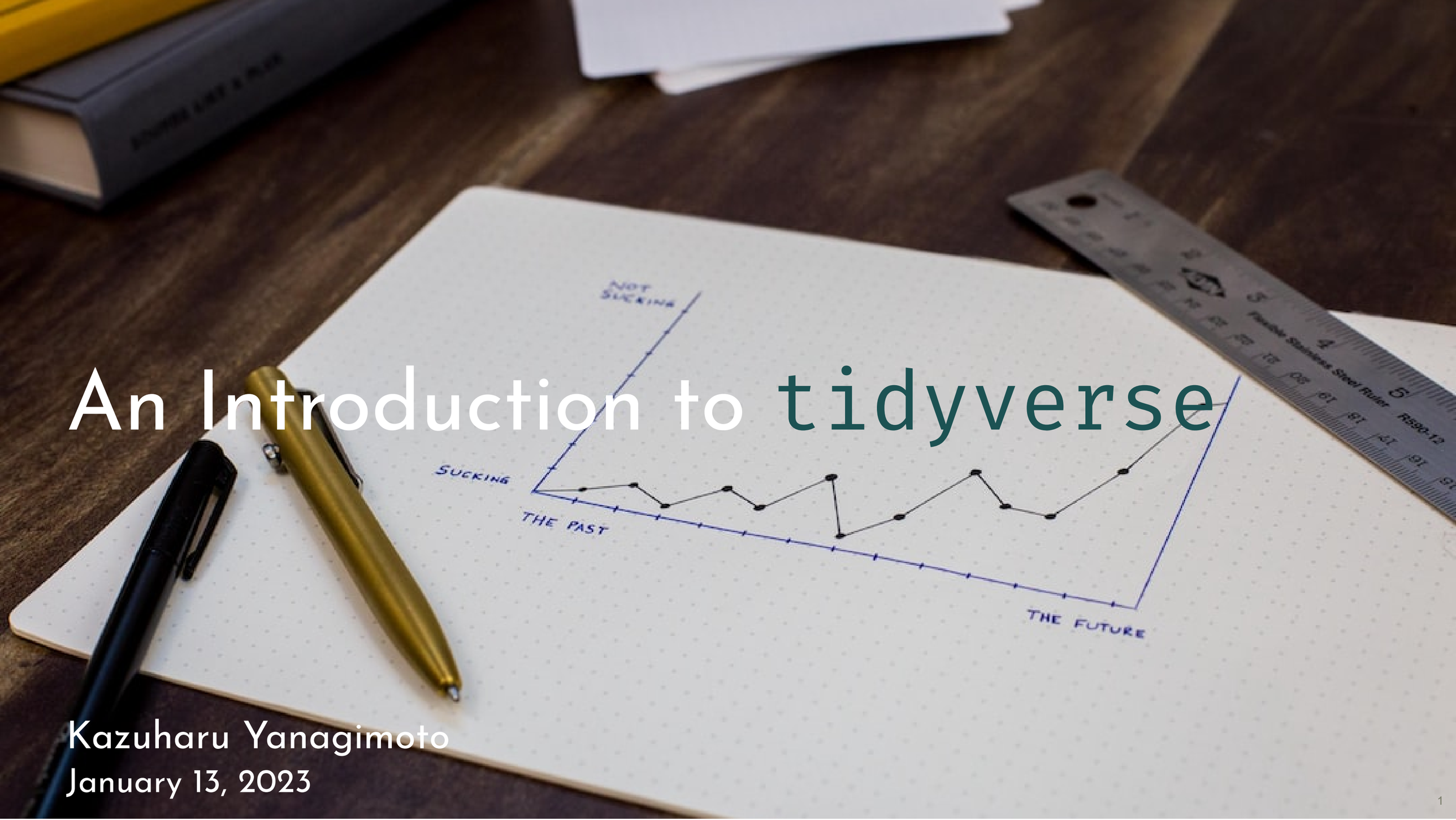


An Introduction to `tidyverse`

Kazuharu Yanagimoto

January 13, 2023

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")
```

Pipe Operator |>

|> inserts the previous object into the **first argument** of the next function

```
1 mtcars |>
2   subset(cyl==4) |>
3   head()
```

```
                mpg cyl  disp hp drat    wt  qsec vs am
gear carb
Datsun 710     22.8   4 108.0 93 3.85 2.320 18.61  1  1
4    1
Merc 240D      24.4   4 146.7 62 3.69 3.190 20.00  1  0
4    2
Merc 230       22.8   4 140.8 95 3.92 3.150 22.90  1  0
4    2
Fiat 128       32.4   4  78.7 66 4.08 2.200 19.47  1  1
4    1
Honda Civic    30.4   4  75.7 52 4.93 1.615 18.52  1  1
4    2
Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1
4    1
```

```
1 head(subset(mtcars, cyl==4))
```

```
                mpg cyl  disp hp drat    wt  qsec vs am
gear carb
Datsun 710     22.8   4 108.0 93 3.85 2.320 18.61  1  1
4    1
Merc 240D      24.4   4 146.7 62 3.69 3.190 20.00  1  0
4    2
Merc 230       22.8   4 140.8 95 3.92 3.150 22.90  1  0
4    2
Fiat 128       32.4   4  78.7 66 4.08 2.200 19.47  1  1
4    1
Honda Civic    30.4   4  75.7 52 4.93 1.615 18.52  1  1
4    2
Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1
4    1
```

Pipe Operator |>

To insert it to the second or later argument, you can use _

```
mtcars |>
  lm(mpg ~ disp, data = _)
```

```
Call:
lm(formula = mpg ~ disp, data = mtcars)

Coefficients:
(Intercept)         disp
   29.59985     -0.04122
```

- `%>%` in `tidyverse` used to be used
- `|>` is officially supported and recommended after R version 4.1
- `|>` works without loading any package, but `%>%` needs `magnitr`

tibble

In **tidyverse** packages, data frames are handled by **tibble** format.

```
tibble(
  x = 1:5,
  y = 1,
  z = x ^ 2 + y
)
```

```
# A tibble: 5 × 3
      x     y     z
  <int> <dbl> <dbl>
1     1     1     2
2     2     1     5
3     3     1    10
4     4     1    17
5     5     1    26
```

```
tibble::tribble(
  ~x, ~y, ~z,
  "a", 2, 3.6,
  "b", 1, 8.5
)
```

```
# A tibble: 2 × 3
  x         y     z
  <chr> <dbl> <dbl>
1 a         2   3.6
2 b         1   8.5
```

For detail, see Tibbles vs. data.frame in Wickham and Grolemund (2017)

dplyr Basics

glimpse

```r
library(palmerpenguins) # Example Dataset
penguins |>
  head()
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_l…¹ body_…² sex    year
  <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
1 Adelie  Torgersen           39.1          18.7         181    3750 male   2007
2 Adelie  Torgersen           39.5          17.4         186    3800 fema…  2007
3 Adelie  Torgersen           40.3          18           195    3250 fema…  2007
4 Adelie  Torgersen           NA            NA            NA      NA <NA>   2007
5 Adelie  Torgersen           36.7          19.3         193    3450 fema…  2007
6 Adelie  Torgersen           39.3          20.6         190    3650 male   2007
# … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

```r
penguins |>
  glimpse()
```

```
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel…
$ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse…
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, …
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, …
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186…
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, …
$ sex               <fct> male, female, female, NA, female, male, female, male…
$ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007…
```

select

```
penguins |>
  select(species, body_mass_g:year, starts_with("bill")) |>
  head()
```

```
# A tibble: 6 × 6
  species body_mass_g sex     year bill_length_mm bill_depth_mm
  <fct>         <int> <fct>  <int>          <dbl>         <dbl>
1 Adelie         3750 male    2007           39.1          18.7
2 Adelie         3800 female  2007           39.5          17.4
3 Adelie         3250 female  2007           40.3          18
4 Adelie           NA <NA>    2007           NA            NA
5 Adelie         3450 female  2007           36.7          19.3
6 Adelie         3650 male    2007           39.3          20.6
```

```
penguins |>
  select(-year) |>
  head()
```

```
# A tibble: 6 × 7
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_…¹ 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…
4 Adelie  Torgersen           NA            NA                  NA      NA <NA>
5 Adelie  Torgersen           36.7          19.3               193    3450 fema…
6 Adelie  Torgersen           39.3          20.6               190    3650 male
# … with abbreviated variable name ¹body_mass_g
```

filter

```
penguins |>
  filter(year >= 2008, between(bill_length_mm, 39, 41)) |>
  head()
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_l…¹ body_…² sex    year
  <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
1 Adelie  Biscoe              39.6          17.7         186    3500 fema…  2008
2 Adelie  Biscoe              40.1          18.9         188    4300 male   2008
3 Adelie  Biscoe              39            17.5         186    3550 fema…  2008
4 Adelie  Biscoe              40.6          18.8         193    3800 male   2008
5 Adelie  Torgersen           39.7          18.4         190    3900 male   2008
6 Adelie  Torgersen           39.6          17.2         196    3550 fema…  2008
# … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

```
penguins |>
  filter(!is.na(sex)) |>
  head()
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_l…¹ body_…² sex    year
  <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
1 Adelie  Torgersen           39.1          18.7         181    3750 male   2007
2 Adelie  Torgersen           39.5          17.4         186    3800 fema…  2007
3 Adelie  Torgersen           40.3          18           195    3250 fema…  2007
4 Adelie  Torgersen           36.7          19.3         193    3450 fema…  2007
5 Adelie  Torgersen           39.3          20.6         190    3650 male   2007
6 Adelie  Torgersen           38.9          17.8         181    3625 fema…  2007
# … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

rename

```
penguins |>
  rename(gender = sex) |> # I know it is penguin
  head()
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_…¹ body_…² gender  year
  <fct>   <fct>              <dbl>         <dbl>      <int>   <int> <fct>  <int>
1 Adelie  Torgersen           39.1          18.7        181    3750 male    2007
2 Adelie  Torgersen           39.5          17.4        186    3800 female  2007
3 Adelie  Torgersen           40.3          18          195    3250 female  2007
4 Adelie  Torgersen           NA            NA           NA      NA <NA>    2007
5 Adelie  Torgersen           36.7          19.3        193    3450 female  2007
6 Adelie  Torgersen           39.3          20.6        190    3650 male    2007
# … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

You can rename when you select

```
penguins |>
  select(time = year, gender = sex, starts_with("bill")) |>
  head()
```

```
# A tibble: 6 × 4
   time gender bill_length_mm bill_depth_mm
  <int> <fct>           <dbl>         <dbl>
1  2007 male             39.1          18.7
2  2007 female           39.5          17.4
3  2007 female           40.3          18
4  2007 <NA>             NA            NA
5  2007 female           36.7          19.3
6  2007 male             39.3          20.6
```

mutate

```
penguins |>
  mutate(
    flipper_length_cm = flipper_length_mm / 10,
    sex = recode(sex, male = "Male", female = "Female"),
    species_by_sex = case_when(
      species == "Adelie" & sex == "Male" ~ "Am",
      species == "Adelie" & sex == "Female" ~ "Af",
      species == "Chinstrap" & sex == "Memale" ~ "Cm",
      species == "Chinstrap" & sex == "Female" ~ "Cf",
      species == "Gentoo" & sex == "Male" ~ "Gm",
      species == "Gentoo" & sex == "Female" ~ "Gf",
    )) |>
  head()
```

```
# A tibble: 6 × 10
  species island    bill_l…¹ bill_…² flipp…³ body_…⁴ sex    year flipp…⁵ speci…⁶
  <fct>   <fct>        <dbl>   <dbl>   <int>   <int> <fct> <int>   <dbl> <chr>
1 Adelie  Torgersen     39.1    18.7     181    3750 Male   2007    18.1 Am
2 Adelie  Torgersen     39.5    17.4     186    3800 Fema…  2007    18.6 Af
3 Adelie  Torgersen     40.3    18       195    3250 Fema…  2007    19.5 Af
4 Adelie  Torgersen     NA      NA        NA      NA <NA>   2007    NA   <NA>
5 Adelie  Torgersen     36.7    19.3     193    3450 Fema…  2007    19.3 Af
6 Adelie  Torgersen     39.3    20.6     190    3650 Male   2007    19   Am
# … with abbreviated variable names ¹bill_length_mm, ²bill_depth_mm,
#   ³flipper_length_mm, ⁴body_mass_g, ⁵flipper_length_cm, ⁶species_by_sex
```

summarize & group_by

```r
penguins |>
  filter(!is.na(sex)) |>
  group_by(species, sex) |>
  summarize(
    mean_bill_length_mm = mean(bill_length_mm, na.rm = TRUE),
    sd_bill_length_mm = sd(bill_length_mm, na.rm = TRUE)
  )
```

```
# A tibble: 6 × 4
# Groups:   species [3]
  species   sex    mean_bill_length_mm sd_bill_length_mm
  <fct>     <fct>                <dbl>              <dbl>
1 Adelie    female                37.3               2.03
2 Adelie    male                  40.4               2.28
3 Chinstrap female                46.6               3.11
4 Chinstrap male                  51.1               1.56
5 Gentoo    female                45.6               2.05
6 Gentoo    male                  49.5               2.72
```

across & mutate

```
penguins |>
  mutate(across(bill_length_mm:bill_depth_mm, ~.x / 10)) |>
  rename_at(vars(bill_length_mm:bill_depth_mm), ~str_replace(.x, "mm", "cm")) |>
  head()
```

```
# A tibble: 6 × 8
  species island    bill_length_cm bill_depth_cm flipper_l…¹ body_…² sex    year
  <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
1 Adelie  Torgersen           3.91          1.87         181    3750 male   2007
2 Adelie  Torgersen           3.95          1.74         186    3800 fema…  2007
3 Adelie  Torgersen           4.03          1.8          195    3250 fema…  2007
4 Adelie  Torgersen          NA            NA             NA      NA <NA>   2007
5 Adelie  Torgersen           3.67          1.93         193    3450 fema…  2007
6 Adelie  Torgersen           3.93          2.06         190    3650 male   2007
# … with abbreviated variable names ¹flipper_length_mm, ²body_mass_g
```

- You can replace existing columns at once by `across()`
- `~` creates an anonymous (temporal) function. `.x` is the argument

across & summarize

```
penguins |>
  filter(!is.na(sex)) |>
  group_by(species, sex) |>
  summarize(
    across(bill_length_mm:body_mass_g,
           list(mean = ~mean(.x, na.rm = TRUE),
                sd = ~sd(.x, na.rm = TRUE)))
  )
```

```
# A tibble: 6 × 10
# Groups:   species [3]
  species   sex    bill_…¹ bill_…² bill_…³ bill_…⁴ flipp…⁵ flipp…⁶ body_…⁷ body_…⁸
  <fct>     <fct>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 Adelie    fema…     37.3    2.03    17.6   0.943    188.    5.60   3369.    269.
2 Adelie    male      40.4    2.28    19.1   1.02     192.    6.60   4043.    347.
3 Chinstr… fema…     46.6    3.11    17.6   0.781    192.    5.75   3527.    285.
4 Chinstr… male      51.1    1.56    19.3   0.761    200.    5.98   3939.    362.
5 Gentoo    fema…     45.6    2.05    14.2   0.540    213.    3.90   4680.    282.
6 Gentoo    male      49.5    2.72    15.7   0.741    222.    5.67   5485.    313.
# … with abbreviated variable names ¹bill_length_mm_mean, ²bill_length_mm_sd,
#   ³bill_depth_mm_mean, ⁴bill_depth_mm_sd, ⁵flipper_length_mm_mean,
#   ⁶flipper_length_mm_sd, ⁷body_mass_g_mean, ⁸body_mass_g_sd
```

- You can summarize multiple columns by multiple functions at once
- The names in the list becomes the suffix of the new columns

janitor::tabyl()

```
1 penguins |>
2   janitor::tabyl(species)
```

```
   species   n   percent
    Adelie 152 0.4418605
 Chinstrap  68 0.1976744
    Gentoo 124 0.3604651
```

```
1 penguins |>
2   janitor::tabyl(species) |>
3   janitor::adorn_percentages()
```

```
   species         n     percent
    Adelie 0.9971014 0.002898551
 Chinstrap 0.9971014 0.002898551
    Gentoo 0.9971014 0.002898551
```

```
1 penguins |>
2   janitor::tabyl(species, sex)
```

```
   species female male NA_
    Adelie     73   73   6
 Chinstrap     34   34   0
    Gentoo     58   61   5
```

```
1 penguins |>
2   janitor::tabyl(species, sex) |>
3   janitor::adorn_percentages(denominator = "row")
```

```
   species    female      male        NA_
    Adelie 0.4802632 0.4802632 0.03947368
 Chinstrap 0.5000000 0.5000000 0.00000000
    Gentoo 0.4677419 0.4919355 0.04032258
```

`janitor::tabyl()` gives frequency tables in `tibble` format

`*_join()`

- `inner_join(df1, df2)`
- `left_join(df1, df2)`
- `right_join(df1, df2)`
- `full_join(df1, df2)`
- `semi_join(df1, df2)`
- `anti_join(df1, df2)`

In most of cases, **`left_join`** is enough. For details of other functions, read Rational data section in Wickham and Grolemund (2017).

left_join()

```r
library(nycflights13) # Dataset
flights |> glimpse()
```

```
Rows: 336,776
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2…
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, …
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, …
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1…
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,…
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,…
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1…
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "…
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4…
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394…
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",…
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",
```

```r
planes |> glimpse()
```

```
Rows: 3,322
Columns: 9
$ tailnum      <chr> "N10156", "N102UW", "N103US", "N104UW", "N10575", "N105UW…
$ year         <int> 2004, 1998, 1999, 1999, 2002, 1999, 1999, 1999, 1999, 199…
$ type         <chr> "Fixed wing multi engine", "Fixed wing multi engine", "Fi…
$ manufacturer <chr> "EMBRAER", "AIRBUS INDUSTRIE", "AIRBUS INDUSTRIE", "AIRBU…
$ model        <chr> "EMB-145XR", "A320-214", "A320-214", "A320-214", "EMB-145…
$ engines      <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
$ seats        <int> 55, 182, 182, 182, 55, 182, 182, 182, 182, 182, 55, 55, 5…
$ speed        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ engine       <chr> "Turbo-fan", "Turbo-fan", "Turbo-fan", "Turbo-fan", "Turb…
```

left_join()

```
flights |>
  left_join(planes, by = "tailnum", suffix = c("", "_built")) |>
  select(year, month, day, dep_time, arr_time, carrier, flight, tailnum, year_built, type, model)
```

```
# A tibble: 336,776 × 11
    year month   day dep_time arr_t…¹ carrier flight tailnum year_…² type  model
   <int> <int> <int>    <int>   <int> <chr>    <int> <chr>     <int> <chr> <chr>
 1  2013     1     1      517     830 UA        1545 N14228     1999 Fixe… 737-…
 2  2013     1     1      533     850 UA        1714 N24211     1998 Fixe… 737-…
 3  2013     1     1      542     923 AA        1141 N619AA     1990 Fixe… 757-…
 4  2013     1     1      544    1004 B6         725 N804JB     2012 Fixe… A320…
 5  2013     1     1      554     812 DL         461 N668DN     1991 Fixe… 757-…
 6  2013     1     1      554     740 UA        1696 N39463     2012 Fixe… 737-…
 7  2013     1     1      555     913 B6         507 N516JB     2000 Fixe… A320…
 8  2013     1     1      557     709 EV        5708 N829AS     1998 Fixe… CL-6…
 9  2013     1     1      557     838 B6          79 N593JB     2004 Fixe… A320…
10  2013     1     1      558     753 AA         301 N3ALAA       NA <NA>  <NA>
# … with 336,766 more rows, and abbreviated variable names ¹arr_time,
#   ²year_built
```

If the key name is different, you can write `by = c("LEFT_KEY" = "RIGHT_KEY)`

tidyr Basics

Tidy Data Approach

ⓘ Tidy Data (Wickham (2014), Wickham and Grolemund (2017))

1. Each *variable* must have its own column.
2. Each *observation* must have its own row.
3. Each *value* must have its own cell.

Hence, *tidy data* is in long format rather than in wide format (unlike Stata...)

pivot_longer & pivot_wider

This data is not *tidy*

```
  1  tb1
```

```
# A tibble: 3 × 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2666
2 Brazil       37737  80488
3 China       212258 213766
```

```
  1  tb2
```

```
# A tibble: 6 × 4
  country      year type            count
  <chr>       <dbl> <chr>           <dbl>
1 Afghanistan  1999 cases             745
2 Afghanistan  1999 population  19987071
3 Afghanistan  2000 cases            2666
4 Afghanistan  2000 population  20595360
5 Brazil       1999 cases           37737
6 Brazil       1999 population 172006362
```

Make it *tidy*

```
  1  tb1 |>
  2    pivot_longer(c(`1999`, `2000`),
  3      names_to = "year", values_to = "cases")
```

```
# A tibble: 6 × 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Afghanistan 2000    2666
3 Brazil      1999   37737
4 Brazil      2000   80488
5 China       1999  212258
6 China       2000  213766
```

```
  1  tb2 |>
  2    pivot_wider(names_from = type, values_from = count)
```

```
# A tibble: 3 × 4
  country      year cases population
  <chr>       <dbl> <dbl>      <dbl>
1 Afghanistan  1999   745   19987071
2 Afghanistan  2000  2666   20595360
3 Brazil       1999 37737  172006362
```

separate & unite

This data is not *tidy*

```
tb1
```

```
# A tibble: 6 × 3
  country      year rate
  <chr>       <dbl> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
```

```
tb2
```

```
# A tibble: 6 × 5
  country     century year_d2  cases population
  <chr>         <dbl>   <dbl>  <dbl>      <dbl>
1 Afghanistan      19      99    745   19987071
2 Afghanistan      20       0   2666   20595360
3 Brazil           19      99  37737  172006362
4 Brazil           20       0  80488  174504898
5 China            19      99 212258 1272915272
6 China            20       0 213766 1280428583
```

Make it *tidy*

```
tb1 |>
  separate(rate, into = c("cases", "population")) |>
  mutate(across(c("cases", "population"), as.numeric))
```

```
# A tibble: 6 × 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

```
tb2 |>
  unite(year, century, year_d2, sep = "") |>
  mutate(year = as.integer(year))
```

```
# A tibble: 6 × 4
  country      year  cases population
  <chr>       <int>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan   200   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil        200  80488  174504898
5 China        1999 212258 1272915272
6 China         200 213766 1280428583
```

ggplot2 Basics

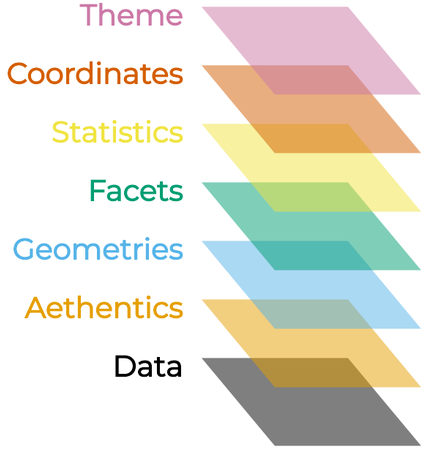
Grammer of Graphics

ⓘ Grammar of Graphics (Wilkinson 1999)

1. Graphics are made of distinct layers of grammatical elements
2. Plots are built with appropriate aesthetic mappings to make these plots meaningful

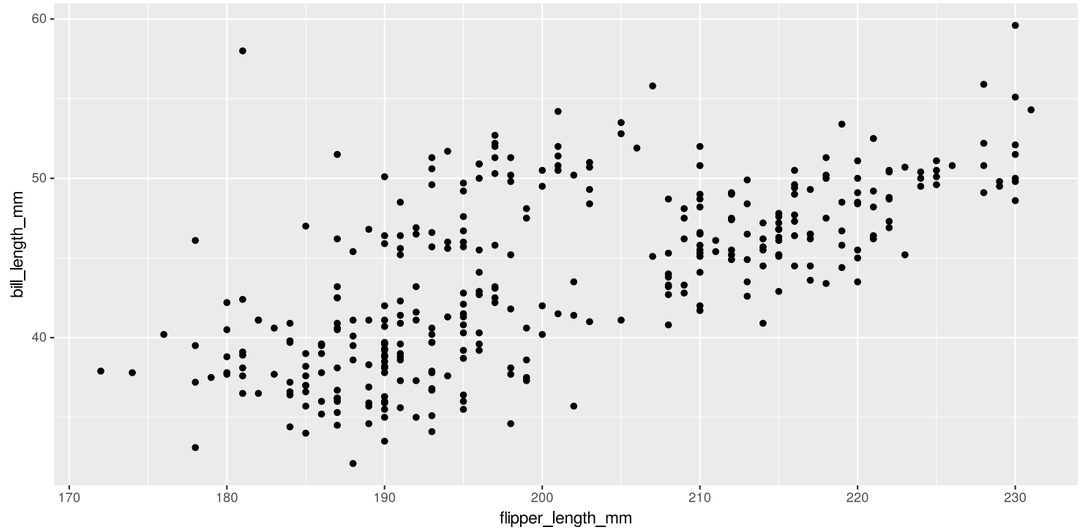
`ggplot2`

- designed to represent the grammar of graphics
- connects layers by `+` instead of `|>`

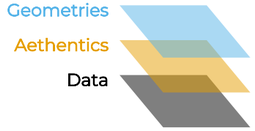


Aethentics & Geometries

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point()
```

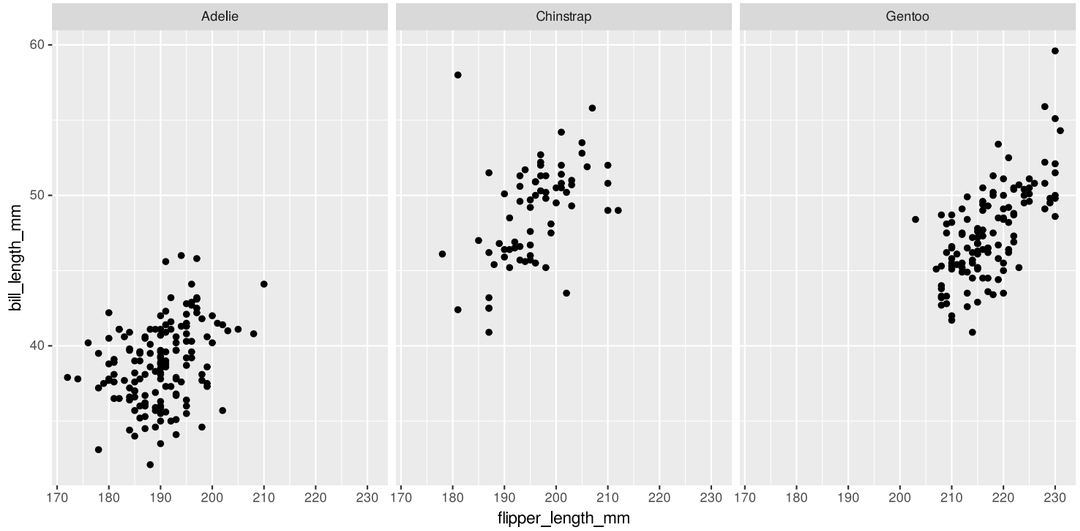


`geom_*()` functions specify the type of plots (density, bar, scatter,...)



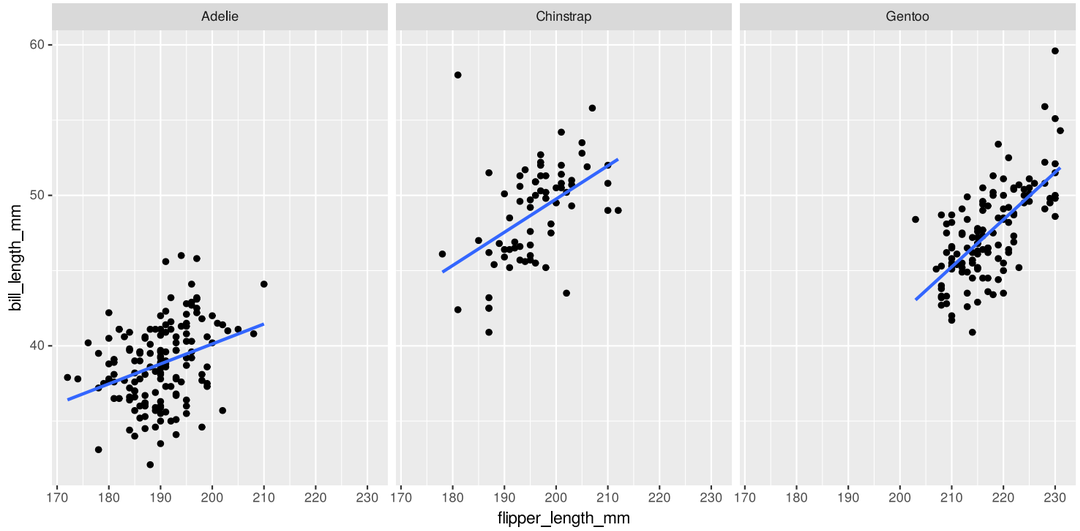
Facets

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point() +
  facet_wrap(~species)
```



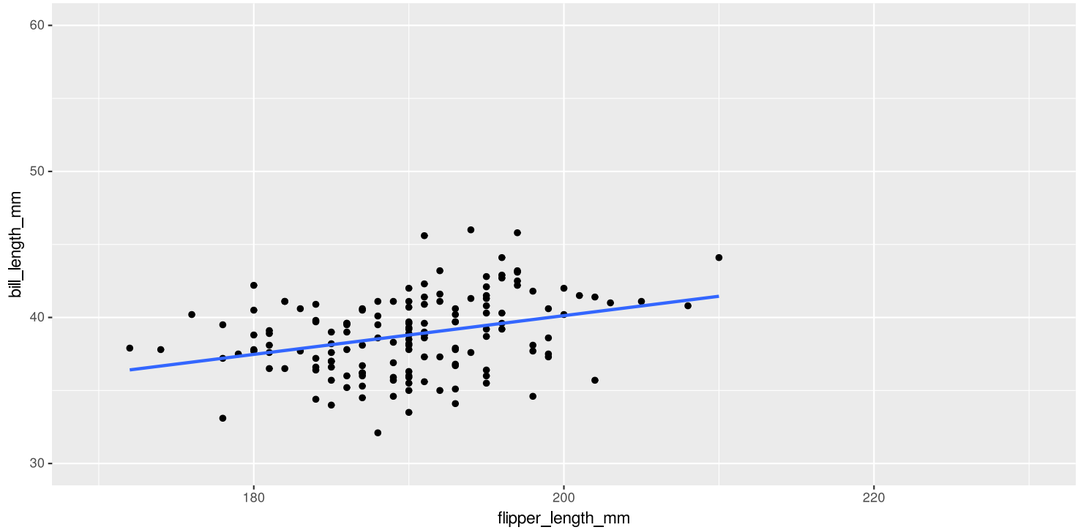
Statistics

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point() +
  facet_wrap(~species) +
  stat_smooth(formula = 'y ~ x',
              method = "lm", se = FALSE)
```

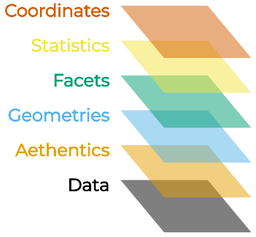


Coordinates

```
penguins |>
  filter(species == "Adelie") |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point() +
  stat_smooth(formula = 'y ~ x',
              method = "lm", se = FALSE) +
  coord_cartesian(xlim = c(170, 230),
                  ylim = c(30, 60))
```

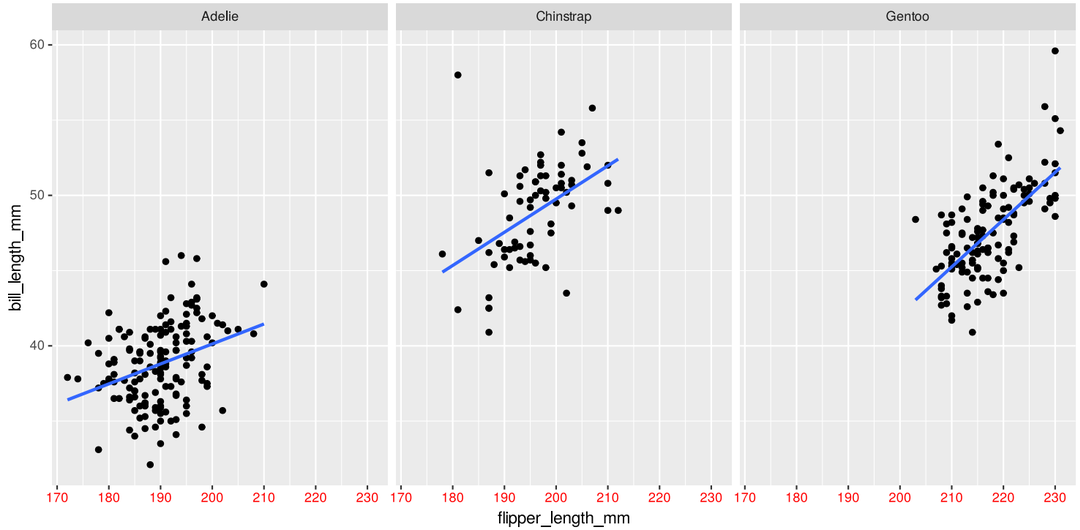


- `coord_cartesan()` is a zoom-in and zoom-out function
- It is more recommended than `scale_*_continuous()`, which removes points if they are out of `xlim` (`ylim`)

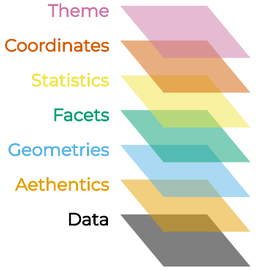


Themes

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point() +
  facet_wrap(~species) +
  stat_smooth(formula = 'y ~ x',
              method = "lm", se = FALSE) +
  theme(axis.text.x = element_text(color = "red"))
```

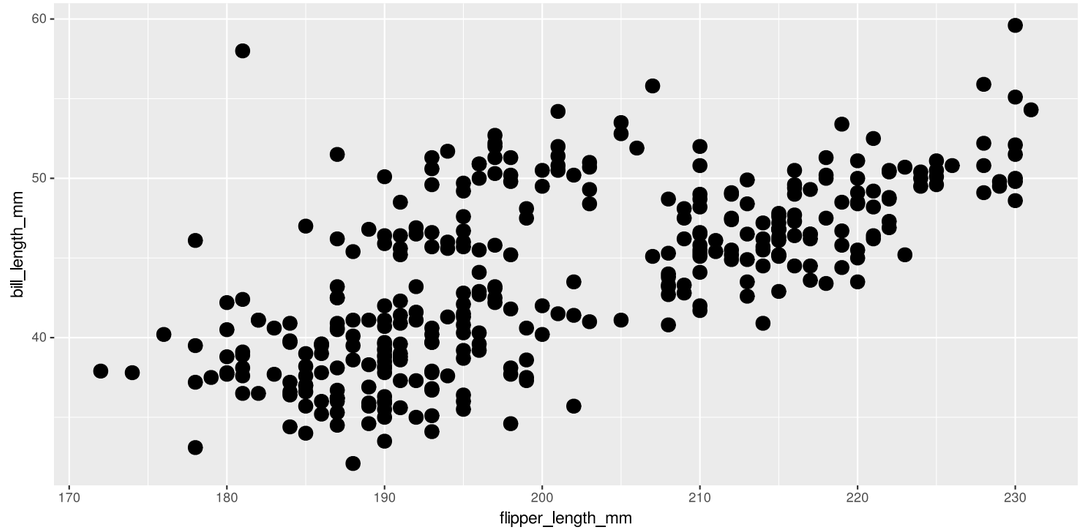


You can configure all the elements in a plot. For more detail, read documents



Size of Points

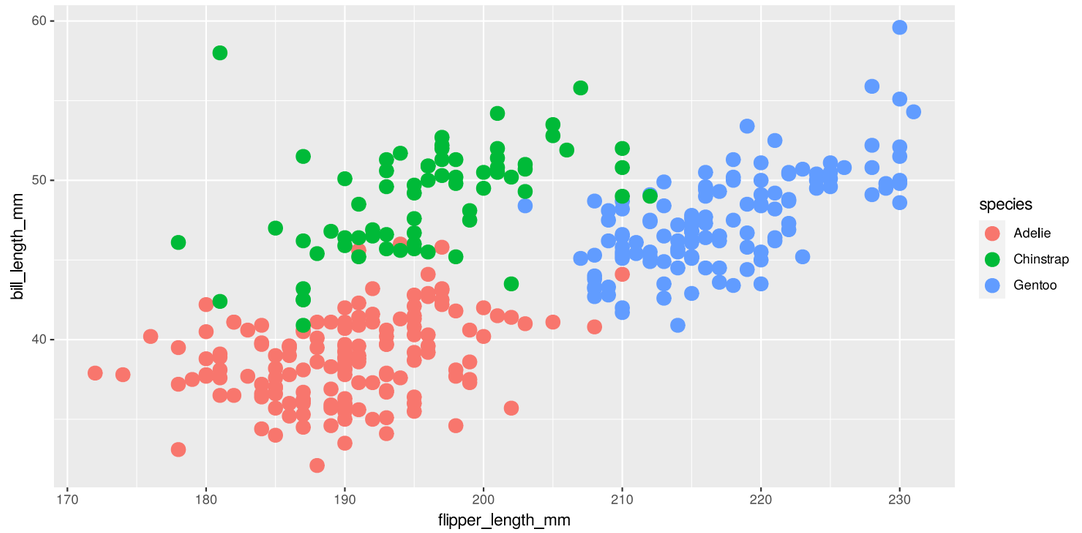
```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm)) +
  geom_point(size = 4)
```



For `geom_line()`, use `linewidth = 4`

Color

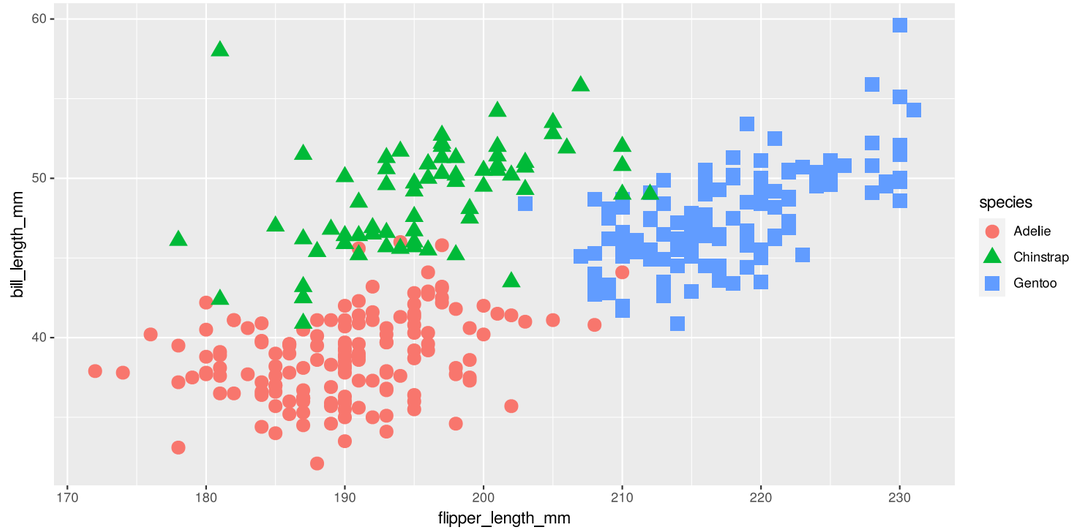
```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm,
             color = species)) +
  geom_point(size = 4)
```



For `geom_bar()`, use `fill = species`

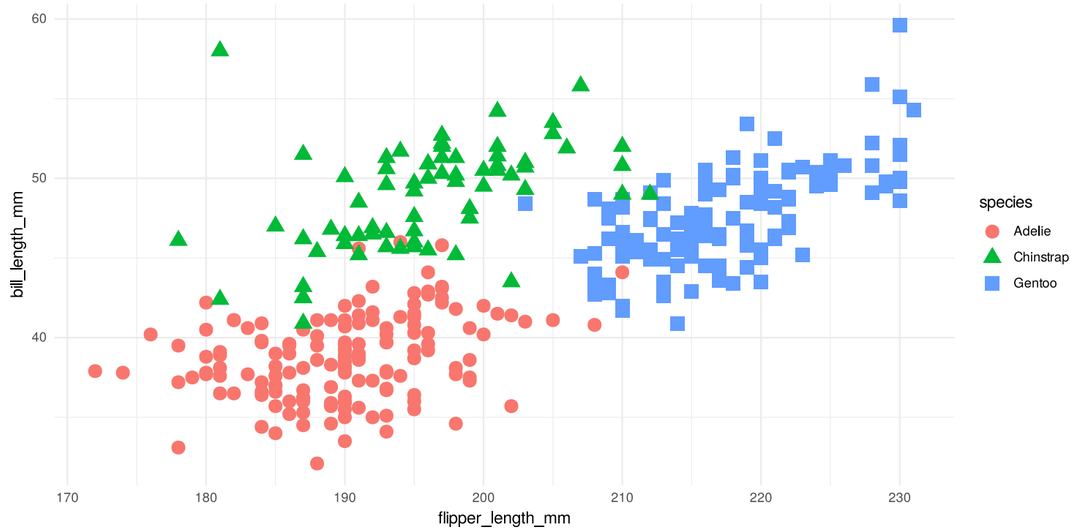
Color & Shape

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm,
             color = species,
             shape = species)) +
  geom_point(size = 4)
```



Built-in Themes

```
penguins |>
  ggplot(aes(x = flipper_length_mm,
             y = bill_length_mm,
             color = species,
             shape = species)) +
  geom_point(size = 4) +
  theme_minimal()
```



For more built-in themes, you can find in ggtheme

Handson

- Open the Quarto Notebook `code/10_notebooks/handson_tidyverse.qmd`
- Answer the question in the notebook
- You can find my answer as a qmd file or webpage
- The webpage version folds codes. It allows you to read only the code where you get stacked

References

Wickham, Hadley. 2014. “Tidy Data.” *Journal of Statistical Software* 59 (10): 1–23. https://doi.org/10.18637/jss.v059.i10.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st edition. Sebastopol, CA: O’Reilly Media. https://r4ds.had.co.nz/.

Wilkinson, Leland. 1999. *The Grammar of Graphics*. Springer New York.