Week 13: *Data Visualization*

EMSE 4574: Intro to Programming for Analytics

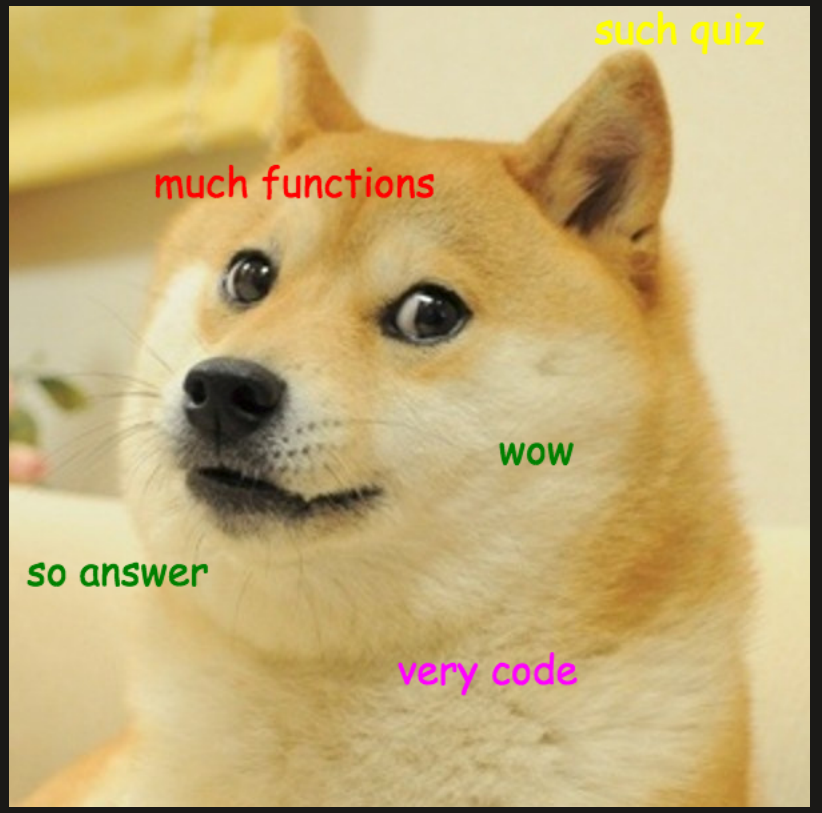
John Paul Helveston

November 24, 2020

Quiz 6

- Go to `#classroom` channel in Slack for link
- Open up RStudio before you start - you'll probably want to use it.

05:00



Before we start

Make sure you have the "tidyverse" installed and loaded, and import these two data frames

```
library(tidyverse)
library(here)

birds <- read_csv(here('data', 'wildlife_impacts.csv'))
bears <- read_csv(here('data', 'bear_killings.csv'))
```

(this is at the top of the notes.R file)

The Challenger disaster

On January 28, 1986 the space shuttle Challenger exploded

The Challenger disaster

NASA Engineers had the data on temperature & o-ring failure

TEMPERATURE CONCERN ON

SRM JOINTS

27 JAN 1986

HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

SRM No.	Cross Sectional View: Erosion Depth (in.)	Cross Sectional View: Perimeter Affected (deg)	Cross Sectional View: Nominal Dia. (in.)	Top View: Length Of Max Erosion (in.)	Top View: Total Heat Affected Length (in.)	Clocking Location (deg)
22A	None	None	0.280	None	None	36°-66°
22A	NONE	NONE	0.280	NONE	NONE	338°-18°
15A	0.010	154.0	0.280	4.25	5.25	163
15B	0.038	130.0	0.280	12.50	58.75	354
15B	None	45.0	0.280	None	29.50	354
13B	0.028	110.0	0.280	3.00	None	275
11A	None	None	0.280	None	None	--
10A	0.040	217.0	0.280	3.00	14.50	351
2B	0.053	116.0	0.280	--	--	90

61A LH Center Field**
61A LH ~~CENTER~~ FIELD**
51C LH Forward Field**
51C RH Center Field (prim)***
51C RH Center Field (sec)***
41D RH Forward Field
41C LH Aft Field*
41B LH Forward Field
STS-2 RH Aft Field

*Hot gas path detected in putty. Indication of heat on O-ring, but no damage.
**Soot behind primary O-ring.
***Soot behind primary O-ring, heat affected secondary O-ring.

Clocking location of leak check port - 0 deg.

OTHER SRM-15 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY AND NO SOOT NEAR OR BEYOND THE PRIMARY O-RING.

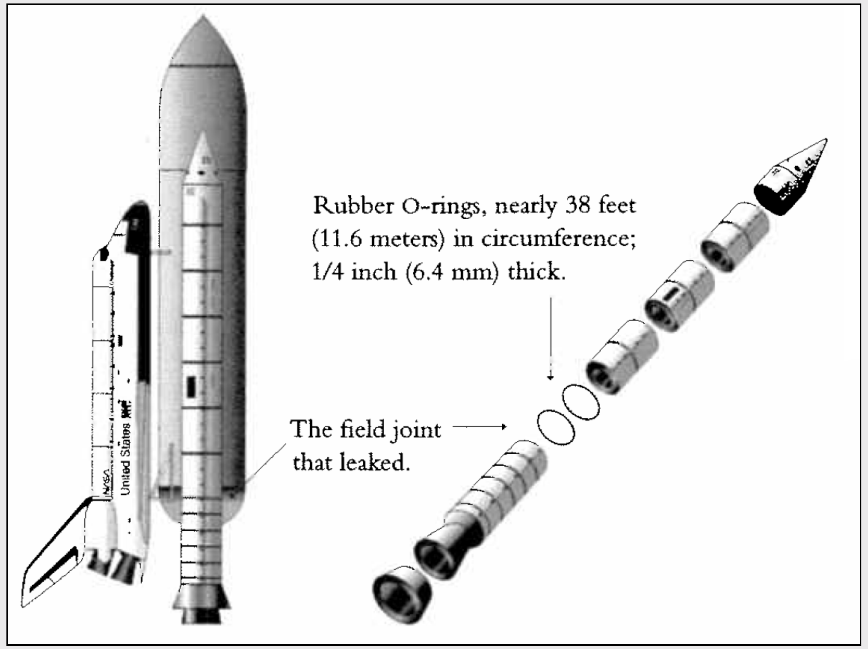
SRM-22 FORWARD FIELD JOINT HAD PUTTY PATH TO PRIMARY O-RING, BUT NO O-RING EROSION AND NO SOOT BLOWBY. OTHER SRM-22 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY.

BLOW BY HISTORY
- SRM-15 WORST BLOW-BY
  - 2 CASE JOINTS (80°), (110°) ARC
  - MUCH WORSE VISUALLY THAN SRM-22
- SRM 22 BLOW-BY
  - 2 CASE JOINTS (30-40°)
- SRM-13A, 15, 16A, 18, 23A 24A
  - NOZZLE BLOW-BY

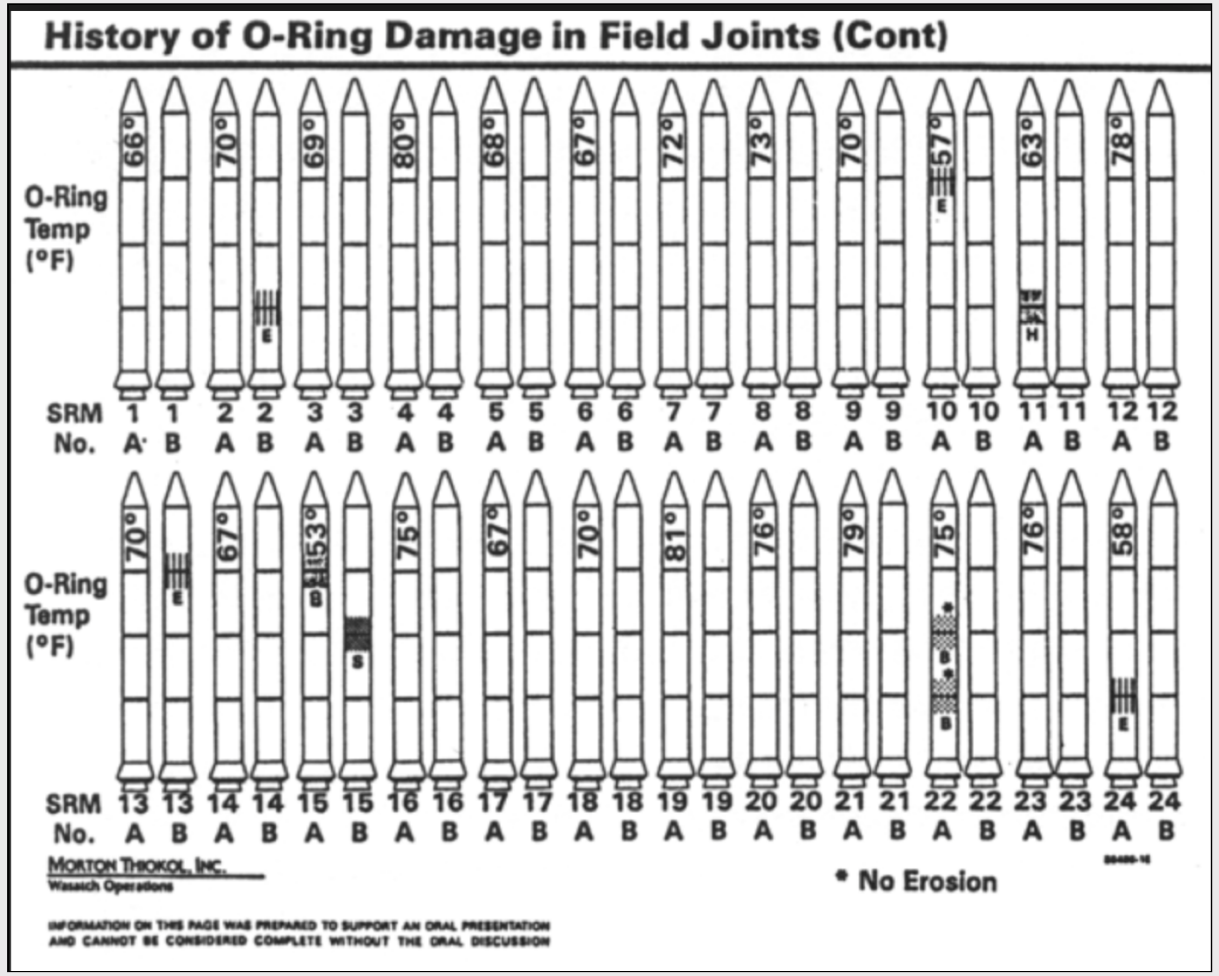
HISTORY OF O-RING TEMPERATURES (DEGREES-F)

MOTOR	MBT	AMB	O-RING	WIND
DM-4	68	36	47	10 MPH
DM-2	76	45	52	10 MPH
QM-3	72.5	40	48	10 MPH
QM-4	76	48	51	10 MPH
SRM-15	52	64	53	10 MPH
SRM-22	77	78	75	10 MPH
SRM-25	55	26	29 27	10 MPH 25 MPH

MOTOR	O-RING
DM-4	47
DM-2	52
QM-3	48
QM-4	51
SRM-15	53
SRM-22	75
SRM-25	29 27

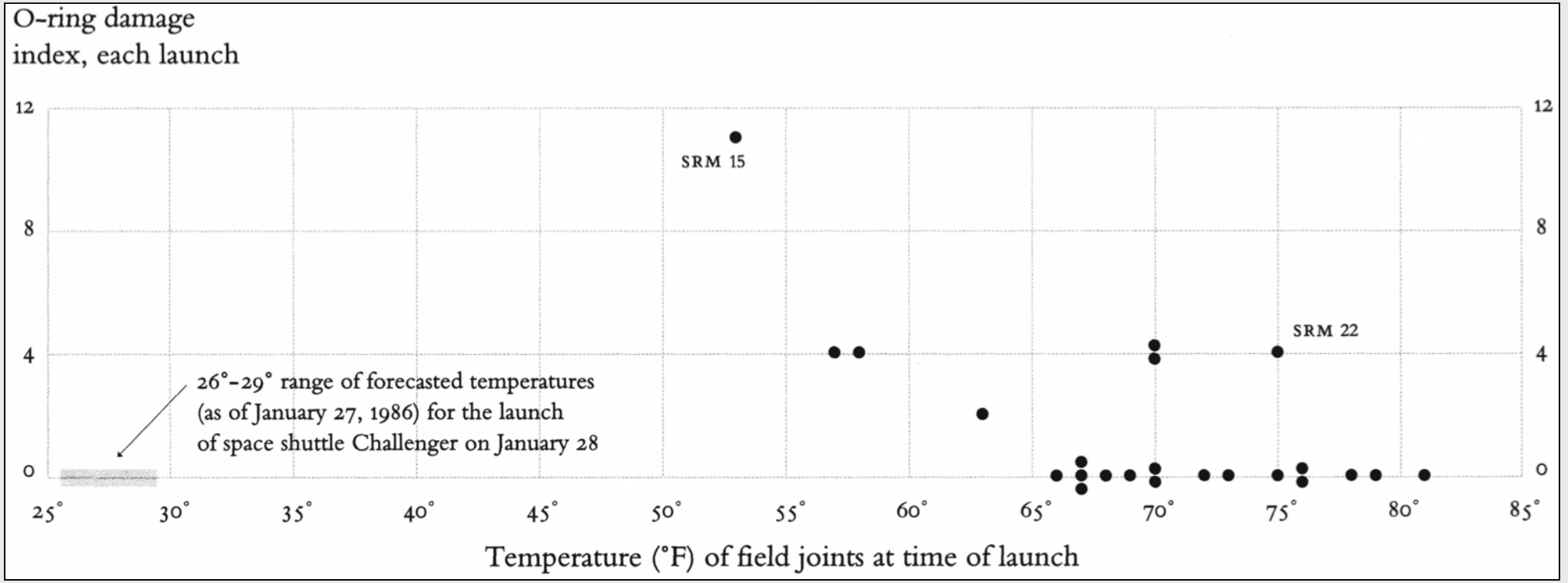


What NASA was shown



Tufte, Edward R. (1997) *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, Cheshire, Connecticut.

What NASA *should* have been shown



Tufte, Edward R. (1997) *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, Cheshire, Connecticut.

Week 13: *Data Visualization*

1. Plotting with Base R
2. Plotting with **ggplot2**
3. Tweaking your ggplot

Week 13: *Data Visualization*

1. Plotting with Base R
2. Plotting with **ggplot2**
3. Tweaking your ggplot

Today's data:

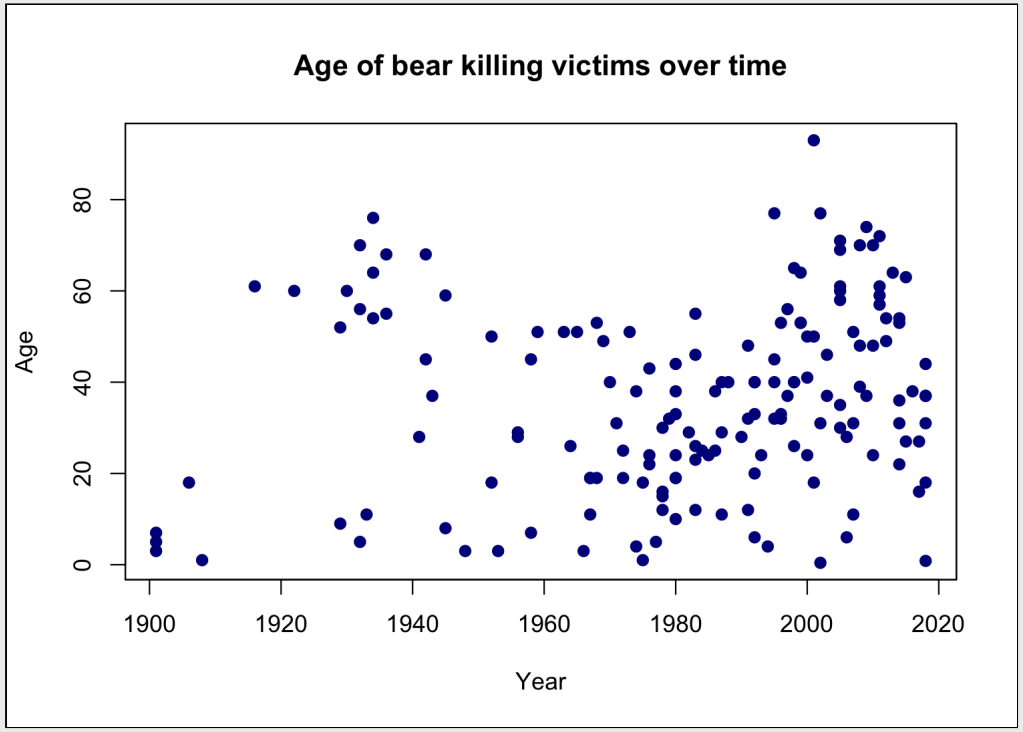
Bear attacks in North America

Explore the `bears` data frame:

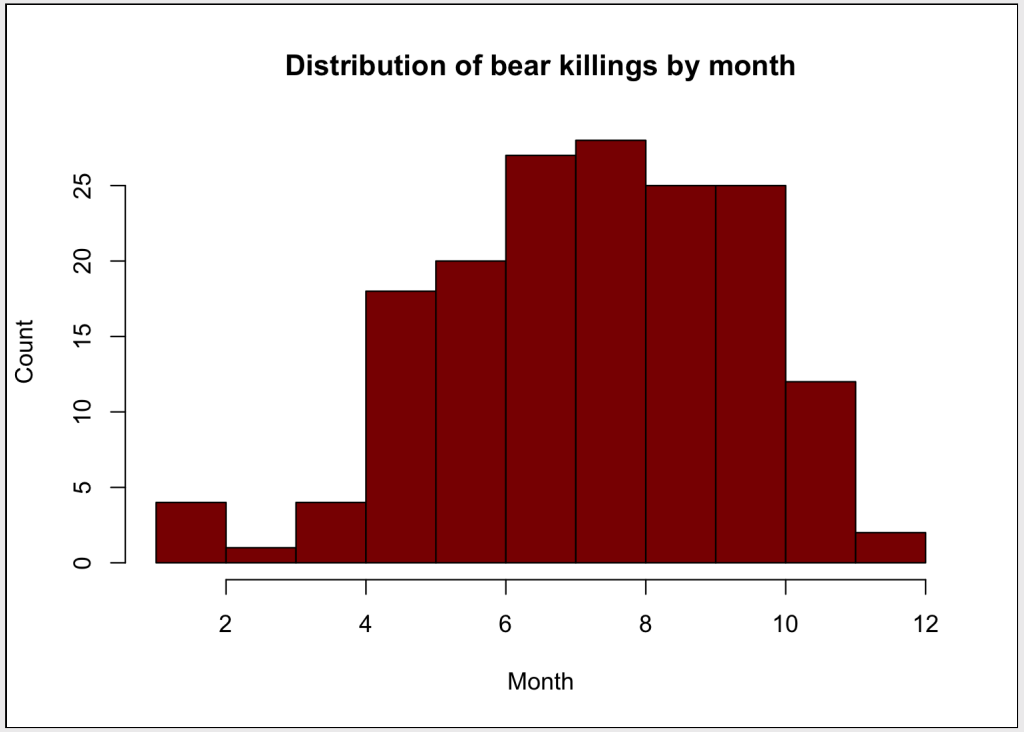
```
glimpse(bears)
head(bears)
```

Two basic plots in R

Scatterplots



Histograms



Scatterplots with `plot()`

Plot relationship between two variables

General syntax:

```
plot(x = x_vector, y = y_vector)
```

Scatterplots with `plot()`

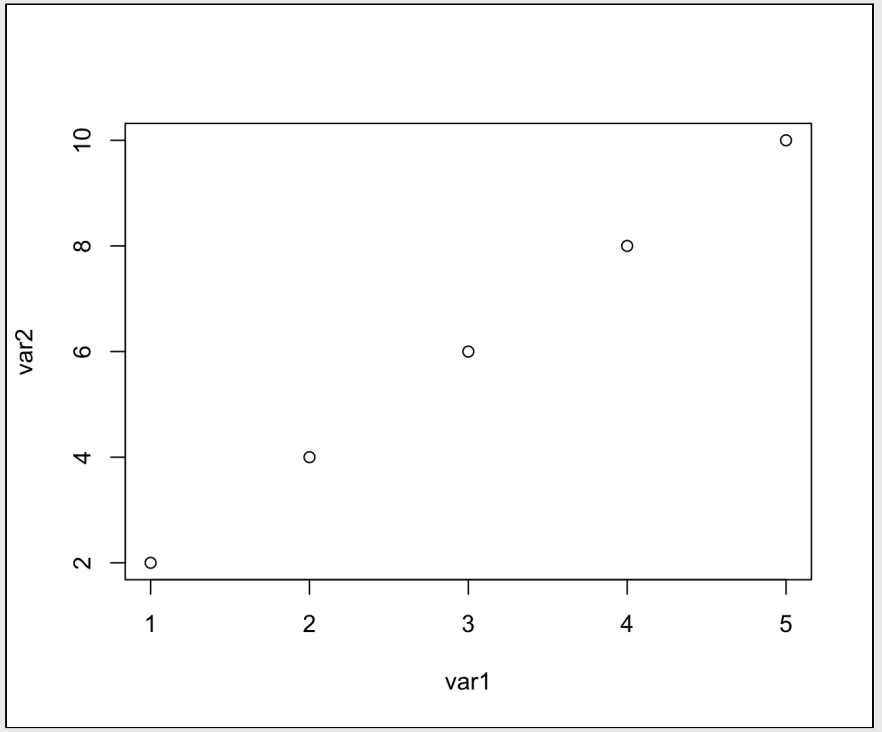
Plot relationship between two variables

General syntax:

```
plot(x = x_vector, y = y_vector)
```

Example:

```
var1 <- seq(1, 5)
var2 <- 2*var1
plot(x = var1, y = var2)
```



Scatterplots with `plot()`

`x` and `y` must have the same length!

```
var2 <- var2[-1]
```

```
length(var1) == length(var2)
```

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

```
plot(x = var1, y = var2)
```

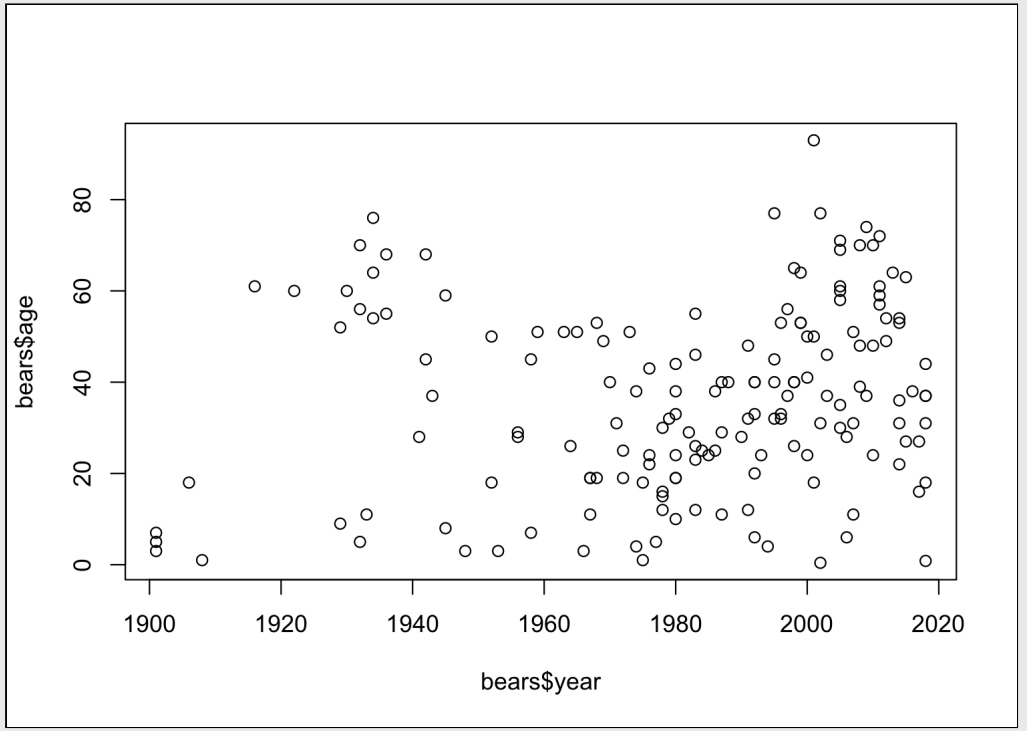
```
## Error in xy.coords(x, y, xlabel, ylabel, log): 'x' and 'y' lengths differ
```

Scatterplots with `plot()`

Plotting variables from a data frame:

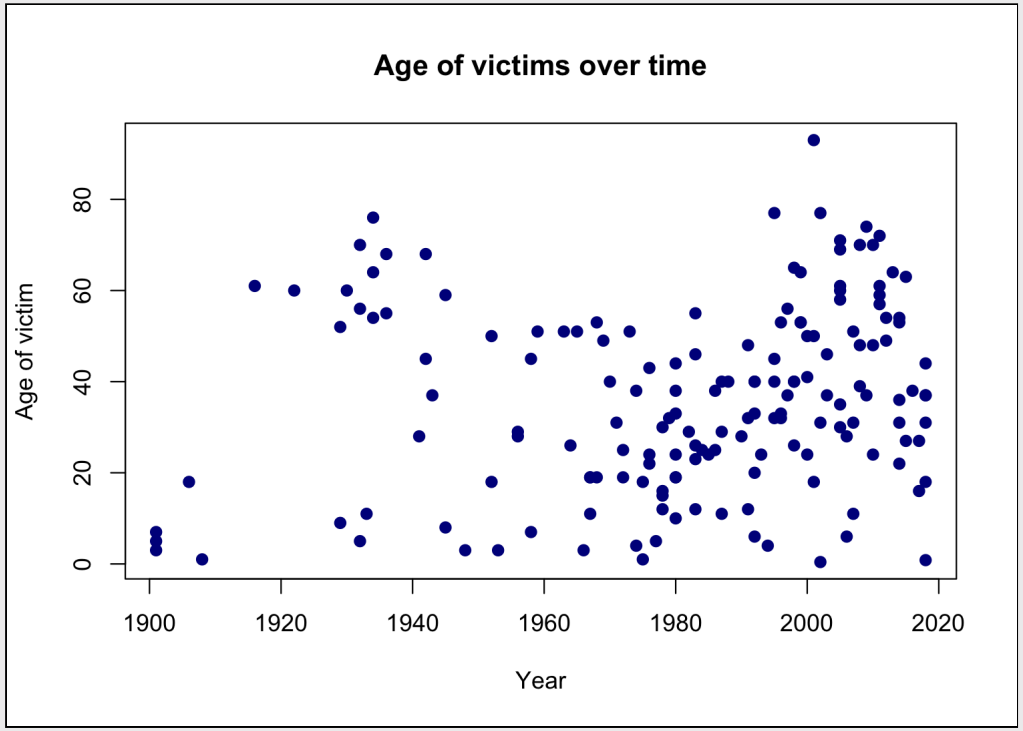
Plot `year` vs. `age`:

```
plot(x = bears$year, y = bears$age)
```



Making `plot()` pretty

```
plot(x    = bears$year,
     y    = bears$age,
     col  = 'darkblue', # Point color
     pch  = 19, # Point shape
     main = "Age of victims over time",
     xlab = "Year",
     ylab = "Age of victim")
```



Think pair share: `plot()`

10:00

Does the annual number of bird impacts appear to be changing over time?

Make a plot using the `birds` data frame to justify your answer

Hint: You may need to create a summary data frame to answer this question!

Bonus points: Make your plot pretty

Histograms with `hist()`

Plot the *distribution* of a single variable

General syntax:

```
hist(x = x_vector)
```

Histograms with `hist()`

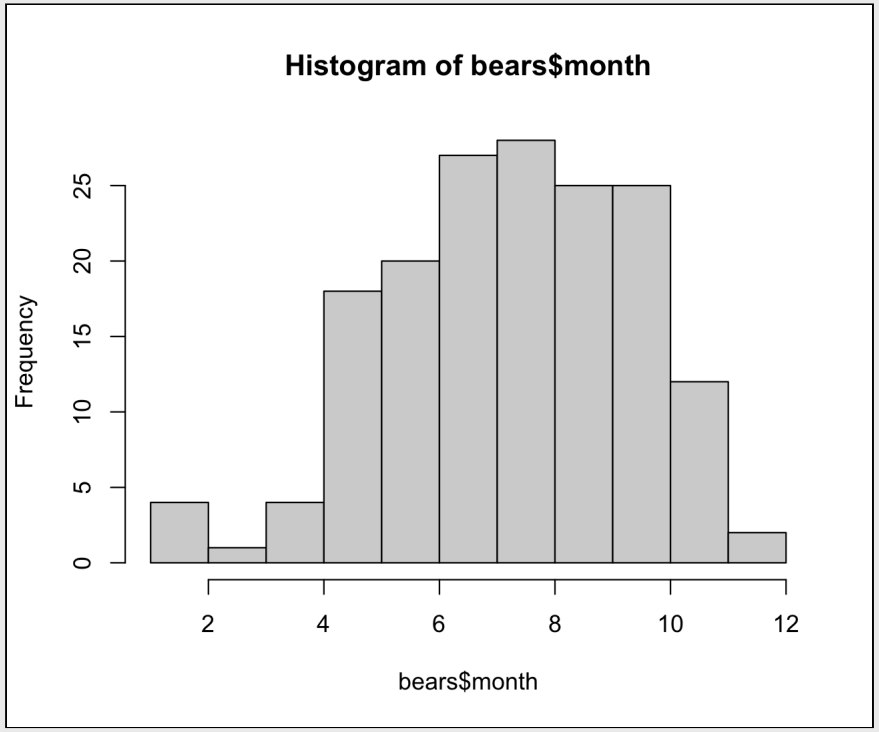
Plot the *distribution* of a single variable

General syntax:

```
hist(x = x_vector)
```

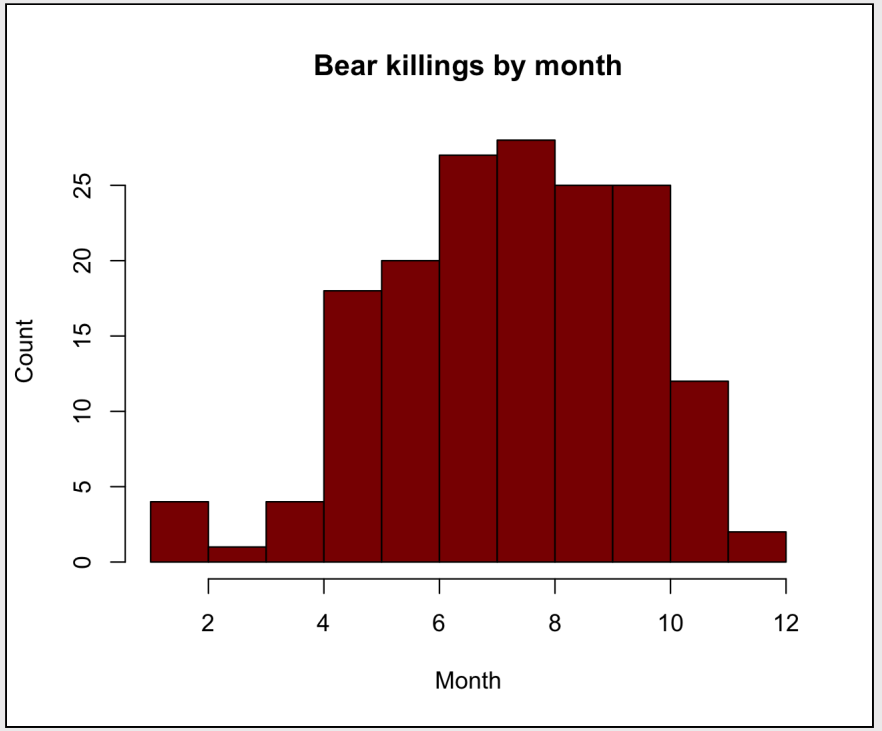
Example:

```
hist(bears$month)
```



Making `hist()` pretty

```
hist(x      = bears$month,
     breaks = 12,
     col    = 'darkred',
     main   = "Bear killings by month",
     xlab   = "Month",
     ylab   = "Count")
```



Think pair share: `hist()`

10:00

Make plots using the `birds` data frame to answer these questions

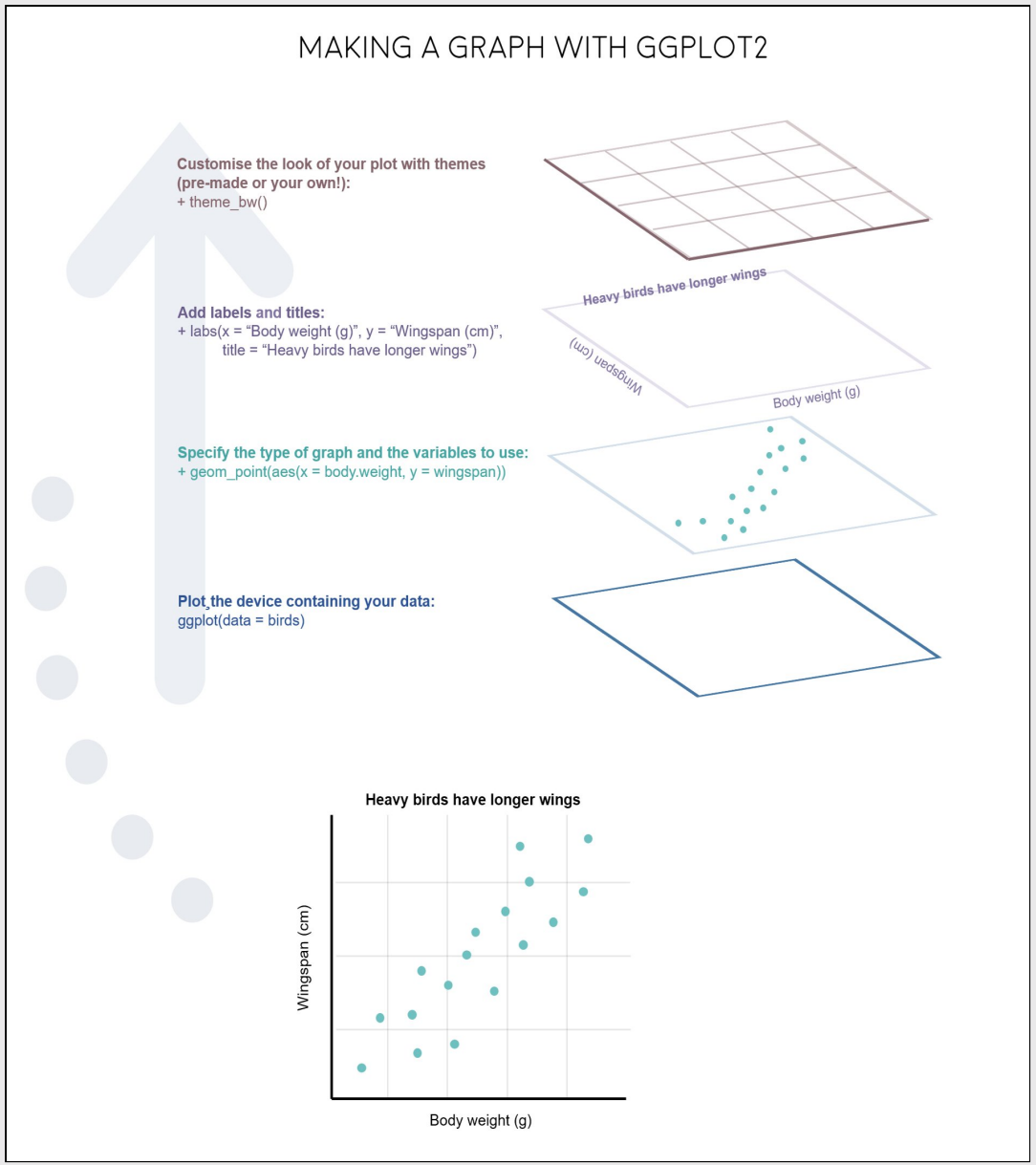
- Which months have the highest and lowest number of bird impacts in the dataset?
- Which aircrafts experience more impacts: 2-engine, 3-engine, or 4-engine?
- At what height do most impacts occur?

Bonus points: Make your plots pretty

Week 13: *Data Visualization*

1. Plotting with Base R
2. Plotting with **ggplot2**
3. Tweaking your ggplot

Advanced figures with ggplot2



"Grammar of Graphics"

Concept developed by Leland Wilkinson (1999)

ggplot2 package developed by Hadley Wickham (2005)

Making plot layers with ggplot2

1. The data (we'll use `bears`)

2. The aesthetic mapping (what goes on the axes?)

3. The geometries (points? bars? etc.)

Layer 1: The data

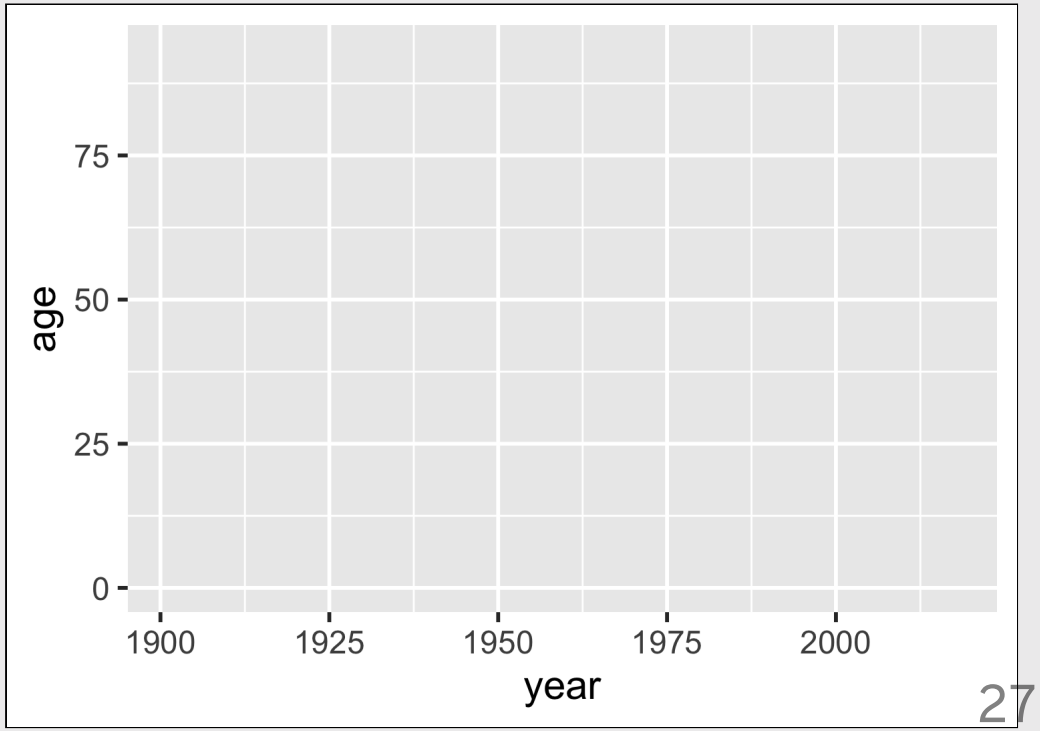
The `ggplot()` function initializes the plot with whatever data you're using

```
ggplot(data = bears)
```

Layer 2: The aesthetic mapping

The `aes()` function determines which variables will be *mapped* to the geometries (e.g. the axes)

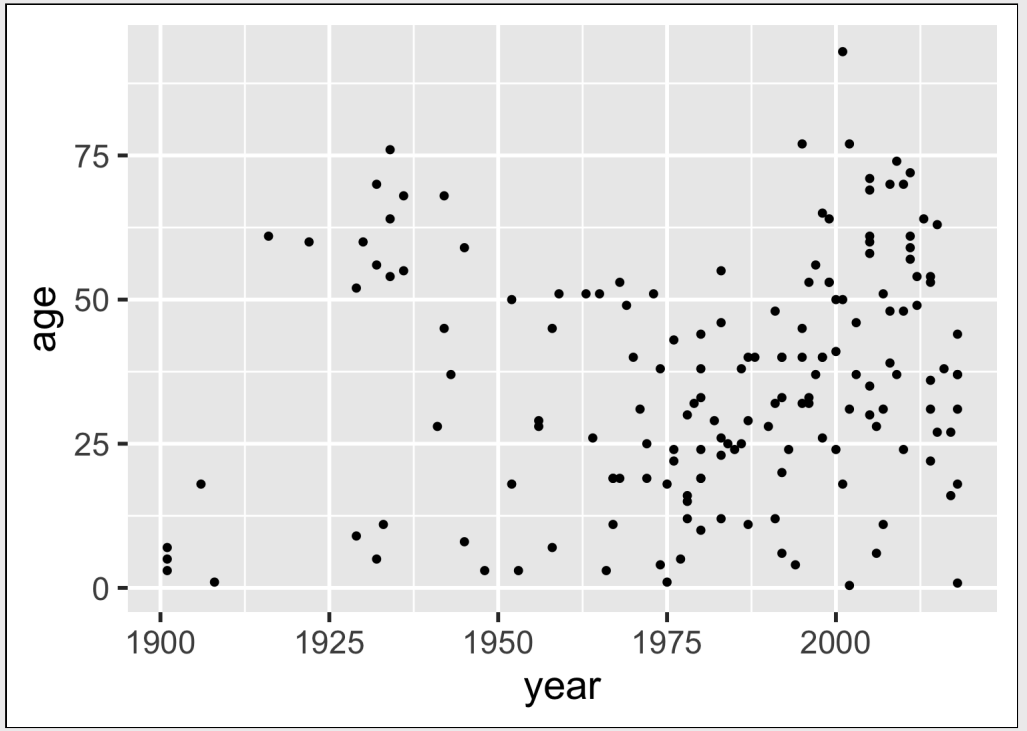
```
ggplot(data = bears,
       aes(x = year, y = age))
```



Layer 3: The geometries

Use + to add geometries (e.g. points)

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point()
```



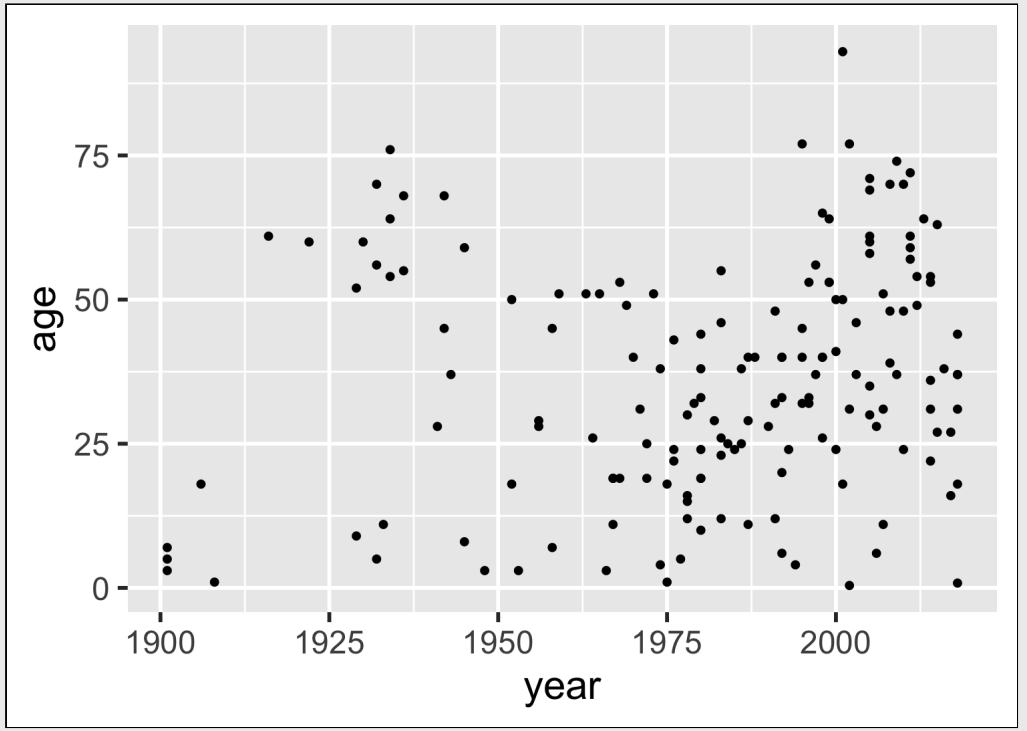
Other common geometries

- `geom_point()`: scatter plots
- `geom_line()`: lines connecting data points
- `geom_col()`: bar charts
- `geom_boxplot()`: boxes for boxplots

Scatterplots with `geom_point()`

Add points:

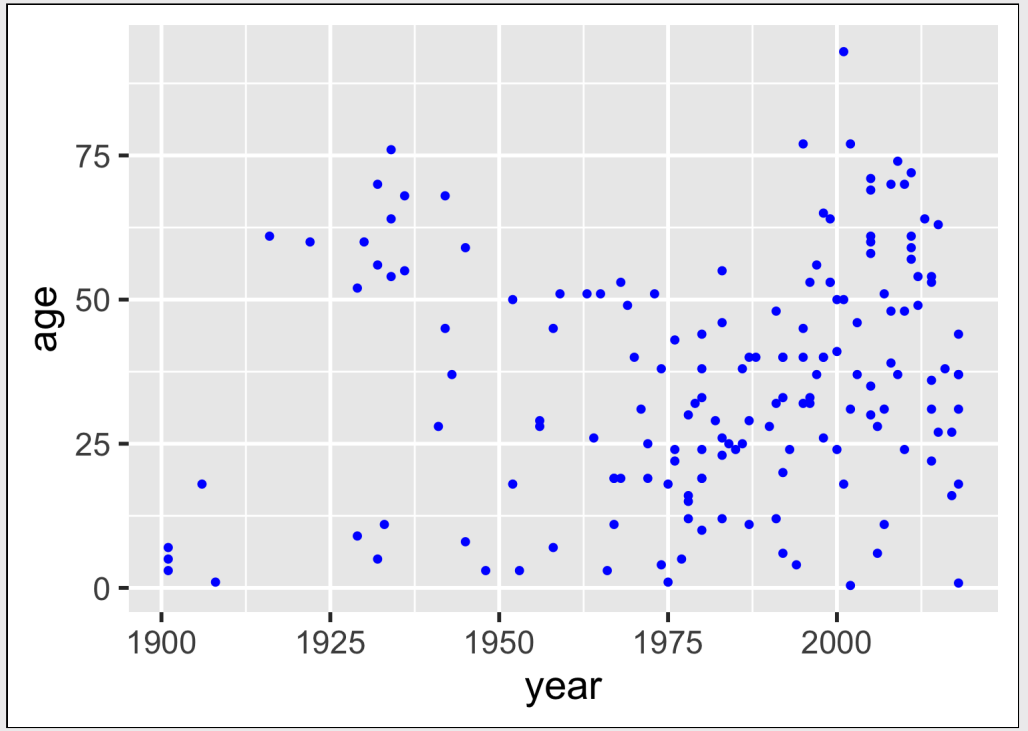
```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point()
```



Scatterplots with `geom_point()`

Change the color of all points:

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point(color = 'blue')
```

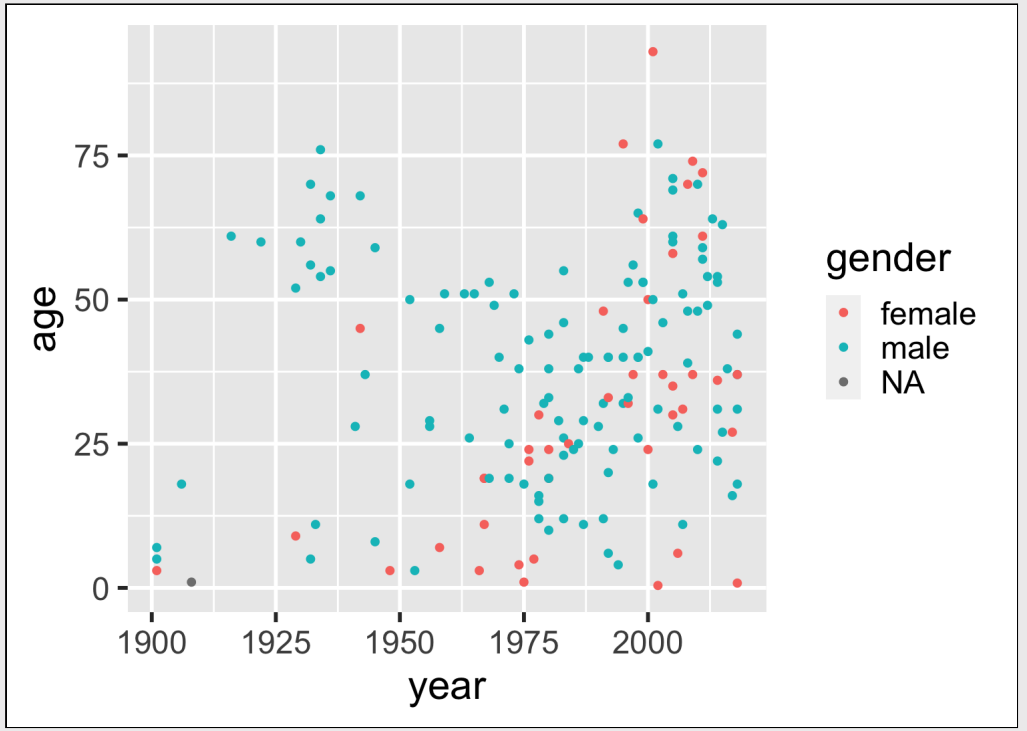


Scatterplots with `geom_point()`

Map the point color to a **variable**:

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point(aes(color = gender))
```

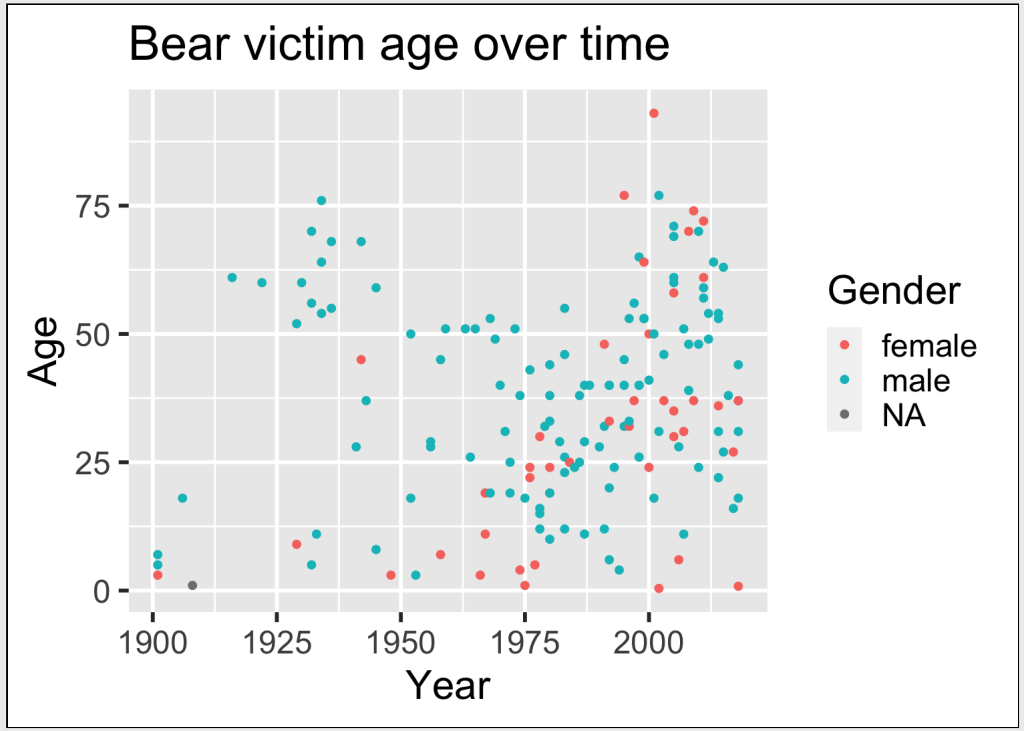
Note that `color = gender` is *inside* `aes()`



Scatterplots with `geom_point()`

Adjust labels with `labs()` layer:

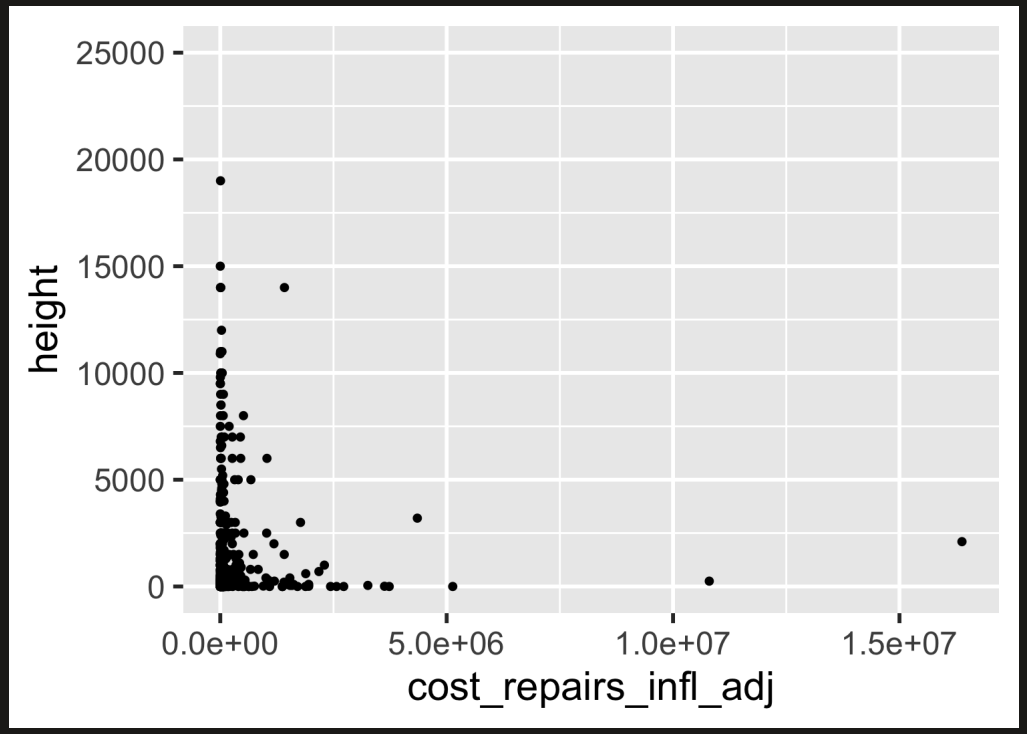
```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point(aes(color = gender)) +
    labs(x = "Year",
         y = "Age",
         title = "Bear victim age over
time",
         color = "Gender")
```

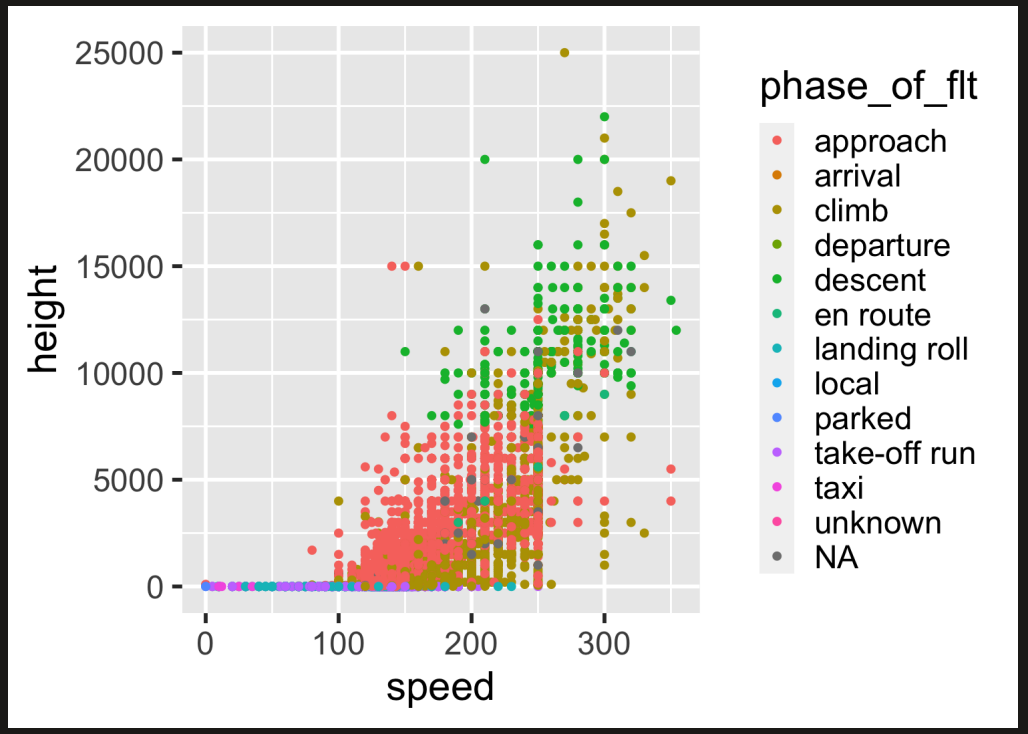


Think pair share: `geom_point()`

10:00

Use the `birds` data frame to create the following plots





Break

05:00

Make bar charts with `geom_col()`

With bar charts, you'll often need to create summary variables to plot

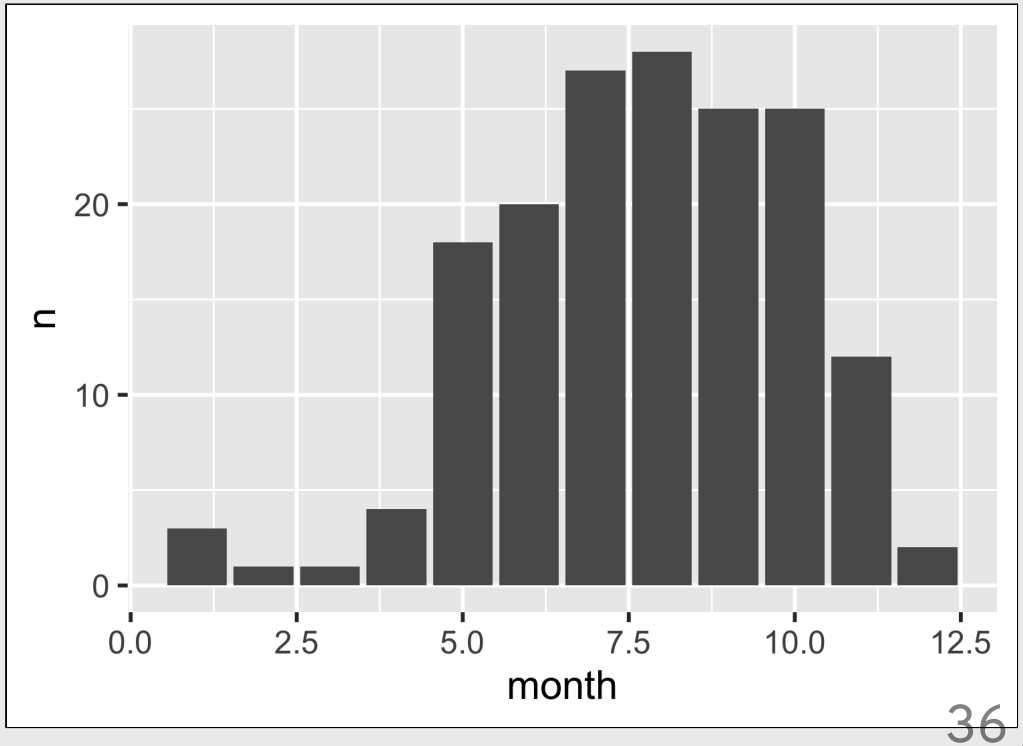
Step 1: Summarize the data

```
bear_months <- bears %>%
  count(month)
```

Step 2: Make the plot

```
ggplot(bear_months) +
  geom_col(aes(x = month, y = n))
```

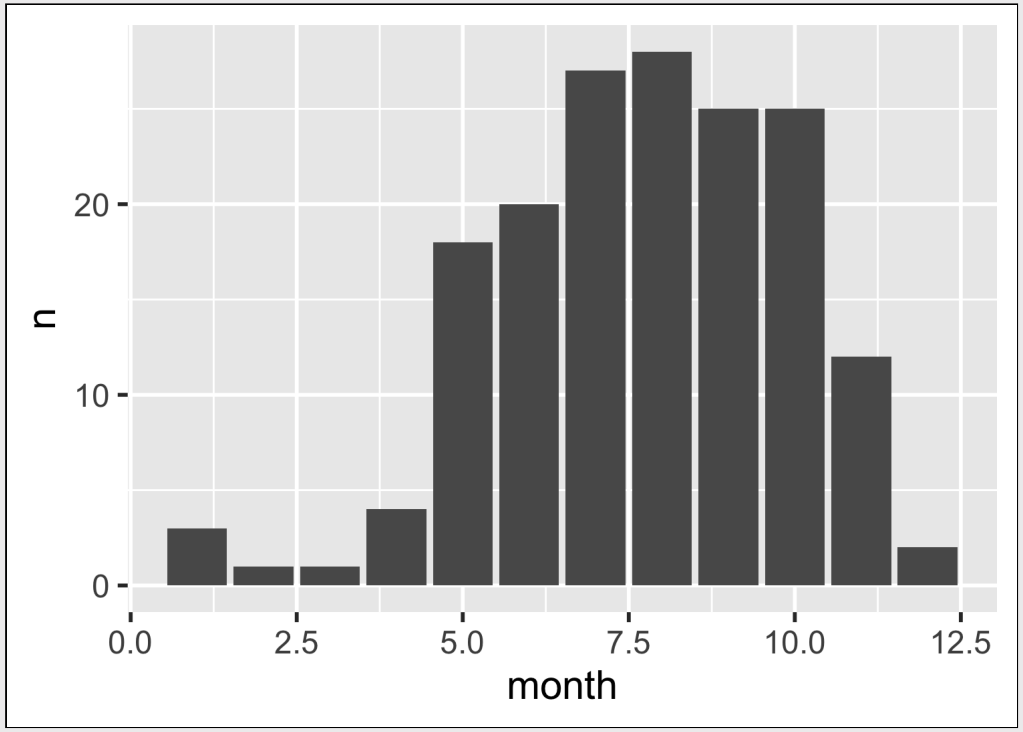
Example: count of attacks by month



Make bar charts with `geom_col()`

Alternative approach: piping directly into ggplot

```
bears %>%
  count(month) %>%  # Pipe into ggplot
  ggplot() +
  geom_col(aes(x = month, y = n))
```



Be careful with `geom_col()` vs. `geom_bar()`

`geom_col()`

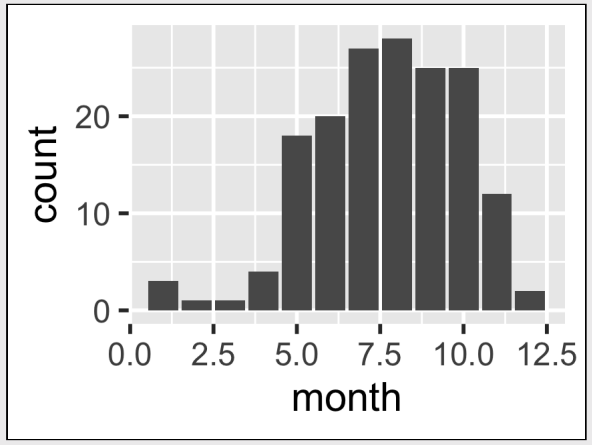
Map both `x` and `y`

```
bears %>%
  count(month) %>%
  ggplot() +
  geom_col(aes(x = month, y = n))
```

`geom_bar()`

Only map `x` (`y` is computed)

```
bears %>%
  ggplot() +
  geom_bar(aes(x = month))
```

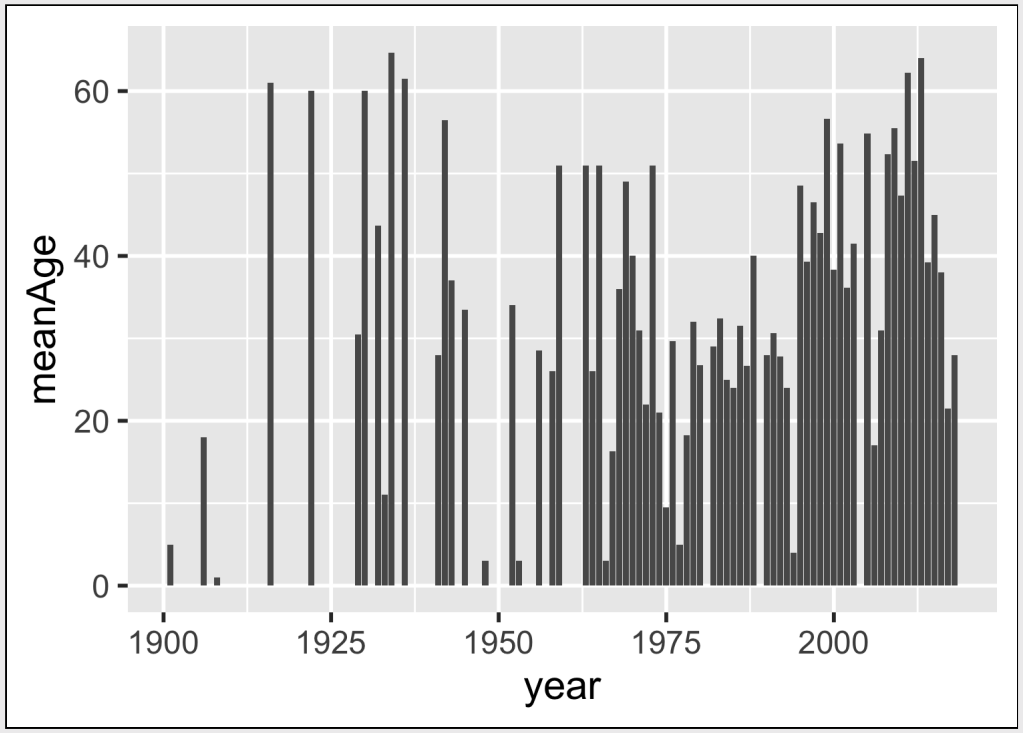


Make bar charts with `geom_col()`

Another example:

Mean age of victim in each year

```
bears %>%
  filter(!is.na(age)) %>%
  group_by(year) %>%
  summarise(meanAge = mean(age)) %>%
  ggplot() +
  geom_col(aes(x = year, y = meanAge))
```

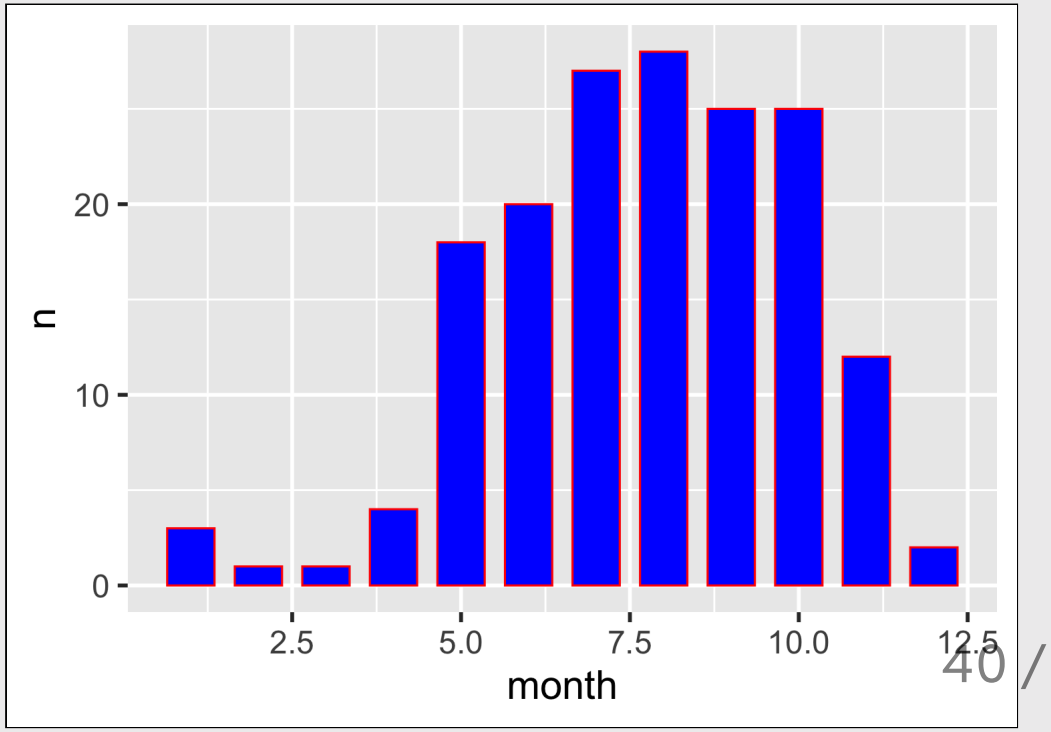


Change bar width: `width`

Change bar color: `fill`

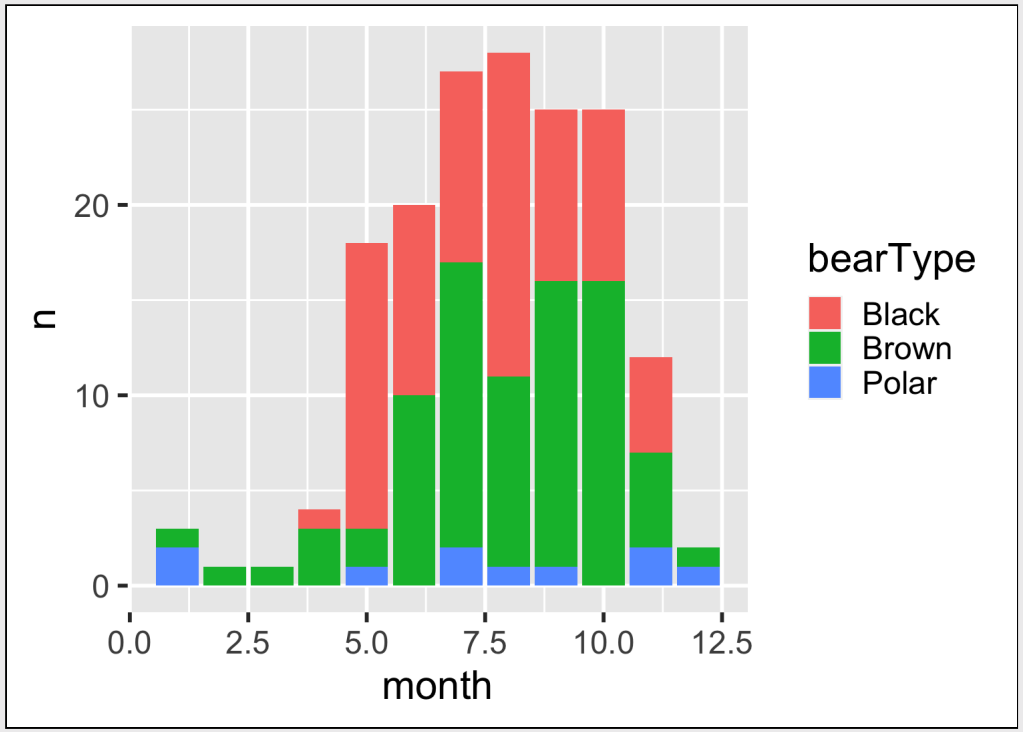
Change bar outline: `color`

```
bears %>%
  count(month) %>%
  ggplot() +
  geom_col(aes(x = month, y = n),
           width = 0.7,
           fill = "blue",
           color = "red")
```



Map the `fill` to `bearType`

```
bears %>%
  count(month, bearType) %>%
  ggplot() +
  geom_col(aes(x = month, y = n,
               fill = bearType))
```



Note that I had to summarize the count by both `month` and `bearType`

```
bears %>%
  count(month, bearType)
```

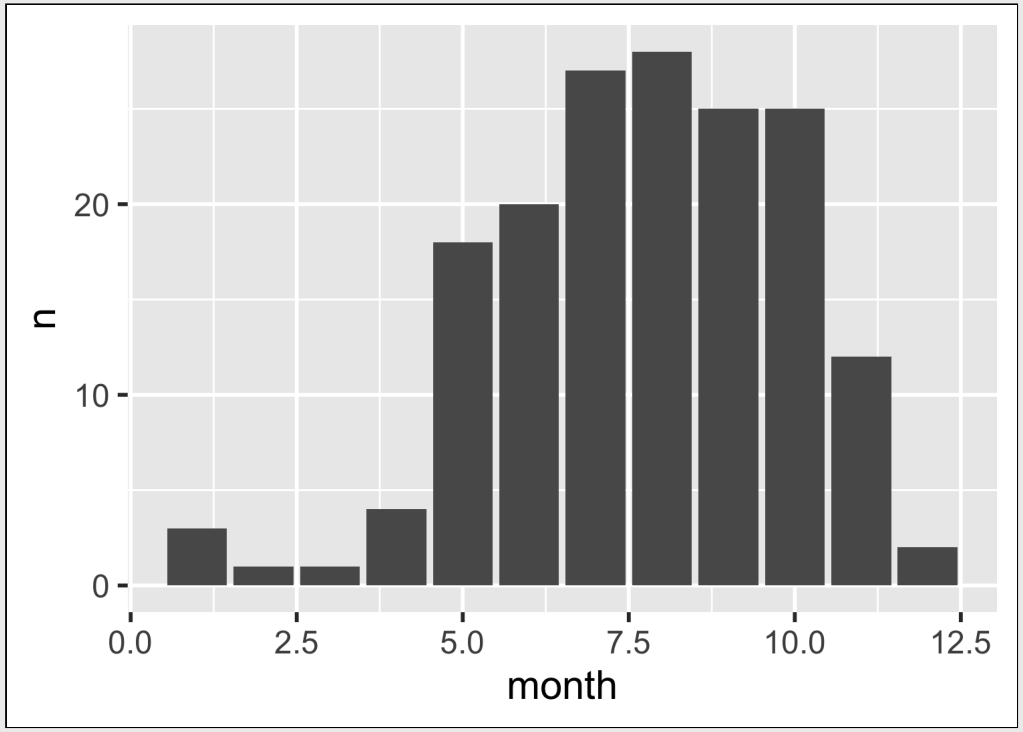
```
## # A tibble: 27 x 3
##    month bearType     n
##    <dbl> <chr>    <int>
##  1     1 Brown        1
##  2     1 Polar        2
##  3     2 Brown        1
##  4     3 Brown        1
##  5     4 Black        1
##  6     4 Brown        3
```

"Factors" = Categorical variables

By default, R makes numeric variables *continuous*

```
bears %>%
  count(month) %>%
  ggplot() +
  geom_col(aes(x = month, y = n))
```

The variable `month` is a *number*

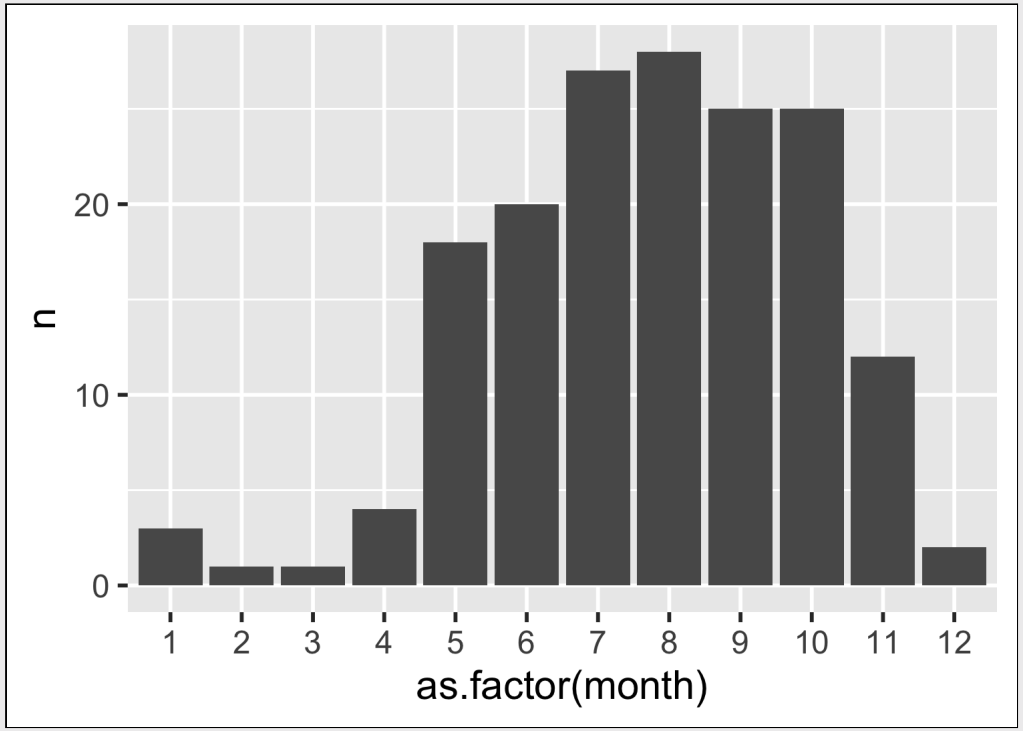


"Factors" = Categorical variables

You can make a continuous variable *categorical* using `as.factor()`

```
bears %>%
  count(month) %>%
  ggplot() +
  geom_col(aes(x = as.factor(month),
               y = n))
```

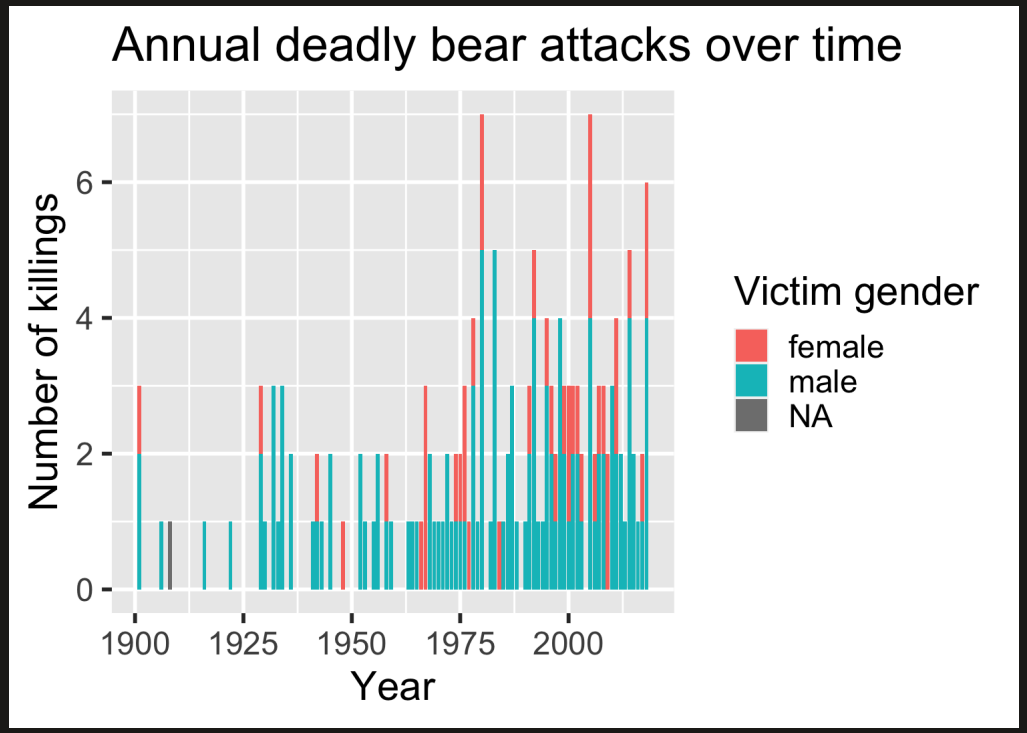
The variable `month` is a *factor*

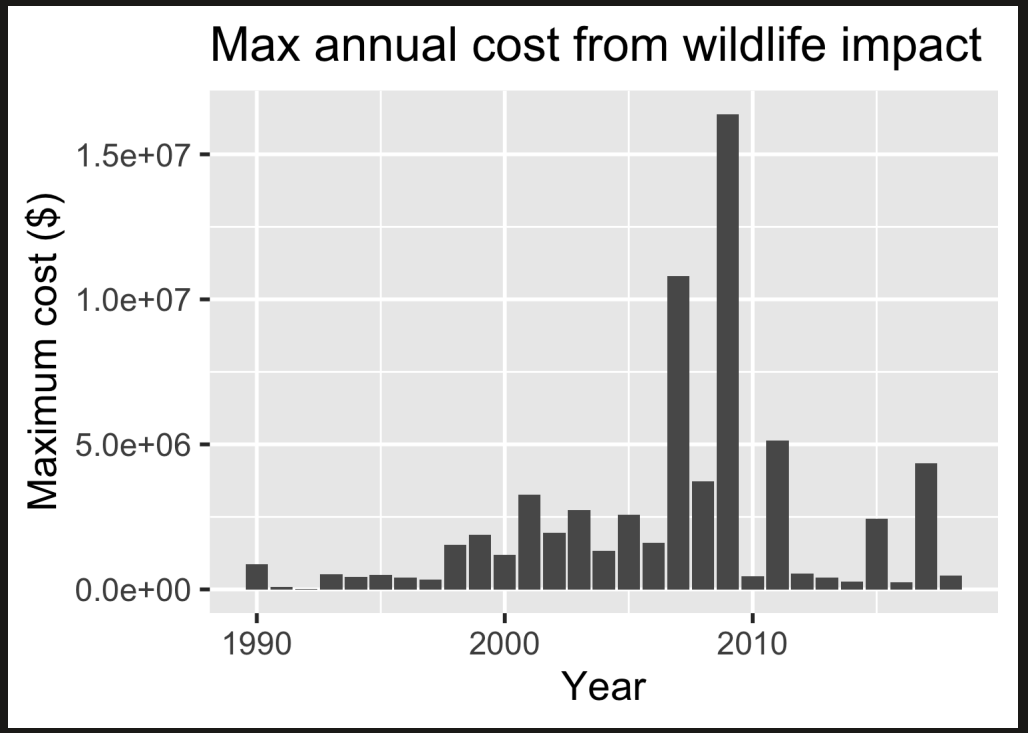


Think pair share: `geom_col()`

15:00

Use the `bears` and `birds` data frame to create the following plots





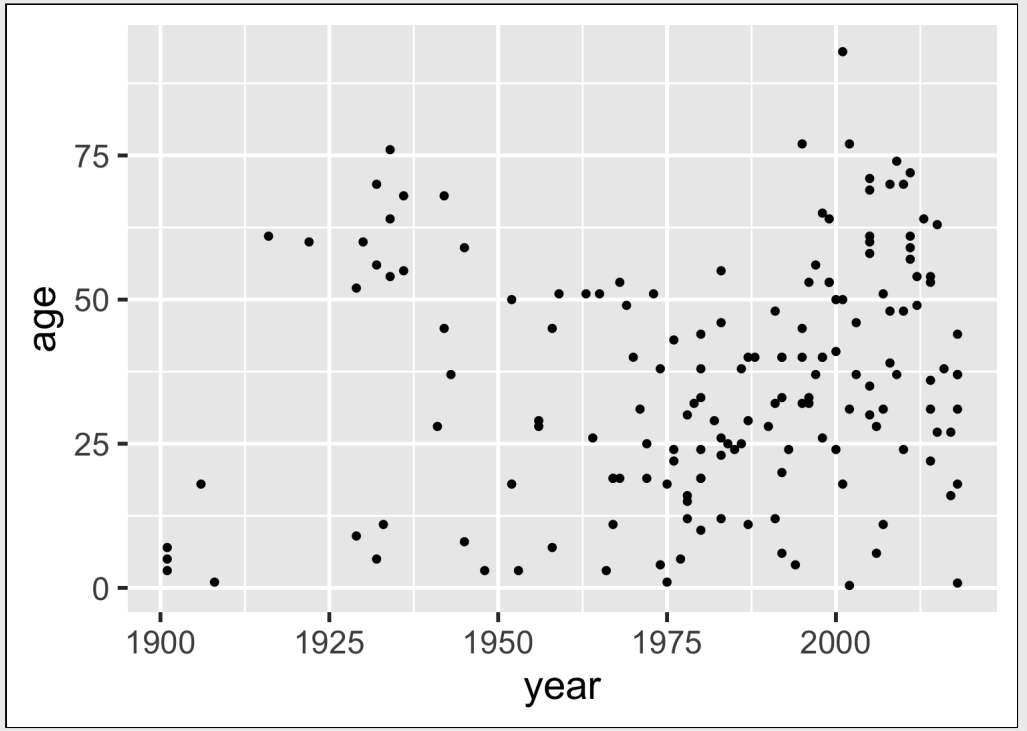
Week 13: *Data Visualization*

1. Plotting with Base R
2. Plotting with **ggplot2**
3. Tweaking your ggplot

Working with themes

Themes change *global* features of your plot, like the background color, grid lines, etc.

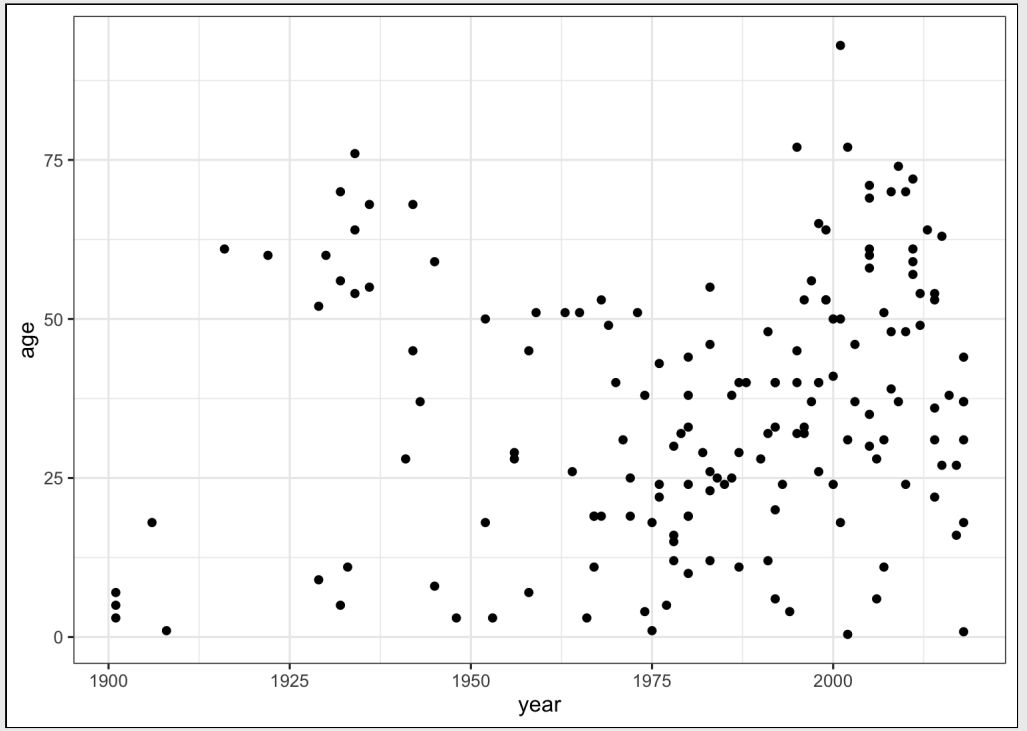
```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point()
```



Working with themes

Themes change *global* features of your plot, like the background color, grid lines, etc.

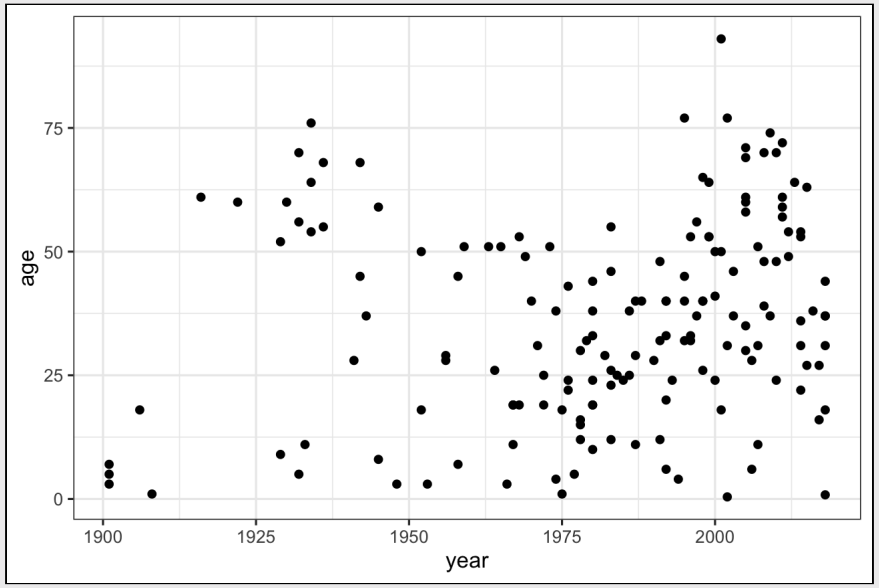
```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_bw()
```



Common themes

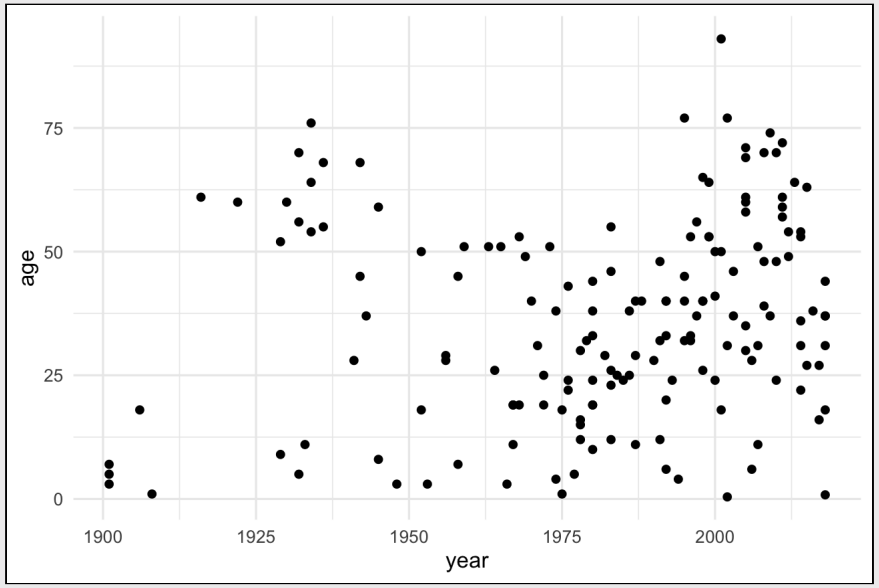
`theme_bw()`

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_bw()
```



`theme_minimal()`

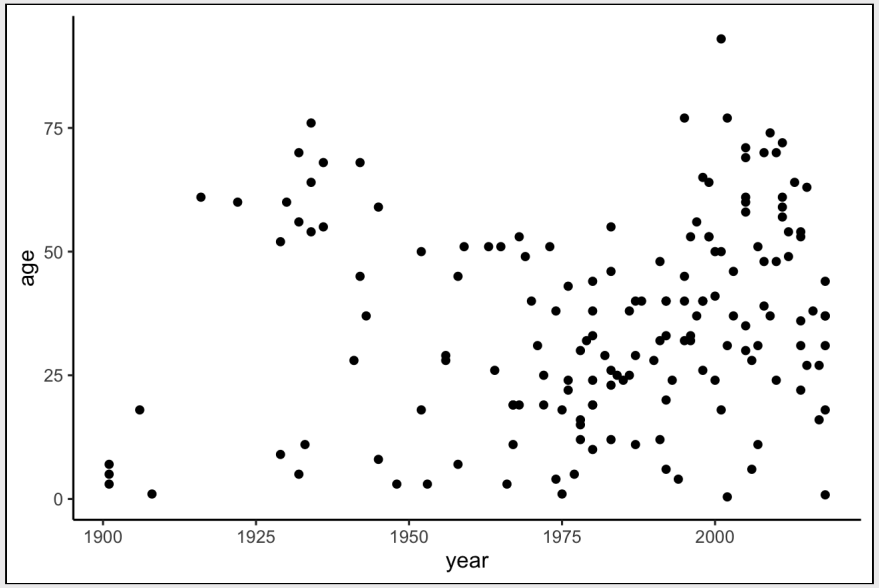
```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_minimal()
```



Common themes

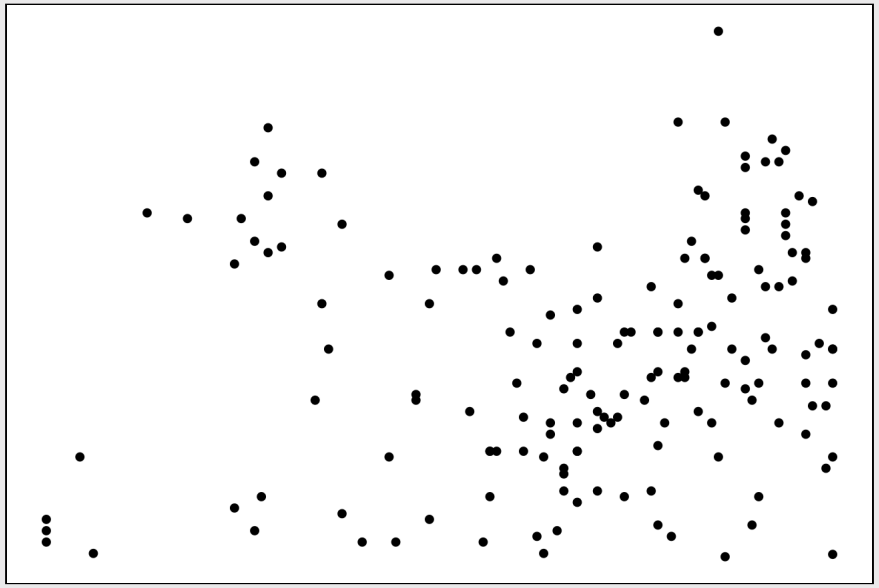
`theme_classic()`

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_classic()
```



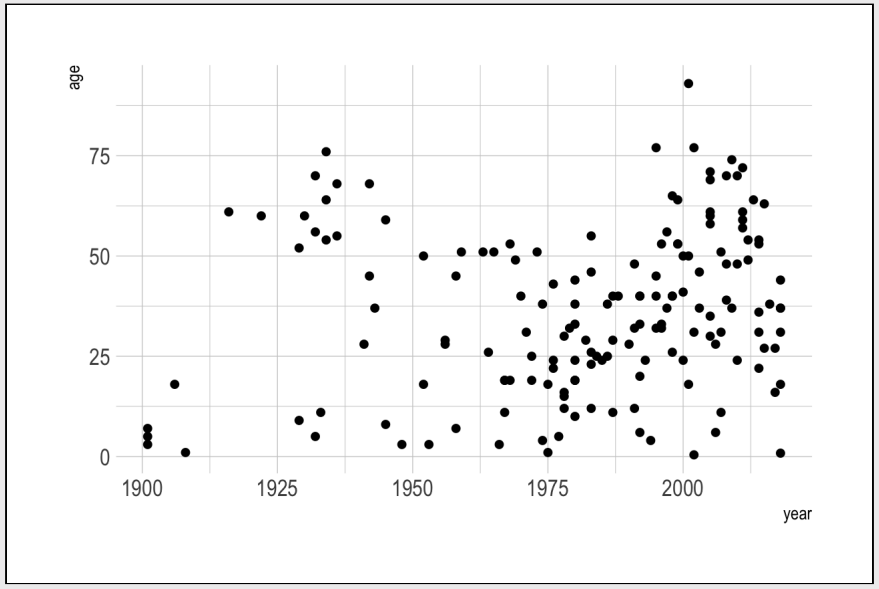
`theme_void()`

```
ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_void()
```

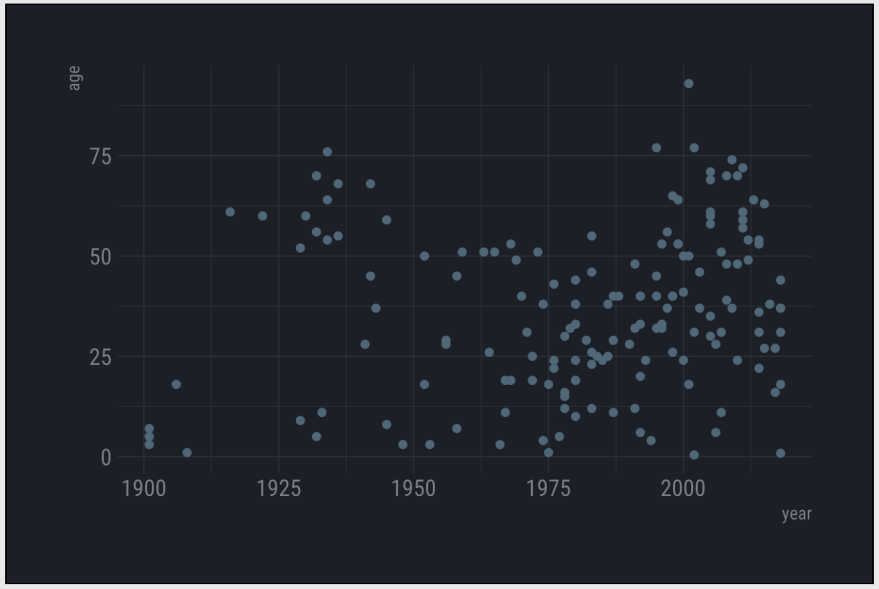


Other themes: **hrbrthemes**

```
library(hrbrthemes)

ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_ipsum()
```

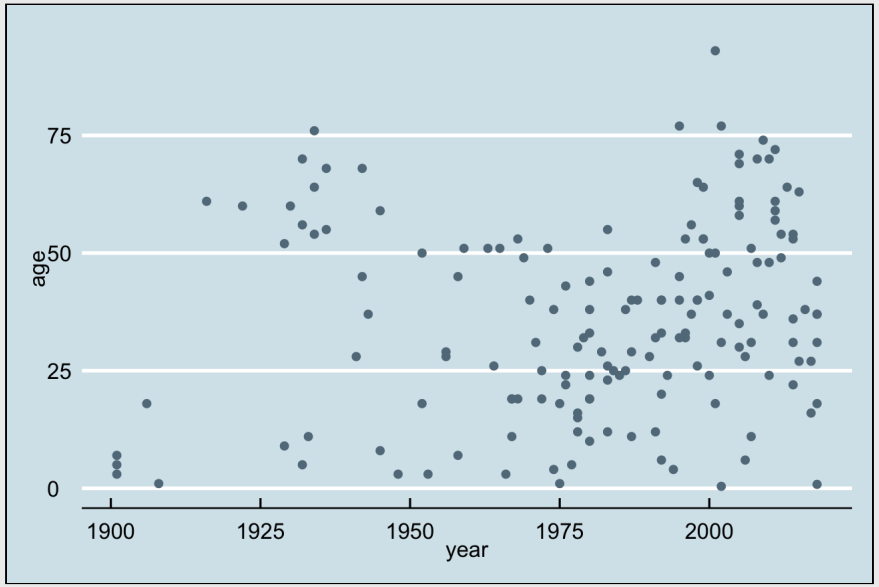



```
library(hrbrthemes)

ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_ft_rc()
```

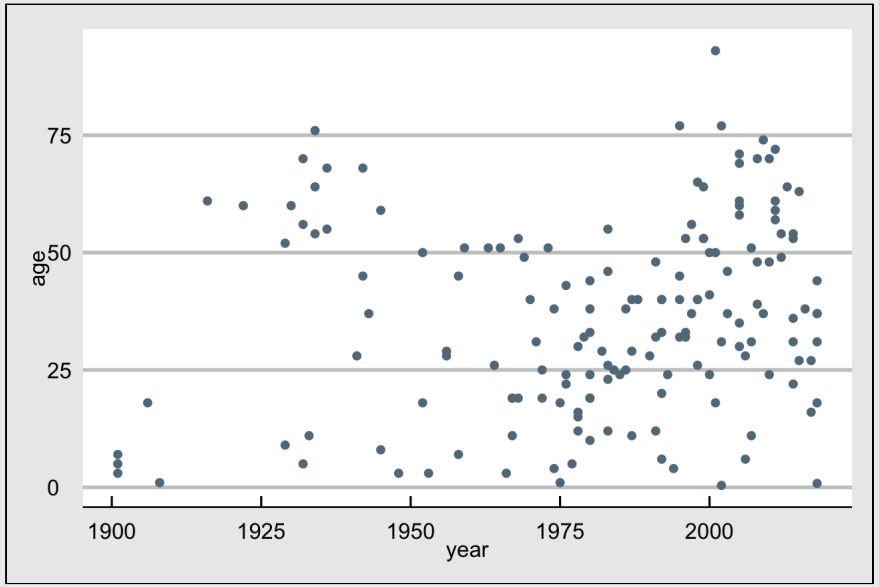


Other themes: **ggthemes**

```
library(ggthemes)

ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_economist()
```




```
library(ggthemes)

ggplot(data = bears,
       aes(x = year, y = age)) +
    geom_point() +
    theme_economist_white()
```



Save figures with `ggsave()`

First, assign the plot to an object name:

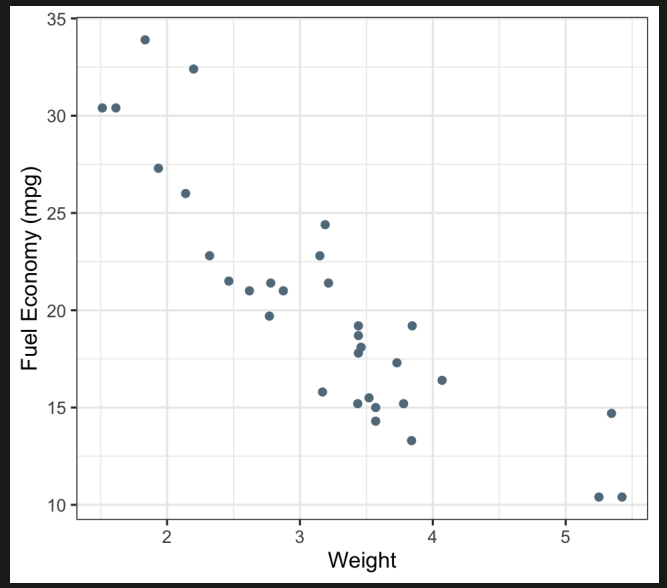
```
scatterPlot <- ggplot(data = bears) +
    geom_point(aes(x = year, y = age))
```

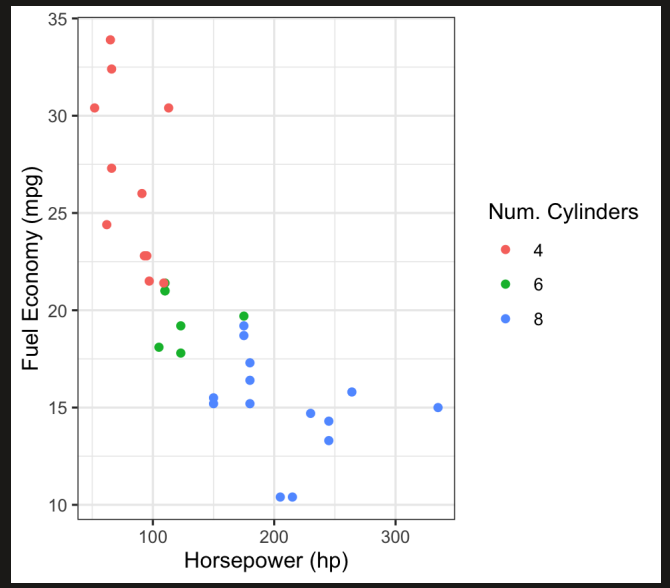
Then use `ggsave()` to save the plot:

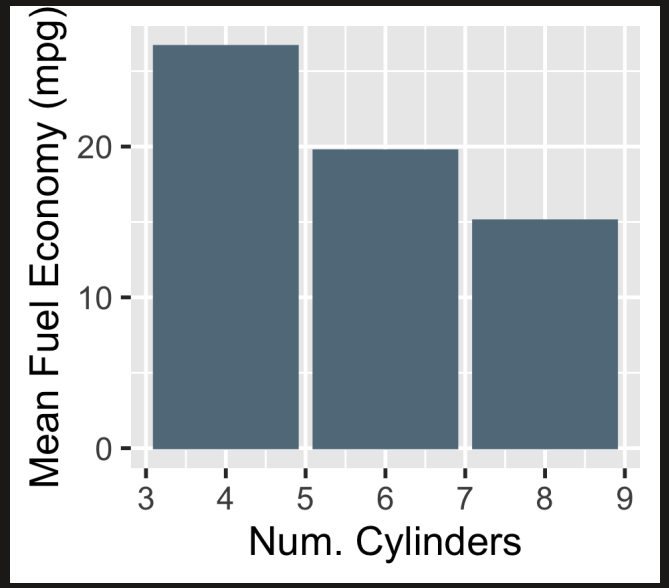
```
ggsave(filename = here('plots', 'scatterPlot.png'),
       plot   = scatterPlot,
       width  = 6, # inches
       height = 4)
```

Extra practice 1

Use the `mtcars` data frame to create the following plots







Extra practice 2

Use the `mpg` data frame to create the following plot