Tidy data and visualisation with R

Richard White

27 Sept 2022

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

Tidyverse

Packages Blog Learn Help Contribute

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 m
```

```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

```
samples <- read_tsv('assets/samples.tsv')
```

```
## Rows: 9 Columns: 2
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: "\t"
## chr (2): sample_name, genotype
##
## 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 m
```

```
head(samples)
```

```
## # A tibble: 6 × 2
##   sample_name genotype
##   <chr>       <chr>
## 1 sample_1    wt
## 2 sample_2    wt
## 3 sample_3    wt
## 4 sample_4    het
## 5 sample_5    het
## 6 sample_6    het
```

Factors

```
samples$genotype
```

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

```
factor(samples$genotype)
```

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

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

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

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

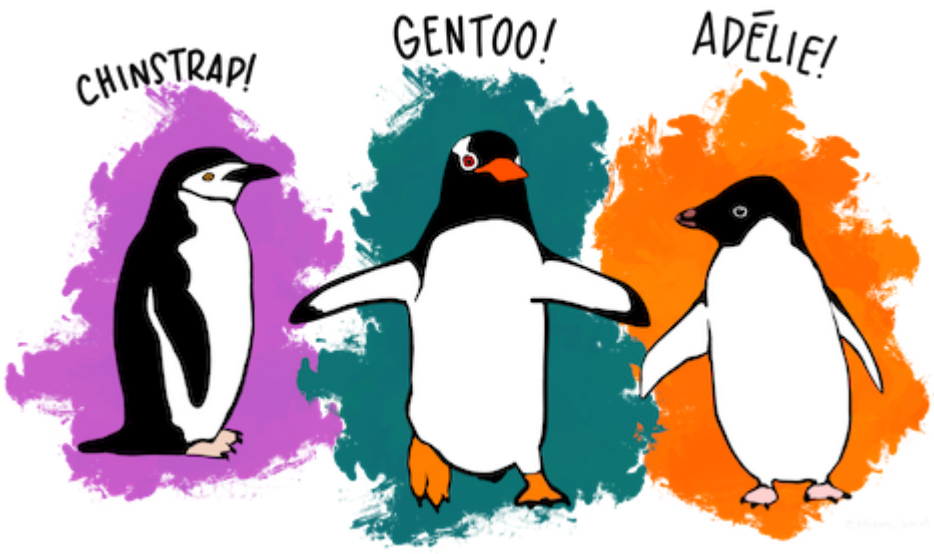
- `select()` - pick variables
- `filter()` - pick rows
- `arrange()` - sort rows
- `mutate()` - create new variables
- `summarise()` - reduce multiple values down to a single summary (mean, min, max etc.)

dplyr

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

```
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

```
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 that are functions of existing 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()

```
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
```

```
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))
```

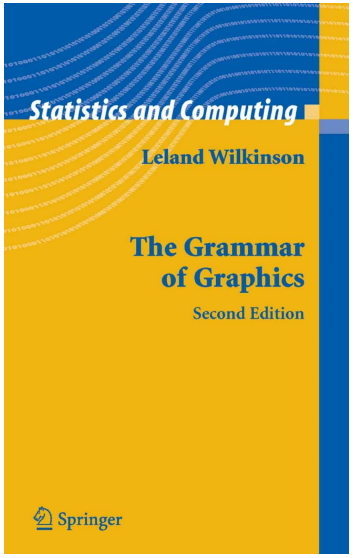
```
## # A tibble: 3 × 4
##   species   min_weight mean_weight max_weight
##   <fct>          <int>       <dbl>      <int>
## 1 Adelie          2850       3701.       4775
## 2 Chinstrap       2700       3733.       4800
## 3 Gentoo          3950       5076.       6300
```

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

ggplot2.tidyverse.org

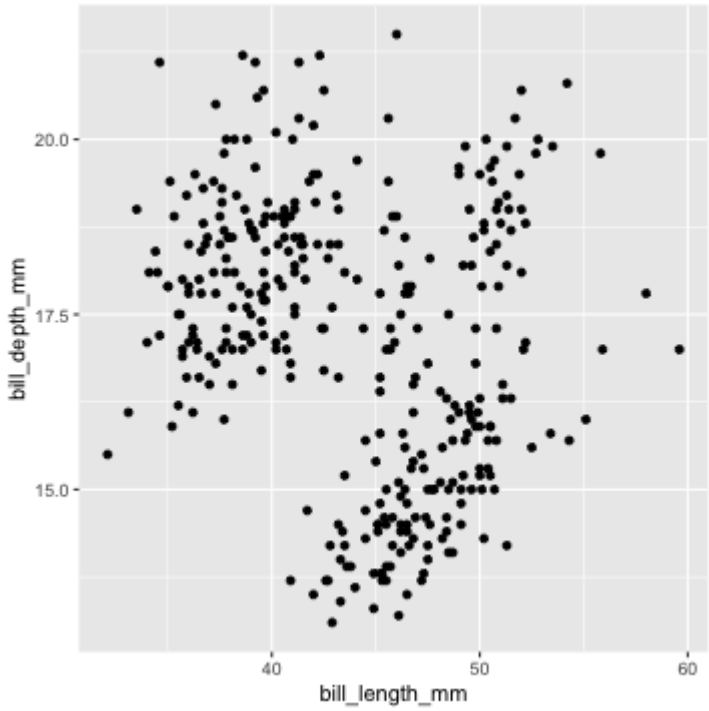
Base plot

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

Scatterplot

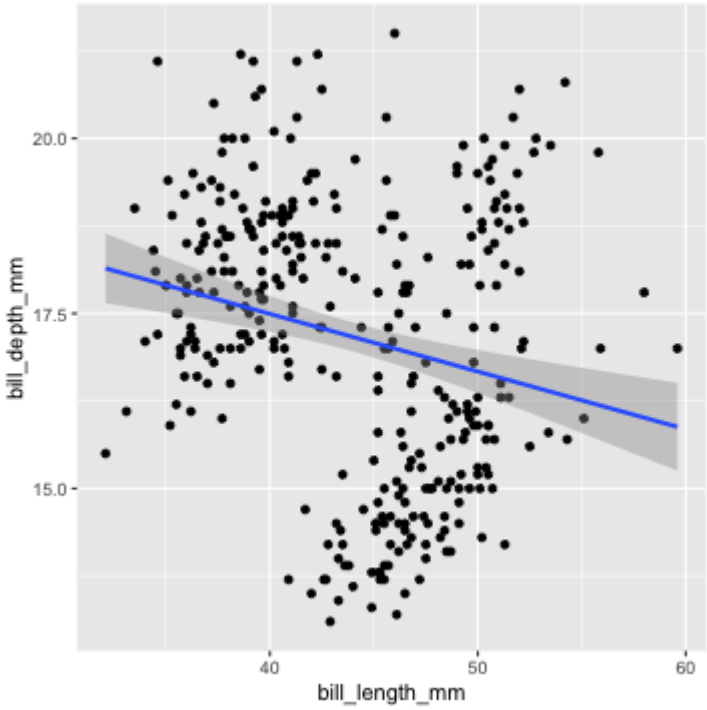
```
ggplot(data = penguins) +
  geom_point(mapping = aes(x = bill_length_mm, y = bill_depth_mm))
```

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



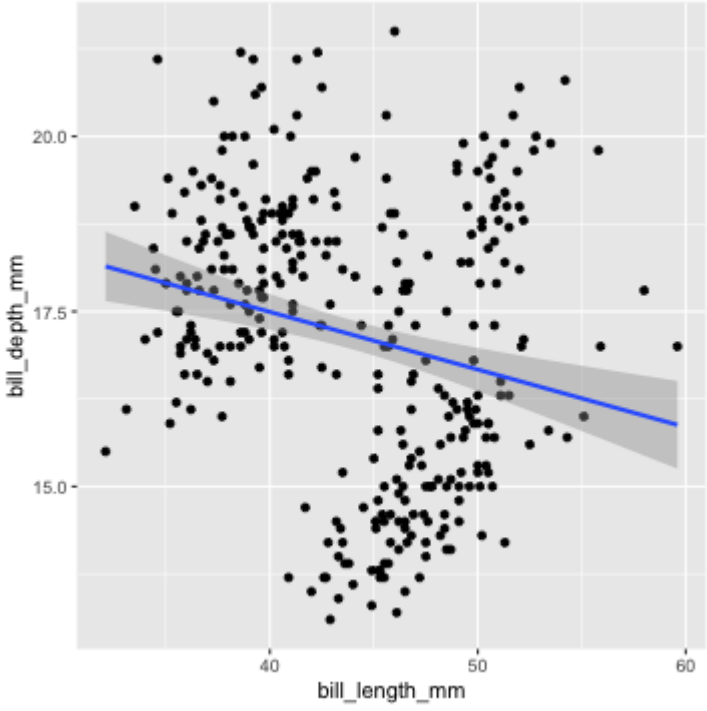
Add another geom

```
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)
```



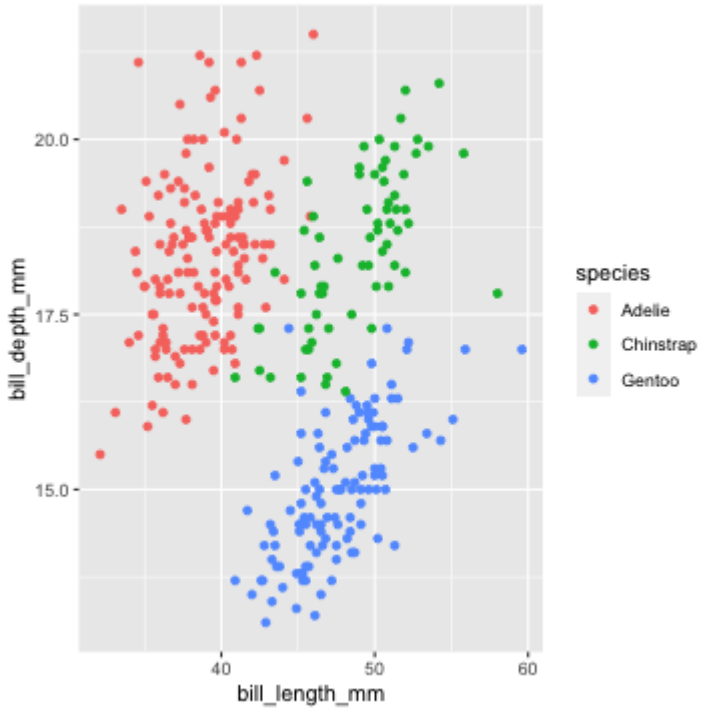
Set the aesthetics for the whole plot

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



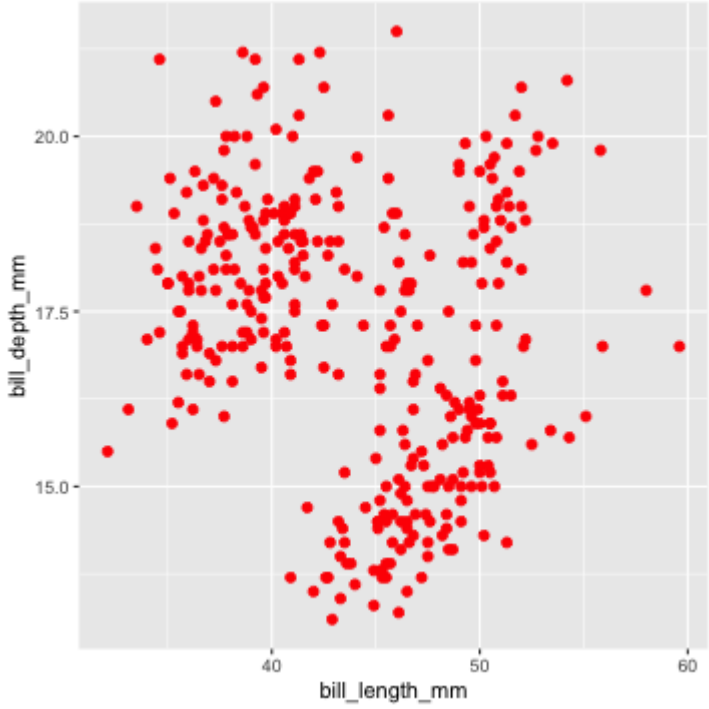
Map categorical variable to colour

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



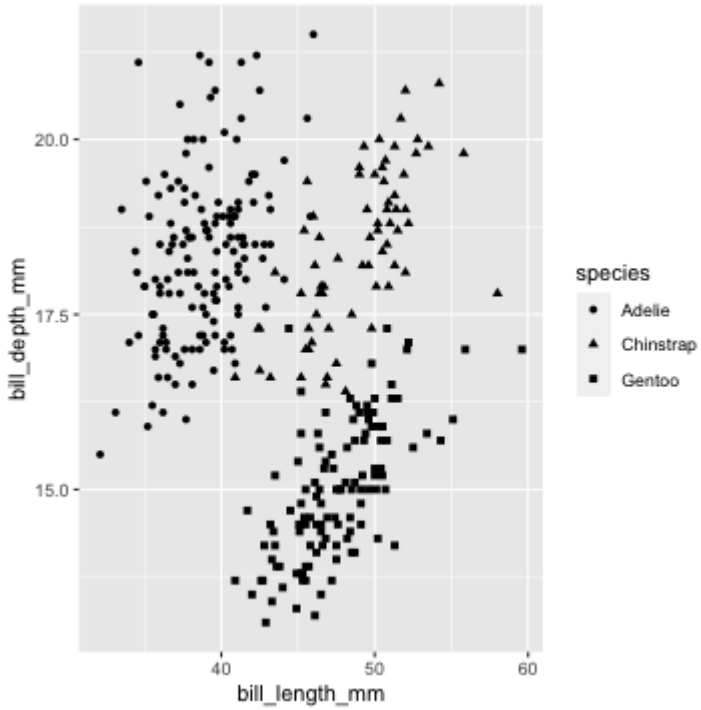
Setting arguments outside of `aes()`

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



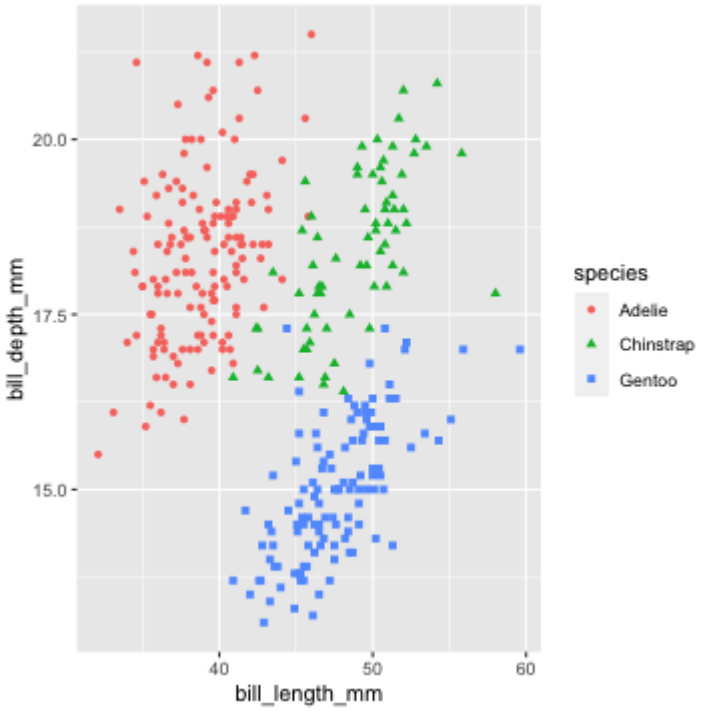
Map categorical variable to shape

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



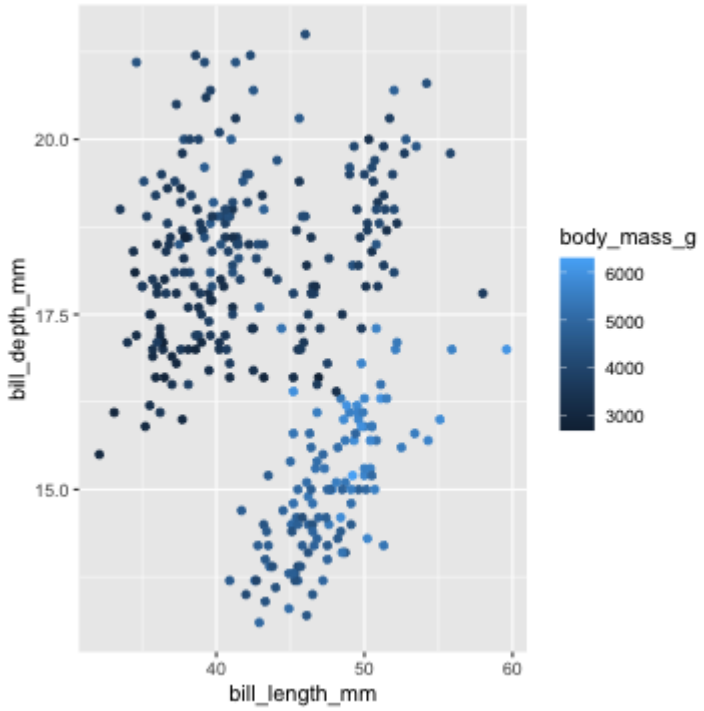
Map to colour and shape

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



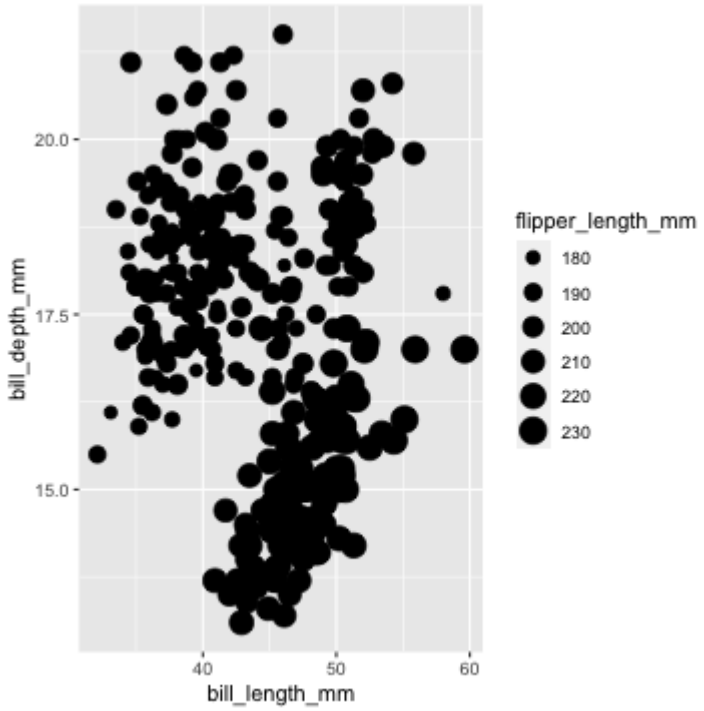
Map continuous variable to colour

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



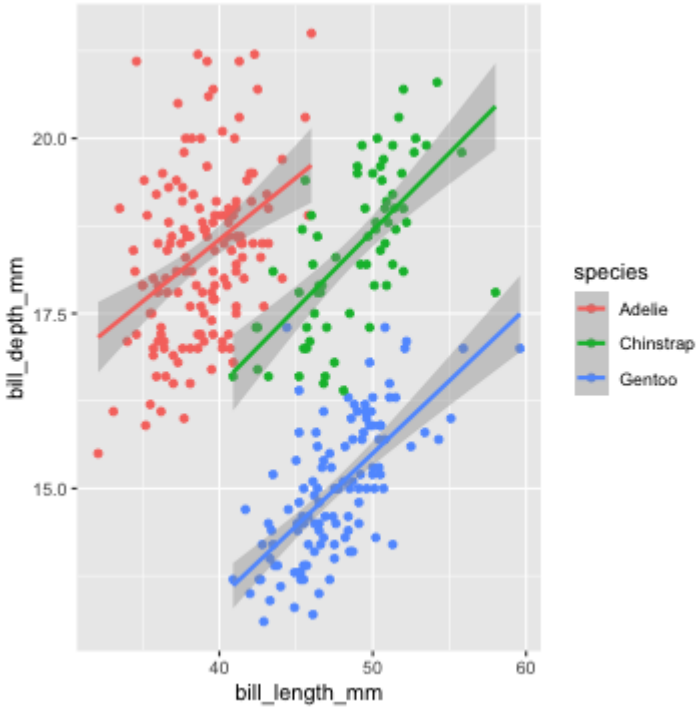
Map continuous variable to size

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



Add extra geoms

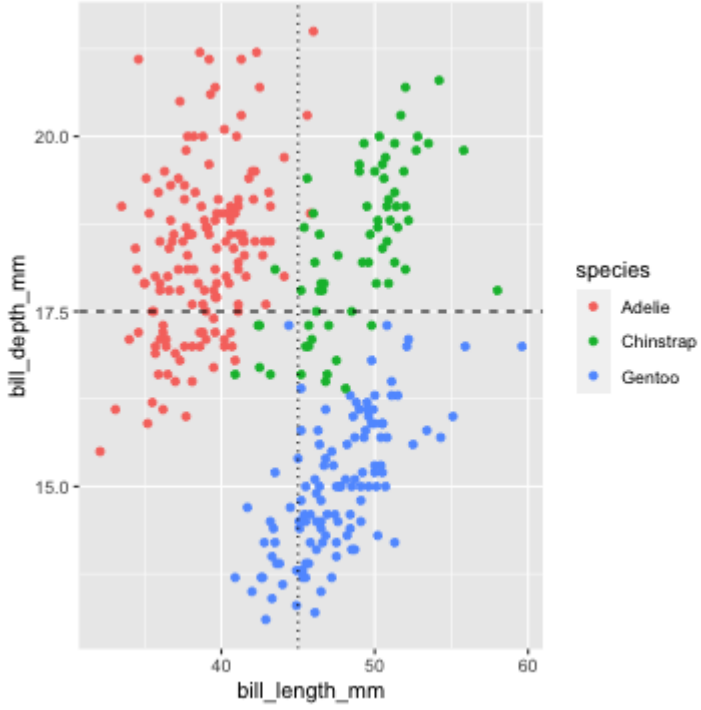
```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                     colour = species)) +
  geom_point() + geom_smooth(method = 'lm', formula = y ~ x)
```



ggplot2.tidyverse.org/reference

Add extra geoms

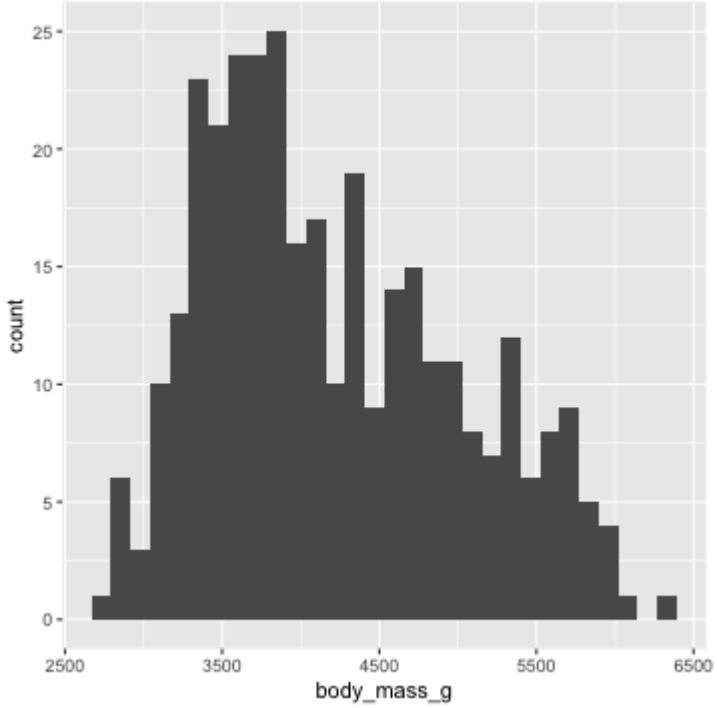
```
ggplot(data = penguins,
       mapping = aes(x = bill_length_mm, y = bill_depth_mm,
                     colour = species)) +
  geom_point() +
  geom_hline(yintercept = 17.5, linetype = "dashed") +
  geom_vline(xintercept = 45, linetype = "dotted")
```



geom_histogram

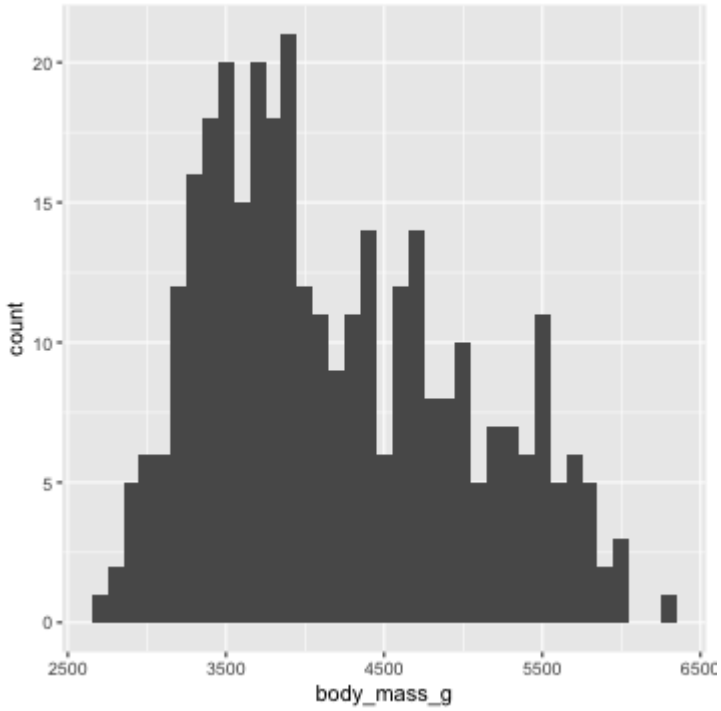
```
ggplot(data = penguins) +
  geom_histogram(mapping = aes(x = body_mass_g))
```

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



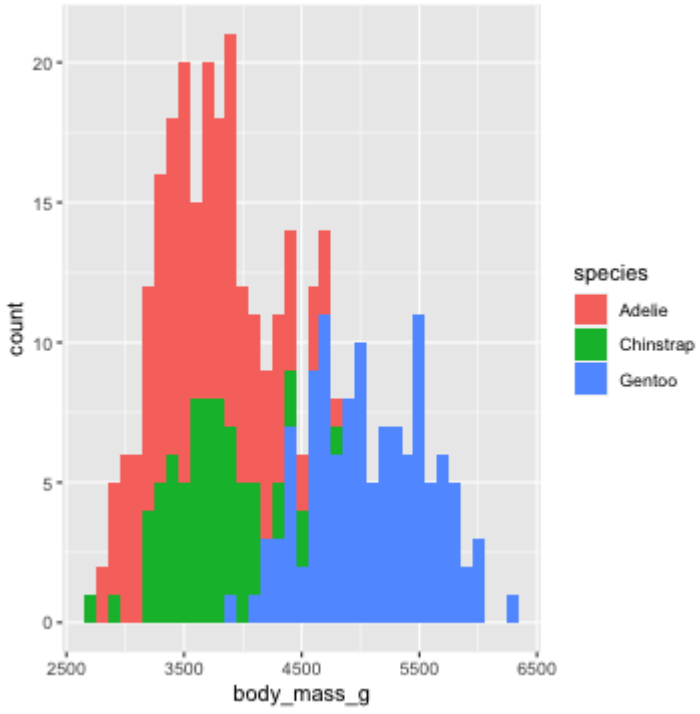
geom_histogram

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



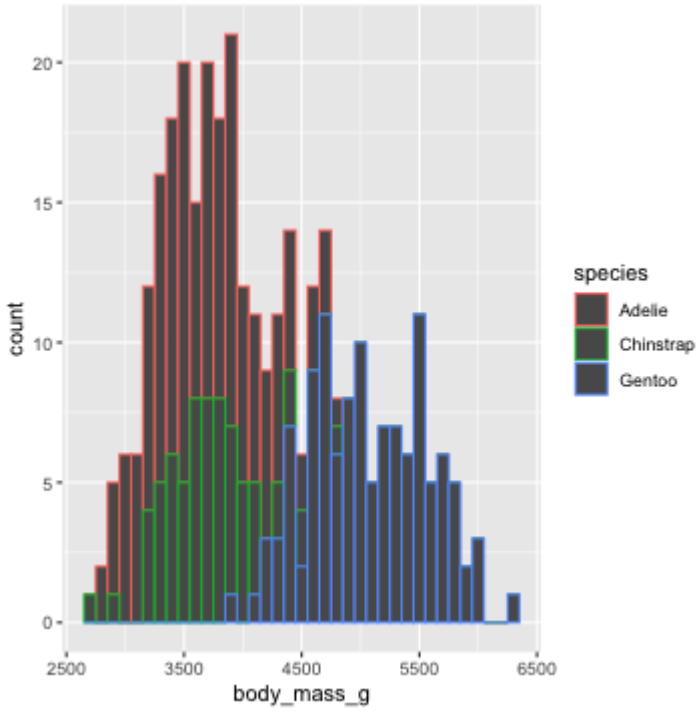
geom_histogram

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



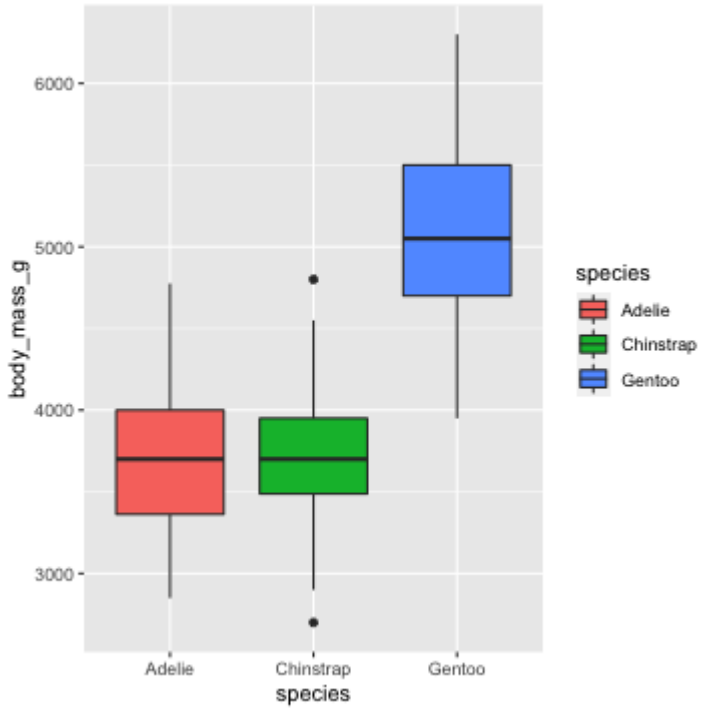
geom_histogram

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



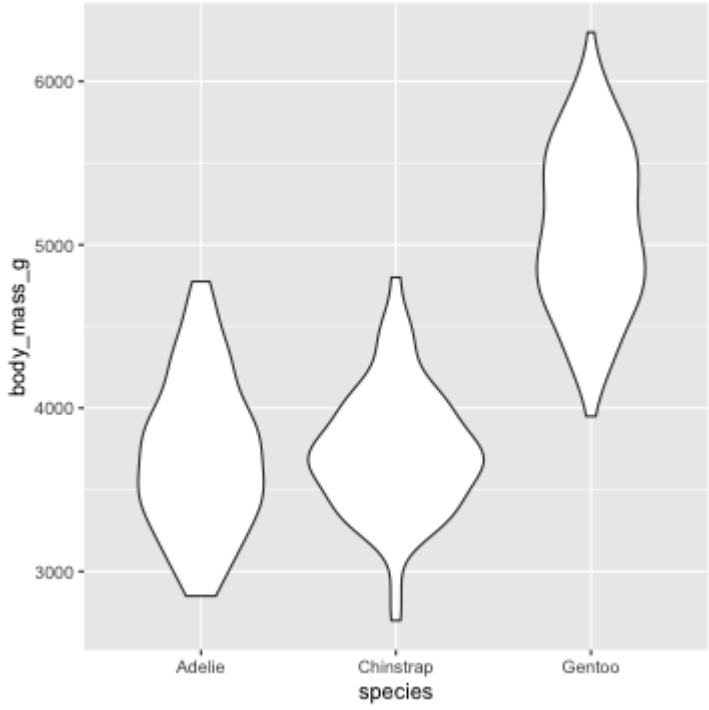
Boxplots

```
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()

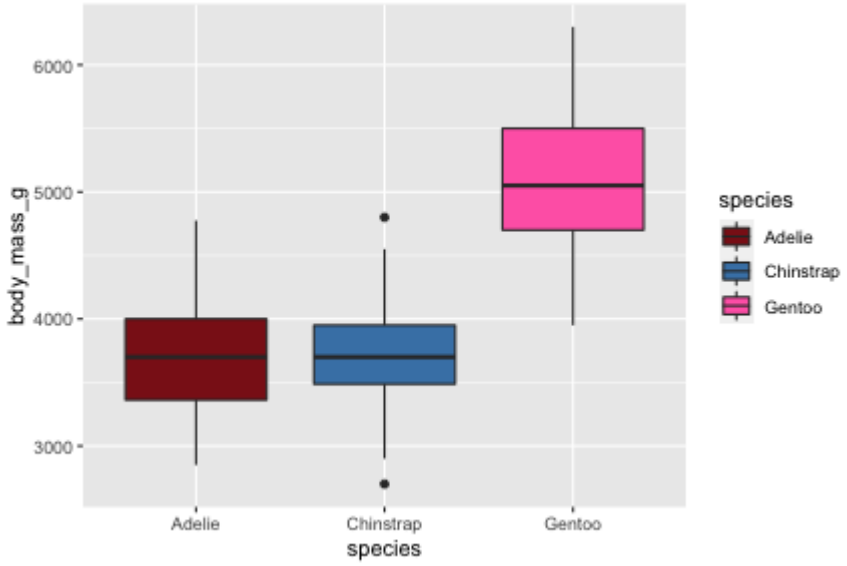
```
colours() %>% head()
```

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

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

Using colours in ggplot

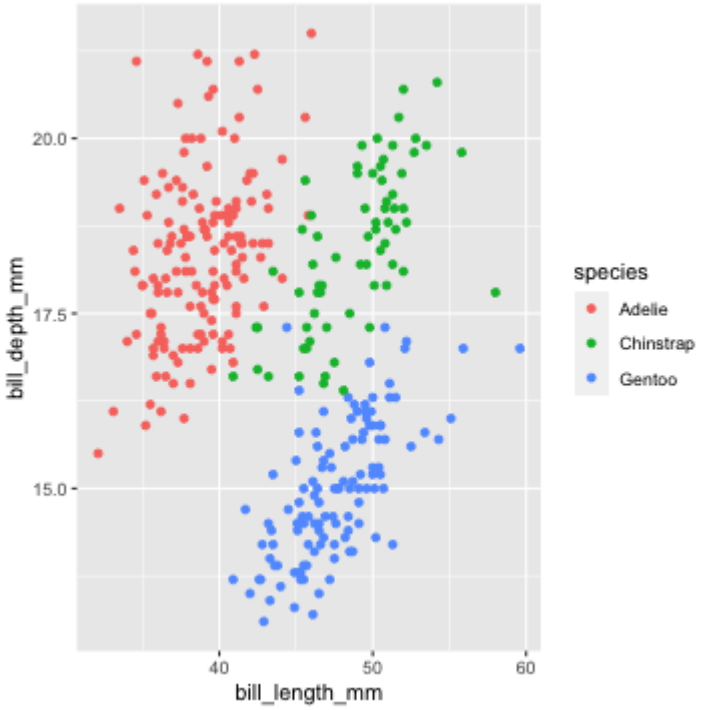
```
ggplot(data = penguins, mapping = aes(x = species, y = body_mass_g,
                                      fill = species)) +
  geom_boxplot() +
  scale_fill_manual(values = c('firebrick4', 'steelblue', 'hotpink'))
```



ggplot2.tidyverse.org/reference/#scales

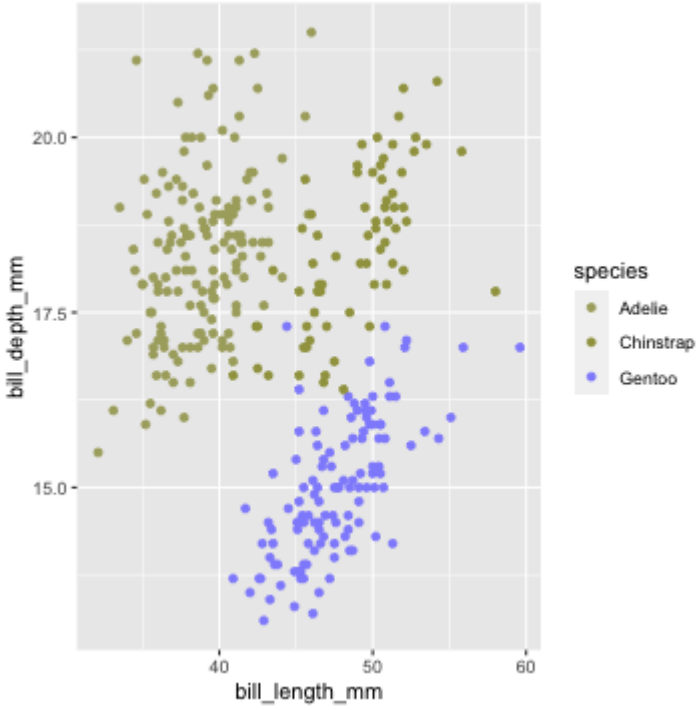
Colour-blind friendly palettes

```
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)))
```



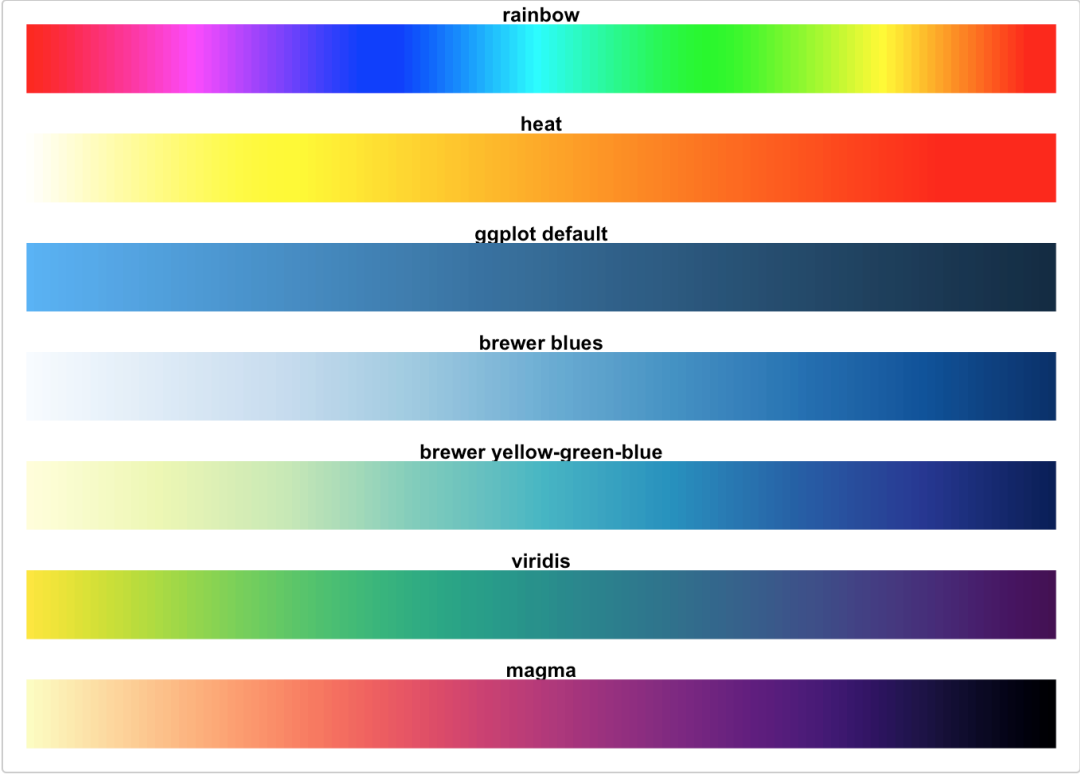
Colour-blind friendly palettes

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

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

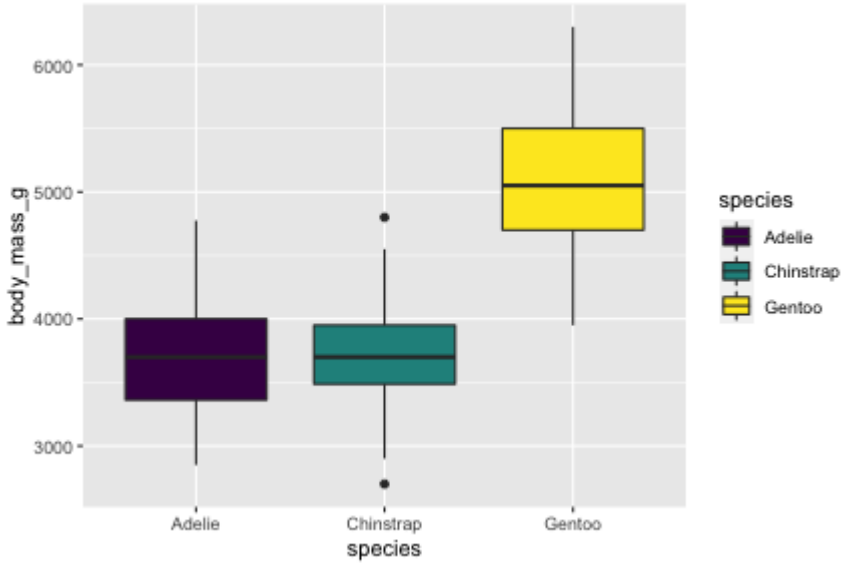
https://jfly.uni-koeln.de/color/#pallet

Viridis: perceptually uniform scales



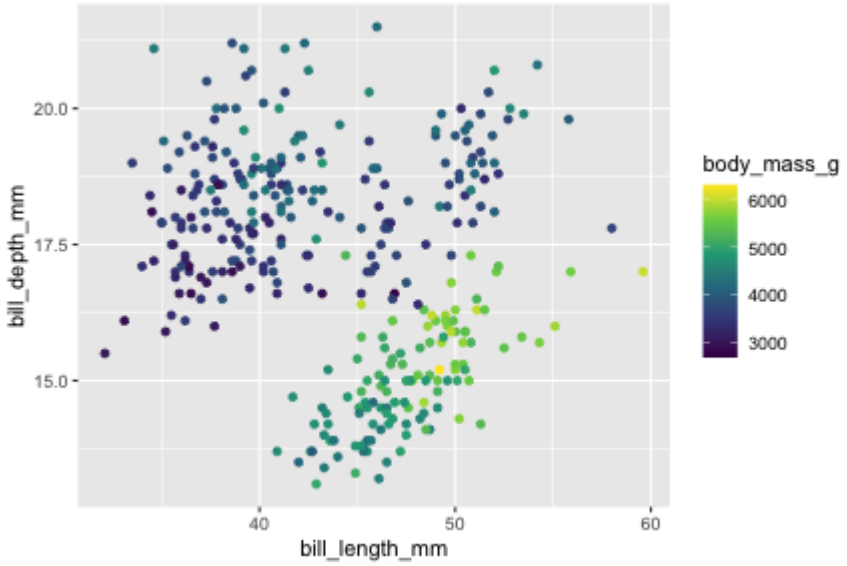
Viridis: perceptually uniform scales

```
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

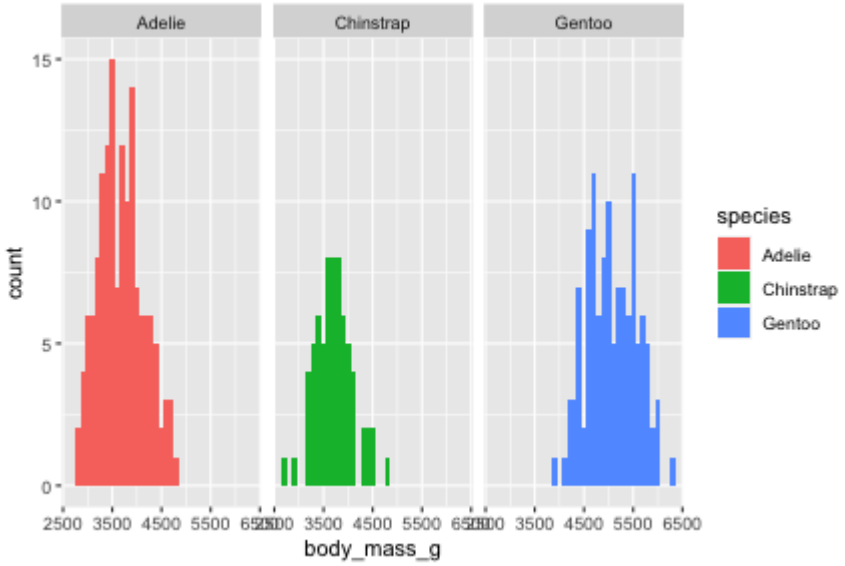
- There are 26 shapes available in R for plotting that are identified by numbers

- 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

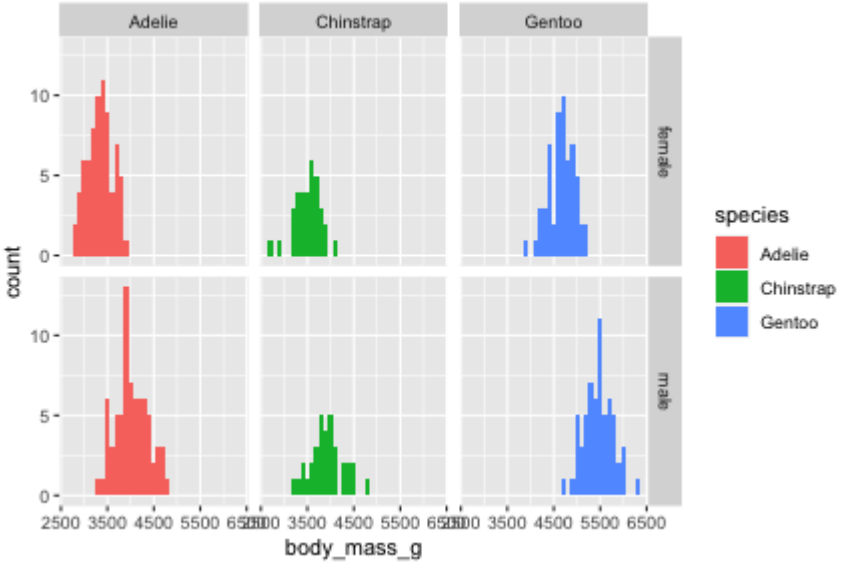
Facets

```
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

ggplot2 part of the tidyverse 3.2.1 Reference

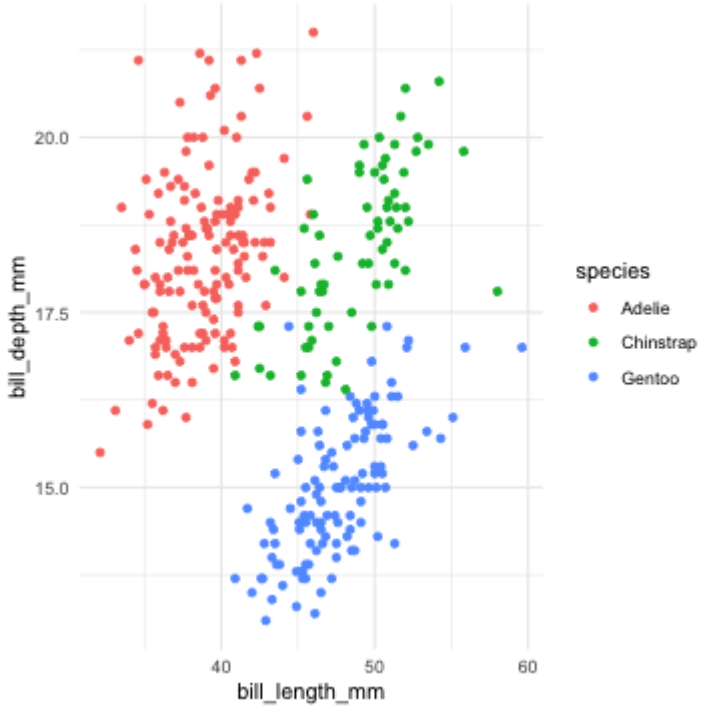
Modify components of a theme

Source: `R/theme.r`

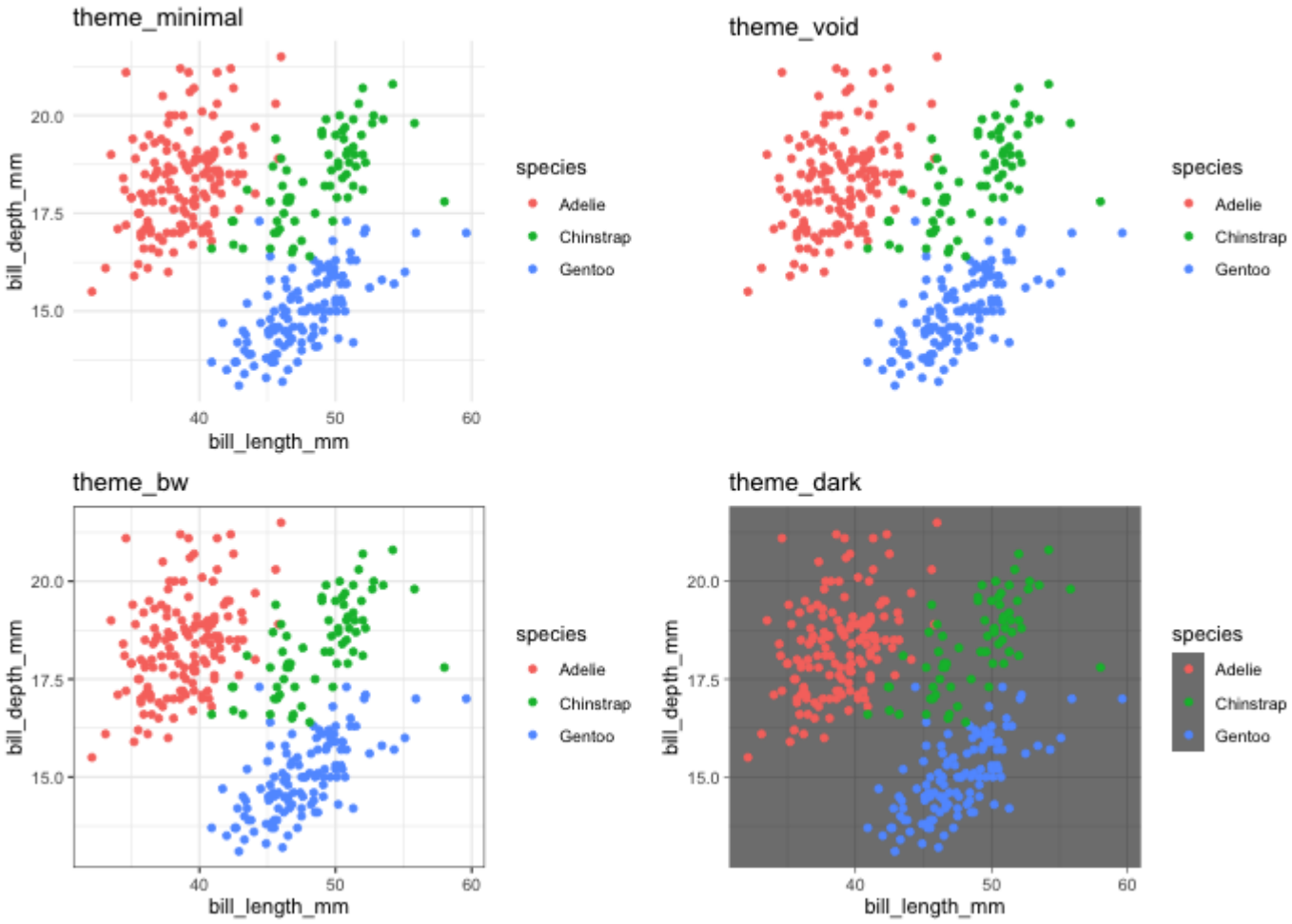
Themes are a powerful way to customize the non-data components of your plots: i.e. titles, labels, fonts, background, gridlines, and legends. Themes can be used to give plots a consistent customized look. Modify a single plot's theme using `theme()`; see `theme_update()` if you want modify the active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

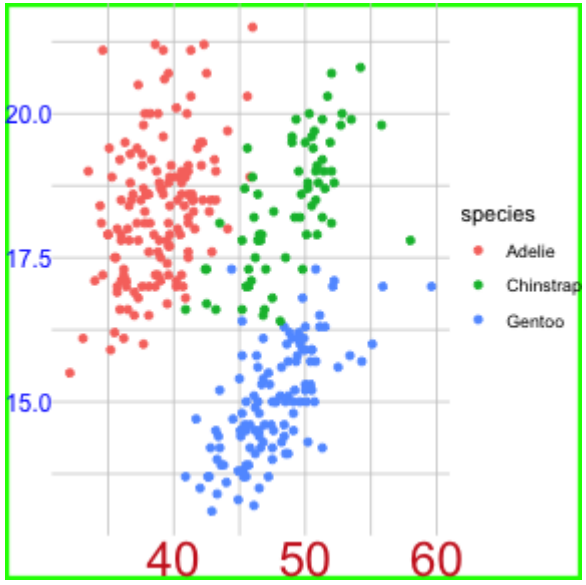


Themes


theme_minimal
theme_void
theme_bw
theme_dark
bill_depth_mm
bill_length_mm
20.0
17.5
15.0
40
50
60
species
Adelie
Chinstrap
Gentoo

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

ggplot2 3.3.5 **Reference** News Articles Extensions

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

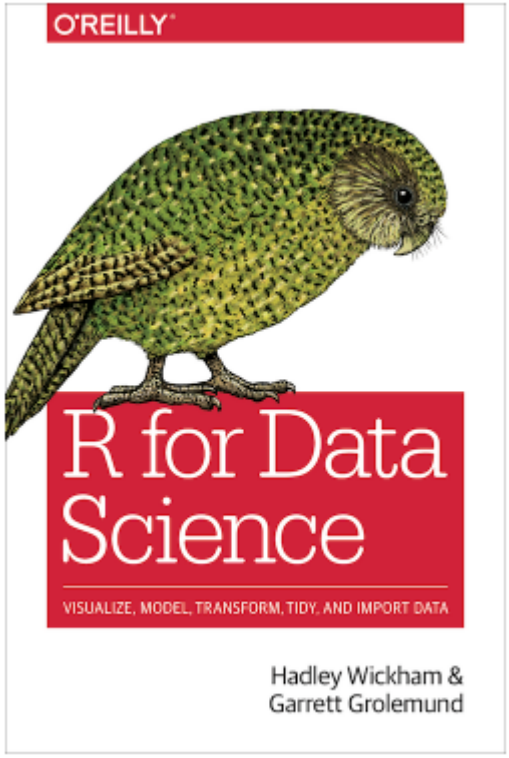
`` `+`(<gg>) `%+%` ``
Add components to a plot

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

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

https://ggplot2.tidyverse.org/reference

Further Reading



r4ds.had.co.nz

Exercises