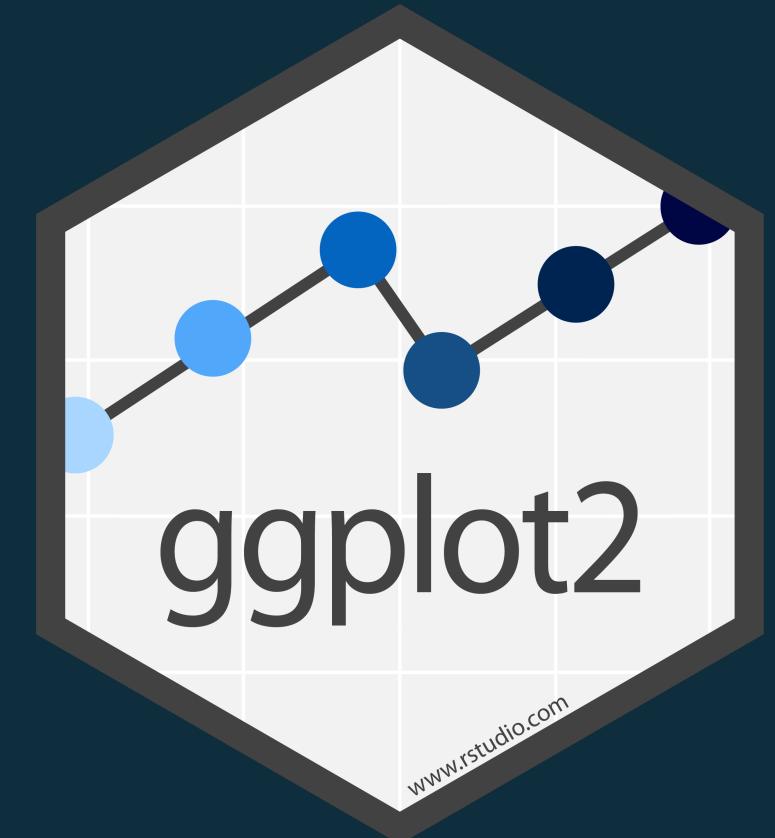


# Data visualization with ggplot2

## Introduction to R - Day 2

Instructor: Selina Baldauf

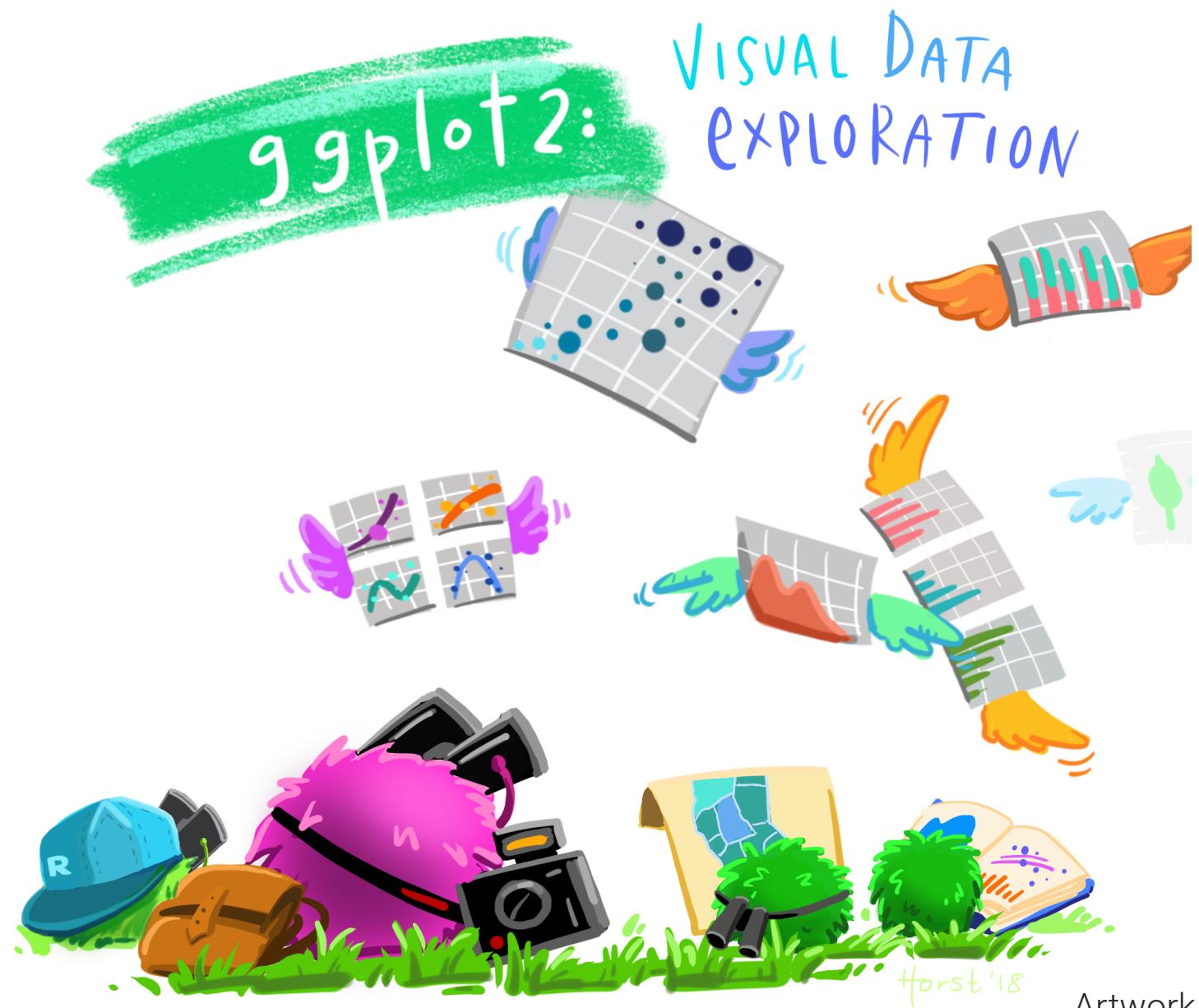
Freie Universität Berlin - Theoretical Ecology



2021-06-15 (updated: 2021-07-31)

# ggplot2

- ggplot2 was developed by Hadley Wickham in 2005
- most popular package for building graphics in R
- one of the most downloaded packages from CRAN



# ggplot2:

Build a data  
**MASTERpiece**



HORST '18

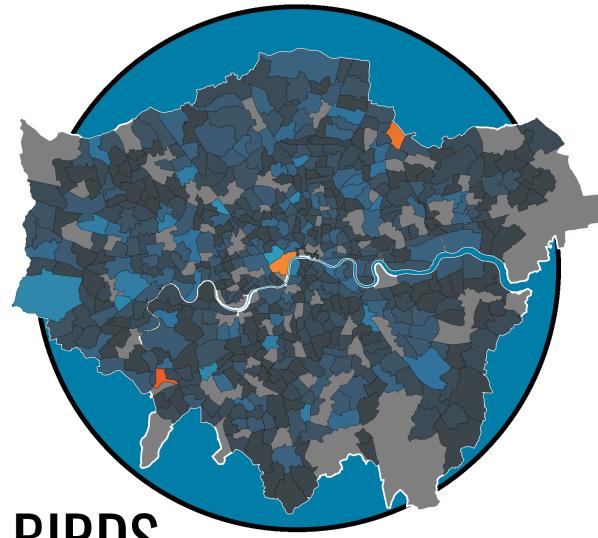
Artwork by Allison Horst 4 / 46

# A ggplot showcase

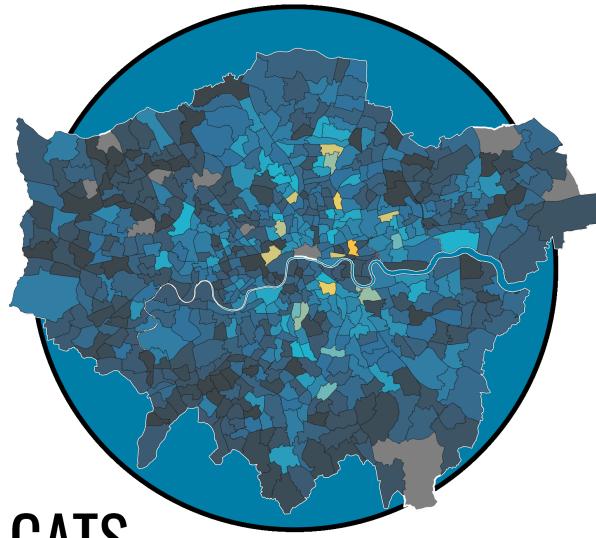
Some examples of plots you can create with ggplot

## Frequency of Rescues of Birds, Cats and Dogs in London from 2009-2021

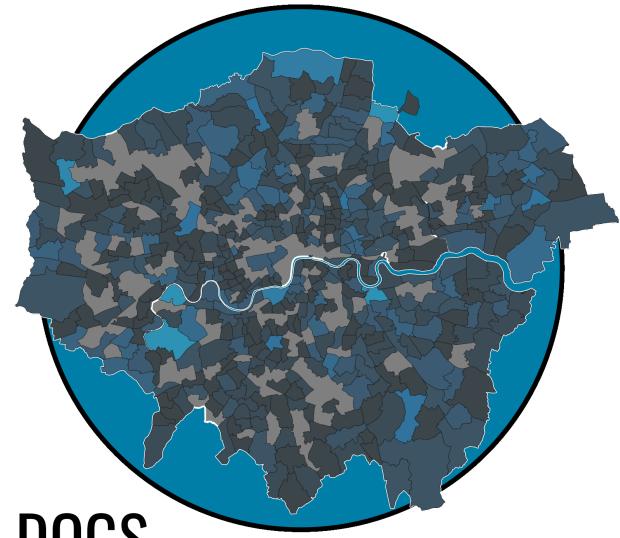
Illustrated below in three choropleth maps are rescues of birds, cats and dogs in London wards. Darker colors indicate lower rescue numbers while brighter colors indicate a greater number of rescues in that ward.



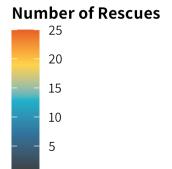
BIRDS



CATS



DOGS

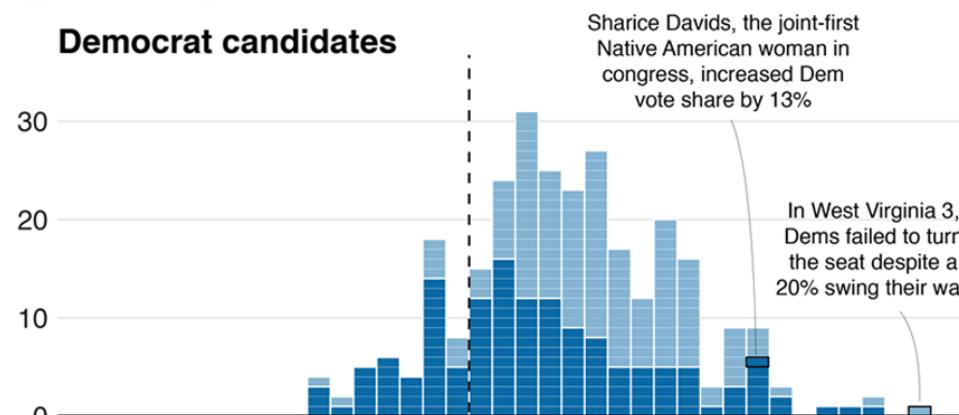


Data: London.gov | Graphic: @jakekaupp

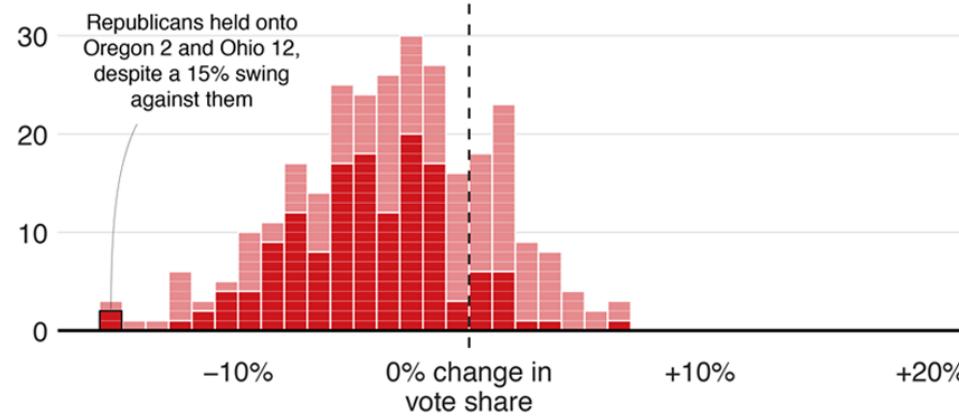
## Blue wave

■ Won seat ■ Didn't win

### Democrat candidates

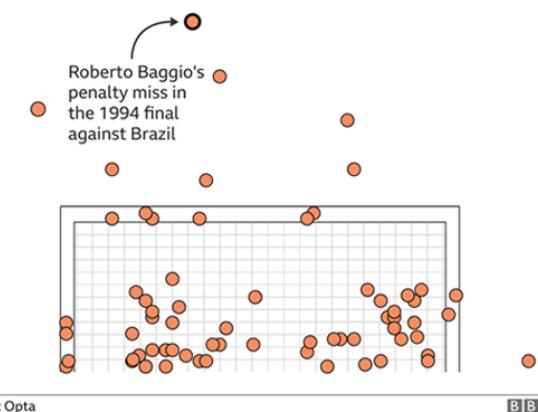


### Republican candidates



## Where penalties are saved

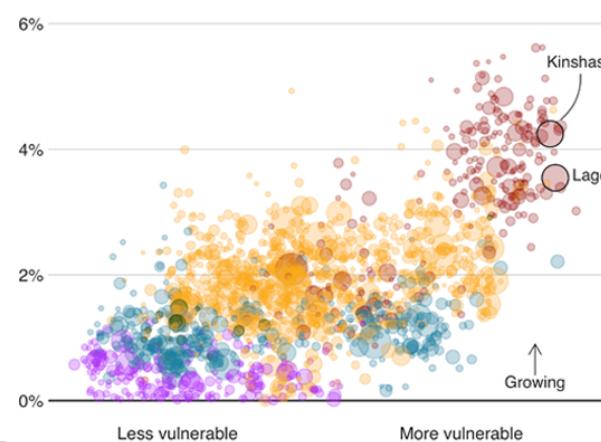
World Cup shootout misses and saves, 1982-2014



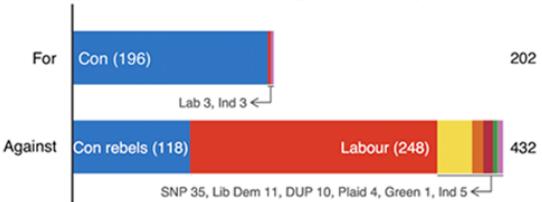
## Fast-growing cities face worse climate risks

Population growth 2018-2035 over climate change vulnerability

■ Africa ■ Asia ■ Americas ■ Europe ■ Oceania

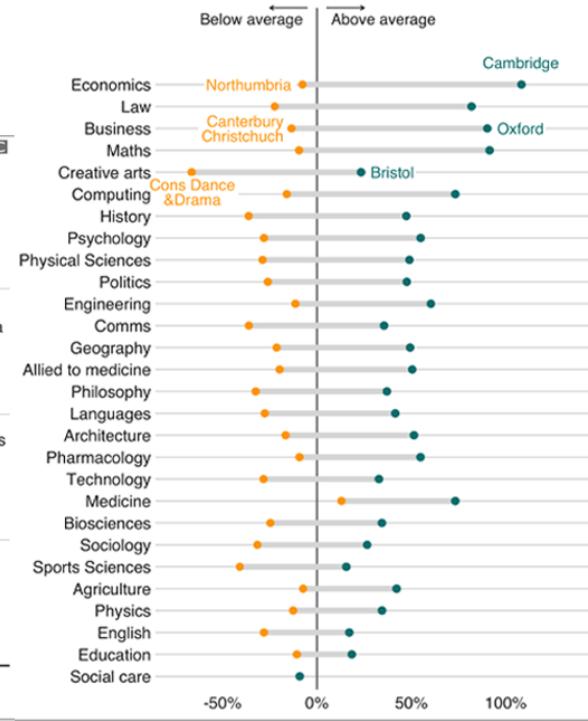


## MPs rejected Theresa May's deal by 230 votes

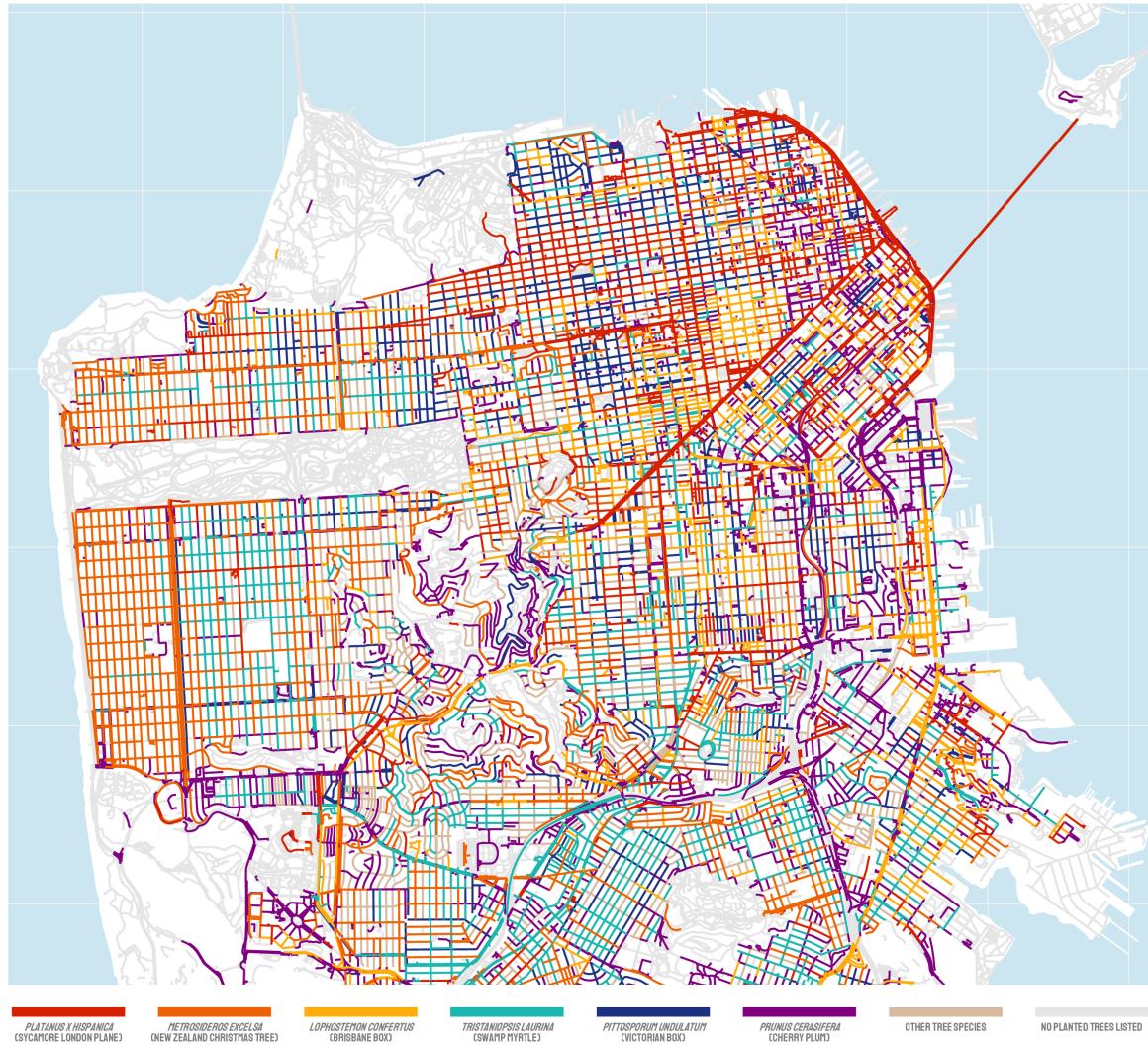


## Earnings vary across unis even within subjects

Impact on men's earnings relative to the average degree

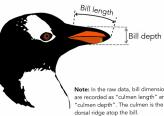


## THE DOMINANT TREE SPECIES' PLANTED ALONG SAN FRANCISCO'S ROADS

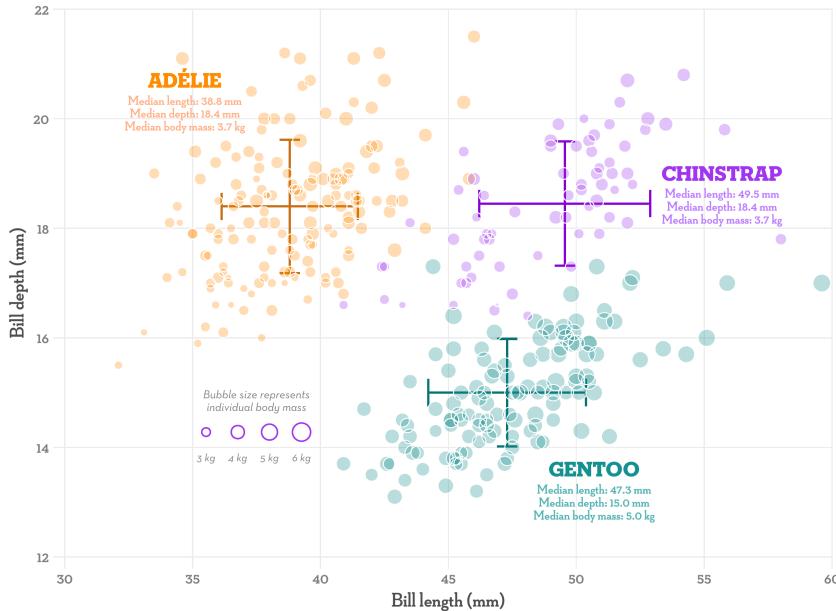


## BILL DIMENSIONS OF BRUSH-TAILED PENGUINS

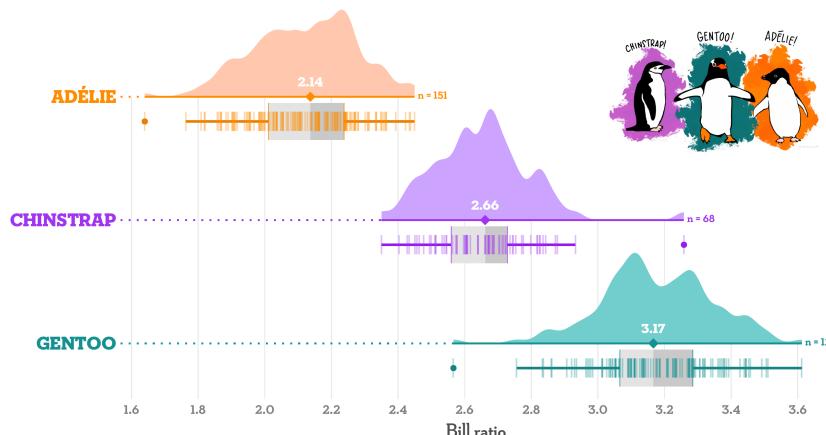
*Pygoscelis adeliae* (Adélie penguin) • *P. antarctica* (Chinstrap penguin) • *P. papua* (Gentoo penguin)



A. Scatterplot of bill length versus bill depth (error bars show median +/- sd)



B. Distribution of the bill ratio, estimated as bill length divided by bill depth



Visualization by Cédric Scherer,  
code available on [Github](#)

# Advantages of ggplot

- consistent grammar/structure
- flexible structure allows you to produce any type of plot
- highly customizable appearance (themes)
- many extension packages that provide
  - Additional plot types
  - Additional themes
  - Color palettes
  - Animation
  - Composition of multiple plots
  - ...
- Active community that provides help and inspiration

# Basic idea of ggplot

Shorten the distance from mind to page

(Hadley Wickham)

`ggplot2` is an implementation of the grammar of graphics by Leland Wilkinson

- coherent system for describing and building graphs
- basic idea: stack distinct **layers of graphical elements** to create a plot

# Layers of a ggplot

Layer	Function	Details
Data	<code>ggplot(data)</code>	The data that you want to plot.
Mapping	<code>aes()</code>	Aesthetic mappings of the geometric and statistical objects.
Geometries	<code>geom_*</code>	Graphical representation of aesthetics as point, line, polygon, ...

# Layers of a ggplot

Layer	Function	Details
Data	<code>ggplot(data)</code>	The data that you want to plot.
Mapping	<code>aes()</code>	Aesthetic mappings of the geometric and statistical objects.
Geometries	<code>geom_*</code> ()	Graphical representation of aesthetics as point, line, polygon, ...
Scales	<code>scale_*</code> ()	Translate between variable values and properties (e.g. categories -> colour)
Coordinates	<code>coord_*</code> ()	Interprets and maps the position values of the data.
Facets	<code>facet_*</code> ()	Split your plot into multiples.
Themes	<code>theme()</code> and <code>theme_*</code> ()	Visual look of the plot not related to the data.

# The data

The `ggplot` package has a built-in data set called `msleep` about the sleep times of 83 mammals.

Data variables (among others):

- **vore**: carnivore, omnivore, herbivore?
- **conservation**: conservation status of the animal
- **sleep\_total**: total amount of sleep [h]
- **brainwt**: brain weight [kg]
- **bodywt**: body weight [kg]

If you want to know more about the data set:

```
library(ggplot2)
str(msleep)
?msleep
```

# ggplot(data)

The `ggplot()` function initializes a ggplot object. Every ggplot needs this function.

```
library(ggplot) # or library(tidyverse)  
ggplot(data = msleep)
```

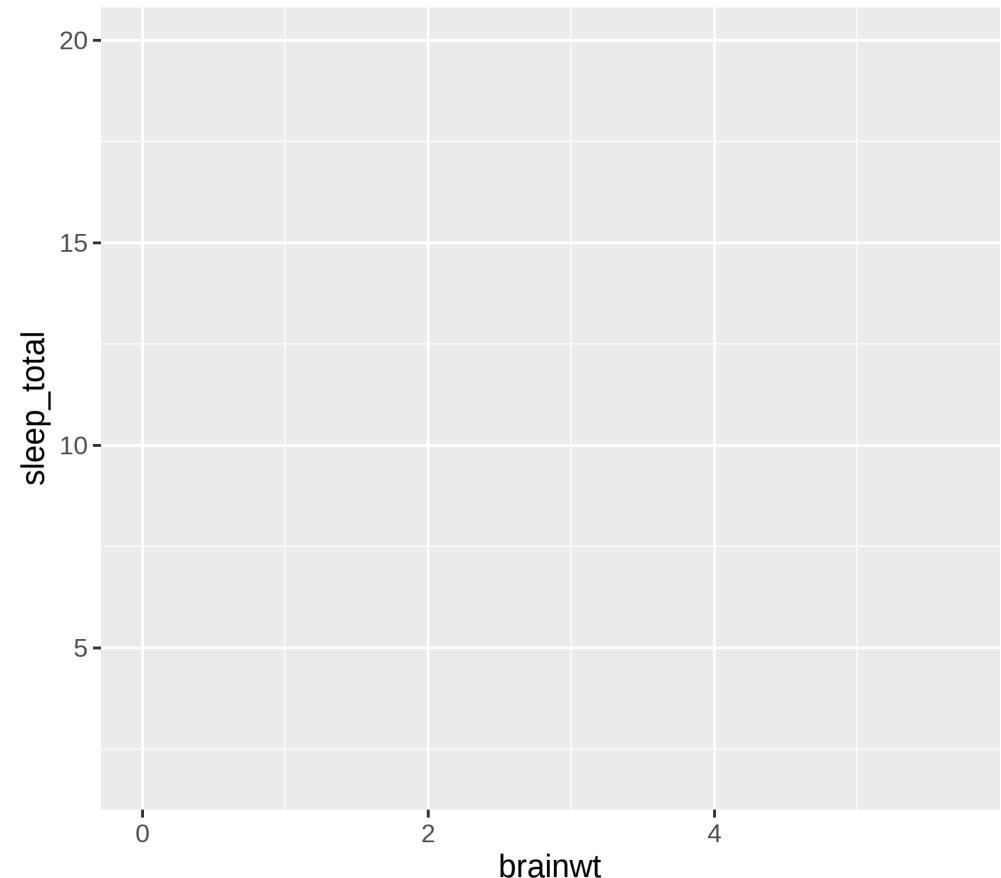
- Empty plot because we did not specify the mapping of data variables

## aes(x, y)

The `aesthetic` mapping defines how variables are mapped to aesthetic properties of the plot.

```
ggplot(data = msleep,  
       mapping = aes(  
           x = brainwt,  
           y = sleep_total))
```

- map variable `brainwt` to x-axis and `sleep_total` to y-axis
- default scales are automatically adapted to range of data



## aes(x, y)

The aesthetic mapping defines how variables are mapped to aesthetic properties of the plot.

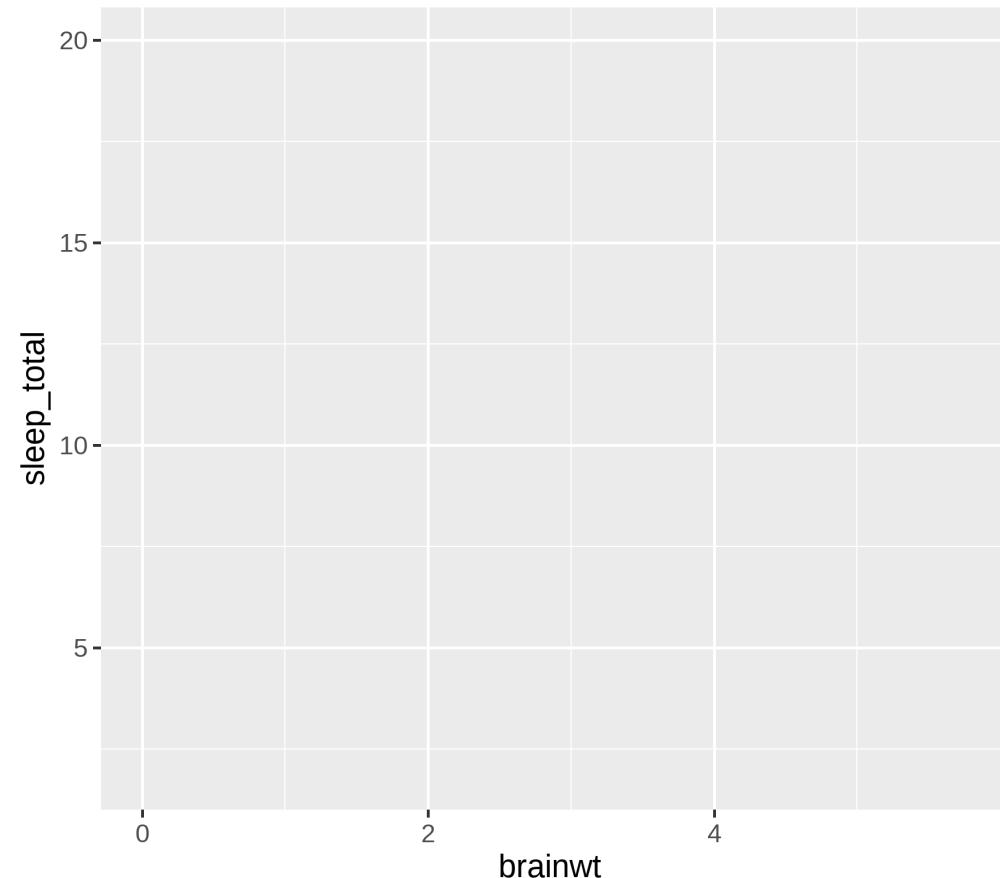
```
ggplot(data = msleep,  
       mapping = aes(  
           x = brainwt,  
           y = sleep_total))
```

- map variable `brainwt` to x-axis and `sleep_total` to y-axis
- default scales are automatically adapted to range of data

This is the same but shorter:

```
ggplot(msleep, aes(brainwt, sleep_total))
```

Remember argument matching by position?



## `geom_*`()

`geoms` defines how to represent the data points. There are many different geoms to chose from

 **a + geom\_blank()**  
(Useful for expanding limits)

 **b + geom\_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1)** - x, yend, alpha, angle, color, curvature, linetype, size

 **a + geom\_path(lineend = "butt", linejoin = "round", linemitre = 1)**  
x, y, alpha, color, group, linetype, size

 **a + geom\_polygon(aes(group = group))**  
x, y, alpha, color, fill, group, linetype, size

 **b + geom\_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1))** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

 **a + geom\_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900))** - x, ymax, ymin, alpha, color, fill, group, linetype, size

## LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

 **b + geom\_abline(aes(intercept = 0, slope = 1))**  
 **b + geom\_hline(aes(yintercept = lat))**  
 **b + geom\_vline(aes(xintercept = long))**

**b + geom\_segment(aes(yend = lat + 1, xend = long + 1))**  
**b + geom\_spoke(aes(angle = 1:1155, radius = 1))**

## ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

 **c + geom\_area(stat = "bin")**  
x, y, alpha, color, fill, linetype, size

 **c + geom\_density(kernel = "gaussian")**  
x, y, alpha, color, fill, group, linetype, size, weight

 **c + geom\_dotplot()**  
x, y, alpha, color, fill

 **c + geom\_freqpoly()** x, y, alpha, color, group, linetype, size

 **c + geom\_histogram(binwidth = 5)** x, y, alpha, color, fill, linetype, size, weight

c + geom\_sf(sf::st\_as\_sf(USArrests) %>% st\_as\_sf)

 **e + geom\_label(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

 **e + geom\_jitter(height = 2, width = 2)**  
x, y, alpha, color, fill, shape, size

 **e + geom\_point()**, x, y, alpha, color, fill, shape, size, stroke

 **e + geom\_quantile()**, x, y, alpha, color, group, linetype, size, weight

 **e + geom\_rug(sides = "bl")**, x, y, alpha, color, linetype, size

 **e + geom\_smooth(method = lm)**, x, y, alpha, color, fill, group, linetype, size, weight

 **e + geom\_text(aes(label = cty), nudge\_x = 1, nudge\_y = 1, check\_overlap = TRUE)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

## discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

 **f + geom\_col()**, x, y, alpha, color, fill, group, linetype, size

 **f + geom\_boxplot()**, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

 **f + geom\_dotplot(binaxis = "y", stackdir = "center")**, x, y, alpha, color, fill, group

 **f + geom\_violin(scale = "area")**, x, y, alpha, color, fill, group, linetype, size, weight

## discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

 **g + geom\_count()**, x, y, alpha, color, fill, shape, size, stroke

 **h + geom\_bin2d(binwidth = c(0.25, 500))**  
x, y, alpha, color, fill, linetype, size, weight

 **h + geom\_density2d()**  
x, y, alpha, colour, group, linetype, size

 **h + geom\_hex()**  
x, y, alpha, colour, fill, size

## continuous function

i <- ggplot(economics, aes(date, unemploy))

 **i + geom\_area()**  
x, y, alpha, color, fill, linetype, size

 **i + geom\_line()**  
x, y, alpha, color, group, linetype, size

 **i + geom\_step(direction = "hv")**  
x, y, alpha, color, group, linetype, size

## visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)  
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

 **j + geom\_crossbar(fatten = 2)**  
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

 **j + geom\_errorbar()**, x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom\_errorbarh()**)

 **j + geom\_linerange()**  
x, ymin, ymax, alpha, color, group, linetype, size

 **j + geom\_pointrange()**  
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

## maps

data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))  
map <- map\_data("state")  
k <- ggplot(data, aes(fill = murder))

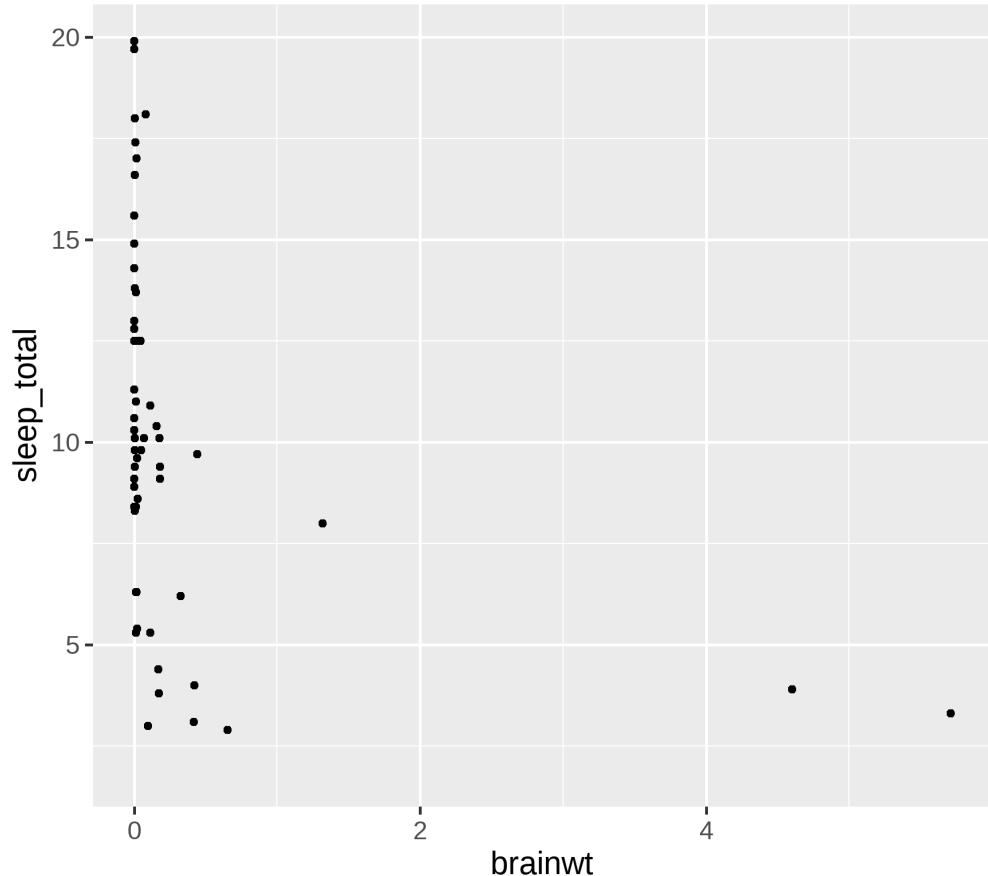
 **k + geom\_map(aes(map\_id = state), map = map) + expand\_limits(x = map\$long, y = map\$lat)**, map\_id, alpha, color, fill, linetype, size

# geom\_point()

```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point()
```

```
## Warning: Removed 27 rows containing missing  
values (geom_point).
```

- new plot layers are added with `+`
- warning that 27 points can not be plotted due to missing values
- `data` and `aes` defined in `ggplot` call are inherited to all plot layers



# geom\_point()

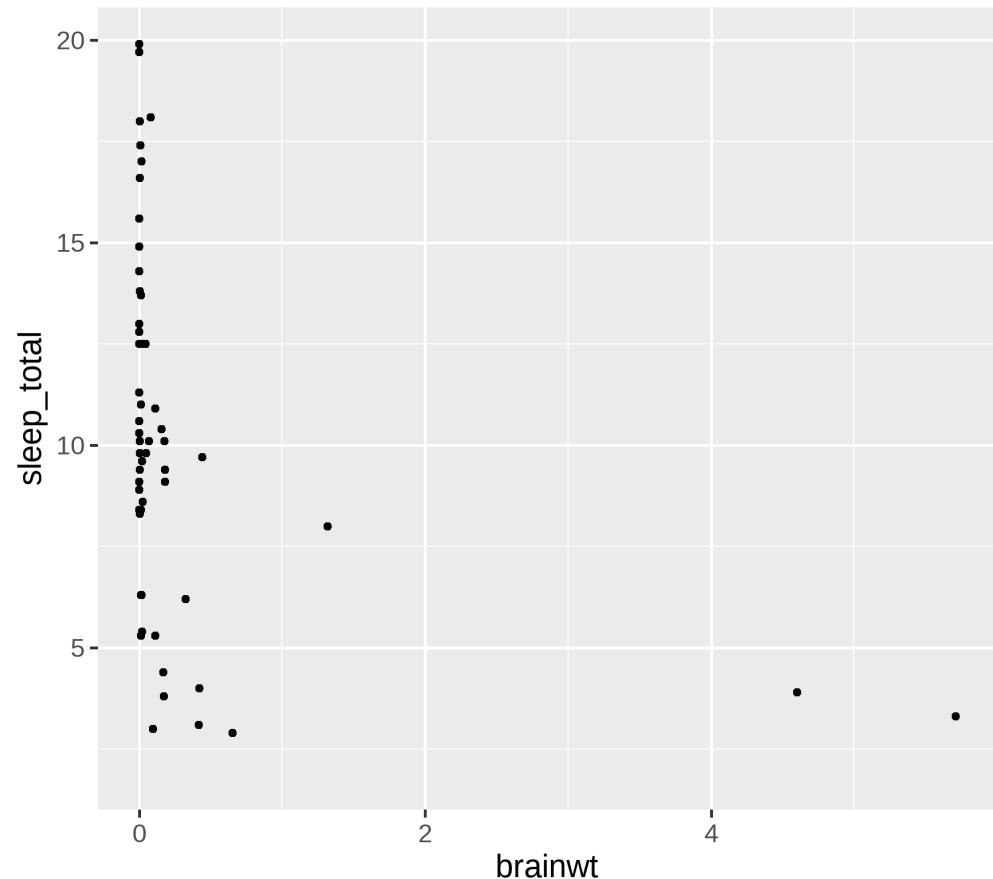
```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point()
```

- new plot layers are added with `+`
- warning that 27 points can not be plotted due to missing values
- `data` and `aes` defined in `ggplot` call are inherited to all plot layers
- `data` and `aes` can be local to a layer:

```
ggplot(msleep) +  
  geom_point(aes(brainwt, sleep_total))
```

Here, it does not make a difference.

```
## Warning: Removed 27 rows containing missing  
values (geom_point).
```

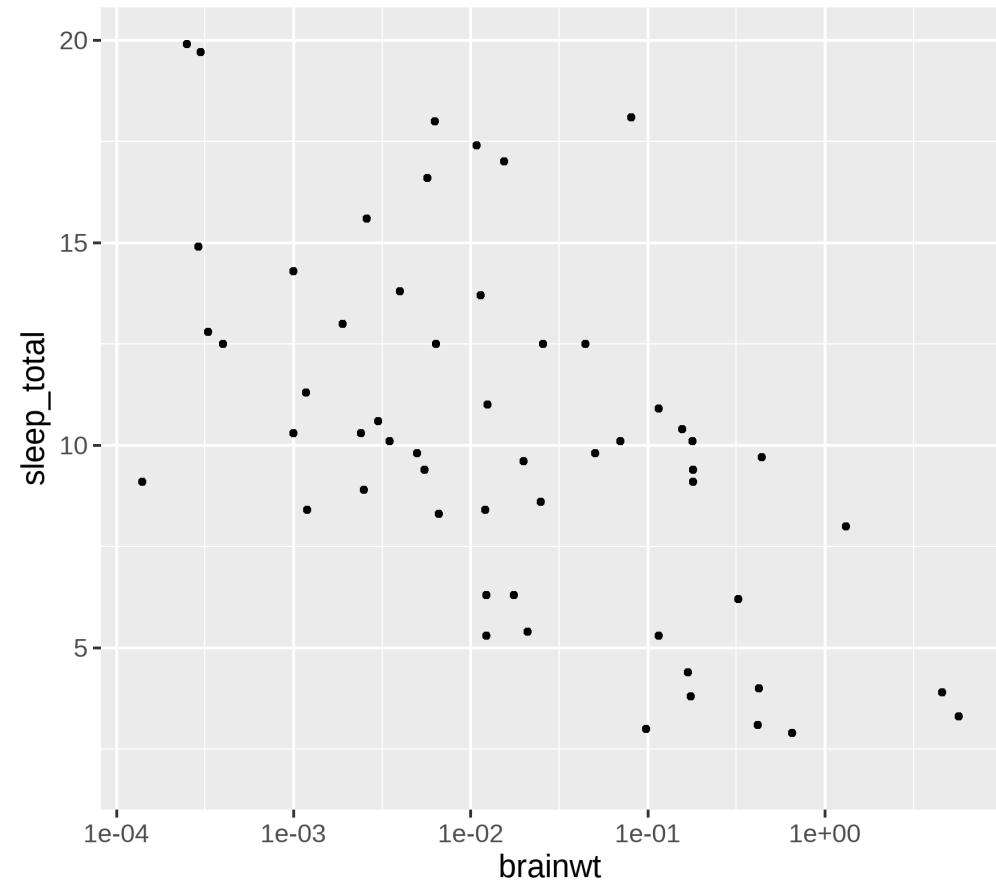


## scale\_x\_log10()

The scales onto which the aesthetic elements are mapped can be changed.

```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point() +  
  scale_x_log10()
```

- scales can be changed for all elements of `aes`



# scale\_x\_log10()

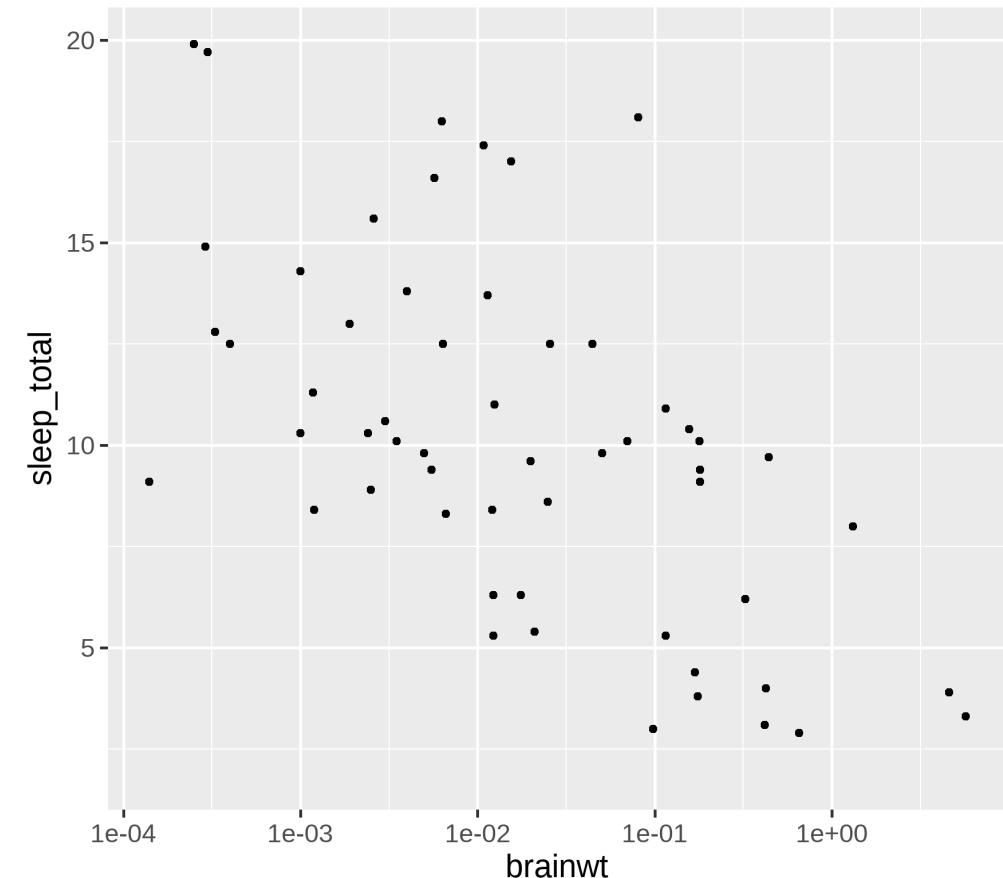
The scales onto which the aesthetic elements are mapped can be changed.

```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point() +  
  scale_x_log10()
```

- scales can be changed for all elements of `aes`
- the general format of scale functions are:

`scale_aes-name_scale-type`

In this example we scale the `x` aesthetic to `log10`.

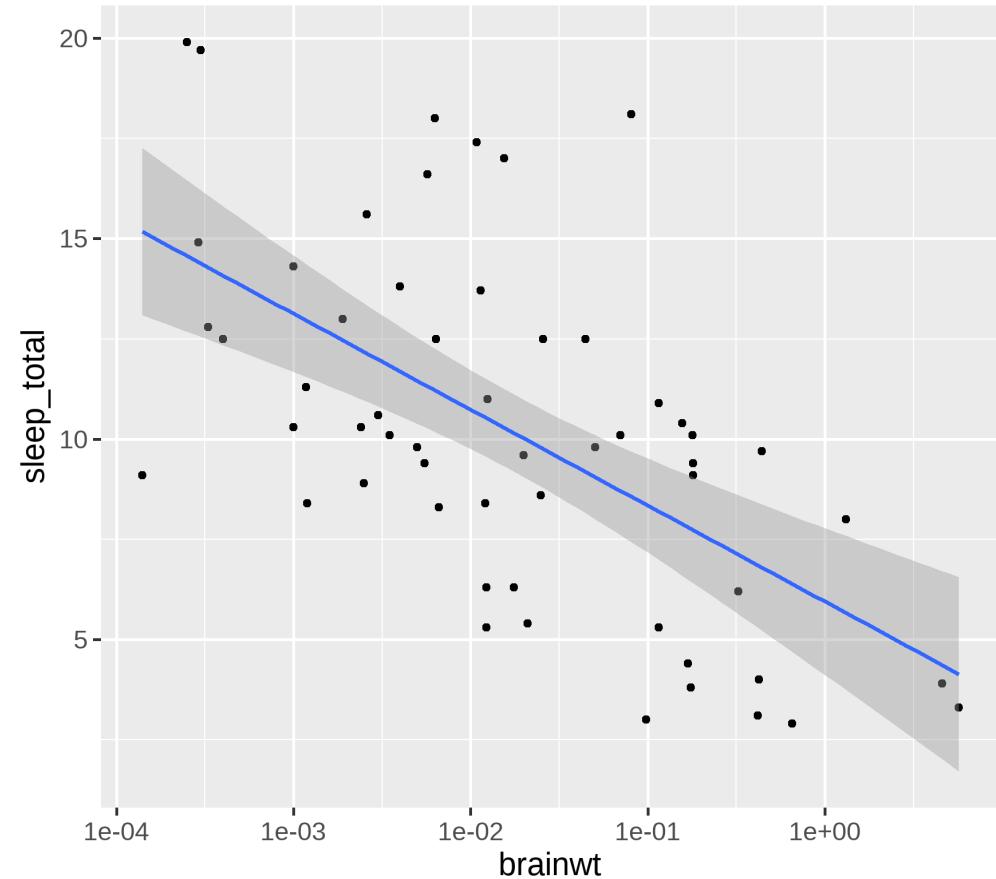


# geom\_smooth()

We can add a smoothing line that helps us see patterns in the data

```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  scale_x_log10()
```

- with `method = lm`, a linear regression line is added
- the ribbon represents confidence interval around the regression
  - turn ribbon off with `se = FALSE` argument

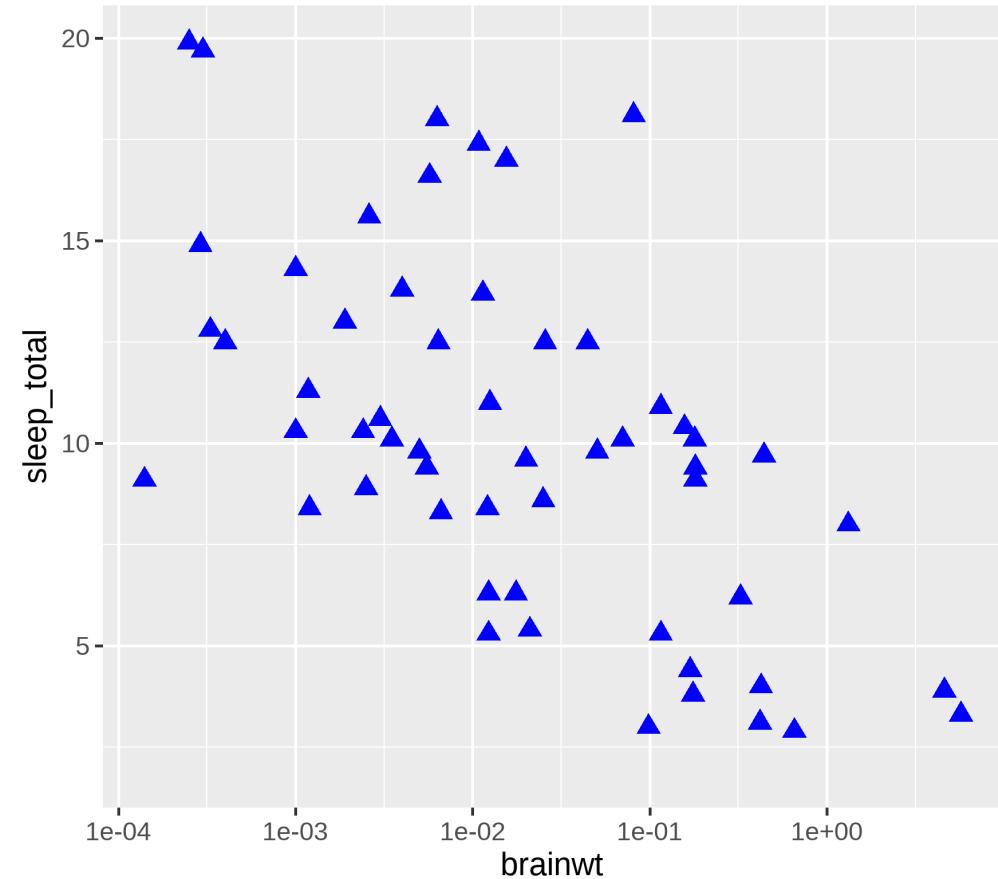


# Change appearance of points

The basic things we can change about points are their **size**, **shape** and **color**:

```
ggplot(data = msleep,  
       aes(x = brainwt,  
            y = sleep_total)) +  
  geom_point(size = 4,  
             shape = 17,  
             color = "blue") +  
  scale_x_log10()
```

- have a look [here](#) for the point shape codes

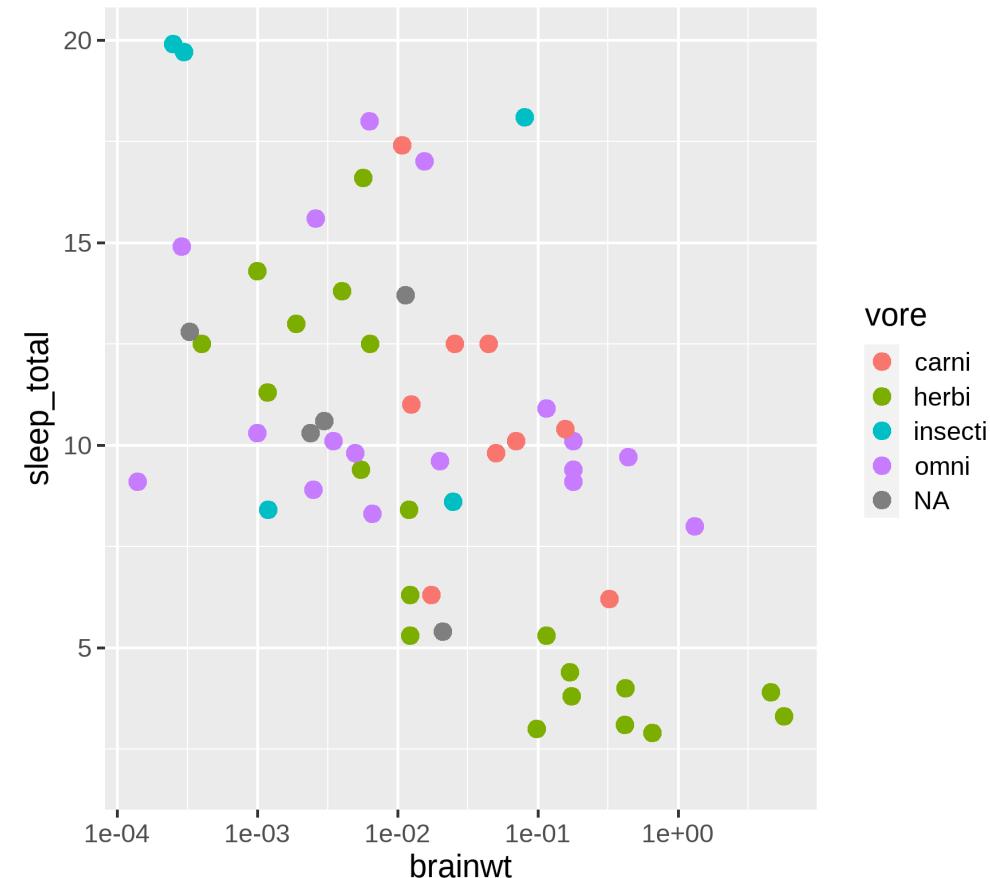


# aes (color): mapping color to a variable

Instead of changing color of all points, **map color to a variable** by adding it to aesthetics:

```
g <- ggplot(  
  data = msleep,  
  aes(  
    x = brainwt,  
    y = sleep_total,  
    color = vore  
)  
) +  
  geom_point(size = 4) +  
  scale_x_log10()  
g
```

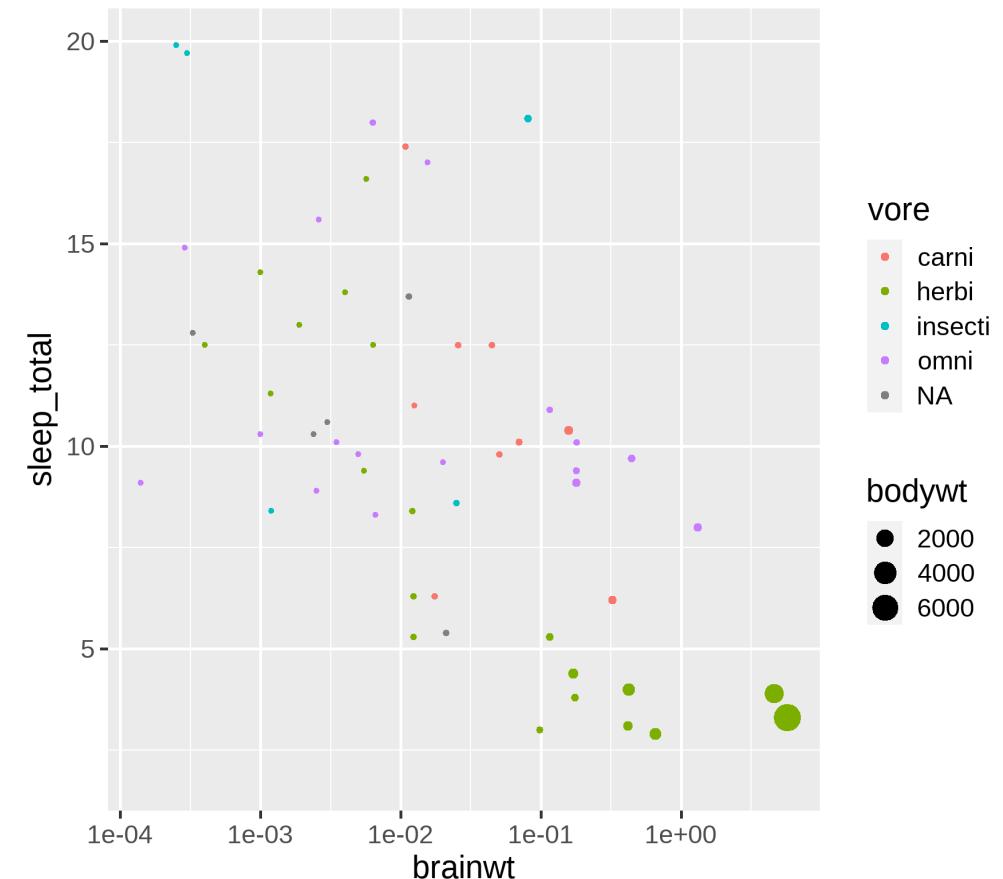
- map the `vore` variable to the color aesthetic of the plot
- plots can be saved in variables



# aes (size): mapping size to a variable

We can do the same with size:

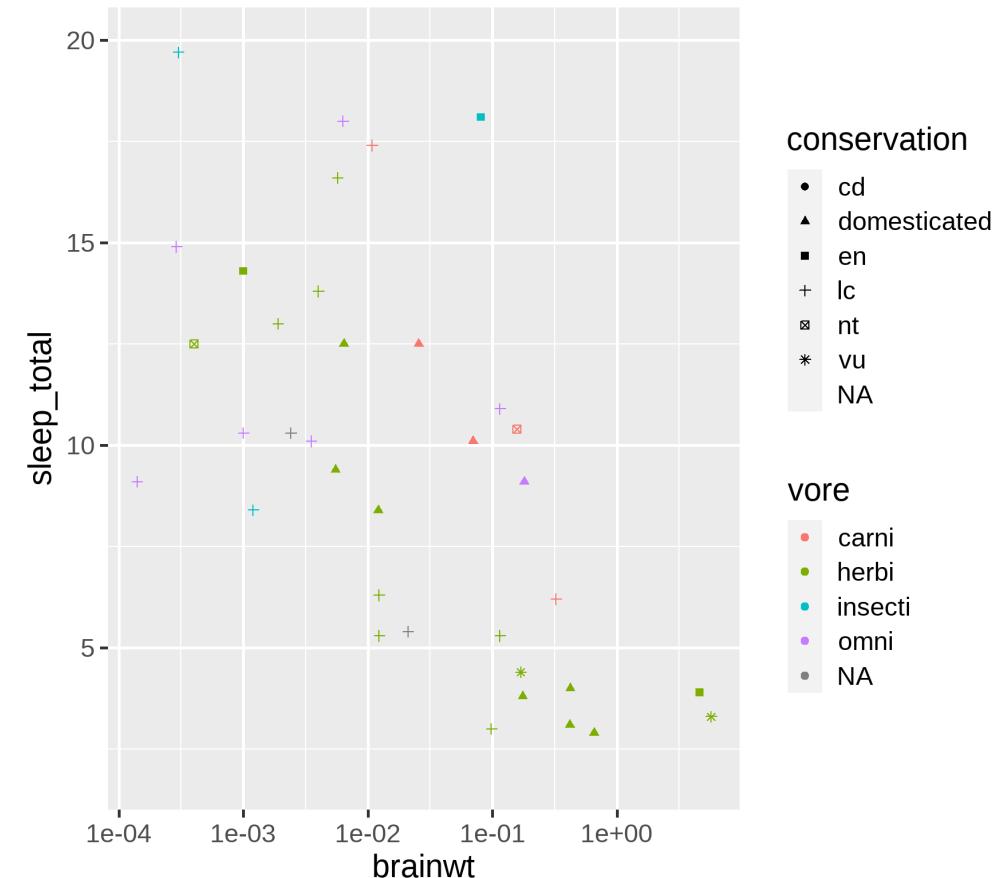
```
ggplot(  
  data = msleep,  
  aes(  
    x = brainwt,  
    y = sleep_total,  
    color = vore,  
    size = bodywt  
  )  
) +  
  geom_point() +  
  scale_x_log10()
```



# aes (shape) : mapping shape to a variable

We can do the same with shape:

```
ggplot(  
  data = msleep,  
  aes(  
    x = brainwt,  
    y = sleep_total,  
    color = vore,  
    shape = conservation  
  )  
) +  
  geom_point() +  
  scale_x_log10()
```

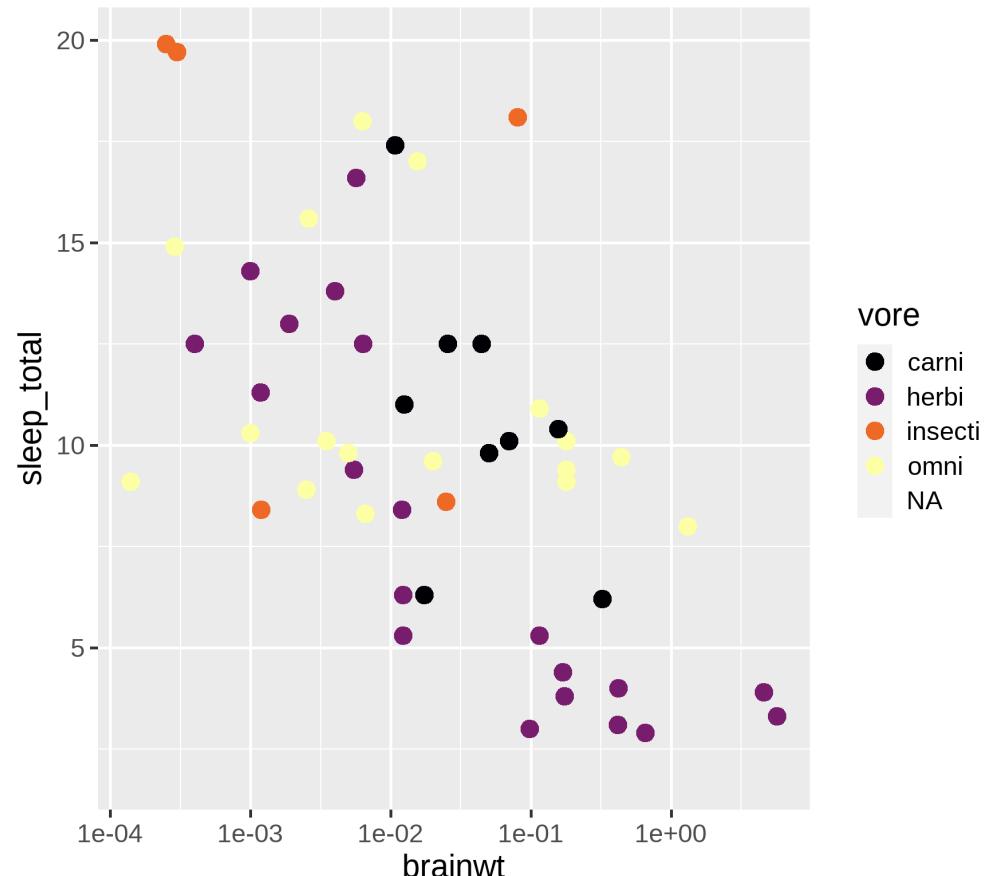


## `scale_color_viridis_d()`

Back to our plot `g` with color as additional aesthetic. We can change the color scale:

```
g +  
  scale_color_viridis_d(option = "inferno")
```

- the viridis color palette is designed for viewers with common forms of color blindness
- different options of viridis color palettes are:
  - "magma", "inferno", "plasma", "viridis", "cividis"



## scale\_color\_viridis\_d()

The `scale_color_viridis_d` function by default does not map color to missing data points (`NA` values). We have to explicitly state the color for `NA`:

```
g +
  scale_color_viridis_d(
    option = "inferno",
    na.value = "grey")
```

- whether or not you have to do this, depends on the default of the `na.value` argument of the color scale that you use
- you can also remove missing values from your data set before plotting (we will learn that later)

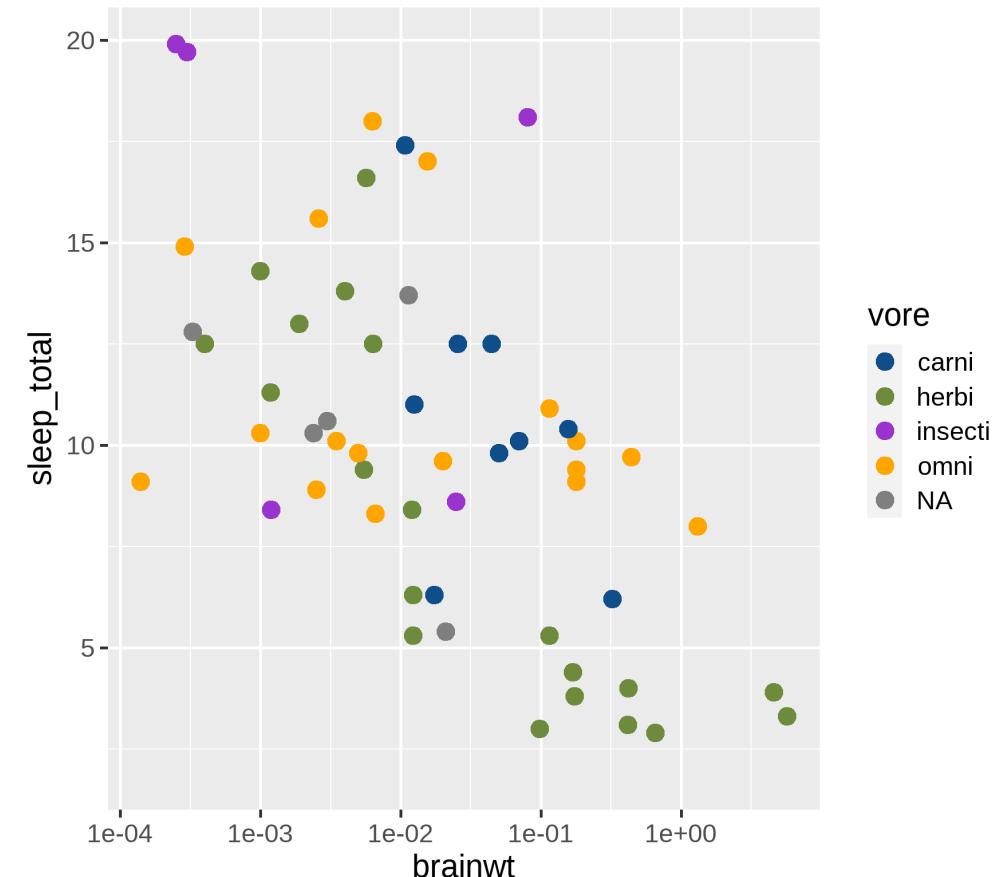
# scale\_color\_manual()

We can also manually specify colors:

```
g <- g +  
  scale_color_manual(  
    values = c("dodgerblue4",  
              "darkolivegreen4",  
              "darkorchid3",  
              "orange",  
              "grey"))
```

g

- length of color vector has to match number of levels in your aesthetic
- specify colors
  - via their name (see [here](#) for all color names)
  - via their Hex color codes



# Other color scales

There are many packages with preset color palettes, e.g.:

**ggsci**: scientific journal and sci-fi themed palettes

```
install.packages("ggsci")
# Examples
ggsci::scale_color_npg() # nature publishing group
ggsci::scale_color_rickandmorty()
```

**ggthemes**: software and publisher themed palettes:

```
install.packages("ggthemes")
# Examples
ggthemes::scale_color_excel_new() # colors from new Excel version
ggthemes::scale_color_economist() # color palette from Economist graphs
```

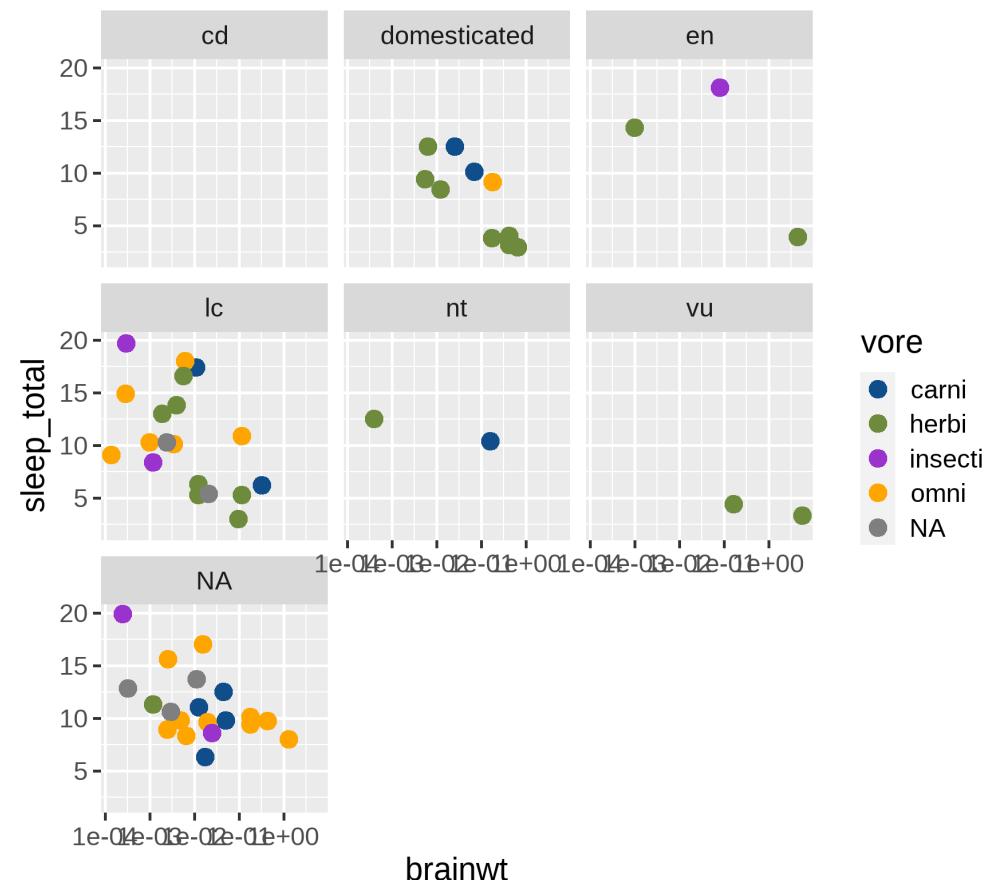
For a comprehensive list of color palettes available, check out [this repository](#) by Emil Hvitfeldt.

# `facet_wrap()` and `facet_grid()`

Facets split a plot into small multiples along values of a variable from the data.

```
g +  
  facet_wrap(~conservation)
```

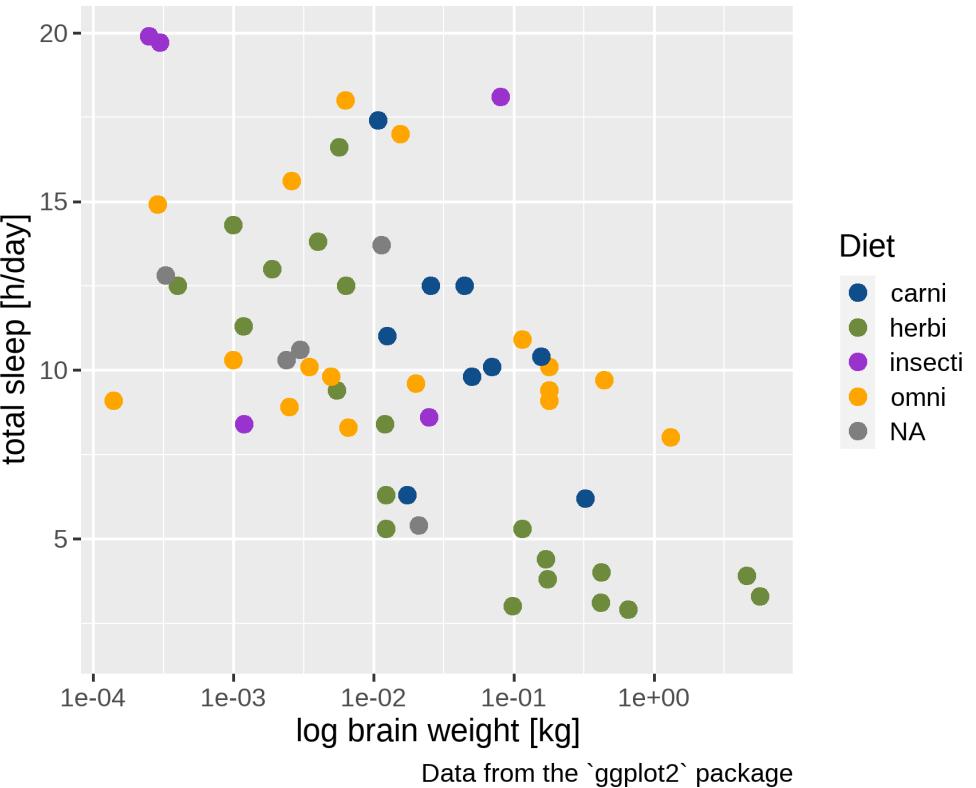
- you can even split plots along two variables using `facet_grid()` instead of `facet_wrap()`
- The basic functioning is `facet_grid(yvar ~ xvar)`
  - `yvar` will be displayed as columns
  - `xvar` will be displayed as rows



# labs(): change axis and legend titles and add plot title

```
g <- g +  
  labs (  
    x = "log brain weight [kg]",  
    y = "total sleep [h/day]",  
    color = "Diet",  
    title = "Brain weight and sleep time in  
mammals",  
    subtitle = "Larger brains seem to sleep  
less",  
    caption = "Data from the ggplot2 package")  
g
```

Brain weight and sleep time in mammals  
Larger brains seem to sleep less



# theme\_\*(): change appearance

ggplot2 offers many pre-defined themes that we can apply to change the appearance of a plot.

```
g +  
  theme_classic()
```

```
g +  
  theme_linedraw()
```

# theme\_\*(): change appearance

ggplot2 offers several pre-defined themes that we can apply to change the appearance of our plot.

```
g +
  theme_dark()
```

```
g +
  theme_minimal()
```

# theme(): customize theme

You can manually change a theme or even create an entire theme yourself. The elements you can control in the theme are:

- titles (plot, axis, legend, ...)
- labels
- background
- borders
- grid lines
- legends

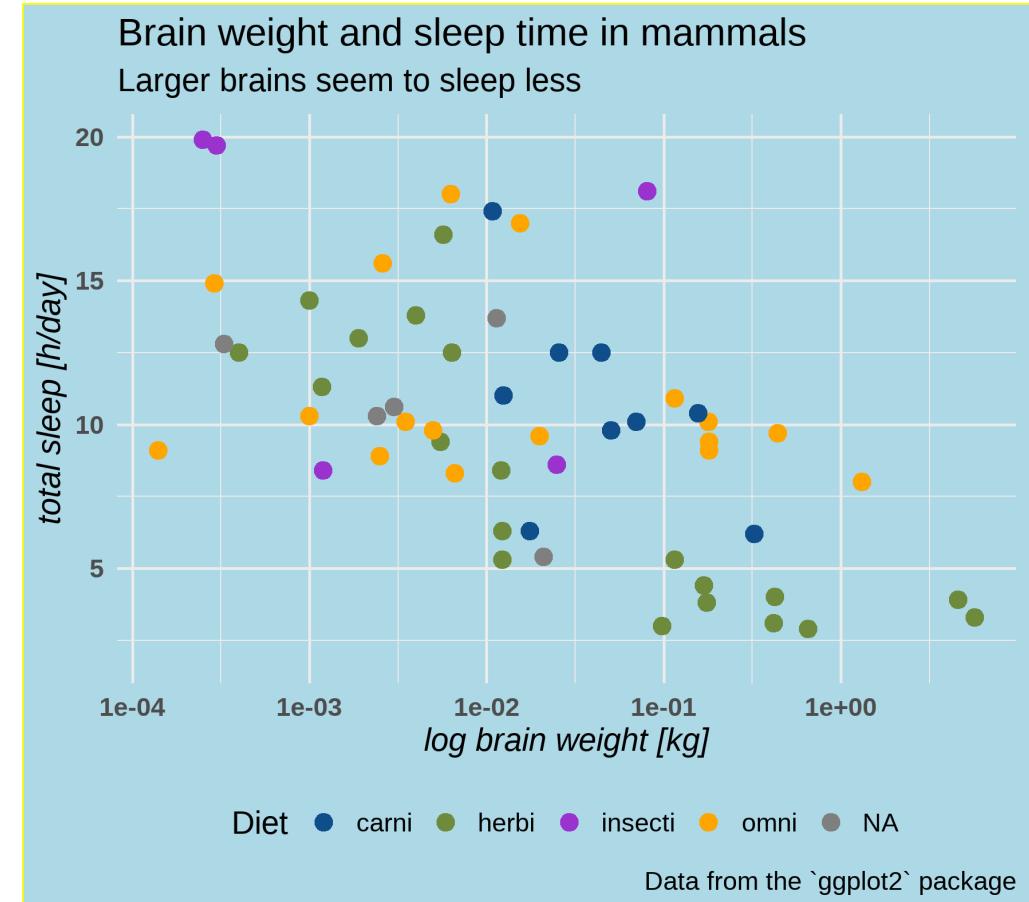
If you want a full list of what you can customize, have a look at

```
?theme
```

# theme(): customize theme

To edit a theme, just add another `theme()` layer to your plot.

```
g +
  theme_minimal() +
  theme(
    axis.text = element_text(face = "bold"),
    axis.title = element_text(face = "italic"),
    legend.position = "bottom",
    plot.background = element_rect(
      fill = "lightblue",
      color = "yellow")
  )
```



# theme\_set(): set global theme

You can set a global theme that will be applied to all ggplot objects in the current R session.

```
# Globally set theme_minimal as the default theme  
theme_set(theme_minimal())
```

Add this to the beginning of your script.

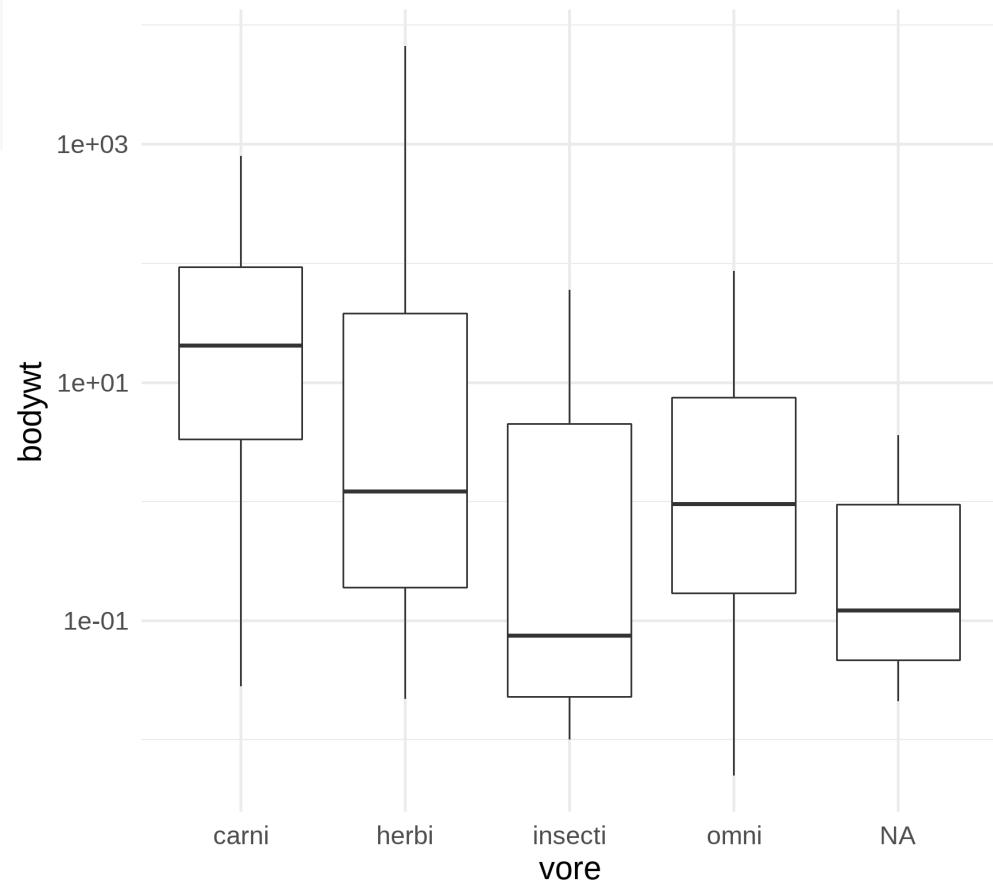
You can also specify some defaults, e.g. the text size:

```
theme_set(theme_minimal(base_size = 16))
```

# geom\_boxplot()

A boxplot can be created with `geom_boxplot`:

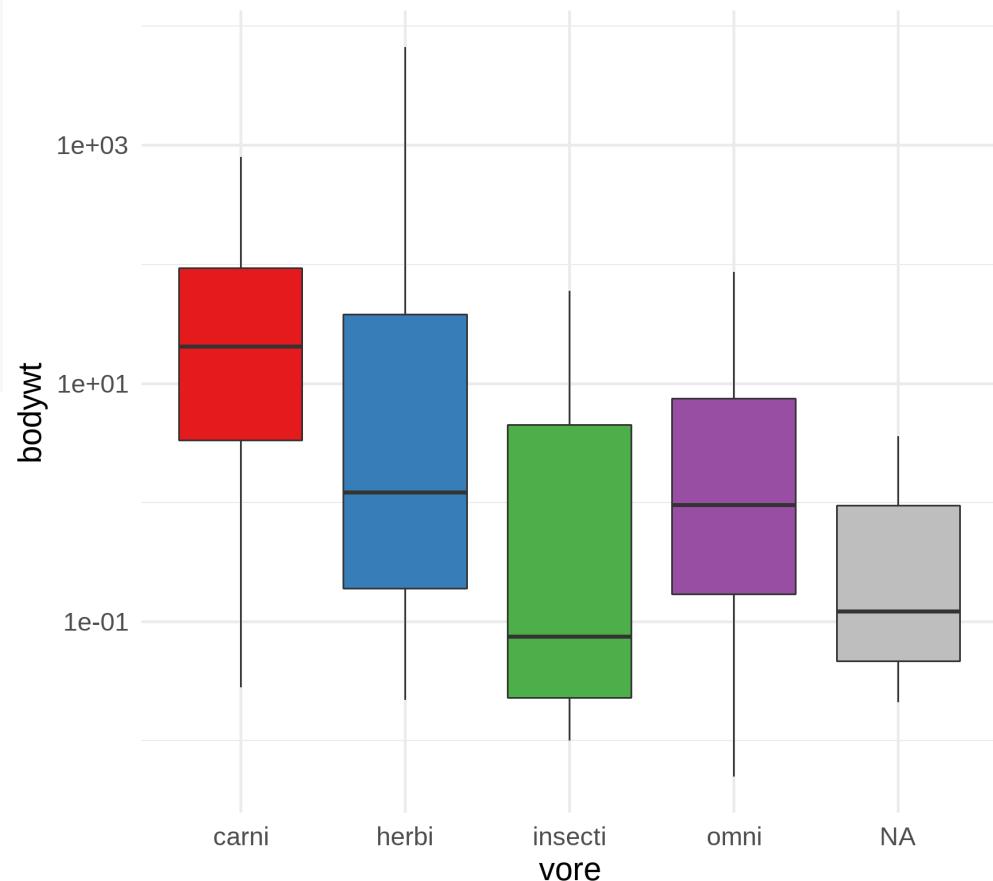
```
ggplot(msleep, aes(x = vore, y = bodywt)) +  
  geom_boxplot() +  
  scale_y_log10()
```



# aes(fill) and scale\_fill\_brewer()

Pay attention to fill vs. color aesthetic

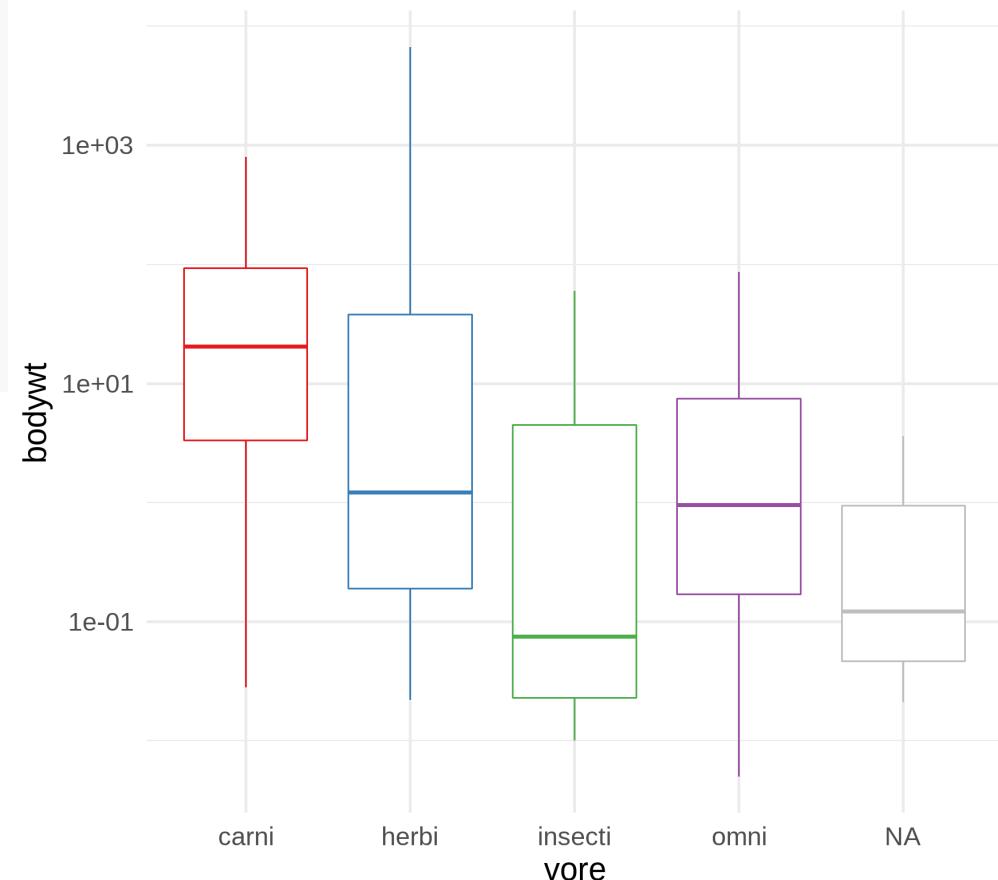
```
ggplot(msleep, aes(x = vore, y = bodywt)) +  
  geom_boxplot(  
    aes(fill = vore)  
  ) +  
  scale_y_log10() +  
  scale_fill_brewer(  
    palette = "Set1",  
    na.value = "gray",  
    guide = "none")
```



# aes(color) and scale\_color\_brewer()

Pay attention to fill vs. color aesthetic

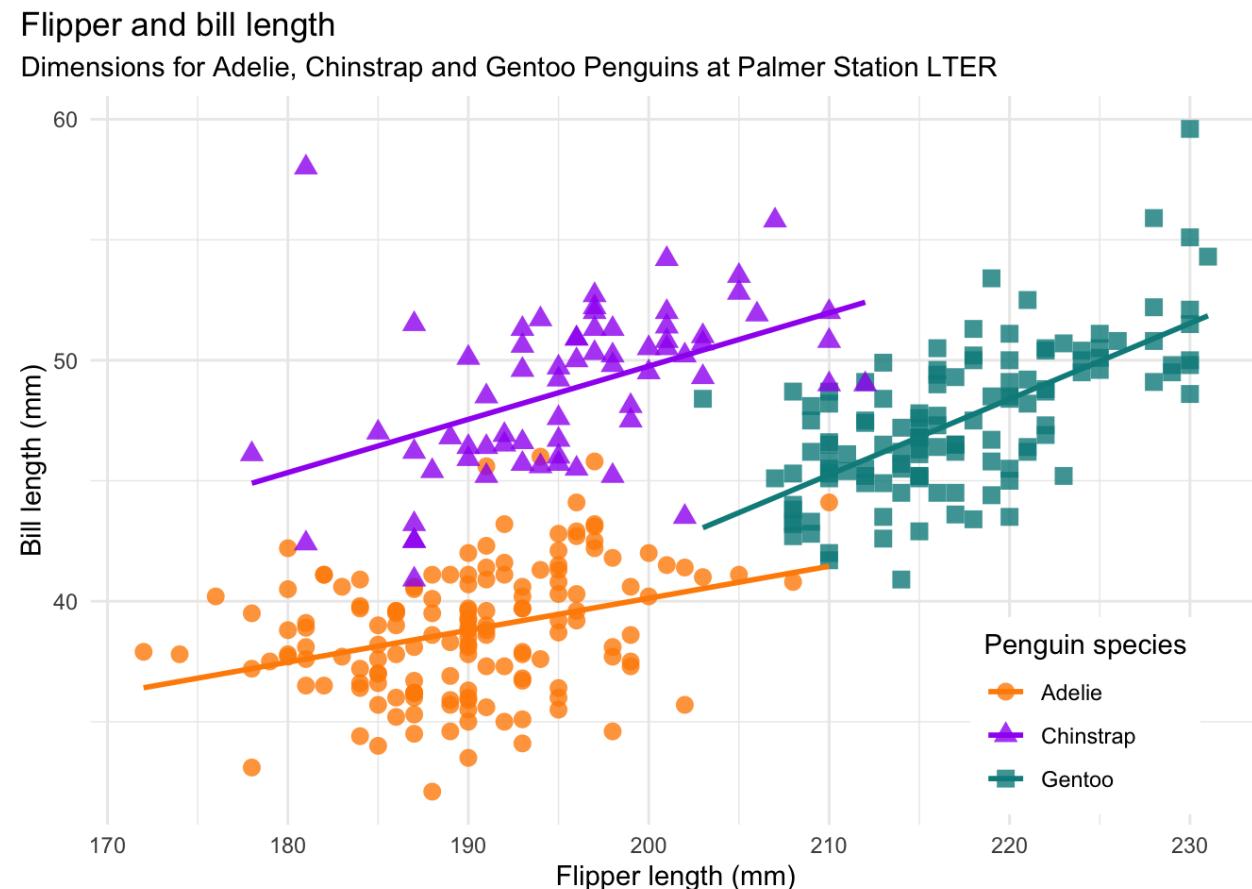
```
ggplot(msleep, aes(x = vore, y = bodywt)) +  
  geom_boxplot(  
    aes(color = vore)  
  ) +  
  scale_y_log10() +  
  scale_color_brewer(  
    palette = "Set1",  
    na.value = "gray",  
    guide = "none")
```



# Let's look at an example

What are the layers in this plot?

- data
- aesthetic mapping
- geoms
- themes



# ggsave()

A ggplot object can be saved on disk in different formats.

Without specifications:

```
# save plot g in img as my_plot.pdf  
ggsave(filename = "./img/my_plot.pdf", plot = g)  
# save plot g in img as my_plot.png  
ggsave(filename = "./img/my_plot.png", plot = g)
```

Or with specifications:

```
# save a plot named g in the img directory under the name my_plot.png with width 16 cm and height  
9 cm  
ggsave(filename = "./img/my_plot.png",  
       plot = g,  
       width = 16,  
       height = 9,  
       units = "cm")
```

Have a look at `?ggsave` to see all options.

# Summary I

## Tidyverse and ggplot

- The tidyverse is a collections of R packages for data analysis, including `ggplot2`, `readr`, `tidyverse`, `dplyr`, `tibble`
- All tidyverse packages are designed to work together seamlessly
- basic idea of `ggplot` is to stack distinct layers of graphical elements to create a plot
- Check out the [ggplot cheatsheet](#) for an overview of how to create a ggplot

Now you

Task 2-1: Create your own penguin ggplots

Find the task description [here](#)