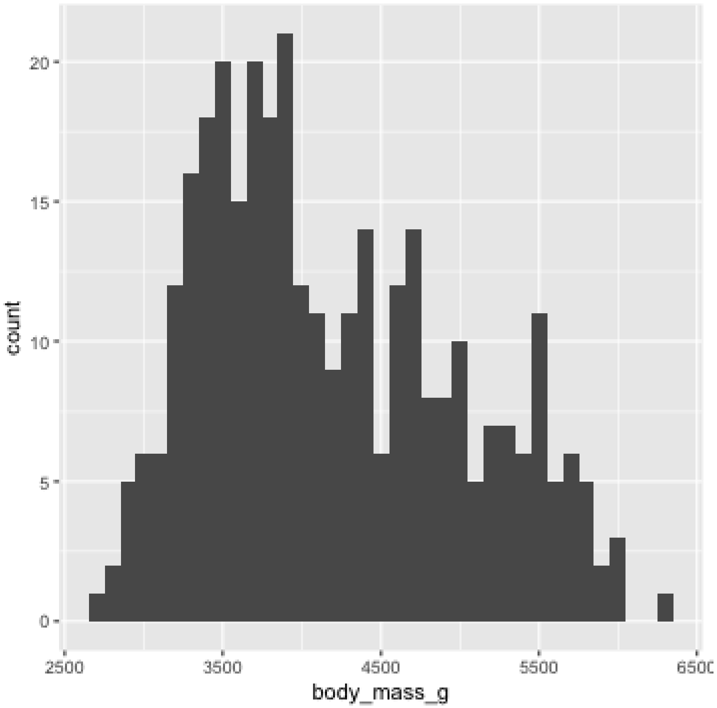
```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g))
```

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



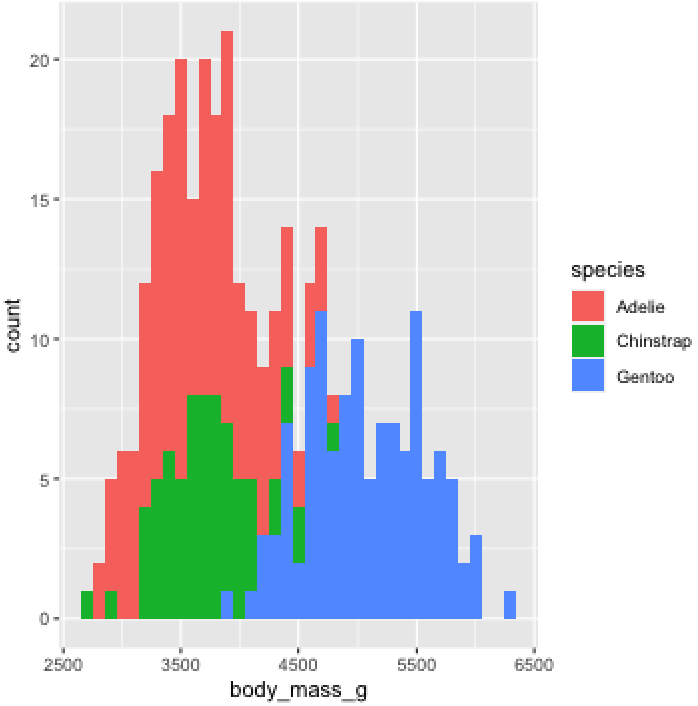
geom_histogram

```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g),
                 binwidth = 100)
```



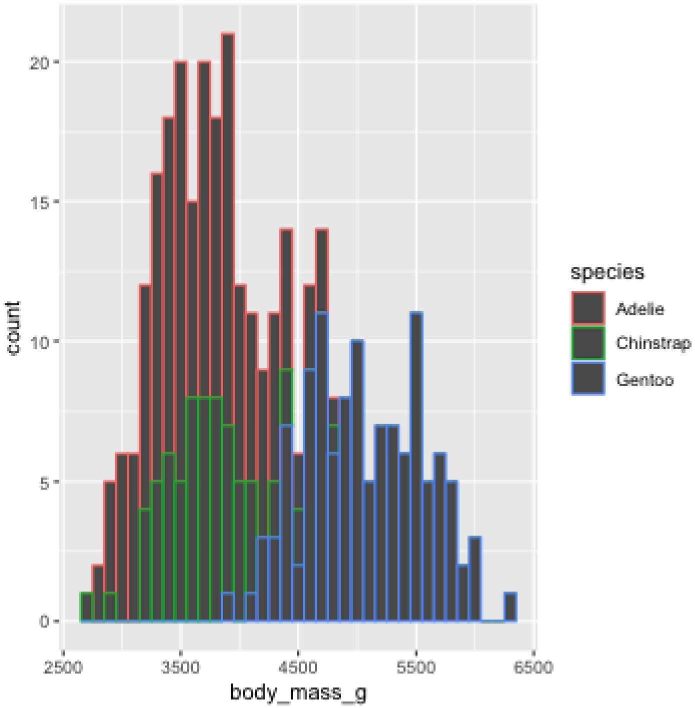
geom_histogram

```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g, fill = species),
                 binwidth = 100)
```



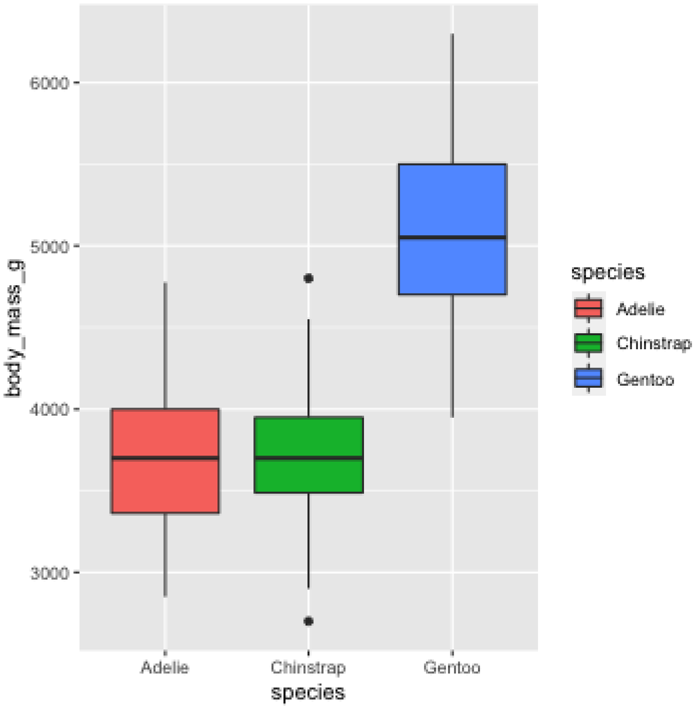
geom_histogram

```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g, colour = species),
                 binwidth = 100)
```



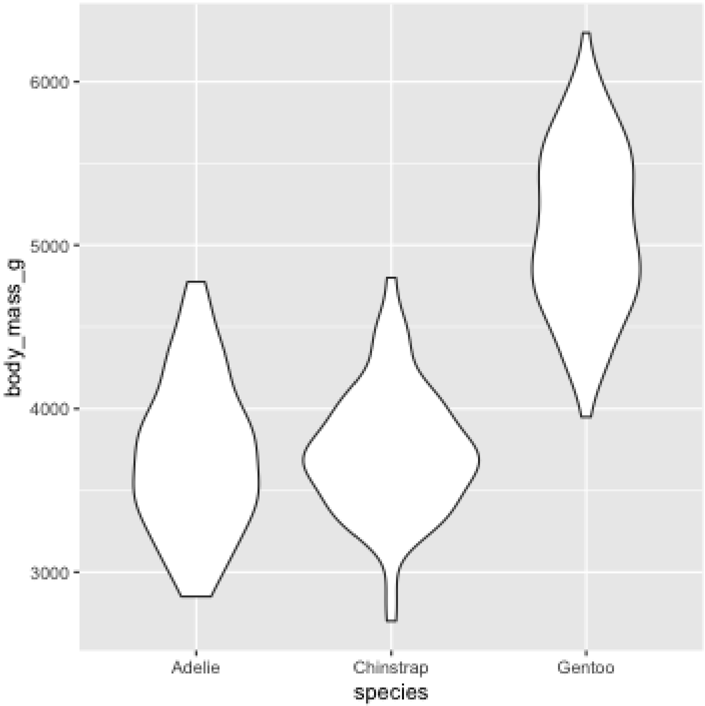
Boxplots

```
ggplot(data = penguins) +
  geom_boxplot(mapping = aes(x = species, y = body_mass_g,
                             fill = species))
```



Violin plot

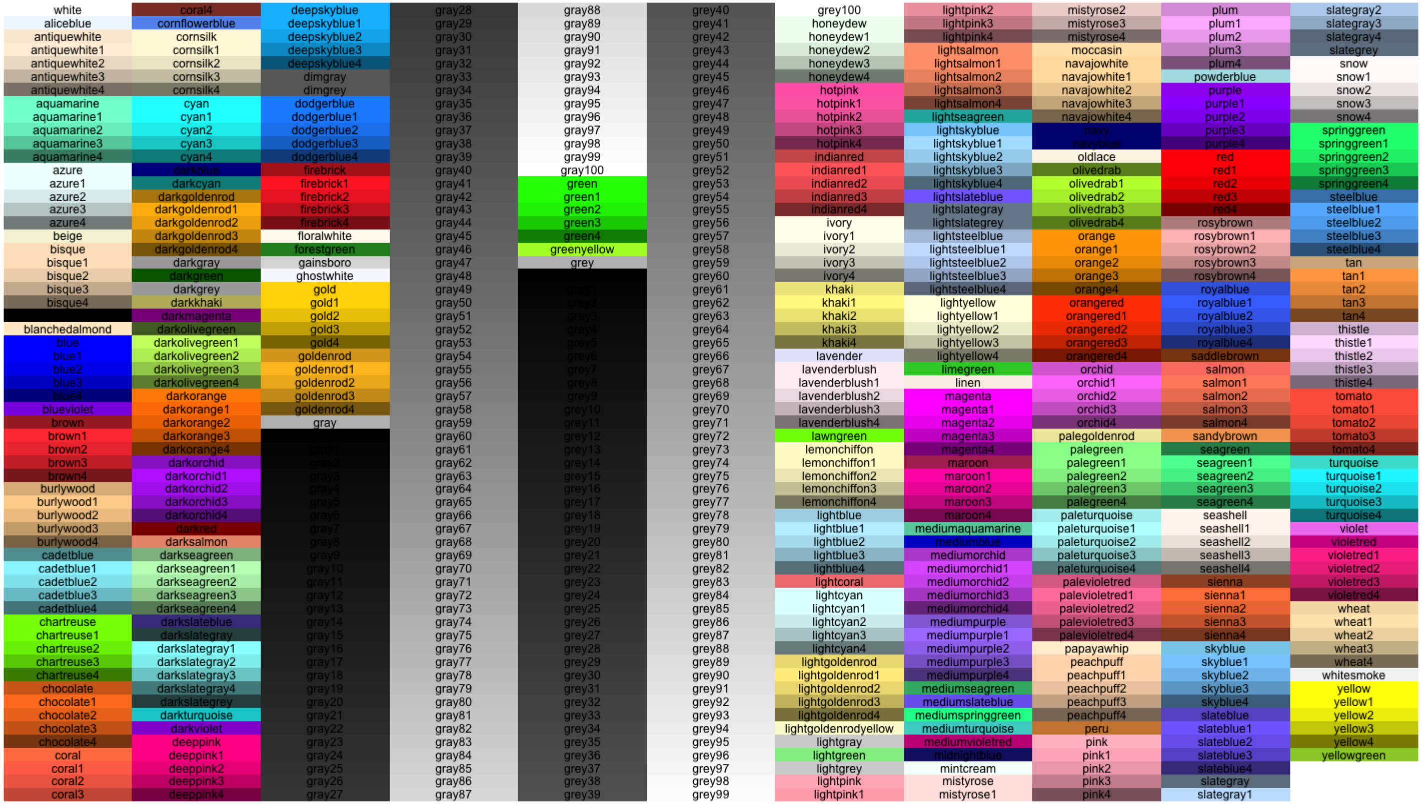
```
ggplot(data = penguins) +
  geom_violin(mapping = aes(x = species, y = body_mass_g))
```



colours()

```
colours() %>% head()
```

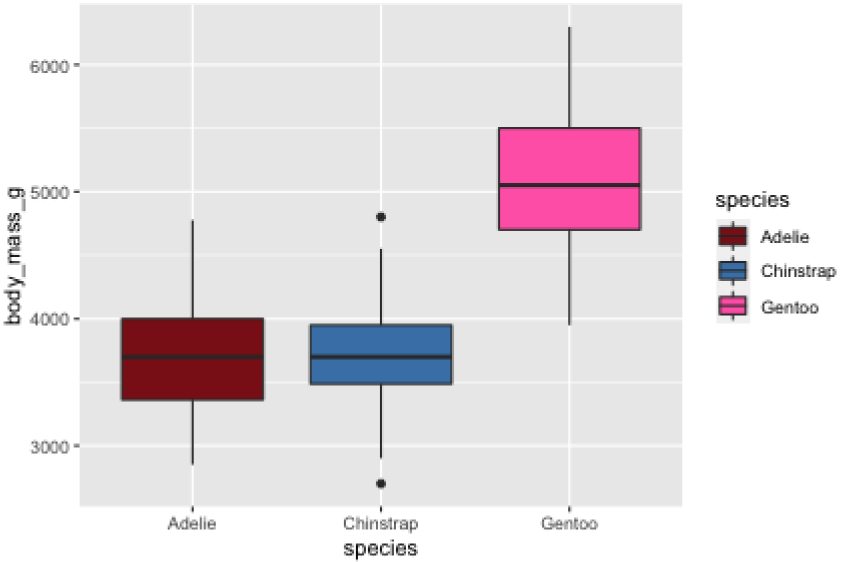
```
## [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
## [5] "antiquewhite2" "antiquewhite3"
```



white
aliceblue
antiquewhite
antiquewhite1
antiquewhite2
antiquewhite3
antiquewhite4
aquamarine
aquamarine1
aquamarine2
aquamarine3
aquamarine4
azure
azure1
azure2
azure3
azure4
beige
bisque
bisque1
bisque2
bisque3
bisque4
blanchedalmond
blue
blue1
blue2
blue3
blue4
blueviolet
brown
brown1
brown2
brown3
brown4
burlywood
burlywood1
burlywood2
burlywood3
burlywood4
cadetblue
cadetblue1
cadetblue2
cadetblue3
cadetblue4
chartreuse
chartreuse1
chartreuse2
chartreuse3
chartreuse4
chocolate
chocolate1
chocolate2
chocolate3
chocolate4
coral
coral1
coral2
coral3
coral4
cornflowerblue
cornsilk
cornsilk1
cornsilk2
cornsilk3
cornsilk4
cyan
cyan1
cyan2
cyan3
cyan4
darkblue
darkcyan
darkgoldenrod
darkgoldenrod1
darkgoldenrod2
darkgoldenrod3
darkgoldenrod4
darkgray
darkgreen
darkgrey
darkkhaki
darkmagenta
darkolivegreen
darkolivegreen1
darkolivegreen2
darkolivegreen3
darkolivegreen4
darkorange
darkorange1
darkorange2
darkorange3
darkorange4
darkorchid
darkorchid1
darkorchid2
darkorchid3
darkorchid4
darkred
darksalmon
darkseagreen
darkseagreen1
darkseagreen2
darkseagreen3
darkseagreen4
darkslateblue
darkslategray
darkslategray1
darkslategray2
darkslategray3
darkslategray4
darkslategrey
darkturquoise
darkviolet
deeppink
deeppink1
deeppink2
deeppink3
deeppink4
deepskyblue
deepskyblue1
deepskyblue2
deepskyblue3
deepskyblue4
dimgray
dimgrey
dodgerblue
dodgerblue1
dodgerblue2
dodgerblue3
dodgerblue4
firebrick
firebrick1
firebrick2
firebrick3
firebrick4
floralwhite
forestgreen
gainsboro
ghostwhite
gold
gold1
gold2
gold3
gold4
goldenrod
goldenrod1
goldenrod2
goldenrod3
goldenrod4
gray
gray1
gray2
gray3
gray4
gray5
gray6
gray7
gray8
gray9
gray10
gray11
gray12
gray13
gray14
gray15
gray16
gray17
gray18
gray19
gray20
gray21
gray22
gray23
gray24
gray25
gray26
gray27
gray28
gray29
gray30
gray31
gray32
gray33
gray34
gray35
gray36
gray37
gray38
gray39
gray40
gray41
gray42
gray43
gray44
gray45
gray46
gray47
gray48
gray49
gray50
gray51
gray52
gray53
gray54
gray55
gray56
gray57
gray58
gray59
gray60
gray61
gray62
gray63
gray64
gray65
gray66
gray67
gray68
gray69
gray70
gray71
gray72
gray73
gray74
gray75
gray76
gray77
gray78
gray79
gray80
gray81
gray82
gray83
gray84
gray85
gray86
gray87
gray88
gray89
gray90
gray91
gray92
gray93
gray94
gray95
gray96
gray97
gray98
gray99
gray100
green
green1
green2
green3
green4
greenyellow
grey
grey1
grey2
grey3
grey4
grey5
grey6
grey7
grey8
grey9
grey10
grey11
grey12
grey13
grey14
grey15
grey16
grey17
grey18
grey19
grey20
grey21
grey22
grey23
grey24
grey25
grey26
grey27
grey28
grey29
grey30
grey31
grey32
grey33
grey34
grey35
grey36
grey37
grey38
grey39
grey40
grey41
grey42
grey43
grey44
grey45
grey46
grey47
grey48
grey49
grey50
grey51
grey52
grey53
grey54
grey55
grey56
grey57
grey58
grey59
grey60
grey61
grey62
grey63
grey64
grey65
grey66
grey67
grey68
grey69
grey70
grey71
grey72
grey73
grey74
grey75
grey76
grey77
grey78
grey79
grey80
grey81
grey82
grey83
grey84
grey85
grey86
grey87
grey88
grey89
grey90
grey91
grey92
grey93
grey94
grey95
grey96
grey97
grey98
grey99
grey100
honeydew
honeydew1
honeydew2
honeydew3
honeydew4
hotpink
hotpink1
hotpink2
hotpink3
hotpink4
indianred
indianred1
indianred2
indianred3
indianred4
ivory
ivory1
ivory2
ivory3
ivory4
khaki
khaki1
khaki2
khaki3
khaki4
lavender
lavenderblush
lavenderblush1
lavenderblush2
lavenderblush3
lavenderblush4
lawngreen
lemonchiffon
lemonchiffon1
lemonchiffon2
lemonchiffon3
lemonchiffon4
lightblue
lightblue1
lightblue2
lightblue3
lightblue4
lightcoral
lightcyan
lightcyan1
lightcyan2
lightcyan3
lightcyan4
lightgoldenrod
lightgoldenrod1
lightgoldenrod2
lightgoldenrod3
lightgoldenrod4
lightgoldenrodyellow
lightgray
lightgreen
lightgrey
lightpink
lightpink1
lightpink2
lightpink3
lightpink4
lightsalmon
lightsalmon1
lightsalmon2
lightsalmon3
lightsalmon4
lightseagreen
lightskyblue
lightskyblue1
lightskyblue2
lightskyblue3
lightskyblue4
lightslateblue
lightslategray
lightslategrey
lightsteelblue
lightsteelblue1
lightsteelblue2
lightsteelblue3
lightsteelblue4
lightyellow
lightyellow1
lightyellow2
lightyellow3
lightyellow4
limegreen
linen
magenta
magenta1
magenta2
magenta3
magenta4
maroon
maroon1
maroon2
maroon3
maroon4
mediumaquamarine
mediumblue
mediumorchid
mediumorchid1
mediumorchid2
mediumorchid3
mediumorchid4
mediumpurple
mediumpurple1
mediumpurple2
mediumpurple3
mediumpurple4
mediumseagreen
mediumslateblue
mediumspringgreen
mediumturquoise
mediumvioletred
midnightblue
mintcream
mistyrose
mistyrose1
mistyrose2
mistyrose3
mistyrose4
moccasin
navajowhite
navajowhite1
navajowhite2
navajowhite3
navajowhite4
navy
navyblue
oldlace
olivedrab
olivedrab1
olivedrab2
olivedrab3
olivedrab4
orange
orange1
orange2
orange3
orange4
orangered
orangered1
orangered2
orangered3
orangered4
orchid
orchid1
orchid2
orchid3
orchid4
palegoldenrod
palegreen
palegreen1
palegreen2
palegreen3
palegreen4
paleturquoise
paleturquoise1
paleturquoise2
paleturquoise3
paleturquoise4
palevioletred
palevioletred1
palevioletred2
palevioletred3
palevioletred4
papayawhip
peachpuff
peachpuff1
peachpuff2
peachpuff3
peachpuff4
peru
pink
pink1
pink2
pink3
pink4
plum
plum1
plum2
plum3
plum4
powderblue
purple
purple1
purple2
purple3
purple4
red
red1
red2
red3
red4
rosybrown
rosybrown1
rosybrown2
rosybrown3
rosybrown4
royalblue
royalblue1
royalblue2
royalblue3
royalblue4
saddlebrown
salmon
salmon1
salmon2
salmon3
salmon4
sandybrown
seagreen
seagreen1
seagreen2
seagreen3
seagreen4
seashell
seashell1
seashell2
seashell3
seashell4
sienna
sienna1
sienna2
sienna3
sienna4
skyblue
skyblue1
skyblue2
skyblue3
skyblue4
slateblue
slateblue1
slateblue2
slateblue3
slateblue4
slategray
slategray1
slategray2
slategray3
slategray4
slategrey
snow
snow1
snow2
snow3
snow4
springgreen
springgreen1
springgreen2
springgreen3
springgreen4
steelblue
steelblue1
steelblue2
steelblue3
steelblue4
tan
tan1
tan2
tan3
tan4
thistle
thistle1
thistle2
thistle3
thistle4
tomato
tomato1
tomato2
tomato3
tomato4
turquoise
turquoise1
turquoise2
turquoise3
turquoise4
violet
violetred
violetred1
violetred2
violetred3
violetred4
wheat
wheat1
wheat2
wheat3
wheat4
whitesmoke
yellow
yellow1
yellow2
yellow3
yellow4
yellowgreen

Using colours in ggplot

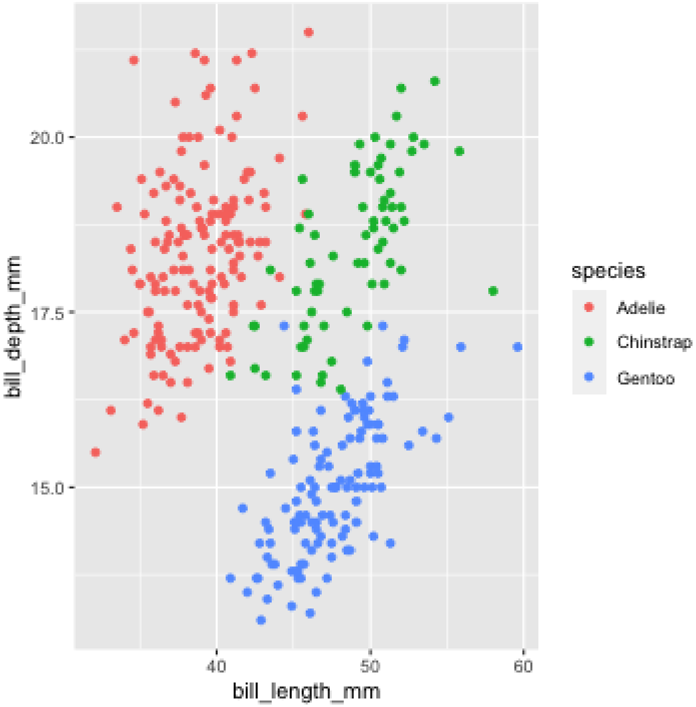
```
ggplot(data = penguins, mapping = aes(x = species, y = body_mass_g,
                                      fill = species)) +
  geom_boxplot() +
  scale_fill_manual(values = c('firebrick4', 'steelblue', 'hotpink'))
```



ggplot2.tidyverse.org/reference/#scales

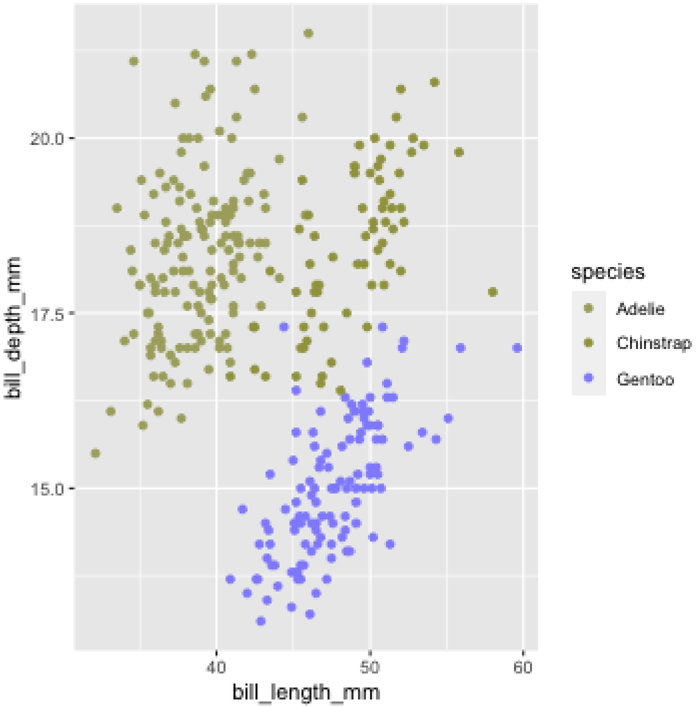
Colour-blind friendly palettes

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm,
                           y = bill_depth_mm,
                           colour = species))
```

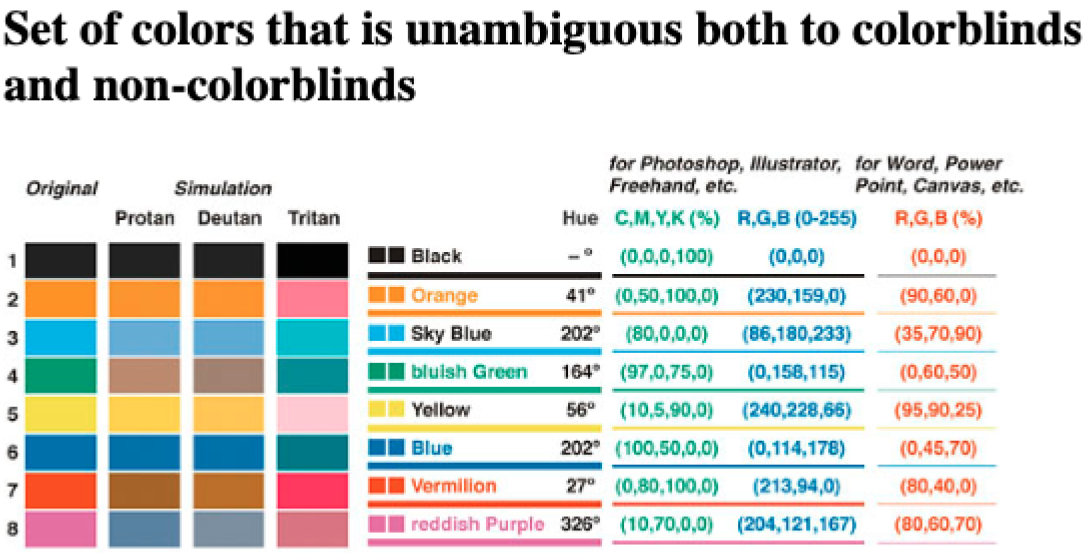


Colour-blind friendly palettes

```
library(dichromat)
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           colour = species)) +
  scale_colour_manual(values = dichromat(scales::hue_pal()(3)))
```



Colour-blind friendly palettes



Set of colors that is unambiguous both to colorblinds and non-colorblinds

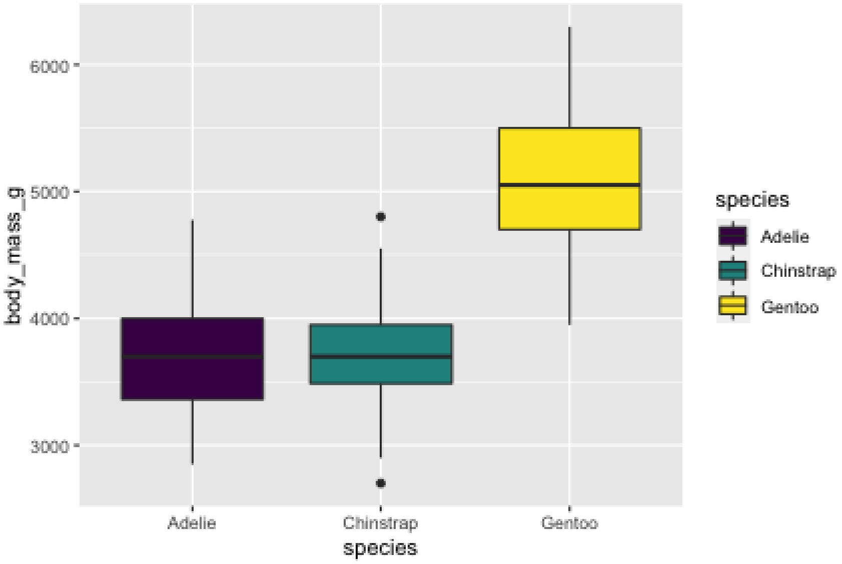
	Original	*Simulation* Protan	 Deutan	 Tritan		Hue	*for Photoshop, Illustrator, Freehand, etc.* C,M,Y,K (%)	 R,G,B (0-255)	*for Word, Power Point, Canvas, etc.* R,G,B (%)
1	■	■	■	■	■■ Black	–°	(0,0,0,100)	(0,0,0)	(0,0,0)
2	■	■	■	■	■■ Orange	41°	(0,50,100,0)	(230,159,0)	(90,60,0)
3	■	■	■	■	■■ Sky Blue	202°	(80,0,0,0)	(86,180,233)	(35,70,90)
4	■	■	■	■	■■ bluish Green	164°	(97,0,75,0)	(0,158,115)	(0,60,50)
5	■	■	■	■	■■ Yellow	56°	(10,5,90,0)	(240,228,66)	(95,90,25)
6	■	■	■	■	■■ Blue	202°	(100,50,0,0)	(0,114,178)	(0,45,70)
7	■	■	■	■	■■ Vermilion	27°	(0,80,100,0)	(213,94,0)	(80,40,0)
8	■	■	■	■	■■ reddish Purple	326°	(10,70,0,0)	(204,121,167)	(80,60,70)

https://jfly.uni-koeln.de/color/#pallet

Viridis: perceptually uniform scales

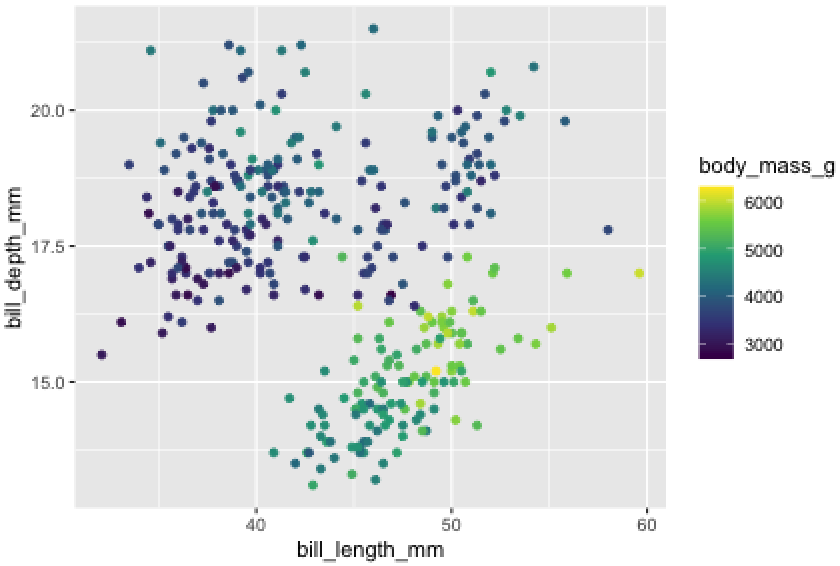
Viridis: perceptually uniform scales

```
ggplot(data = penguins,
       mapping = aes(x = species, y = body_mass_g, fill = species))
  geom_boxplot() +
  scale_fill_viridis_d()
```



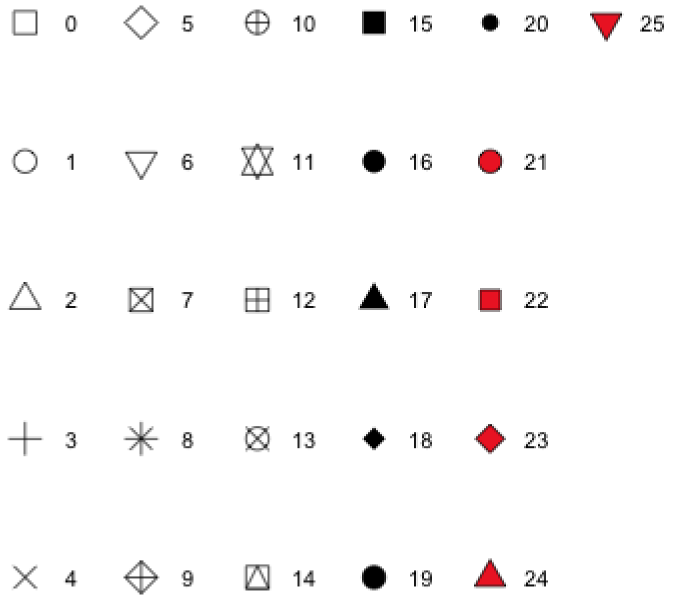
Viridis: perceptually uniform scales

```
ggplot(data = penguins, mapping = aes(x = bill_length_mm,
       y = bill_depth_mm, colour = body_mass_g)) +
  geom_point() +
  scale_colour_viridis_c()
```



Shapes

- There are 26 shapes available in R for plotting that are identified by numbers

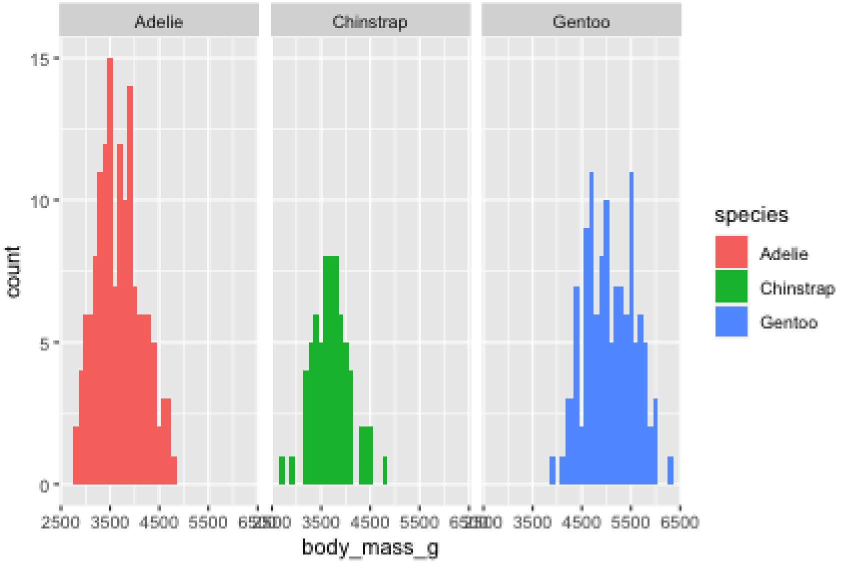


- 0-14 are hollow. The border colour is determined by the `colour` aesthetic
- 15-20 are solid. The colour is determined by the `colour` aesthetic
- 21-25 are filled shapes that have a border `colour` and a `fill` colour

ggplot2.tidyverse.org/reference/aes_linetype_size_shape.html

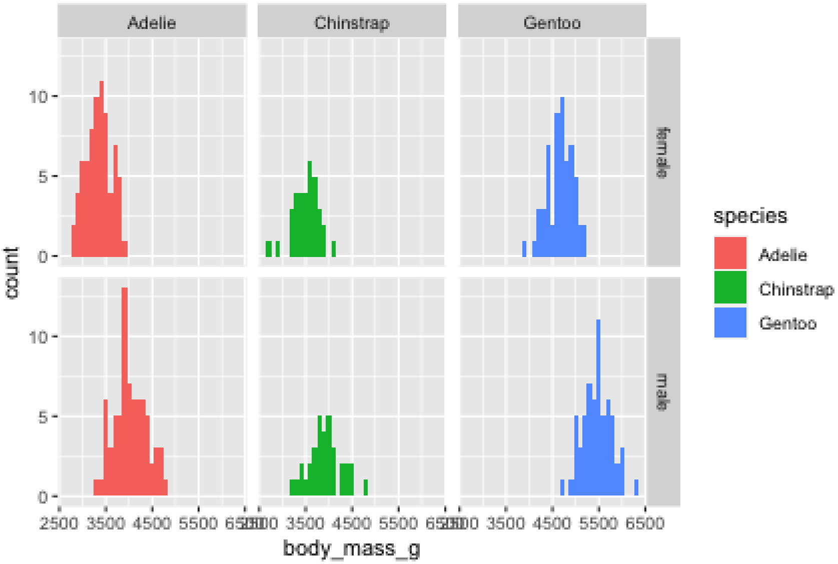
Facets

```
ggplot(data = penguins,
       mapping = aes(x = body_mass_g, fill = species)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(vars(species))
```



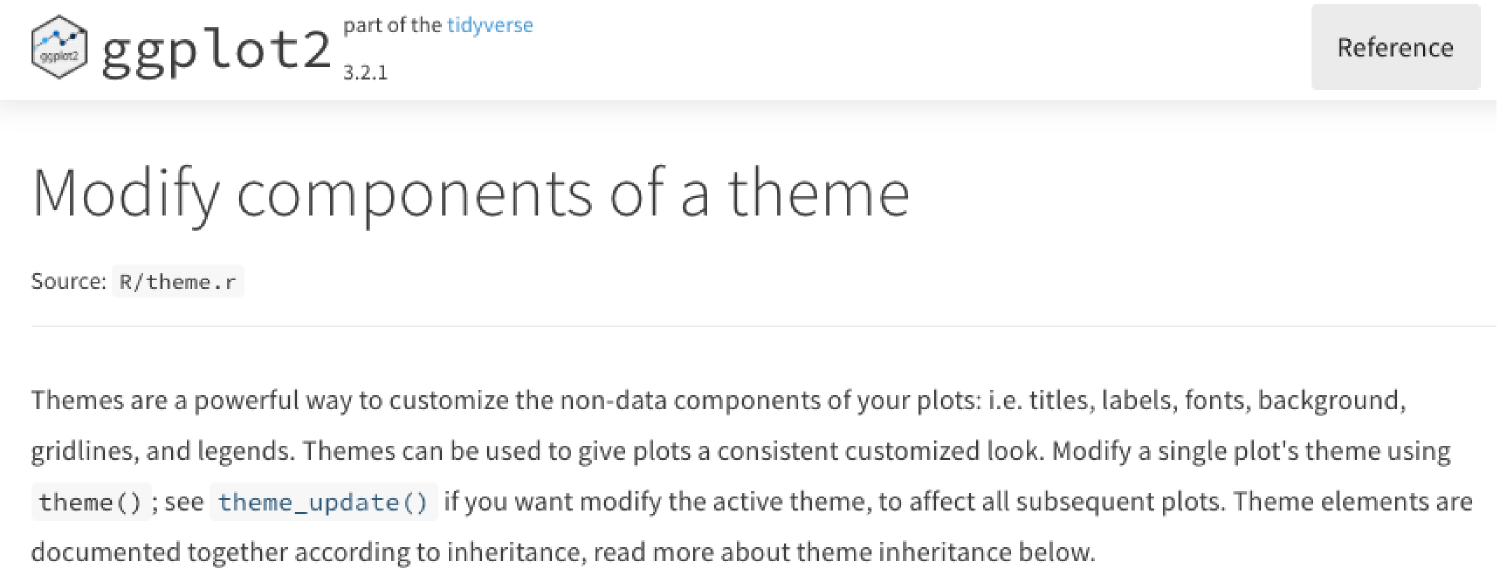
Facets

```
ggplot(data = penguins,
       mapping = aes(x = body_mass_g, fill = species)) +
  geom_histogram(binwidth = 100) +
  facet_grid(cols = vars(species),
             rows = vars(sex))
```



Themes

- `theme()` is ggplot's way of controlling the overall look of a plot
- Change axis titles, labels, lines and ticks
- Change the look of the legend (title, text, position, direction)
- Change the look of the panels (title, background, grid lines)
- A set of complete themes already exist



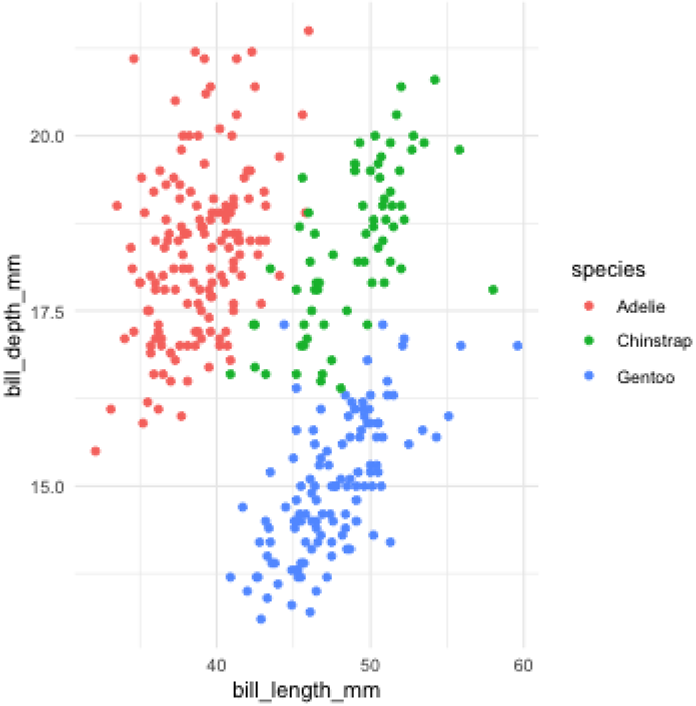
Modify components of a theme

Source: `R/theme.r`

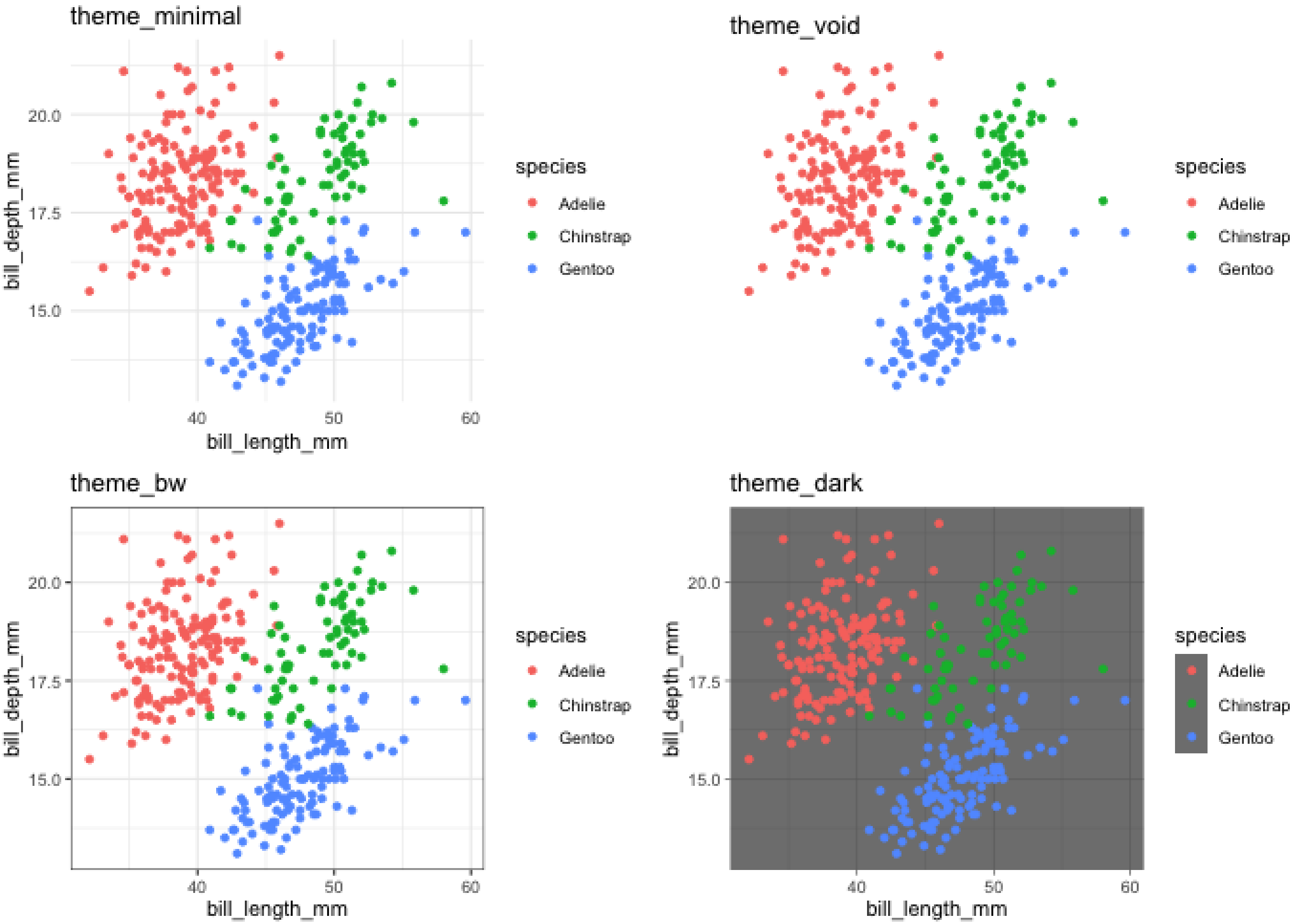
Themes are a powerful way to customize the non-data components of your plots: i.e. titles, labels, fonts, background, gridlines, and legends. Themes can be used to give plots a consistent customized look. Modify a single plot's theme using `theme()`; see `theme_update()` if you want modify the active theme, to affect all subsequent plots. Theme elements are documented together according to inheritance, read more about theme inheritance below.

Themes

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm,
                           y = bill_depth_mm,
                           colour = species)) +
  theme_minimal()
```



Themes



theme_minimal

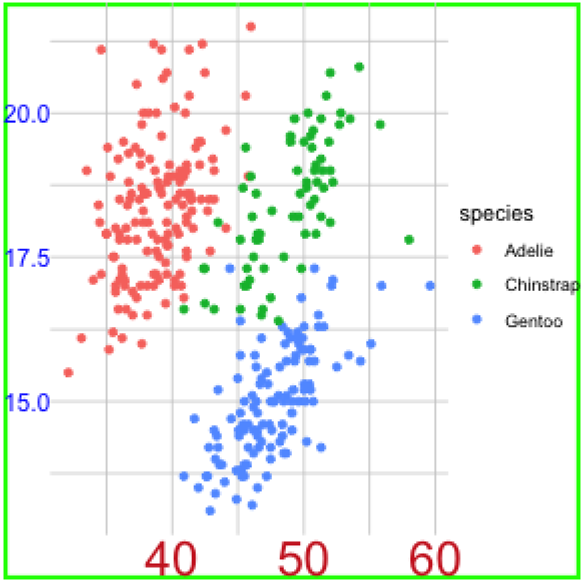
theme_void

theme_bw

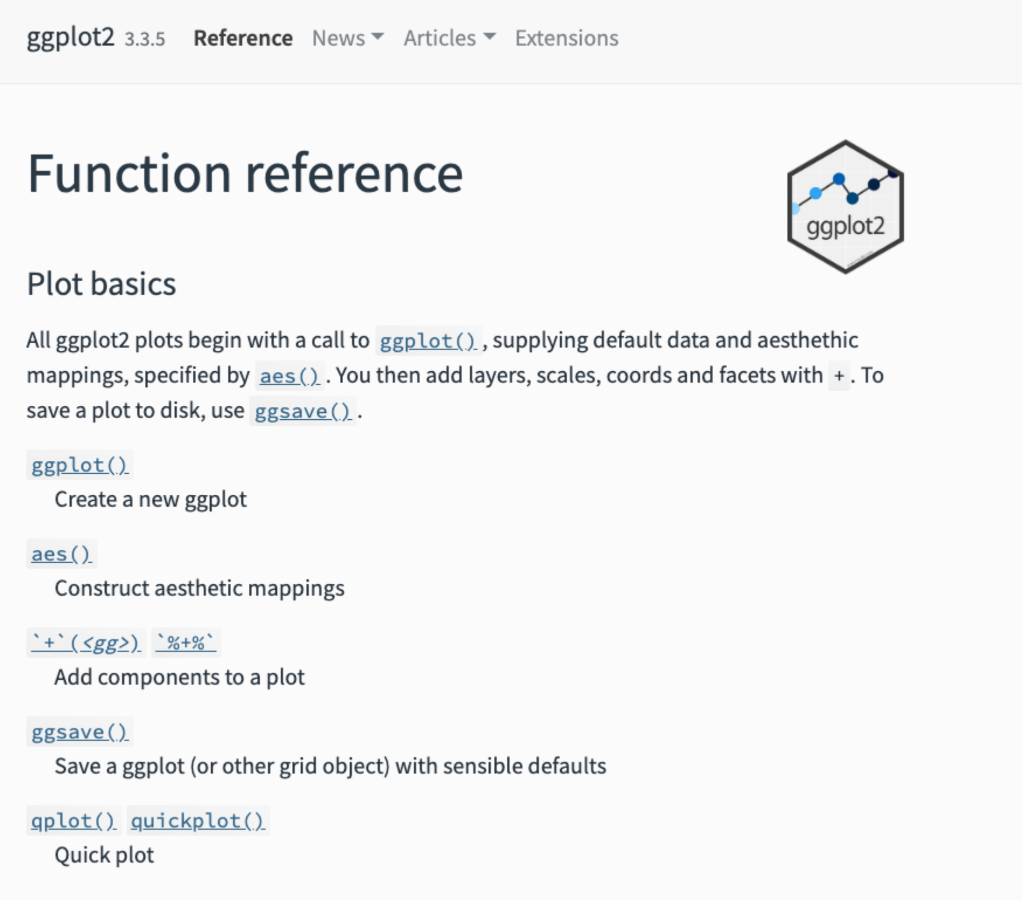
theme_dark

Customising themes

```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                           colour = species)) +
  theme_void() +
  theme(axis.text = element_text(colour = "blue", size = 12),
    axis.text.x = element_text(colour = "firebrick3", size = 24),
    panel.grid = element_line(colour = "grey80"),
    plot.background = element_rect(colour = "green", size = 2))
```



Further Reading



Function reference

Plot basics

All ggplot2 plots begin with a call to `ggplot()`, supplying default data and aesthethic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

`ggplot()`
Create a new ggplot

`aes()`
Construct aesthetic mappings

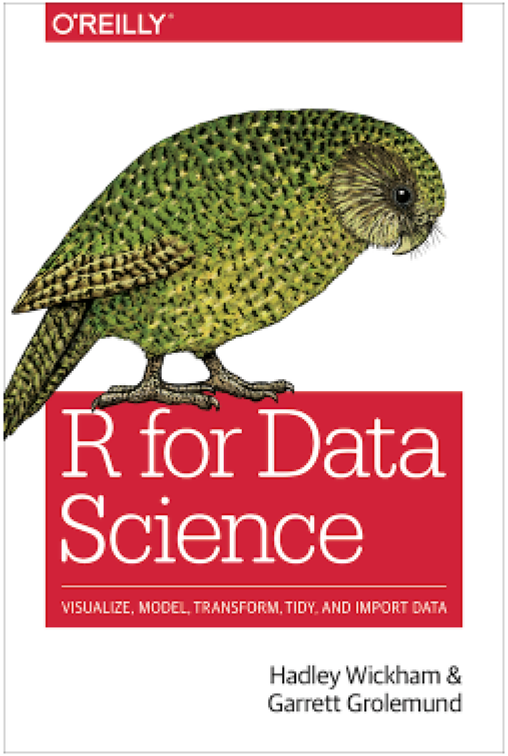
`` `+`(<gg>) `` `` `%+%` ``
Add components to a plot

`ggsave()`
Save a ggplot (or other grid object) with sensible defaults

`qplot()` `quickplot()`
Quick plot

https://ggplot2.tidyverse.org/reference

Further Reading


O'REILLY

R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

Hadley Wickham &
Garrett Grolemund

r4ds.had.co.nz

Exercises