Exploratory Data Analysis and Data Visualization

Emile Latour, Nicky Wakim, Meike Niederhausen

January 21, 2026

Today's plan

1. Tools for exploratory data analysis (EDA)
2. Data wrangling with `dplyr`
3. Summary tables with `janitor` and `rstatix`
4. Data visualization with `ggplot2`

What is exploratory data analysis?

Exploratory Data Analysis (EDA) is the process of:

- Inspecting and summarizing data
- Identifying patterns, outliers, and relationships
- Understanding the structure before formal analysis

EDA helps answer questions like:

- What variables do I have?
- What does the distribution look like?
- Are there missing values?
- How do variables relate to each other?

The tidyverse ecosystem

Artwork by @allison_horst

The tidyverse philosophy

The **tidyverse** is a collection of R packages designed for data science.

Key principles:

- Consistent syntax across packages
- Functions designed to work together
- Human-readable code
- Works well with the pipe operator %>%

- **ggplot2** - data visualisation
- **dplyr** - data manipulation
- **tidyr** - tidy data
- **readr** - read rectangular data
- **purrr** - functional programming
- **tibble** - modern data frames
- **stringr** - string manipulation
- **forcats** - factors
- and many more ...

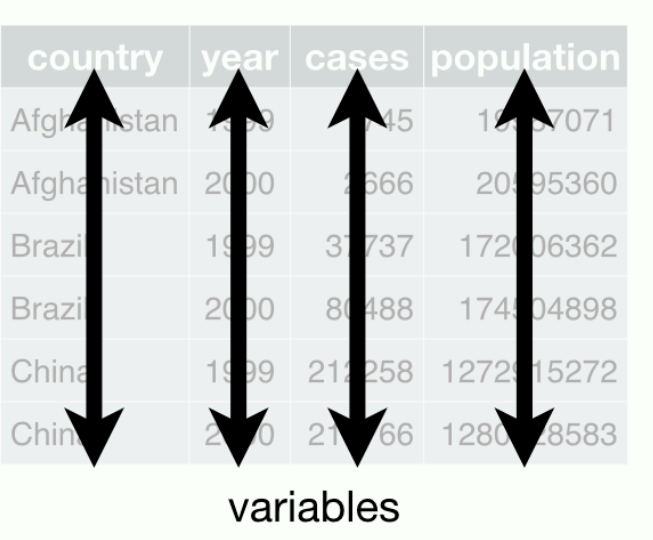
Packages for EDA

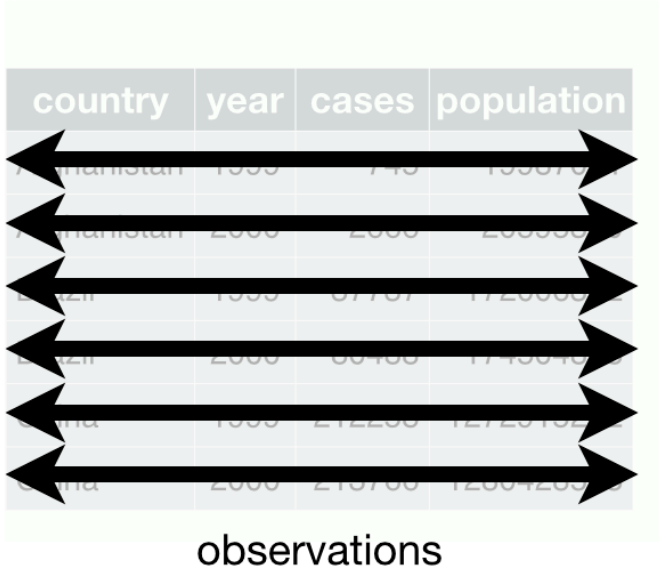
Packages we'll use today:

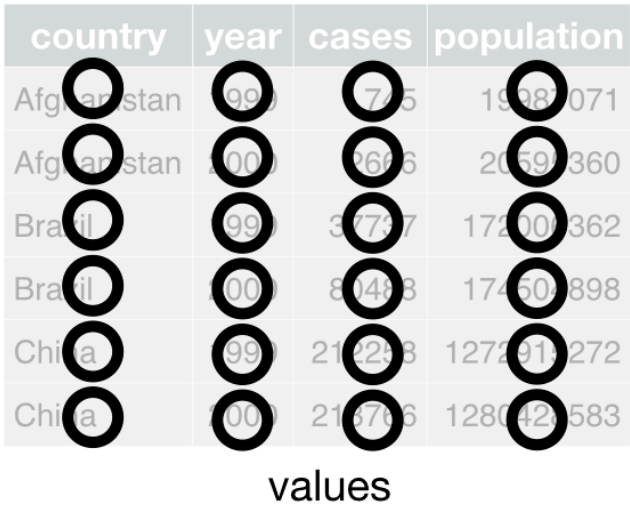
- `dplyr` - data manipulation
- `ggplot2` - visualization
- `janitor` - clean tables
- `rstatix` - summary statistics
- `skimr` - quick data summaries

Not all from the tidyverse, but came after and follow the philosophy.

Tidy data[1]







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

1. Source: R for Data Science. Grolemund and Wickham.

Today's dataset: NHANES

National Health and Nutrition Examination Survey (NHANES)

- CDC survey assessing health and nutritional status of adults and children in the US
- Combines interviews, physical examinations, and laboratory tests
- Today we'll use a sample of adult participants

Loading the data:

```
1 library(oibiostat)
2 data("nhanes.samp.adult")
```

First look at the data

Always start by getting a sense of what's in your dataset.

How many rows and columns?

```
1 dim(nhanes.samp.adult)
```

```
[1] 135  76
```

What are the variable names?

```
1 names(nhanes.samp.adult)
```

```
 [1] "ID"               "SurveyYr"         "Gender"           "Age"
 [5] "AgeDecade"        "AgeMonths"        "Race1"            "Race3"
 [9] "Education"        "MaritalStatus"    "HHIncome"         "HHIncomeMid"
[13] "Poverty"          "HomeRooms"        "HomeOwn"          "Work"
[17] "Weight"           "Length"           "HeadCirc"         "Height"
[21] "BMI"              "BMICatUnder20yrs" "BMI_WHO"          "Pulse"
[25] "BPSysAve"         "BPDiaAve"         "BPSys1"           "BPDia1"
[29] "BPSys2"           "BPDia2"           "BPSys3"           "BPDia3"
[33] "Testosterone"     "DirectChol"       "TotChol"          "UrineVol1"
[37] "UrineFlow1"       "UrineVol2"        "UrineFlow2"       "Diabetes"
[41] "DiabetesAge"      "HealthGen"        "DaysPhysHlthBad"
"DaysMentHlthBad"
[45] "LittleInterest"   "Depressed"        "nPregnancies"     "nBabies"
```

What does the first few rows look like?

```
1 head(nhanes.samp.adult, 3)
```

```
# A tibble: 3 × 76
     ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
  <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
1 63147 2011_12  male      41 " 40-49"         NA White White Some College
2 57165 2009_10  male      48 " 40-49"        586 Black <NA>  High School
3 69465 2011_12  female    50 " 50-59"         NA White White College Grad
# ℹ 67 more variables: MaritalStatus <fct>, HHIncome <fct>, HHIncomeMid
<int>,
#   Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>, Work <fct>, Weight <dbl>,
#   Length <dbl>, HeadCirc <dbl>, Height <dbl>, BMI <dbl>,
#   BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>, BPSysAve <int>,
#   BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>, BPDia2 <int>,
#   BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>, DirectChol <dbl>,
```

Tibbles: tidyverse data frames

A **tibble** is the tidyverse version of a data frame.

```
1 nhanes.samp.adult
```

```
# A tibble: 135 × 76
      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1   Race3 Education
 * <int> <fct>    <fct>  <int> <fct>         <int> <fct>   <fct> <fct>
 1 63147 2011_12  male      41 " 40-49"         NA White   White Some College
 2 57165 2009_10  male      48 " 40-49"        586 Black   <NA>  High School
 3 69465 2011_12  female    50 " 50-59"         NA White   White College Grad
 4 57313 2009_10  female    74 " 70+"          889 White   <NA>  College Grad
 5 56047 2009_10  female    27 " 20-29"        329 White   <NA>  9 - 11th Grade
 6 57056 2009_10  male      26 " 20-29"        316 Mexican <NA>  High School
 7 54643 2009_10  female    41 " 40-49"        503 White   <NA>  College Grad
 8 53095 2009_10  female    52 " 50-59"        632 White   <NA>  9 - 11th Grade
 9 67434 2011_12  male      56 " 50-59"         NA White   White College Grad
10 60273 2009_10  male      39 " 30-39"        472 Black   <NA>  9 - 11th Grade
```

Benefits of tibbles:

- Prints only first 10 rows (doesn't flood console)
- Shows column types
- Never converts strings to factors automatically

glimpse() - quick data structure

`glimpse()` from the `dplyr` package shows you the structure of your data in a compact format.

```
1 glimpse(nhanes.samp.adult)
```

```
Rows: 135
Columns: 76
$ ID              <int> 63147, 57165, 69465, 57313, 56047, 57056, 54643, 5309…
$ SurveyYr        <fct> 2011_12, 2009_10, 2011_12, 2009_10, 2009_10, 2009_10,…
$ Gender          <fct> male, male, female, female, female, male, female, fem…
$ Age             <int> 41, 48, 50, 74, 27, 26, 41, 52, 56, 39, 32, 72, 35, 2…
$ AgeDecade       <fct>  40-49,  40-49,  50-59,  70+,  20-29,  20-29,  40-49,…
$ AgeMonths       <int> NA, 586, NA, 889, 329, 316, 503, 632, NA, 472, 391, 8…
$ Race1           <fct> White, Black, White, White, White, Mexican, White, Wh…
$ Race3           <fct> White, NA, White, NA, NA, NA, NA, NA, White, NA, NA, …
$ Education       <fct> Some College, High School, College Grad, College Grad…
$ MaritalStatus   <fct> Married, Married, Divorced, Widowed, NeverMarried, Ne…
$ HHIncome        <fct> 25000-34999, 25000-34999, 35000-44999, 45000-54999, 7…
```

- Each row shows: variable name, type, and first few values
- Easier to read than `str()` for large datasets

skim() - comprehensive summary

`skim()` from the `skimr` package gives you detailed summaries and mini visualizations.

```
1 skim(nhanes.samp.adult)
```

```
> skim(nhanes.samp.adult)
── Data Summary ────────────────────
                           Values
Name                       nhanes.samp.adult
Number of rows             135
Number of columns          76
_______________________
Column type frequency:
  factor                   31
  numeric                  45
_______________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   skim_variable    n_missing complete_rate ordered n_unique top_counts
 1 SurveyYr                 0         1     FALSE          2 "201: 69, 200: 66"
 2 Gender                   0         1     FALSE          2 "fem: 70, mal: 65"
 3 AgeDecade                2         0.985 FALSE          6 " 20: 32,  30: 29,  50: 25,  40: 23"
 4 Race1                    0         1     FALSE          5 "Whi: 95, Bla: 12, Mex: 11, Oth: 9"
 5 Race3                   66         0.511 FALSE          6 "Whi: 49, His: 6, Mex: 6, Bla: 5"
 6 Education                0         1     FALSE          5 "Col: 46, Som: 39, Hig: 27, 9 -: 16"
 7 MaritalStatus            0         1     FALSE          6 "Mar: 78, Nev: 25, Div: 13, Liv: 13"
 8 HHIncome                10         0.926 FALSE         12 "mor: 27, 350: 18, 250: 14, 750: 14"
 9 HomeOwn                  0         1     FALSE          3 "Own: 88, Ren: 43, Oth: 4"
10 Work                     0         1     FALSE          3 "Wor: 90, Not: 40, Loo: 5"
11 BMICatUnder20yrs       135         0     FALSE          0 "Und: 0, Nor: 0, Ove: 0, Obe: 0"
12 BMI_WHO                  1         0.993 FALSE          4 "30.: 49, 25.: 46, 18.: 35, 12.: 4"
13 Diabetes                 0         1     FALSE          2 "No: 122, Yes: 13"
14 HealthGen               11         0.919 FALSE          5 "Goo: 45, Vgo: 32, Fai: 30, Exc: 16"
15 LittleInterest          11         0.919 FALSE          3 "Non: 92, Sev: 27, Mos: 5"
16 Depressed               11         0.919 FALSE          3 "Non: 94, Sev: 21, Mos: 9"
17 SleepTrouble             0         1     FALSE          2 "No: 99, Yes: 36"
18 PhysActive               0         1     FALSE          2 "Yes: 76, No: 59"
19 TVHrsDay                66         0.511 FALSE          7 "1_h: 15, 2_h: 15, 3_h: 15, 4_h: 10"
20 CompHrsDay              66         0.511 FALSE          7 "0_h: 16, 0_t: 14, 1_h: 14, 2_h: 12"
21 Alcohol12PlusYr         11         0.919 FALSE          2 "Yes: 102, No: 22"
22 SmokeNow                70         0.481 FALSE          2 "Yes: 36, No: 29"
23 Smoke100                 0         1     FALSE          2 "No: 70, Yes: 65"
24 Smoke100n                0         1     FALSE          2 "Non: 70, Smo: 65"
25 Marijuana               38         0.719 FALSE          2 "Yes: 58, No: 39"
26 RegularMarij            38         0.719 FALSE          2 "No: 65, Yes: 32"
27 HardDrugs               26         0.807 FALSE          2 "No: 87, Yes: 22"
28 SexEver                 26         0.807 FALSE          2 "Yes: 106, No: 3"
29 SameSex                 26         0.807 FALSE          2 "No: 103, Yes: 6"
30 SexOrientation          41         0.696 FALSE          2 "Het: 93, Bis: 1, Hom: 0"
31 PregnantNow             95         0.296 FALSE          2 "No: 38, Yes: 2, Unk: 0"

── Variable type: numeric ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   skim_variable   n_missing complete_rate      mean        sd       p0       p25       p50       p75      p100 hist
 1 ID                      0         1      62140.     5723.    51780     57277     62229     67475     71581 "▇▇▇▇▇"
 2 Age                     0         1         44.5      16.1       21        30.5      42        57        80   "▇▆▅▃▂"
 3 AgeMonths              70         0.481    520.      189.      259       332       486       683       910   "▇▆▅▃▂"
 4 HHIncomeMid            10         0.926  57180     31608.     2500     30000     50000     87500    100000 "▂▃▂▃▇"
 5 Poverty                11         0.919      2.78      1.70      0         1.28      2.71      4.82      5   "▇▃▂▂▂"
 6 HomeRooms               0         1          5.96      2.31      1         4.5       6         7        13   "▂▇▅▂▁"
 7 Weight                  0         1         84.0      23.7      39.7      67.4      80.7      98.5     188.  "▂▇▃▁▁"
 8 Length                135         0        NaN       NA        NA        NA        NA        NA        NA   " "
 9 HeadCirc              135         0        NaN       NA        NA        NA        NA        NA        NA   " "
```

The pipe operator: %>%

The pipe operator %>% takes the output from one function and passes it as the first argument to the next function.

Think of it as: "and then..."

I want to find my keys, then start my car, then drive to work, then park my car.

Nested

```
park(drive(start_car(find("keys")),
           to = "work"))
```

Piped

```
find("keys") %>%
  start_car() %>%
  drive(to = "work") %>%
  park()
```

The pipe operator: %>%

Without pipes (nested)

Read from inside-out

```
head(nhanes.samp.adult, 3)
```

```
# A tibble: 3 × 76
     ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
  <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
1 63147 2011_12  male      41 " 40-49"         NA White White Some College
2 57165 2009_10  male      48 " 40-49"        586 Black <NA>  High School
3 69465 2011_12  female    50 " 50-59"         NA White White College Grad
# ℹ 67 more variables: MaritalStatus <fct>, HHIncome <fct>, HHIncomeMid
<int>,
#   Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>, Work <fct>, Weight <dbl>,
#   Length <dbl>, HeadCirc <dbl>, Height <dbl>, BMI <dbl>,
#   BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>, BPSysAve <int>,
#   BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>, BPDia2 <int>,
#   BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>, DirectChol <dbl>,
```

With pipes

Read left to right

```
nhanes.samp.adult %>%
  head(3)
```

```
# A tibble: 3 × 76
     ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
  <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
1 63147 2011_12  male      41 " 40-49"         NA White White Some College
2 57165 2009_10  male      48 " 40-49"        586 Black <NA>  High School
3 69465 2011_12  female    50 " 50-59"         NA White White College Grad
# ℹ 67 more variables: MaritalStatus <fct>, HHIncome <fct>, HHIncomeMid
<int>,
#   Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>, Work <fct>, Weight <dbl>,
#   Length <dbl>, HeadCirc <dbl>, Height <dbl>, BMI <dbl>,
#   BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>, BPSysAve <int>,
#   BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>, BPDia2 <int>,
#   BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>, DirectChol <dbl>,
```

The pipe operator: %>%

Without pipes (nested)

Harder to read when operations stack up:

```
mean(filter(nhanes.samp.adult,
            Diabetes == "Yes")$BMI,
     na.rm = TRUE)
```

With pipes

More readable for complex operations:

```
nhanes.samp.adult %>%
  filter(Diabetes == "Yes") %>%
  pull(BMI) %>%
  mean(na.rm = TRUE)
```

dplyr: The grammar of data manipulation

`dplyr` provides a consistent set of **verbs** for data manipulation:

Function	What it does
`filter()`	Keep rows that meet conditions
`select()`	Keep or drop columns
`mutate()`	Create or modify columns
`arrange()`	Sort rows
`group_by()`	Group data for summaries
`summarize()`	Calculate summary statistics

These verbs can be chained together with `%>%` for powerful data transformations.

filter() - subsetting rows

`filter()` keeps rows that meet specified conditions.

Single condition

```
nhanes.samp.adult %>%
  filter(Diabetes == "Yes")
```

Multiple conditions (AND)

```
nhanes.samp.adult %>%
  filter(Gender == "female",
         Age > 50)

# OR

nhanes.samp.adult %>%
  filter(Gender == "female" &
         Age > 50)
```

Using OR (|)

```
nhanes.samp.adult %>%
  filter(Diabetes == "Yes" |
         Age > 70)
```

Numeric comparisons

```
nhanes.samp.adult %>%
  filter(BMI >= 30)
```

Comparison operators in R

When using `filter()`, you'll use these comparison operators:

Operator	Meaning
`==`	Equal to
`!=`	Not equal to
`>`	Greater than
`>=`	Greater than or equal to
`<`	Less than
`<=`	Less than or equal to
`&` or `,`	AND (both conditions must be true)
`|`	OR (at least one condition must be true)
`is.na()`	Check if value is missing

select() - choosing columns

`select()` lets you keep or remove specific columns.

Select specific columns

```
nhanes.samp.adult %>%
  select(Age, Gender, BMI)
```

Drop columns (i.e. de-select)

```
nhanes.samp.adult %>%
  select(-ID)
```

Select a range of columns

```
nhanes.samp.adult %>%
  select(Age:Education)
```

Select by pattern

```
nhanes.samp.adult %>%
  select(starts_with("B"))
```

mutate() - creating new variables

`mutate()` creates new columns or modifies existing ones.

Important: `mutate()` doesn't change your original data unless you save it!

Create a single new column

```
1 nhanes.samp.adult %>%
2   mutate(height_m = Height / 100) %>%
3   select(ID, Height, height_m)
```

```
# A tibble: 135 × 3
      ID Height height_m
   <int>  <dbl>    <dbl>
 1 63147   189.     1.89
 2 57165   168.     1.68
 3 69465   162.     1.62
 4 57313   158.     1.58
 5 56047   168.     1.68
 6 57056   166.     1.66
 7 54643   156.     1.56
 8 53095   156.     1.56
 9 67434   181.     1.81
10 60273   178.     1.78
```

mutate() with case_when()

`case_when()` is useful for creating categorical variables based on conditions.

```
nhanes.samp.adult %>%
  mutate(bmi_category = case_when(
    BMI < 18.5 ~ "Underweight",
    BMI < 25 ~ "Normal",
    BMI < 30 ~ "Overweight",
    BMI >= 30 ~ "Obese"
  )) %>%
  select(ID, BMI, bmi_category) %>%
  head(5)  # Just show first 5 rows
```

```
# A tibble: 5 × 3
     ID   BMI bmi_category
  <int> <dbl> <chr>
1 63147  30   Obese
2 57165  27.4 Overweight
3 69465  26.9 Overweight
4 57313  24.7 Normal
5 56047  24.9 Normal
```

- `case_when()` evaluates conditions in order
- Use `~` to separate condition from result
- First matching condition wins

mutate() - multiple columns at once

You can create several new variables in one `mutate()` call.

```
nhanes.samp.adult %>%
  mutate(
    height_m = Height / 100,
    weight_kg = Weight * 0.453592,
    bmi_calculated = weight_kg / (height_m^2)
  ) %>%
  select(ID, Height, Weight, BMI, height_m, weight_kg, bmi_calculated)
```

```
# A tibble: 135 × 7
      ID Height Weight   BMI height_m weight_kg bmi_calculated
   <int>  <dbl>  <dbl> <dbl>    <dbl>     <dbl>          <dbl>
 1 63147   189.  107.   30       1.89      48.4           13.6
 2 57165   168.   77.4  27.4     1.68      35.1           12.4
 3 69465   162.   70.5  26.9     1.62      32.0           12.2
 4 57313   158.   61.7  24.7     1.58      28.0           11.2
 5 56047   168.   70.3  24.9     1.68      31.9           11.3
 6 57056   166.   92.6  33.7     1.66      42.0           15.3
 7 54643   156.   55.6  23.0     1.56      25.2           10.4
 8 53095   156.   82    33.5     1.56      37.2           15.2
 9 67434   181.   88.2  26.8     1.81      40.0           12.2
10 60273   178.  119    37.5     1.78      54.0           17.0
```

Quick practice (2-3 minutes)

Using the `nhanes.samp.adult` data and pipes:

1. Filter to people with diabetes
2. Select the columns: Age, Gender, BMI, Diabetes
3. Create a new column called `age_group` that is:
    - “Under 50” if Age < 50
    - “50 or older” if Age >= 50

(Hint: Use `filter()`, `select()`, and `mutate()` with `case_when()`)

Quick practice: solution

```r
nhanes.samp.adult %>%
  filter(Diabetes == "Yes") %>%
  select(Age, Gender, BMI, Diabetes) %>%
  mutate(age_group = case_when(
    Age < 50 ~ "Under 50",
    Age >= 50 ~ "50 or older"
  ))
```

```
# A tibble: 13 × 5
     Age Gender   BMI Diabetes age_group
   <int> <fct>  <dbl> <fct>    <chr>
 1    32 female  45.5 Yes      Under 50
 2    63 male    37.7 Yes      50 or older
 3    49 male    34.6 Yes      Under 50
 4    36 male    38.7 Yes      Under 50
 5    38 female  31.7 Yes      Under 50
 6    55 male    22.6 Yes      50 or older
 7    57 female  29.2 Yes      50 or older
 8    59 female  21.6 Yes      50 or older
 9    70 female  31.8 Yes      50 or older
10    61 female  38.7 Yes      50 or older
```

arrange() - sorting rows

`arrange()` sorts rows by one or more variables.

Sort ascending (default)

```
nhanes.samp.adult %>%
  select(ID, Age, BMI) %>%
  arrange(Age) %>%
  head(8)
```

Sort descending

```
nhanes.samp.adult %>%
  select(ID, Age, BMI) %>%
  arrange(desc(Age)) %>%
  head(8)
```

Use `desc()` to sort in descending order.

arrange() - multiple columns

You can sort by multiple variables at once.

```
nhanes.samp.adult %>%
  select(ID, Gender, Age, BMI) %>%
  arrange(Gender, desc(Age)) %>%
  head(8)
```

```
# A tibble: 8 × 4
     ID Gender   Age   BMI
  <int> <fct>  <int> <dbl>
1 61964 female    80  19.2
2 57313 female    74  24.7
3 59171 female    74  32.0
4 58284 female    72  25.2
5 67897 female    71  17.1
6 67285 female    70  36.6
7 67668 female    70  31.8
8 69421 female    68  25.9
```

Sorts by Gender first, then by Age (descending) within each gender group.

group_by() - preparing for summaries

`group_by()` groups your data by one or more variables.

- What if I want to quickly look at group differences?
- It will not change how the data look, but changes the actions of following functions

By itself, it doesn't change how the data looks:

```
1 nhanes.samp.adult %>%
2   group_by(Gender)
```

```
# A tibble: 135 × 76
# Groups:   Gender [2]
      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1   Race3 Education
   <int> <fct>    <fct>  <int> <fct>         <int> <fct>   <fct> <fct>
 1 63147 2011_12  male      41 " 40-49"         NA White   White Some College
 2 57165 2009_10  male      48 " 40-49"        586 Black   <NA>  High School
 3 69465 2011_12  female    50 " 50-59"         NA White   White College Grad
 4 57313 2009_10  female    74 " 70+"          889 White   <NA>  College Grad
 5 56047 2009_10  female    27 " 20-29"        329 White   <NA>  9 - 11th Grade
 6 57056 2009_10  male      26 " 20-29"        316 Mexican <NA>  High School
 7 54643 2009_10  female    41 " 40-49"        503 White   <NA>  College Grad
 8 53095 2009_10  female    52 " 50-59"        632 White   <NA>  9 - 11th Grade
 9 67434 2011_12  male      56 " 50-59"         NA White   White College Grad
```

Notice the "Groups: Gender [2]" at the top - the data is now grouped, but not summarized.

group_by() with summarize()

The real power of `group_by()` comes when combined with `summarize()`.

Without grouping

```
nhanes.samp.adult %>%
  summarize(
    mean_bmi = mean(BMI, na.rm = TRUE),
    sd_bmi = sd(BMI, na.rm = TRUE),
    n = n()
  )
```

```
# A tibble: 1 × 3
  mean_bmi sd_bmi     n
     <dbl>  <dbl> <int>
1     29.1   7.55   135
```

With grouping by Gender

```
nhanes.samp.adult %>%
  group_by(Gender) %>%
  summarize(
    mean_bmi = mean(BMI, na.rm = TRUE),
    sd_bmi = sd(BMI, na.rm = TRUE),
    n = n()
  )
```

```
# A tibble: 2 × 4
  Gender mean_bmi sd_bmi     n
  <fct>     <dbl>  <dbl> <int>
1 female     29.0   8.81    70
2 male       29.2   5.97    65
```

`n()` counts the number of observations in each group.

Chaining multiple dplyr verbs

You can chain operations together to answer complex questions.

Question: What is the mean BMI for people over age 50, by diabetes status?

```
nhanes.samp.adult %>%
  filter(Age > 50) %>%
  select(Age, BMI, Diabetes) %>%
  group_by(Diabetes) %>%
  summarize(
    n = n(),
    mean_bmi = mean(BMI, na.rm = TRUE),
    sd_bmi = sd(BMI, na.rm = TRUE)
  )
```

```
# A tibble: 2 × 4
  Diabetes     n mean_bmi sd_bmi
  <fct>    <int>    <dbl>  <dbl>
1 No          39     28.1   6.30
2 Yes          8     32.0   7.16
```

Summary statistics with rstatix

`get_summary_stats()` in `rstatix` package provides pipe-friendly functions for summary statistics.

Summary for one variable

```
nhanes.samp.adult %>%
  get_summary_stats(BMI,
                    type = "mean_sd")
```

```
# A tibble: 1 × 4
  variable     n  mean    sd
  <fct>    <dbl> <dbl> <dbl>
1 BMI        135  29.1  7.55
```

Summary by groups

```
nhanes.samp.adult %>%
  group_by(Gender) %>%
  get_summary_stats(BMI,
                    type = "mean_sd")
```

```
# A tibble: 2 × 5
  Gender variable     n  mean    sd
  <fct>  <fct>    <dbl> <dbl> <dbl>
1 female BMI         70  29.0  8.82
2 male   BMI         65  29.2  5.97
```

Other types: `"median_iqr"`, `"five_number"`, `"full"`, `"common"`

Multiple variables and groups

`get_summary_stats()` can handle multiple variables at once.

```
1 nhanes.samp.adult %>%
2   group_by(Gender, Diabetes) %>%
3   get_summary_stats(BMI, Height, Weight, type = "mean_sd")
```

```
# A tibble: 12 × 6
   Gender Diabetes variable     n  mean    sd
   <fct>  <fct>    <fct>    <dbl> <dbl> <dbl>
 1 female No       BMI         62  28.3  8.75
 2 female No       Height      62 163.   7.57
 3 female No       Weight      62  75.7 25.9
 4 female Yes      BMI          8  34.4  7.73
 5 female Yes      Height       8 161.   7.11
 6 female Yes      Weight       8  90.0 22.6
 7 male   No       BMI         60  28.8  5.69
 8 male   No       Height      60 176.   6.97
 9 male   No       Weight      60  89.6 18.1
10 male   Yes      BMI          5  34.8  7.15
```

Quick check: putting it together

Using what you've learned, try to answer:

What is the mean age and BMI for each combination of Gender and Diabetes status?

Hints:

- Use `group_by()` with two variables
- Use `get_summary_stats()` or `summarize()`
- Include `n()` to count observations

Quick check: solution

Using rstatix

```
nhanes.samp.adult %>%
  group_by(Gender, Diabetes) %>%
  get_summary_stats(Age, BMI,
                    type = "mean_sd")
```

```
# A tibble: 8 × 6
  Gender Diabetes variable     n  mean    sd
  <fct>  <fct>    <fct>    <dbl> <dbl> <dbl>
1 female No       Age         62  42.2 16.2
2 female No       BMI         62  28.3  8.75
3 female Yes      Age          8  51   14.5
4 female Yes      BMI          8  34.4  7.73
5 male   No       Age         60  45.4 16.4
6 male   No       BMI         60  28.8  5.69
7 male   Yes      Age          5  52   10.2
8 male   Yes      BMI          5  34.8  7.15
```

Using dplyr summarize

```
nhanes.samp.adult %>%
  group_by(Gender, Diabetes) %>%
  summarize(
    n = n(),
    mean_age = mean(Age, na.rm = TRUE),
    mean_bmi = mean(BMI, na.rm = TRUE)
  )
```

```
# A tibble: 4 × 5
# Groups:   Gender [2]
  Gender Diabetes     n mean_age mean_bmi
  <fct>  <fct>    <int>    <dbl>    <dbl>
1 female No          62     42.2     28.3
2 female Yes          8     51       34.4
3 male   No          60     45.4     28.8
4 male   Yes          5     52       34.8
```

Summary tables with janitor

`tabyl()` in the `janitor` package makes frequency tables much easier than base R.

One-way frequency table

```
1 nhanes.samp.adult %>%
2   tabyl(Gender)
```

```
Gender  n   percent
female 70 0.5185185
  male 65 0.4814815
```

Gives you counts (`n`), percentages (`percent`), and includes missing values automatically.

Two-way table (cross-tabulation)

```
1 nhanes.samp.adult %>%
2   tabyl(Gender, Diabetes)
```

```
Gender No Yes
female 62   8
  male 60   5
```

Enhancing tables with janitor adorn functions

`janitor` has “adorn” functions to format tables nicely.

Add percentages

```
1 nhanes.samp.adult %>%
2   tabyl(Gender, Diabetes) %>%
3   adorn_percentages("row") %>%
4   adorn_pct_formatting(digits = 1)
```

```
Gender    No   Yes
female 88.6% 11.4%
  male 92.3%  7.7%
```

Add totals

```
1 nhanes.samp.adult %>%
2   tabyl(Gender, Diabetes) %>%
3   adorn_totals(c("row", "col"))
```

```
Gender  No Yes Total
female  62   8    70
  male  60   5    65
 Total 122  13   135
```

Combining adorn functions

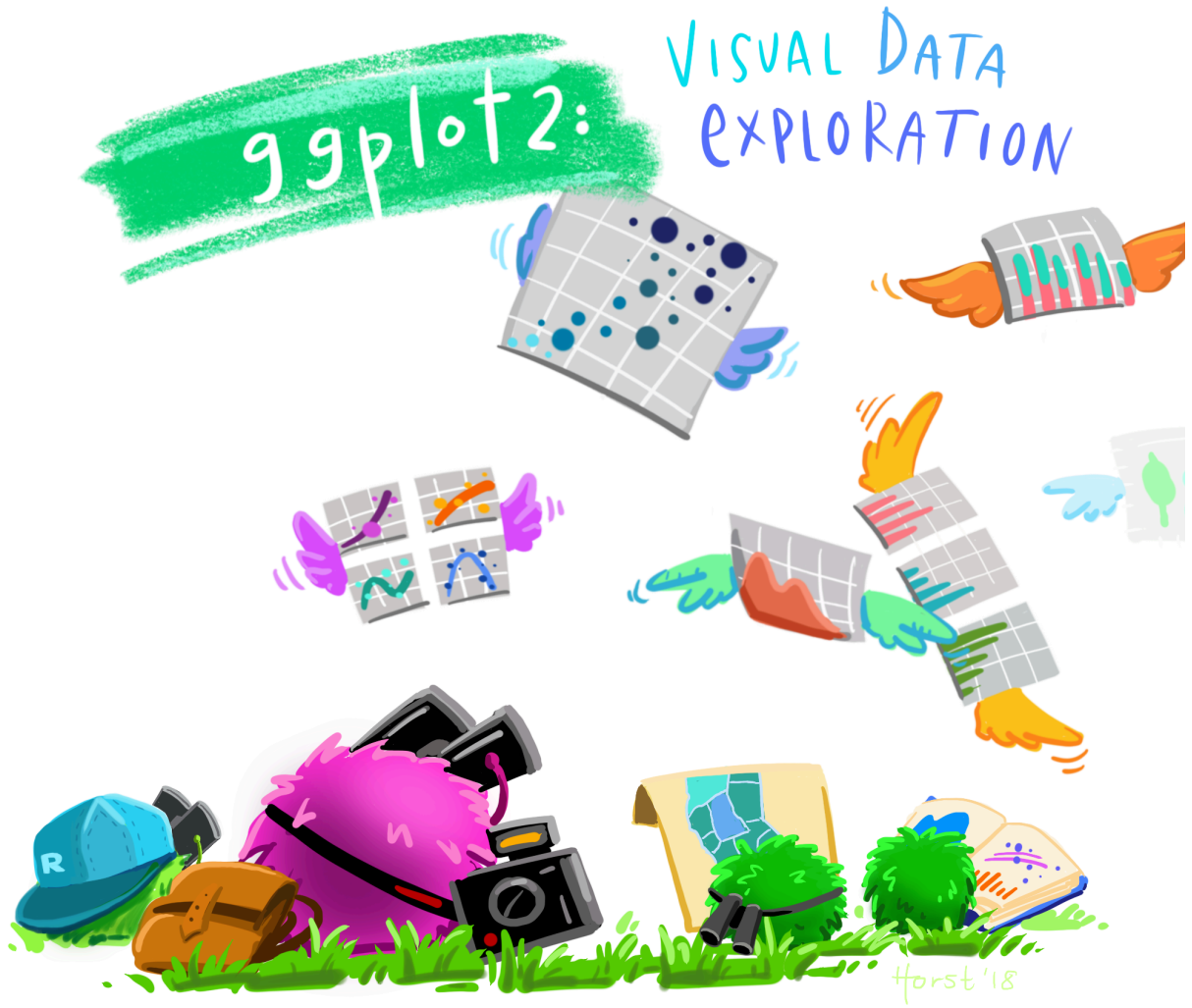
You can chain multiple adorn functions together.

```
nhanes.samp.adult %>%
  tabyl(Gender, Diabetes) %>%
  adorn_totals(c("row", "col")) %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

```
Gender          No        Yes        Total
female 88.6%  (62) 11.4%  (8) 100.0%  (70)
  male 92.3%  (60)  7.7%  (5) 100.0%  (65)
 Total 90.4% (122)  9.6% (13) 100.0% (135)
```

- `adorn_ns()` adds the counts alongside percentages
- Order matters: calculate totals first, then percentages

Data visualization with ggplot2



Artwork by @allison_horst

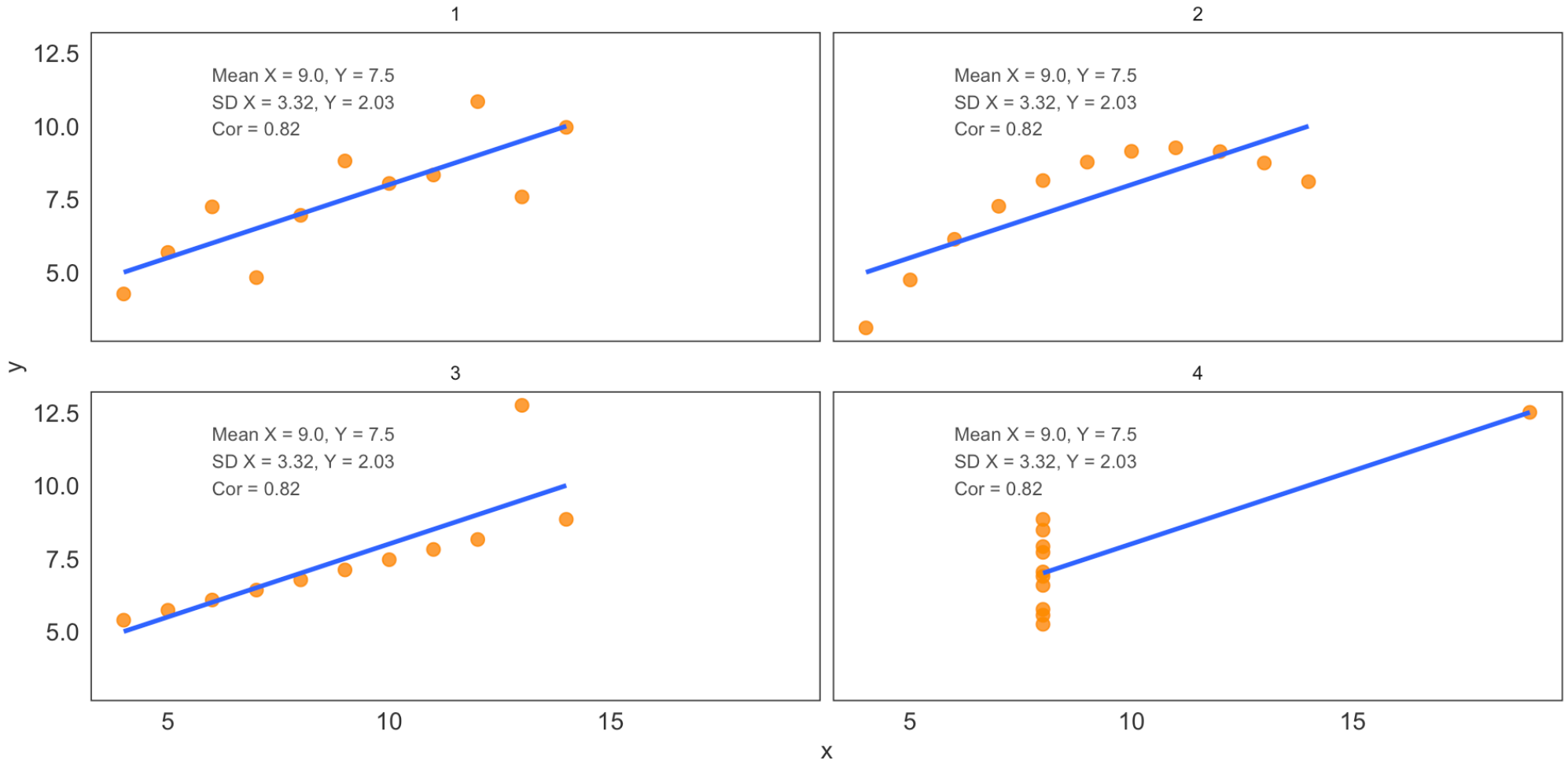
Why visualize data?

Visualizations help us:

- Understand distributions
- Identify outliers and patterns
- Compare groups
- Communicate findings

Anscombe's Quartet

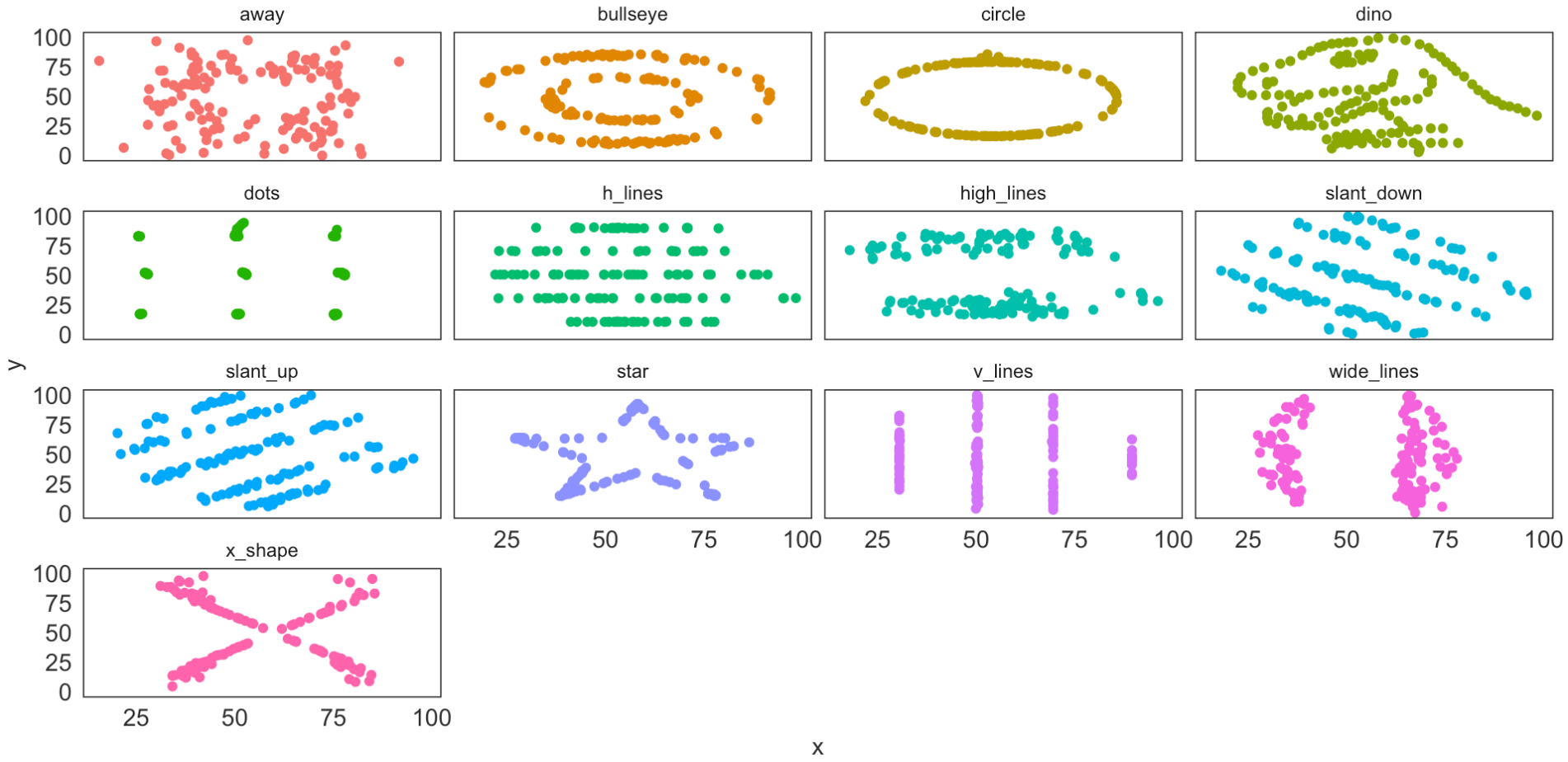
Four datasets with identical summary statistics but very different patterns:



All four have the same mean, SD, and correlation - but look how different they are!

The Datasaurus Dozen

Similar to Anscombe's quartet, The Datasaurus Dozen is a compilation of 13 data sets with similar numerical summaries and radically different visual summaries. See a great discussion of this dataset by the creators, Justin Matejka and George Fitzmaurice here



All 13 datasets have nearly identical summary statistics: Mean X = 54.3, Y = 47.8 SD X = 16.77, Y = 26.94 Cor = -0.06

The grammar of graphics

`ggplot2` implements the **grammar of graphics** - a framework for describing plots.

Every ggplot has three essential components:

1. **Data** - the dataset you're plotting
2. **Aesthetics** (`aes()`) - mappings from data to visual properties (x, y, color, size, etc.)
3. **Geoms** - geometric objects that represent the data (points, lines, bars, etc.)

Basic template:

```
ggplot(data = DATA,
       mapping = aes(x = X_VAR, y = Y_VAR)) +
  geom_SOMETHING()
```

Notice we use `+` (not `%>%`) to add layers to a ggplot.

Basics of a ggplot

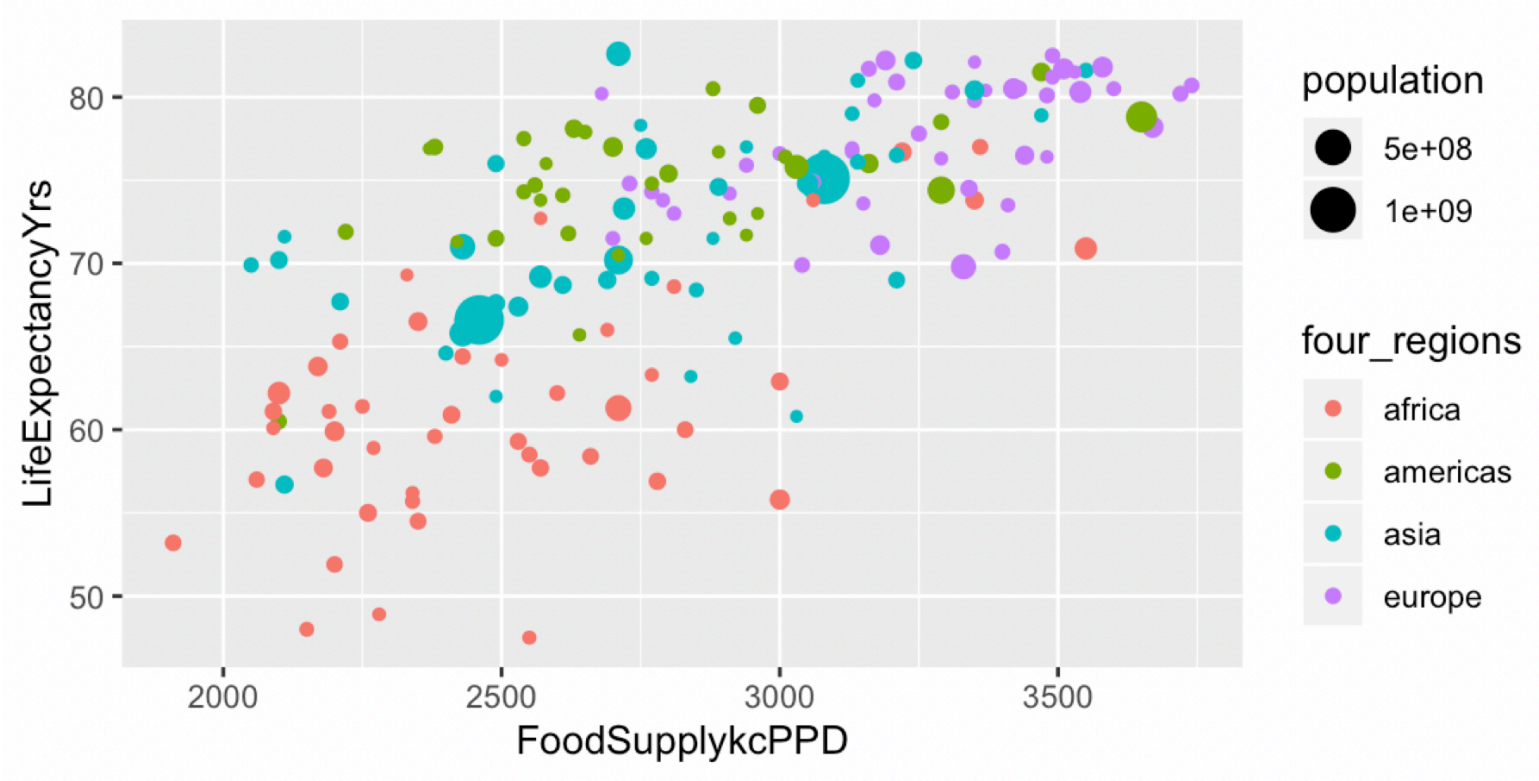
Function

Dataset

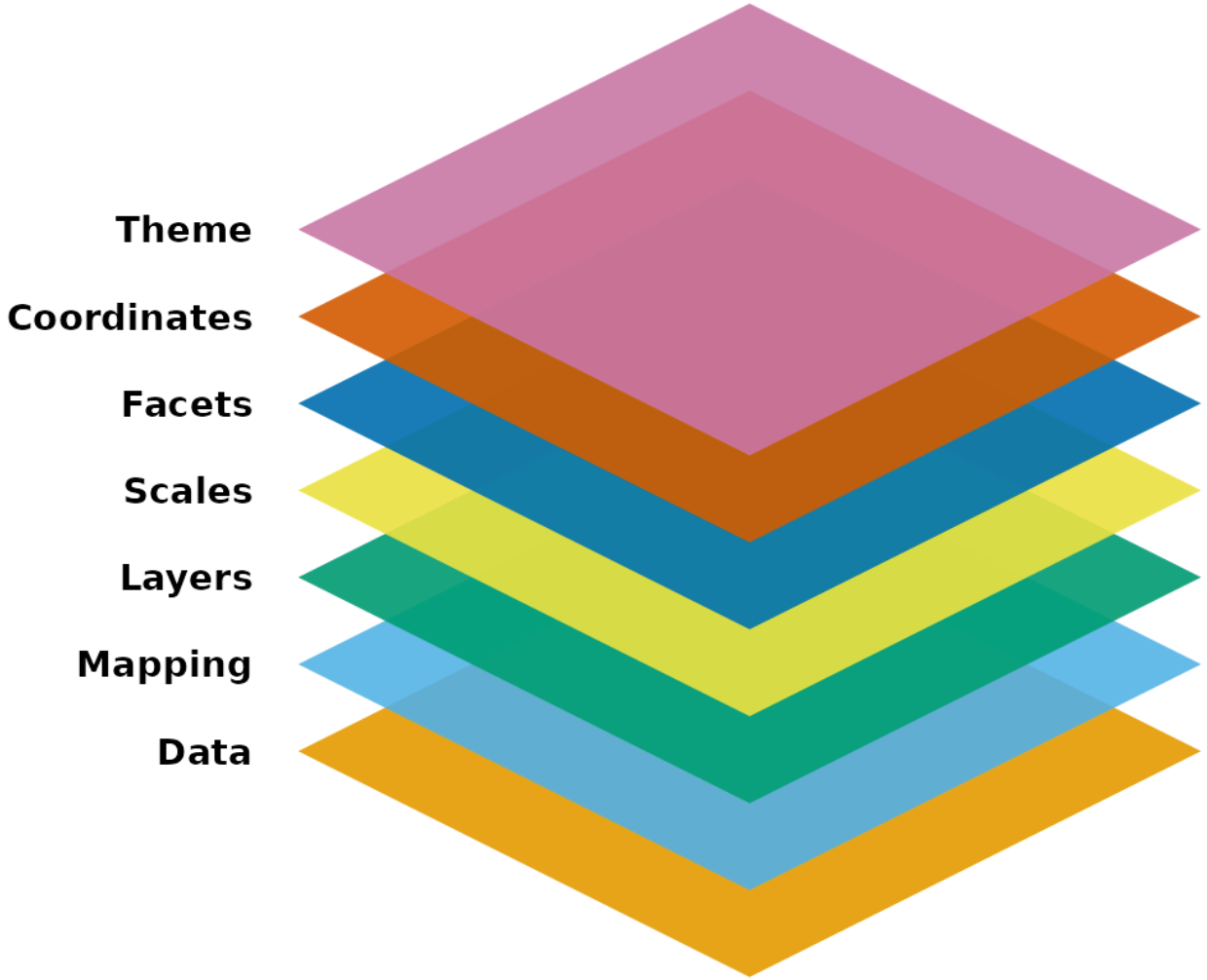
```
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD, y = LifeExpectancyYrs,
           color = four_regions, size = population)) +
       geom_point()
```

Which variables to plot

What kind of plot to make



Grammar of ggplot2



Beyond the basics:

- **Scales** control how data values map to visual properties
- **Facets** create subplots
- **Coordinates** adjust the coordinate system
- **Themes** control visual appearance

Most of these have good defaults - we'll focus on the essential three components today.

Building a ggplot step by step

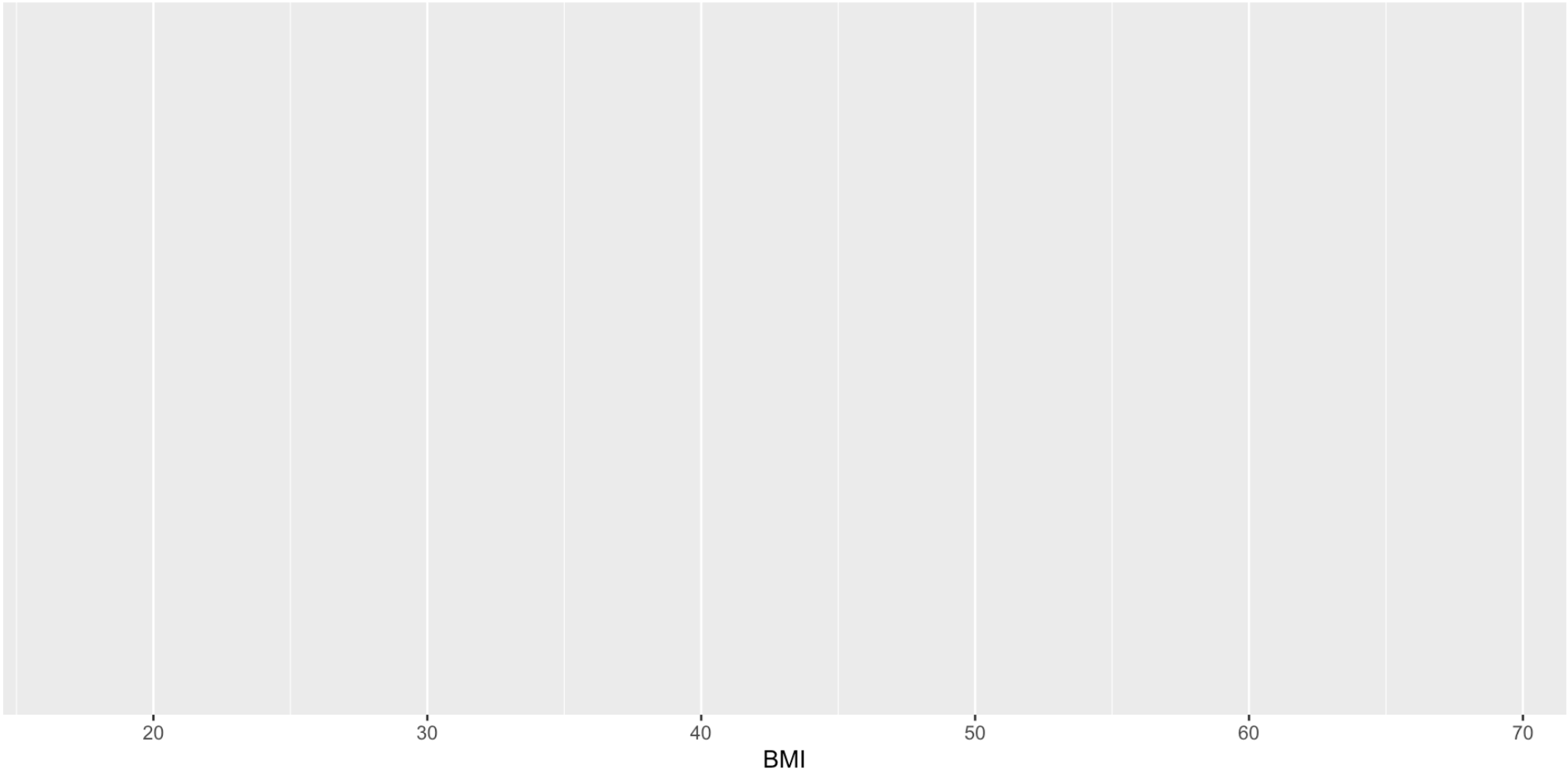
Step 1: Just data

```
1 ggplot(data = nhanes.samp.adult)
```

Building a ggplot step by step

Step 2: Add mapping

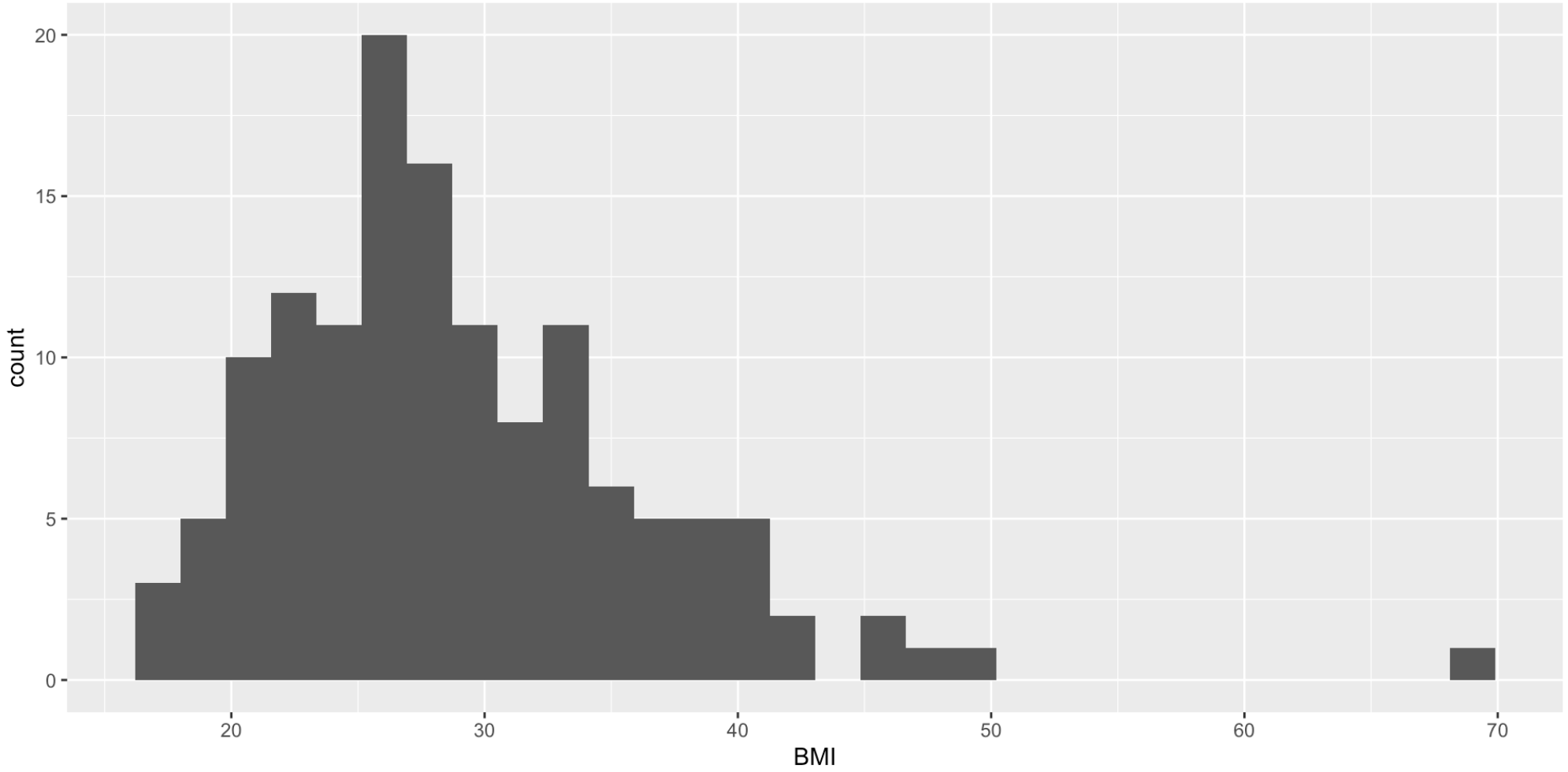
```
1 ggplot(nhanes.samp.adult, aes(x = BMI))
```



Building a ggplot step by step

Step 3: Add geom

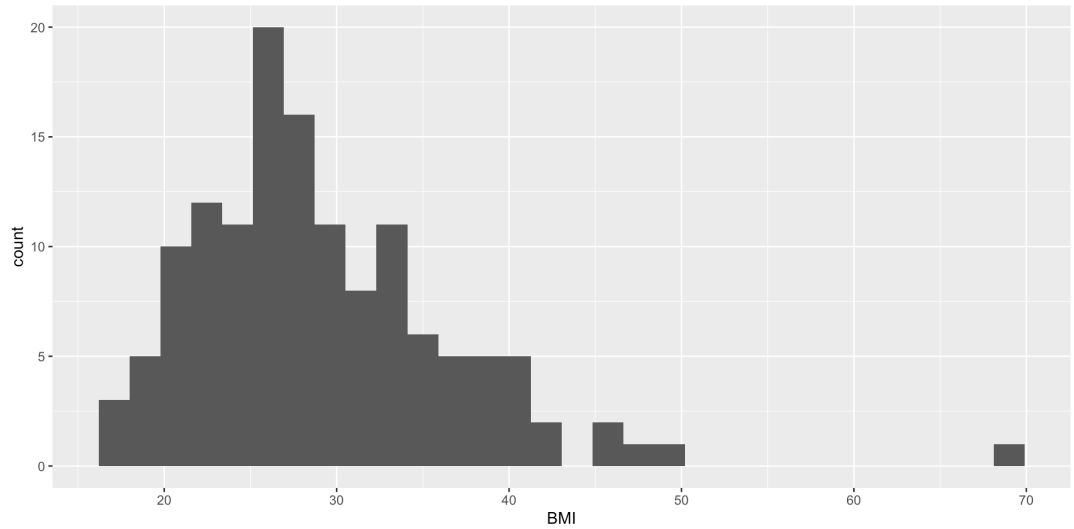
```
1 ggplot(nhanes.samp.adult, aes(x = BMI)) +
2   geom_histogram()
```



Your first ggplot: histogram

Histogram - shows the distribution of a single numeric variable.

```
ggplot(data = nhanes.samp.adult,
       mapping = aes(x = BMI)) +
  geom_histogram()
```



What does this tell us?

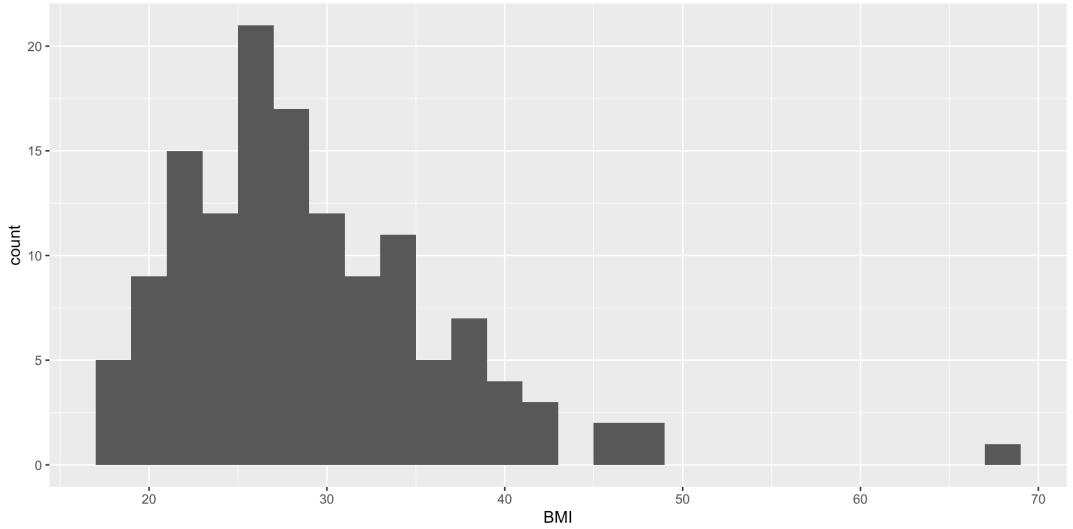
- Most people have BMI between 20-35
- Distribution is slightly right-skewed
- Some people with very high BMI (outliers)

Customizing histograms

You can control the appearance of histograms.

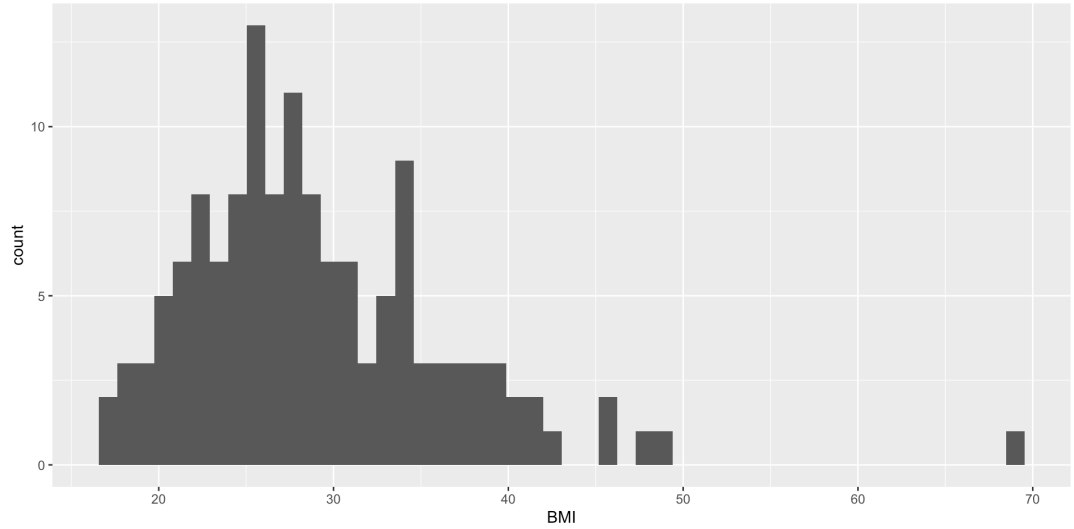
Adjust bin width

```
ggplot(nhanes.samp.adult,
       aes(x = BMI)) +
  geom_histogram(binwidth = 2)
```



Adjust number of bins

```
ggplot(nhanes.samp.adult,
       aes(x = BMI)) +
  geom_histogram(bins = 50)
```

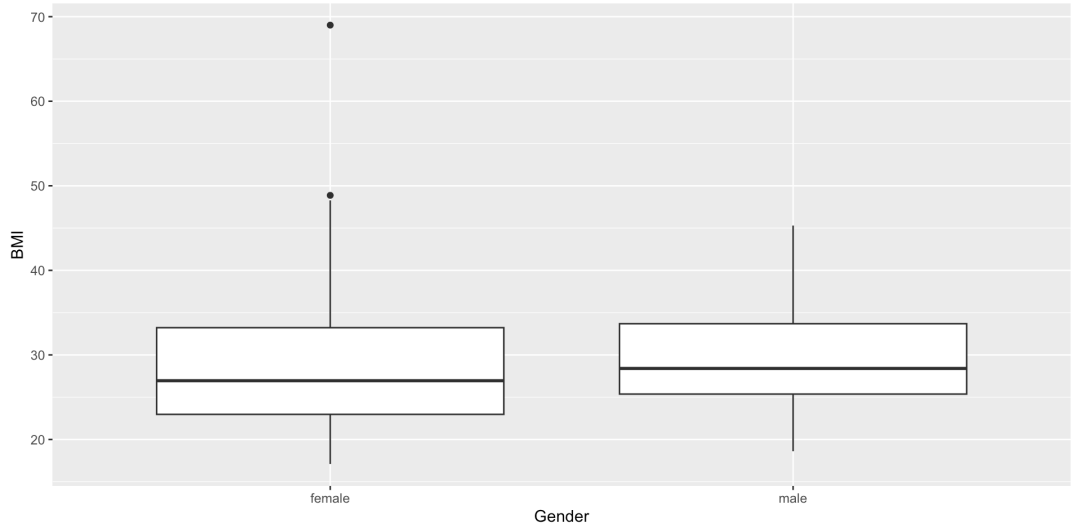


Try different values to find what best reveals patterns in your data.

Boxplots - comparing distributions

Boxplot - shows the distribution across groups.

```
ggplot(nhanes.samp.adult,
       aes(x = Gender, y = BMI)) +
  geom_boxplot()
```



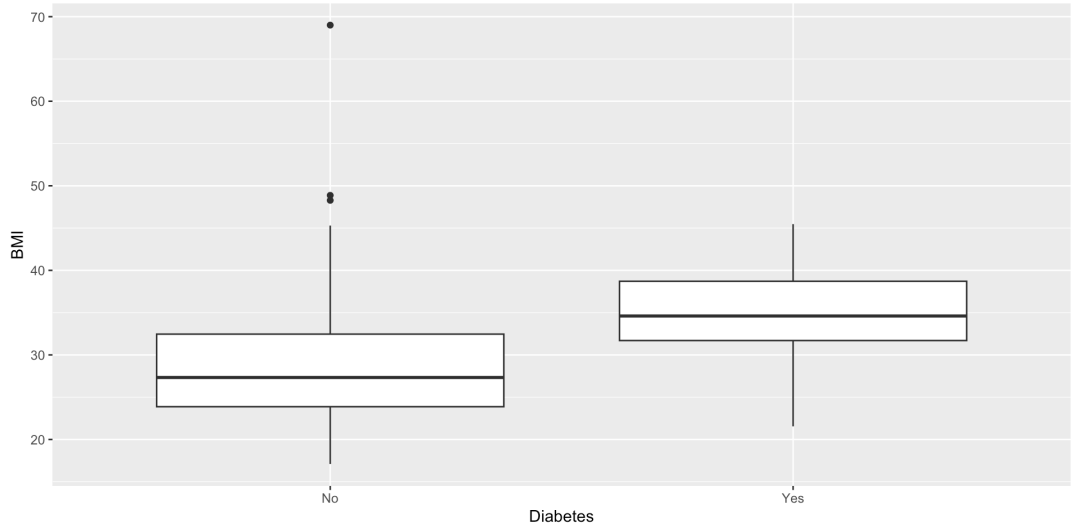
Boxplot components:

- Box: middle 50% of data (IQR)
- Line in box: median
- Whiskers: extend to most extreme non-outlier values
- Points: potential outliers

Boxplot by multiple groups

We can compare distributions across multiple categorical variables.

```
ggplot(nhanes.samp.adult,
       aes(x = Diabetes, y = BMI)) +
  geom_boxplot()
```

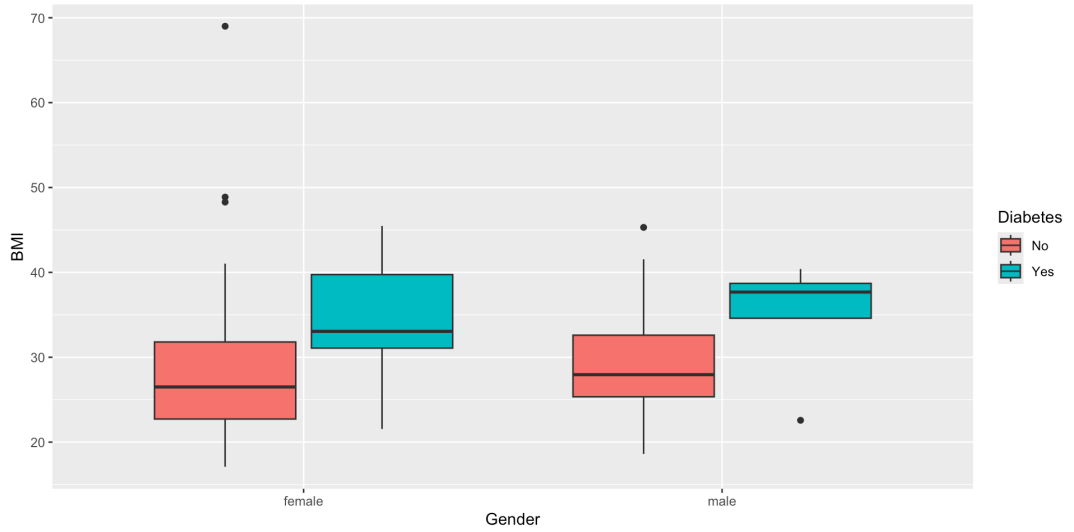


People with diabetes tend to have higher BMI.

Adding color to boxplots

Color can be mapped to a variable using `aes()`.

```
ggplot(nhanes.samp.adult,
       aes(x = Gender,
           y = BMI,
           fill = Diabetes)) +
  geom_boxplot()
```



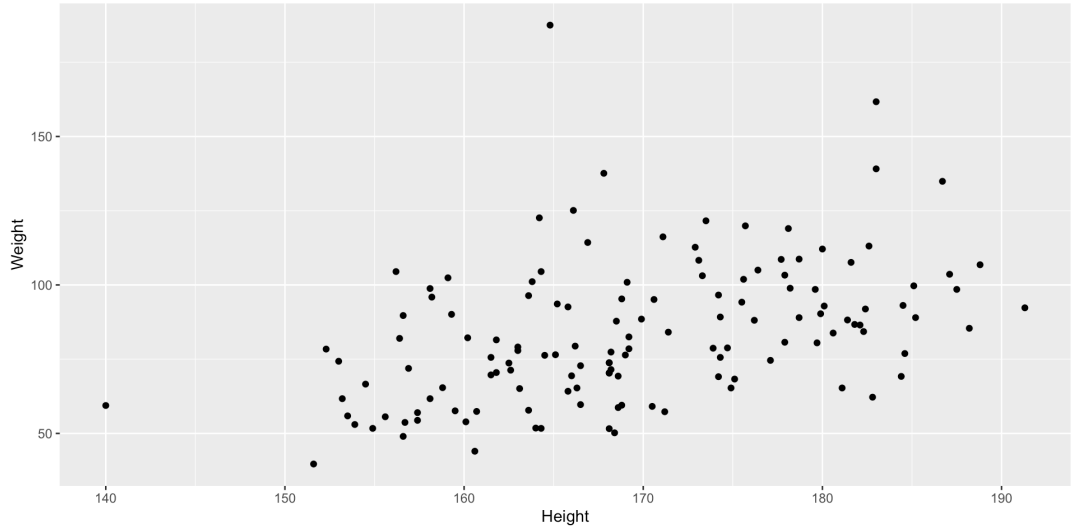
`fill` controls the fill color of shapes.

`color` would control the outline color.

Scatterplots - relationship between two numeric variables

Scatterplot - shows the relationship between two numeric variables.

```
ggplot(nhanes.samp.adult,
       aes(x = Height, y = Weight)) +
  geom_point()
```



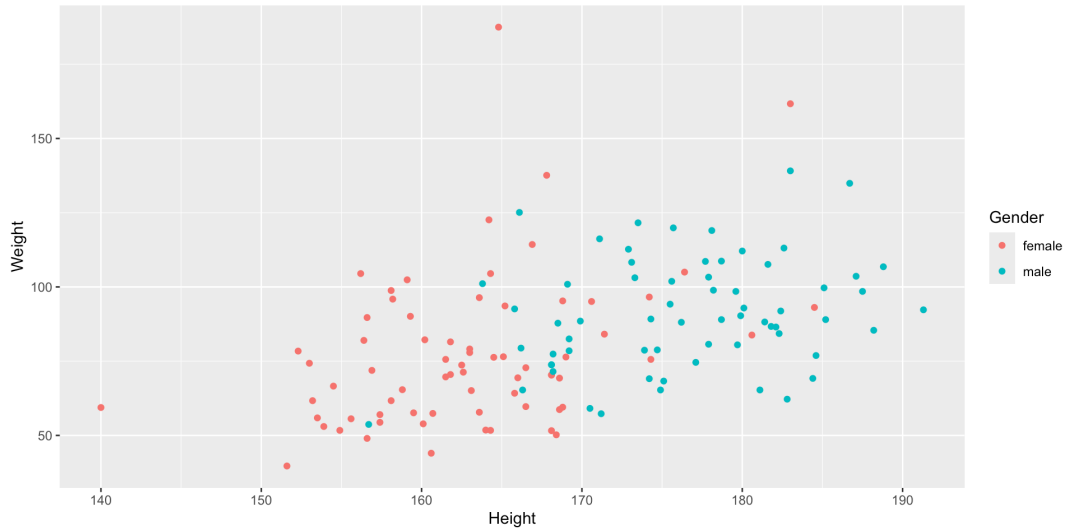
What we see:

- Positive relationship
- Taller people tend to weigh more
- Some variability

Enhancing scatterplots with color

Map a third variable to color to see patterns.

```
ggplot(nhanes.samp.adult,
       aes(x = Height,
           y = Weight,
           color = Gender)) +
  geom_point()
```

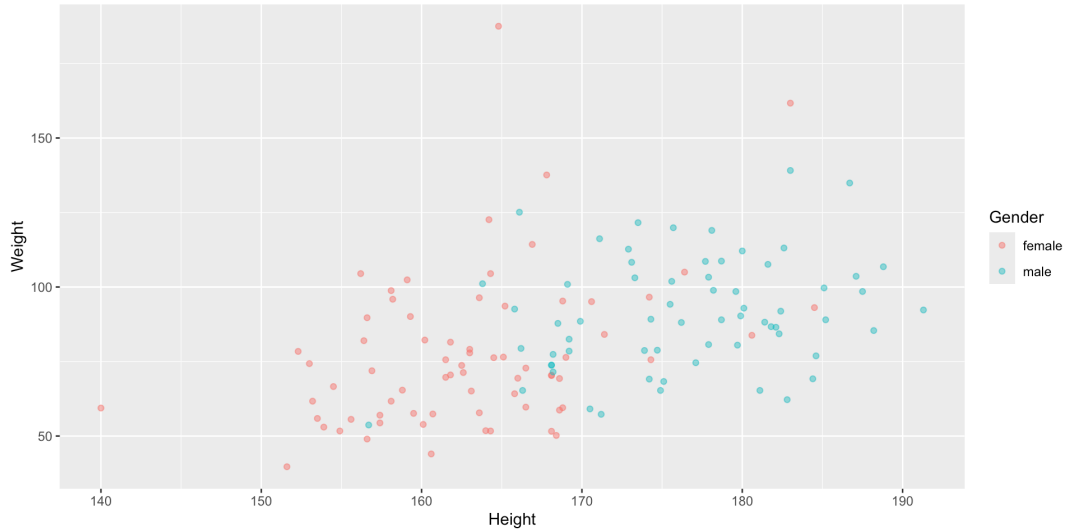


Now we can see if the relationship differs by gender.

Adjusting point transparency

When points overlap, transparency (`alpha`) helps.

```
ggplot(nhanes.samp.adult,
       aes(x = Height,
           y = Weight,
           color = Gender)) +
  geom_point(alpha = 0.5)
```

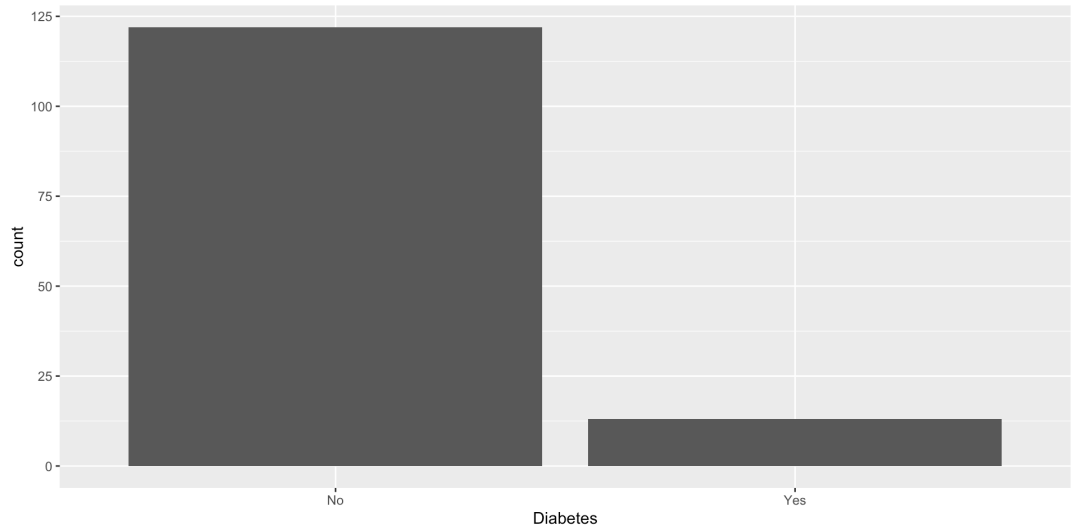


`alpha` ranges from 0 (transparent) to 1 (opaque).

Bar charts - categorical data

Bar chart - shows counts or frequencies of categories.

```
ggplot(nhanes.samp.adult,
       aes(x = Diabetes)) +
  geom_bar()
```

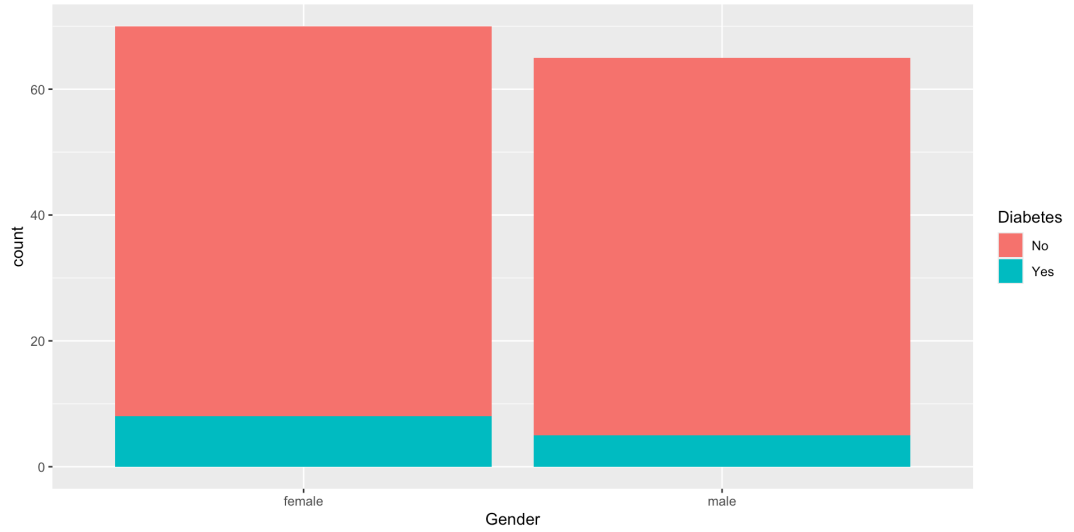


`geom_bar()` automatically counts observations in each category.

Grouped bar charts

Compare categories across groups by mapping a variable to `fill`.

```
ggplot(nhanes.samp.adult,
       aes(x = Gender,
           fill = Diabetes)) +
  geom_bar()
```

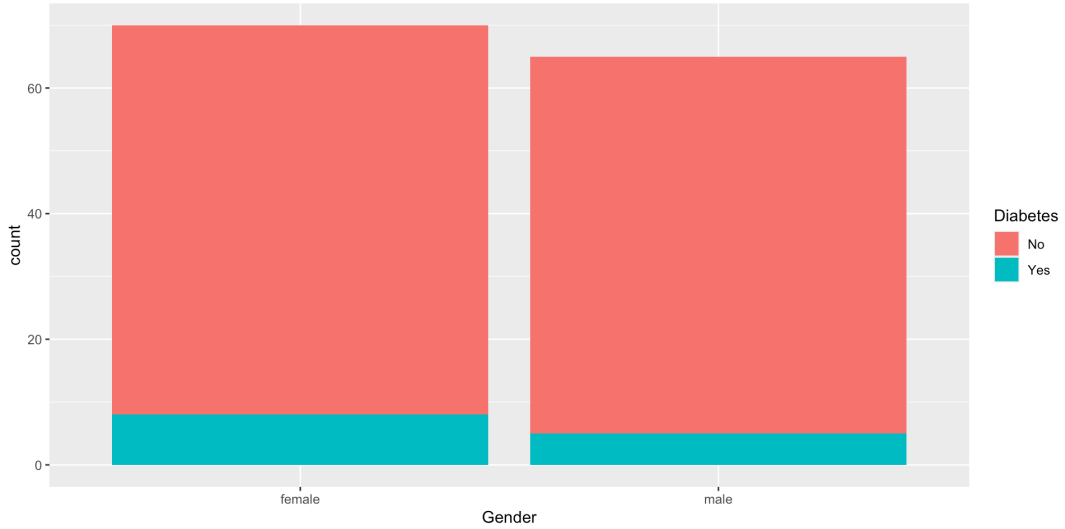


By default, bars are stacked.

Position adjustments for bar charts

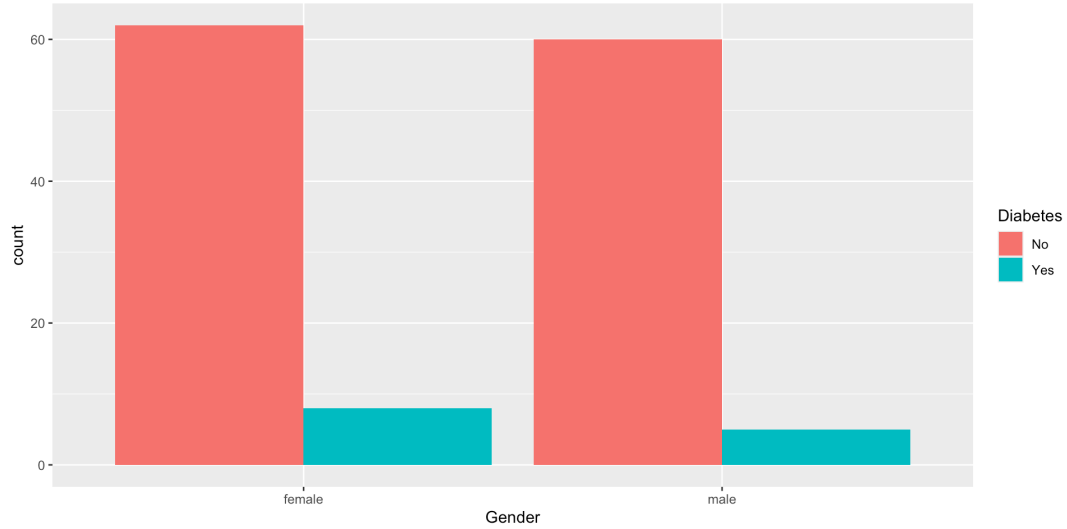
Stacked (default)

```
ggplot(nhanes.samp.adult,
       aes(x = Gender,
           fill = Diabetes)) +
  geom_bar()
```



Side-by-side

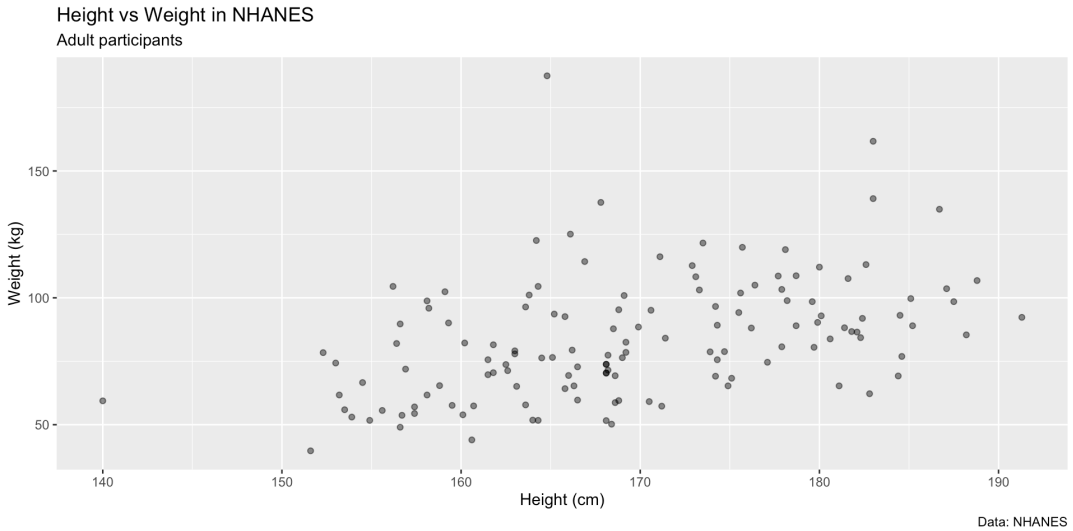
```
ggplot(nhanes.samp.adult,
       aes(x = Gender,
           fill = Diabetes)) +
  geom_bar(position = "dodge")
```



Adding labels and titles

Make your plots more informative with labels.

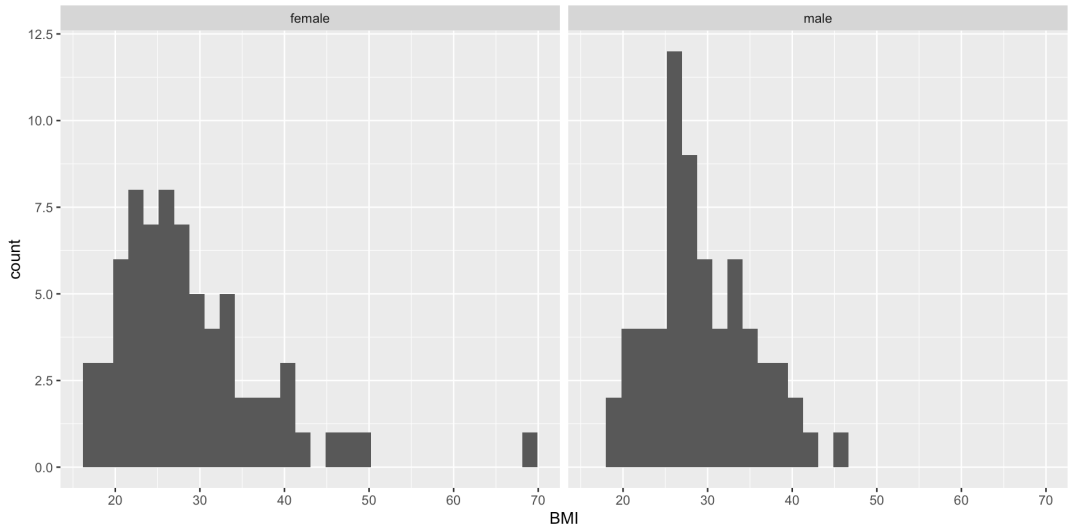
```
ggplot(nhanes.samp.adult,
       aes(x = Height, y = Weight)) +
  geom_point(alpha = 0.5) +
  labs(
    title = "Height vs Weight in NHANES",
    subtitle = "Adult participants",
    x = "Height (cm)",
    y = "Weight (kg)",
    caption = "Data: NHANES"
  )
```



Faceting - small multiples

Facets create separate panels for subgroups.

```
ggplot(nhanes.samp.adult,
       aes(x = BMI)) +
  geom_histogram(bins = 30) +
  facet_wrap(~Gender)
```

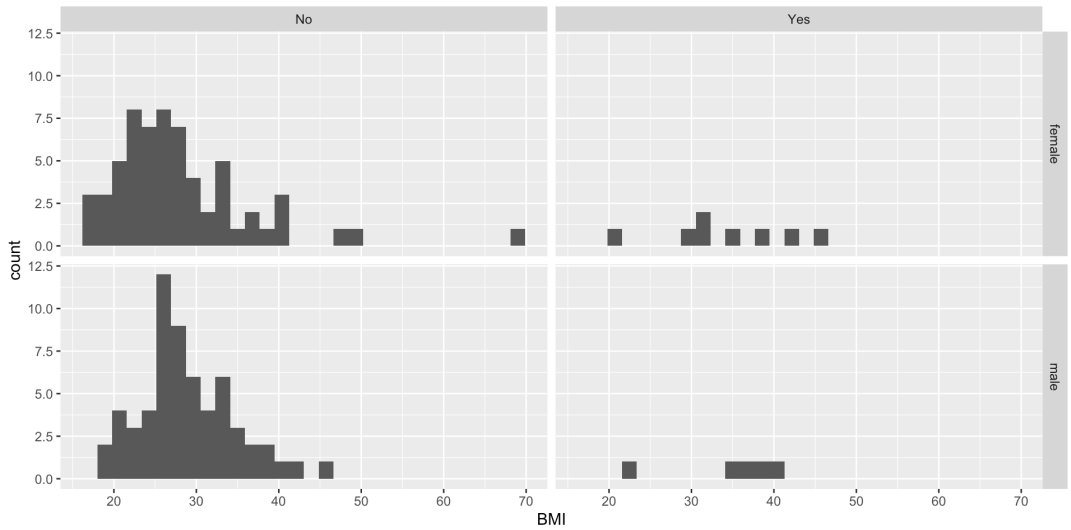


`facet_wrap()` creates a panel for each level of a variable.

Faceting by two variables

Create a grid of panels with two variables.

```
ggplot(nhanes.samp.adult,
       aes(x = BMI)) +
  geom_histogram(bins = 30) +
  facet_grid(Gender ~ Diabetes)
```

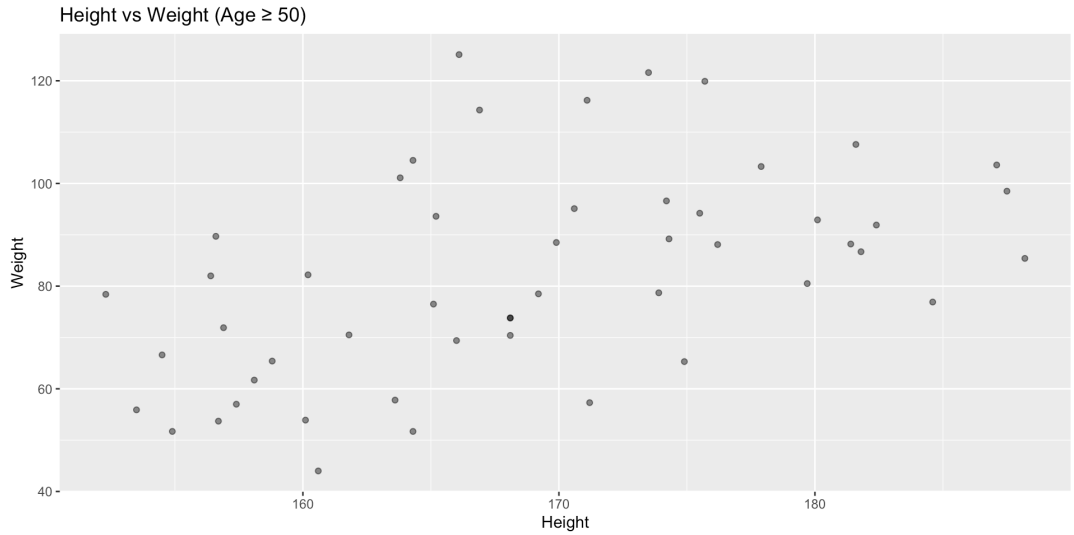


`facet_grid()` creates rows × columns layout.

Combining dplyr and ggplot2

You can pipe data directly into `ggplot()`!

```
nhanes.samp.adult %>%
  filter(Age >= 50) %>%
  ggplot(aes(x = Height, y = Weight)) +
  geom_point(alpha = 0.5) +
  labs(title = "Height vs Weight (Age ≥ 50)")
```



Pipe `%>%` passes data to `ggplot()`.

Then use `+` to add layers.

Saving your work

Remember: **dplyr operations don't modify the original data** unless you save them.

Save filtered data

```
adults_diabetes <- nhanes.samp.adult %>%
  filter(Diabetes == "Yes")

# Check it worked
nrow(adults_diabetes)
```

```
[1] 13
```

Save data with new variables

```
nhanes_with_categories <-
  nhanes.samp.adult %>%
  mutate(
    bmi_category = case_when(
      BMI < 18.5 ~ "Underweight",
      BMI < 25 ~ "Normal",
      BMI < 30 ~ "Overweight",
      BMI >= 30 ~ "Obese"
    )
  )
```

Saving plots

Save plots to files with `ggsave()`.

```
# Create a plot
p <- ggplot(nhanes.samp.adult, aes(x = BMI)) +
  geom_histogram(bins = 30) +
  labs(title = "Distribution of BMI")

# Save it
ggsave("bmi_histogram.png", plot = p,
       width = 8, height = 6, dpi = 300)
```

Tips:

- `ggsave()` saves the last plot if you don't specify `plot =`
- Common formats: `.png`, `.pdf`, `.jpg`
- Adjust `width` and `height` in inches
- Use `dpi = 300` for publication quality

Common ggplot2 geoms

Geom	Use case	Variables needed
`geom_histogram()`	Distribution of one numeric variable	x
`geom_density()`	Smooth distribution curve	x
`geom_boxplot()`	Compare distributions across groups	x (categorical), y (numeric)
`geom_point()`	Relationship between two numeric variables	x, y
`geom_line()`	Trends over time or ordered data	x, y
`geom_bar()`	Counts of categorical data	x
`geom_col()`	Pre-computed values	x, y

Quick practice (5 minutes)

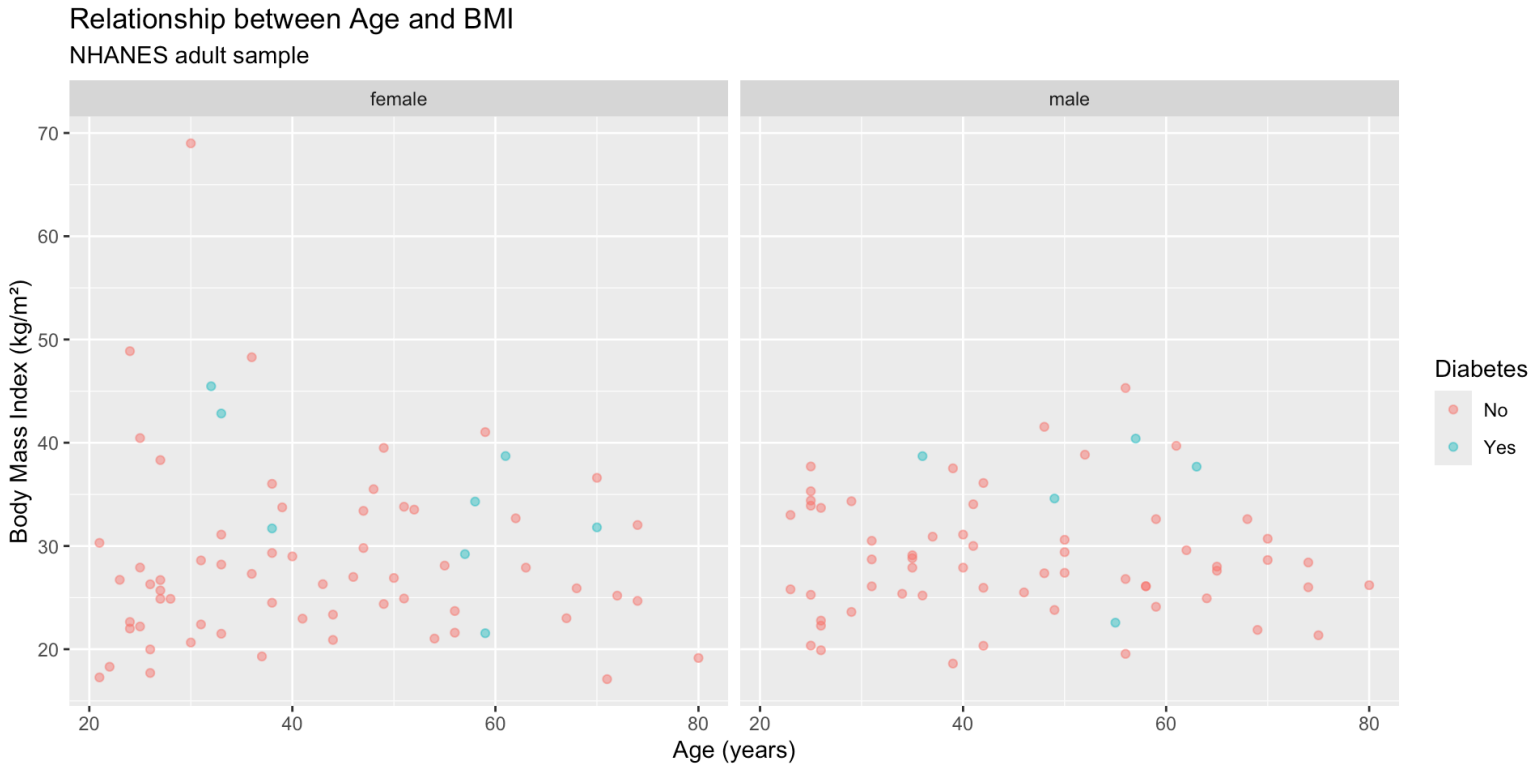
Using what you've learned, create:

1. A scatterplot of Age vs BMI
2. Color the points by Diabetes status
3. Add appropriate labels
4. Add transparency to see overlapping points

Bonus: Facet by Gender

Quick practice: solution

```
ggplot(nhanes.samp.adult,
       aes(x = Age, y = BMI, color = Diabetes)) +
  geom_point(alpha = 0.5) +
  labs(
    title = "Relationship between Age and BMI",
    subtitle = "NHANES adult sample",
    x = "Age (years)",
    y = "Body Mass Index (kg/m²)"
  ) +
  facet_wrap(~Gender)
```



EDA workflow summary

1. Import and inspect

- Load data
- Check dimensions, variable names, and types
- Look for missing values

2. Summarize

- Calculate summary statistics
- Create frequency tables
- Group by relevant variables

3. Visualize

- Plot distributions (histograms, boxplots)
- Explore relationships (scatterplots)
- Compare groups (facets, colors)

4. Iterate

- Ask new questions based on what you find
- Filter, transform, and visualize again

Key takeaways

- **Pipes (`%>%`)** chain operations together for readable code
- **`dplyr` verbs** manipulate data:
  - `filter()` - subset rows
  - `select()` - choose columns
  - `mutate()` - create/modify variables
  - `group_by()` + `summarize()` - calculate summaries by group
- **`janitor::tabyl()`** creates clean frequency tables
- **`rstatix::get_summary_stats()`** provides pipe-friendly summary statistics
- **`ggplot2`** builds visualizations using data + aesthetics + geoms
- **Nothing changes your original data** unless you save it with `<-`

Resources for learning more

Tidyverse documentation:

- https://dplyr.tidyverse.org/
- https://ggplot2.tidyverse.org/

Books (free online):

- R for Data Science (2e) by Hadley Wickham & Garrett Grolemund
- ggplot2: Elegant Graphics for Data Analysis
- Data Visualization: A Practical Introduction by Kieran Healy
- Modern Dive by Chester Ismay & Albert Kim

Cheat sheets:

- Data transformation with dplyr
- Data visualization with ggplot2

Practice exercises

Try these on your own to reinforce what you learned:

1. Filter the data to people with BMI > 30
2. Create a scatterplot of Age vs BMI
3. Calculate mean height by gender
4. Make a histogram of Age
5. Create a two-way table of Gender and Diabetes status