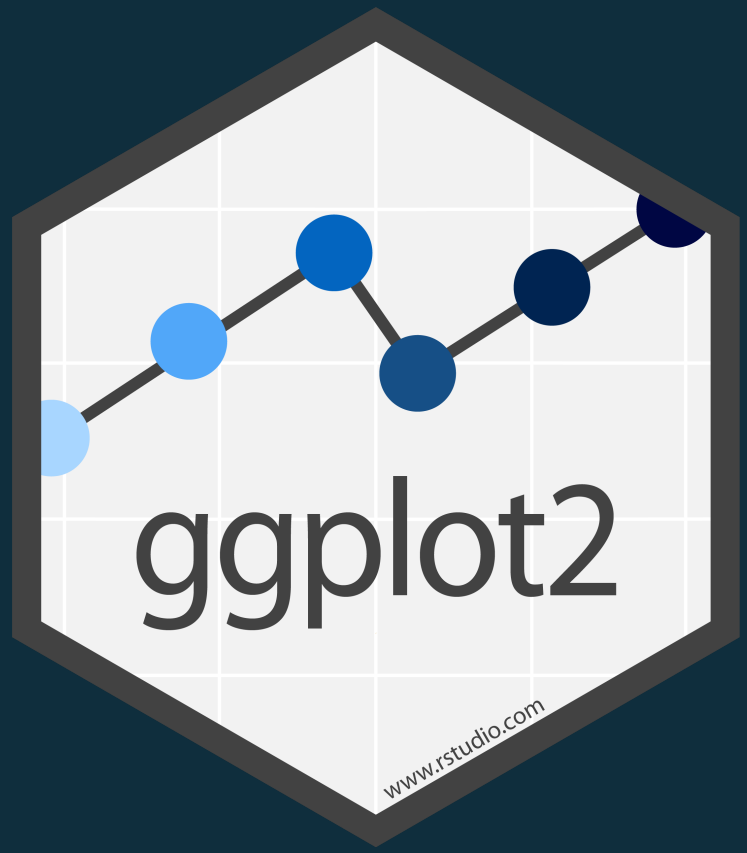
Data visualization with ggplot2

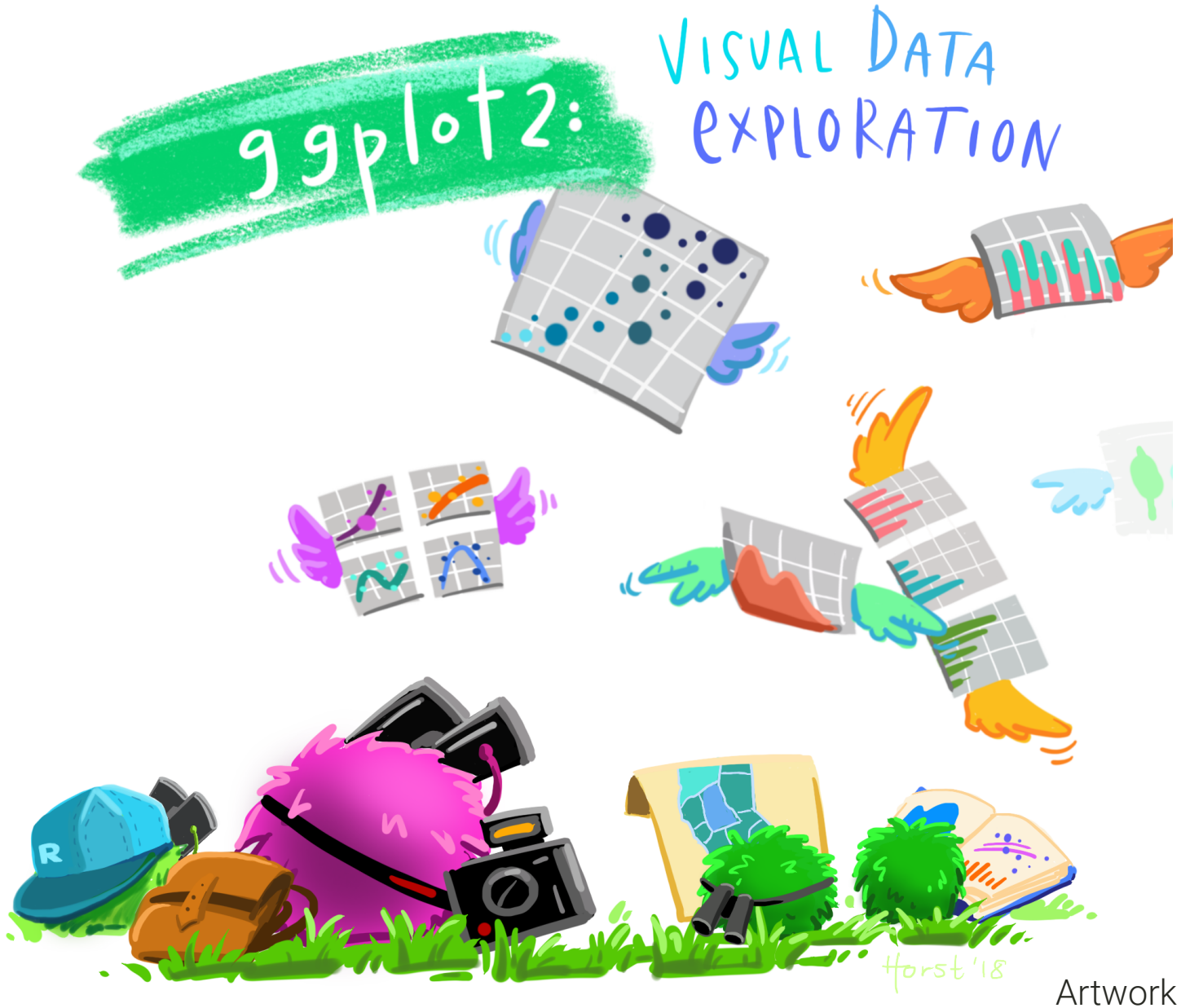
Introduction to R - Day 2

Instructor: Selina Baldauf

Freie Universität Berlin - Theoretical Ecology



2021-08-01 (updated: 2022-08-28)



Artwork by Allison Horst

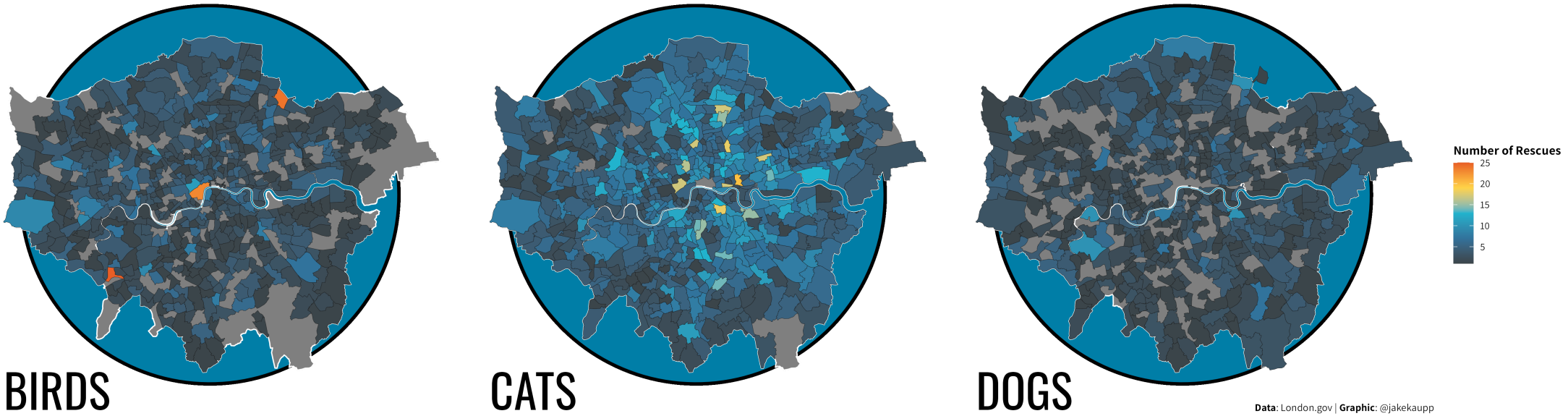
Artwork by Allison Horst

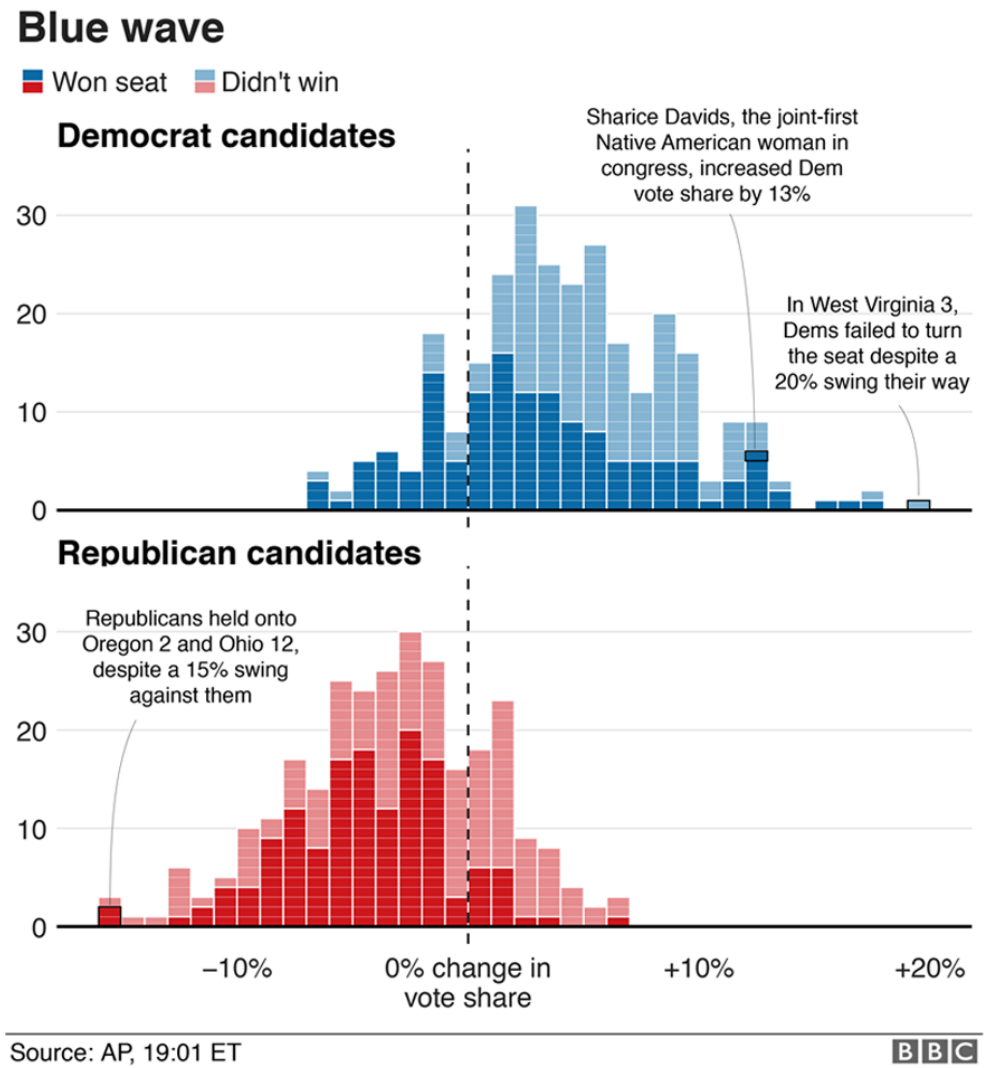
A ggplot showcase

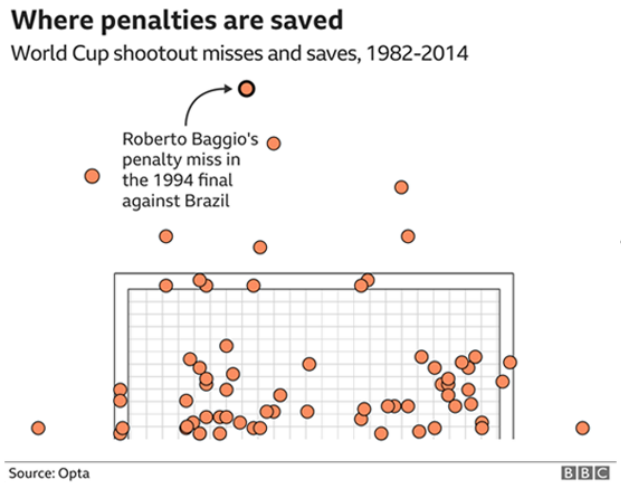
Some examples of plots you can create with ggplot

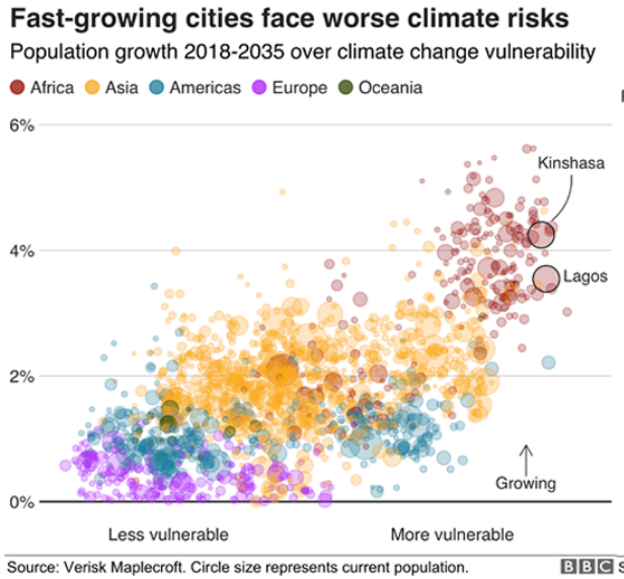
Frequency of Rescues of Birds, Cats and Dogs in London from 2009-2021

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.

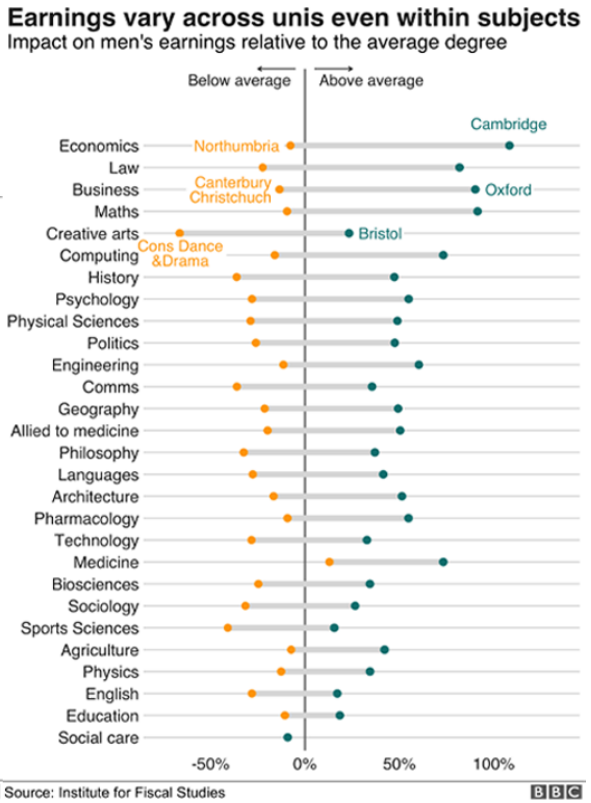



Blue wave
Won seat
Didn't win
Democrat candidates
Sharice Davids, the joint-first Native American woman in congress, increased Dem vote share by 13%
In West Virginia 3, Dems failed to turn the seat despite a 20% swing their way
30
20
10
0
Republican candidates
Republicans held onto Oregon 2 and Ohio 12, despite a 15% swing against them
30
20
10
0
–10%
0% change in vote share
+10%
+20%
Source: AP, 19:01 ET
BBC


Where penalties are saved
World Cup shootout misses and saves, 1982-2014
Roberto Baggio's penalty miss in the 1994 final against Brazil
Source: Opta
BBC

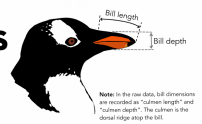

Fast-growing cities face worse climate risks
Population growth 2018-2035 over climate change vulnerability
Africa
Asia
Americas
Europe
Oceania
6%
4%
2%
0%
Kinshasa
Lagos
Growing
Less vulnerable
More vulnerable
Source: Verisk Maplecroft. Circle size represents current population.
BBC


MPs rejected Theresa May's deal by 230 votes
For
Con (196)
202
Lab 3, Ind 3
Against
Con rebels (118)
Labour (248)
432
SNP 35, Lib Dem 11, DUP 10, Plaid 4, Green 1, Ind 5
Source: Commons Votes Services. Excludes 'tellers', the Speaker and deputies
BBC

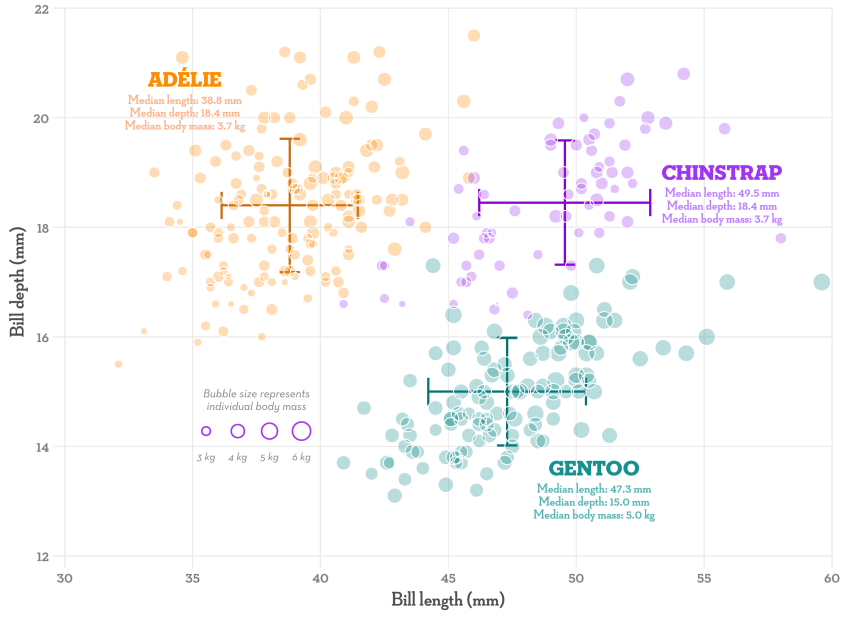

Earnings vary across unis even within subjects
Impact on men's earnings relative to the average degree
Below average
Above average
Economics
Law
Business
Maths
Creative arts
Computing
History
Psychology
Physical Sciences
Politics
Engineering
Comms
Geography
Allied to medicine
Philosophy
Languages
Architecture
Pharmacology
Technology
Medicine
Biosciences
Sociology
Sports Sciences
Agriculture
Physics
English
Education
Social care
Northumbria
Canterbury Christchuch
Cons Dance &Drama
Cambridge
Oxford
Bristol
-50%
0%
50%
100%
Source: Institute for Fiscal Studies
BBC

BILL DIMENSIONS OF BRUSH-TAILED PENGUINS

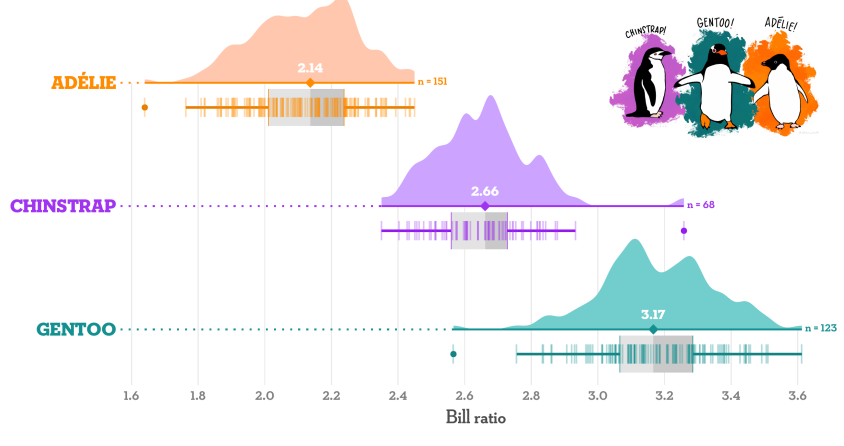
Pygoscelis adéliae (Adélie penguin) • P. antarctica (Chinstrap penguin) • P. papua (Gentoo penguin)



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



Note: In the original data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal (upper) ridge of a bird's bill.
Visualization: Cédric Scherer • Data: Gorman, Williams & Fraser (2014) DOI: 10.1371/journal.pone.0090081 • Illustrations: Allison Horst

Visualization by Cédric Scherer,
code available on Github

Advantages of ggplot

- **Consistent** grammar/structure
- **Flexible** structure allows you to produce any type of plots
- 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

- Basic idea: stack distinct **layers of graphical elements** to create a plot

Layers of a ggplot

Layer	Function	Details
Data	`ggplot(data)`	The data that you want to plot.
Mapping	`aes()`	Aesthetic mappings of the geometric and statistical objects.
Geometries	`geom_*()`	Graphical representation of aesthetics as point, line, polygon, ...
Scales	`scale_*()`	Translate between variable values and properties (e.g. categories -> colour)
Coordinates	`coord_*()`	Interprets and maps the position values of the data.
Facets	`facet_*()`	Split your plot into multiples.
Themes	`theme()` and `theme_*()`	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(ggplot2) # or library(tidyverse)

ggplot(data = msleep)
```

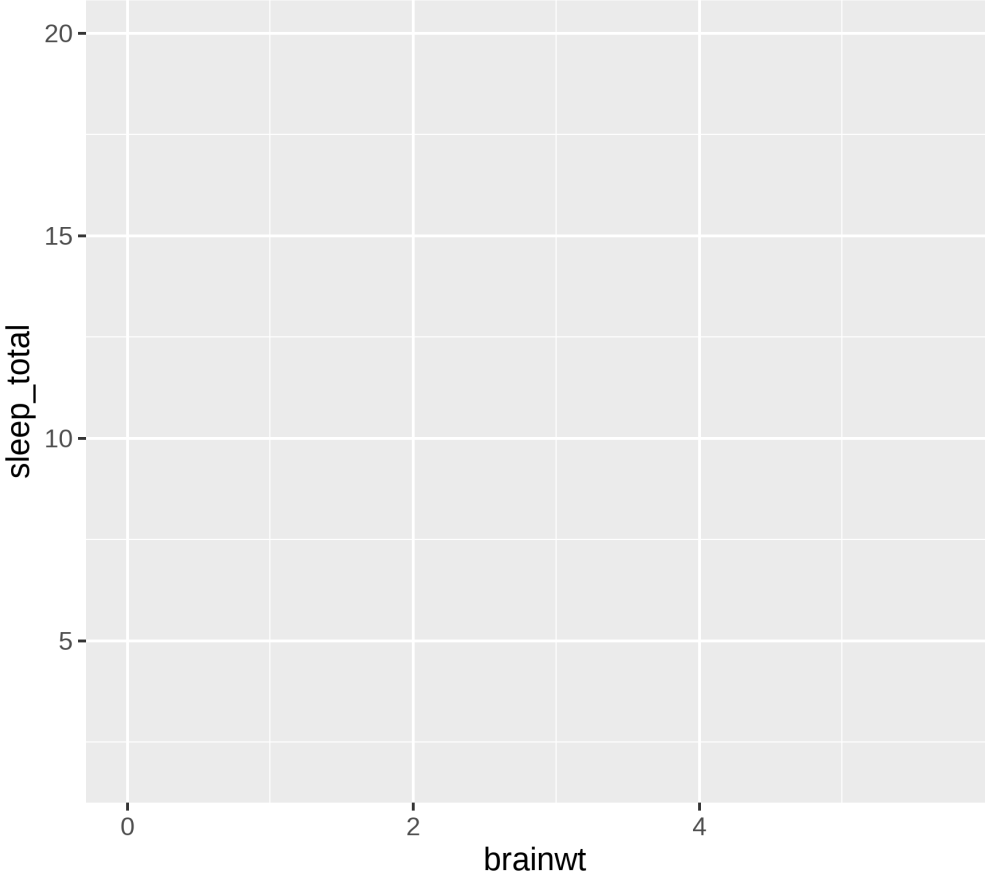
- Empty plot because we did not specify the mapping of data variables

aes(x, y)

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

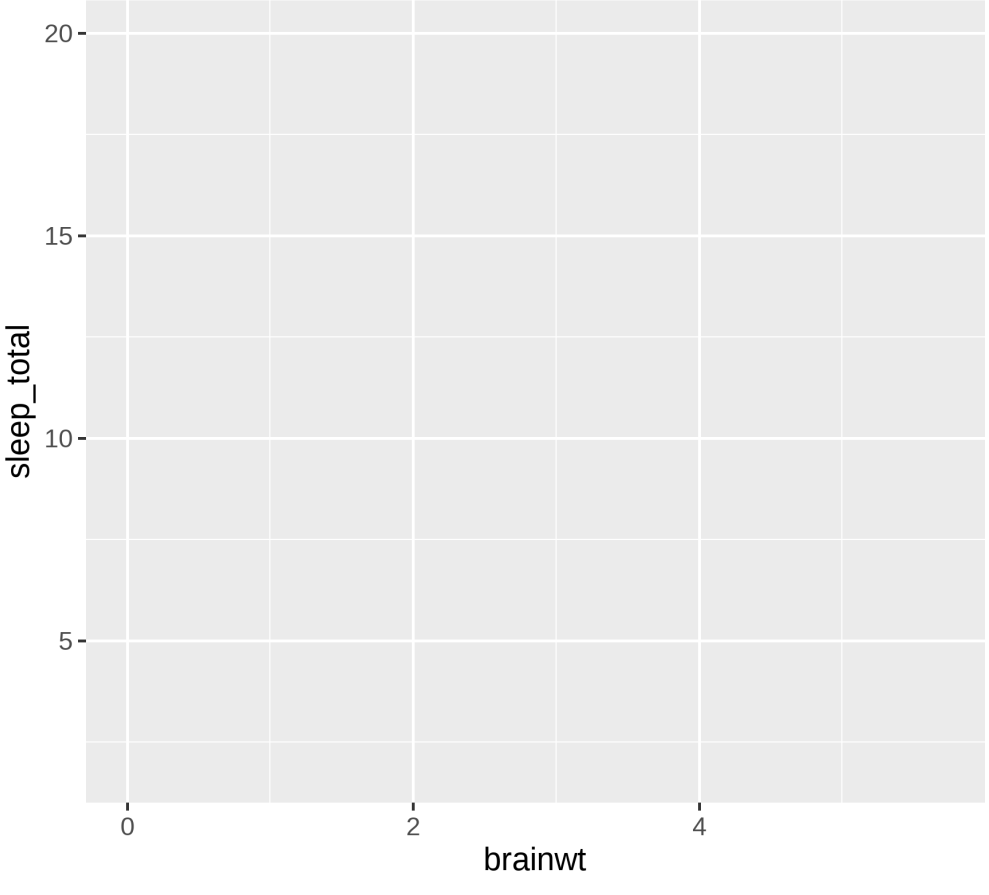
```
ggplot(data = msleep,
       mapping = aes(
         x = brainwt,
         y = sleep_total))
```

- 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` define how data points are represented. 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, xend, y, 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

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))

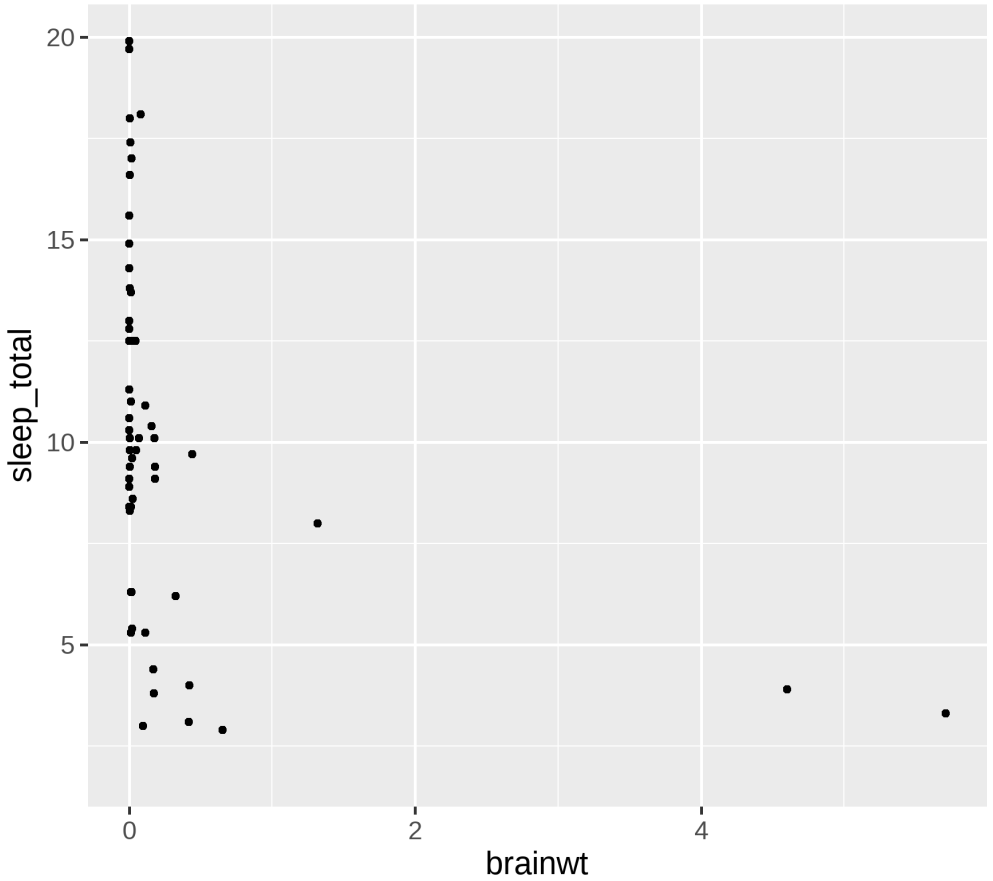
k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

geom_point()

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

- 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

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



geom_point()

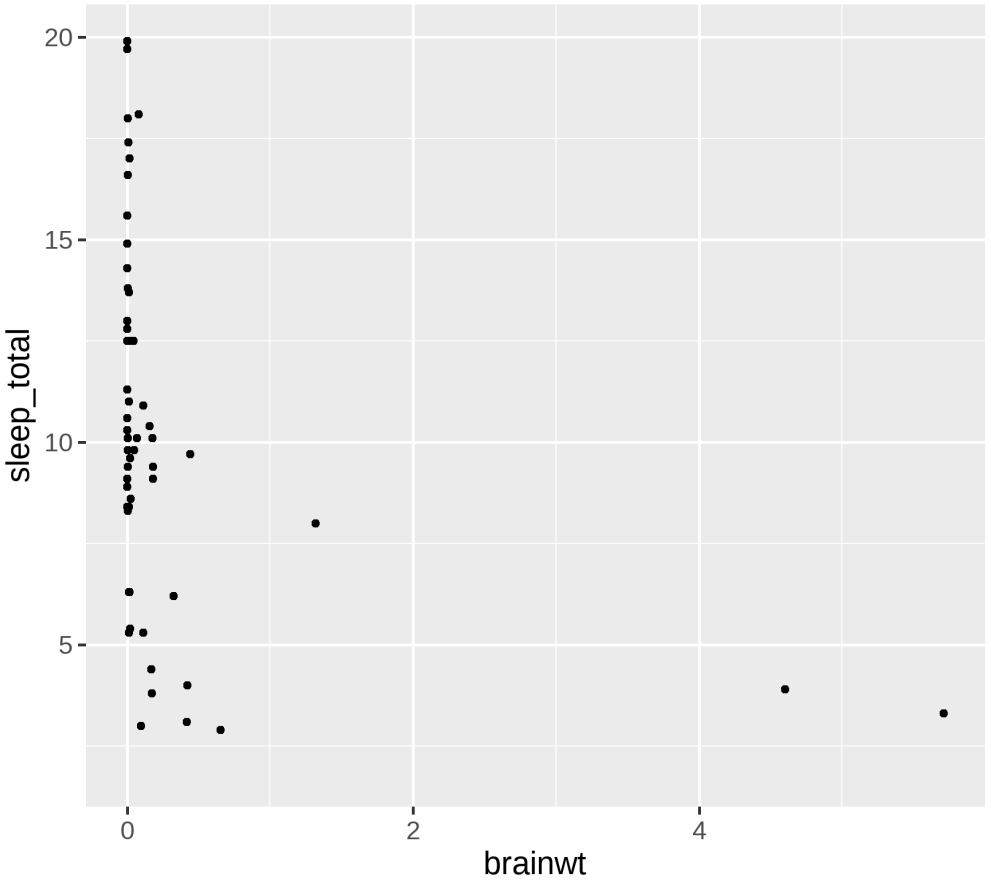
```
ggplot(data = msleep,
       aes(x = brainwt,
           y = sleep_total)) +
  geom_point()
```

- 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).
```

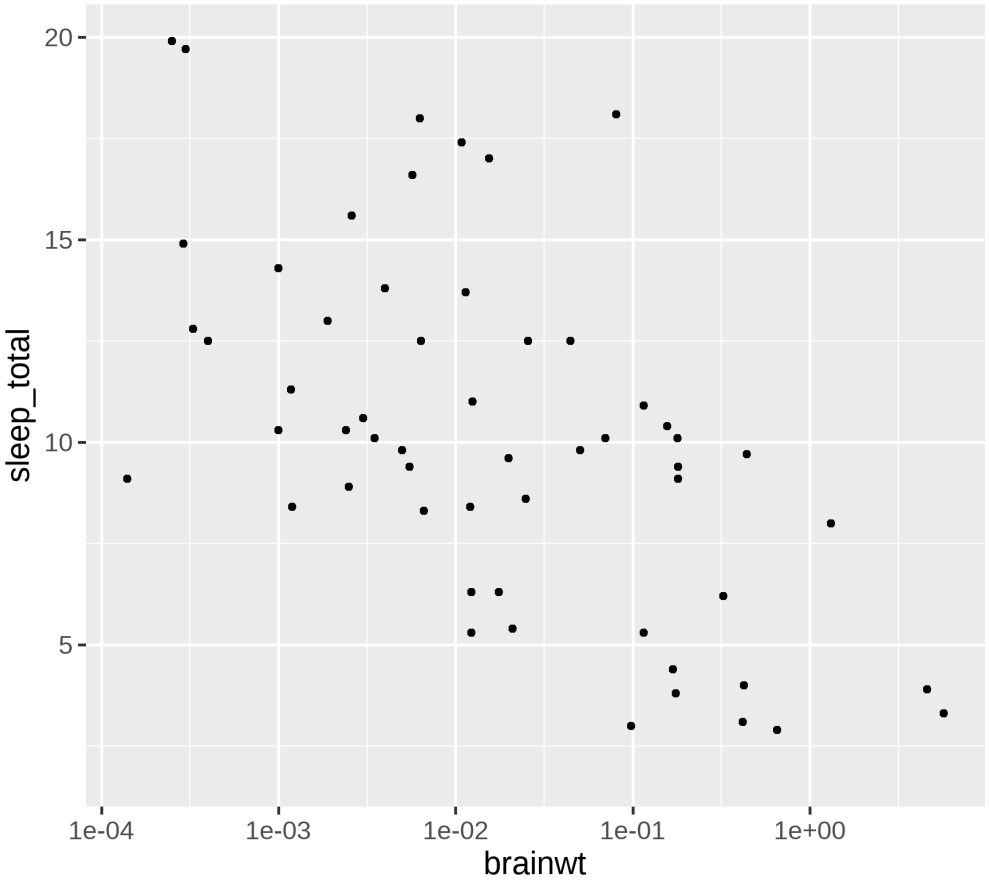


scale_x_log10()

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()

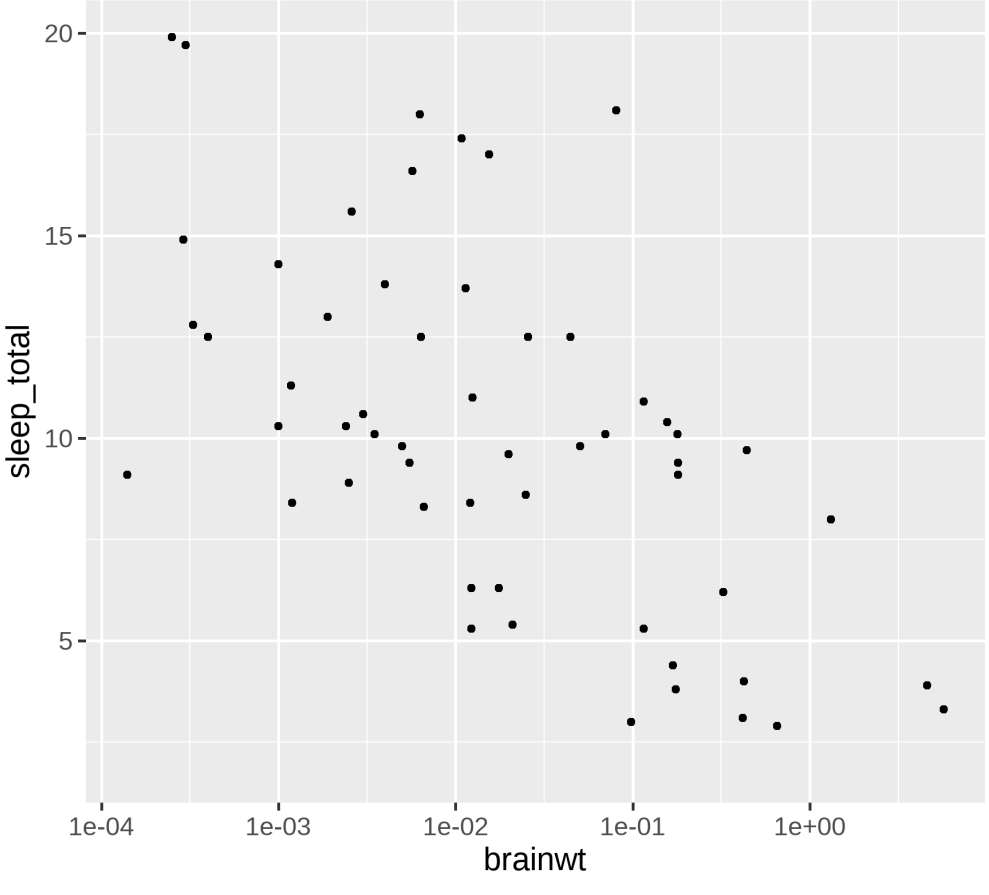
The scales onto which the aesthetic elements are mapped can be changed.

```
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.

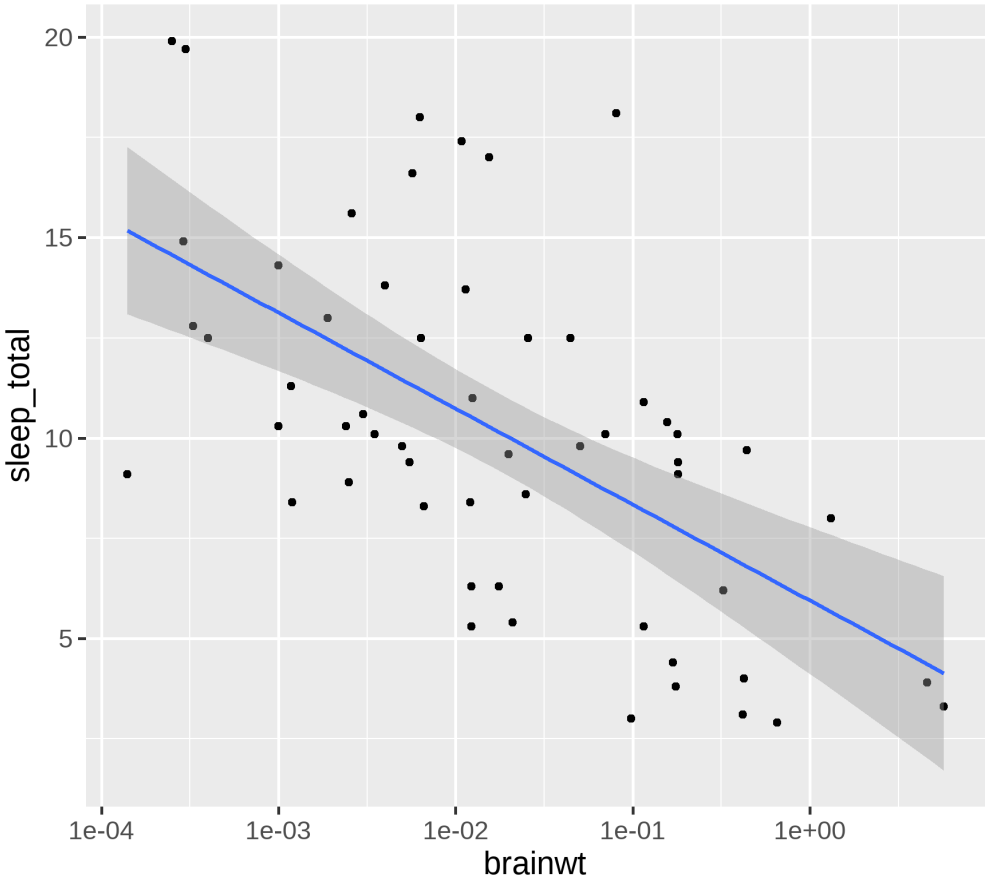


geom_smooth()

Add a smoothing line that helps see patterns in the data

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

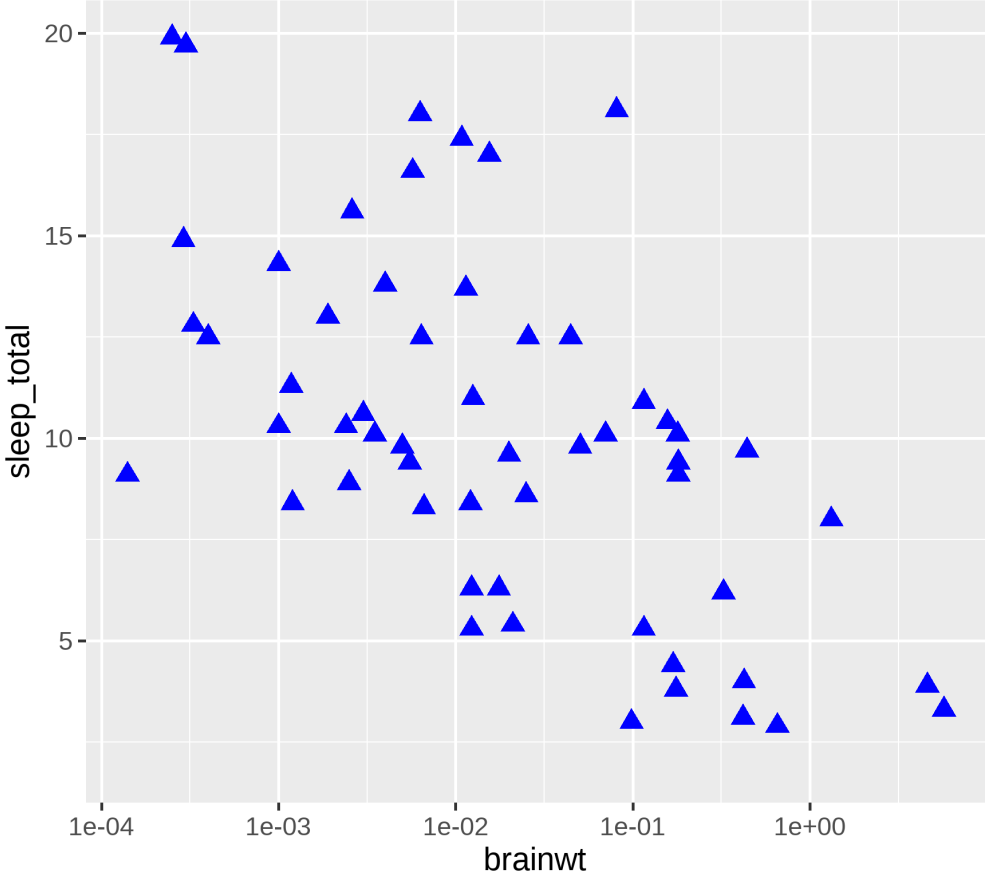
- With `method = "lm"`, a linear regression line is added
- The ribbon represents confidence interval around the regression
  - Turn ribbon off with `se = FALSE` argument



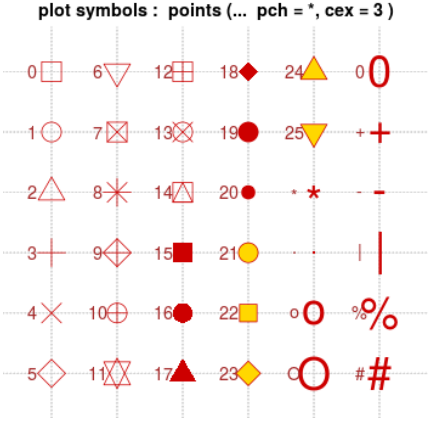
Change appearance of points

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()
```



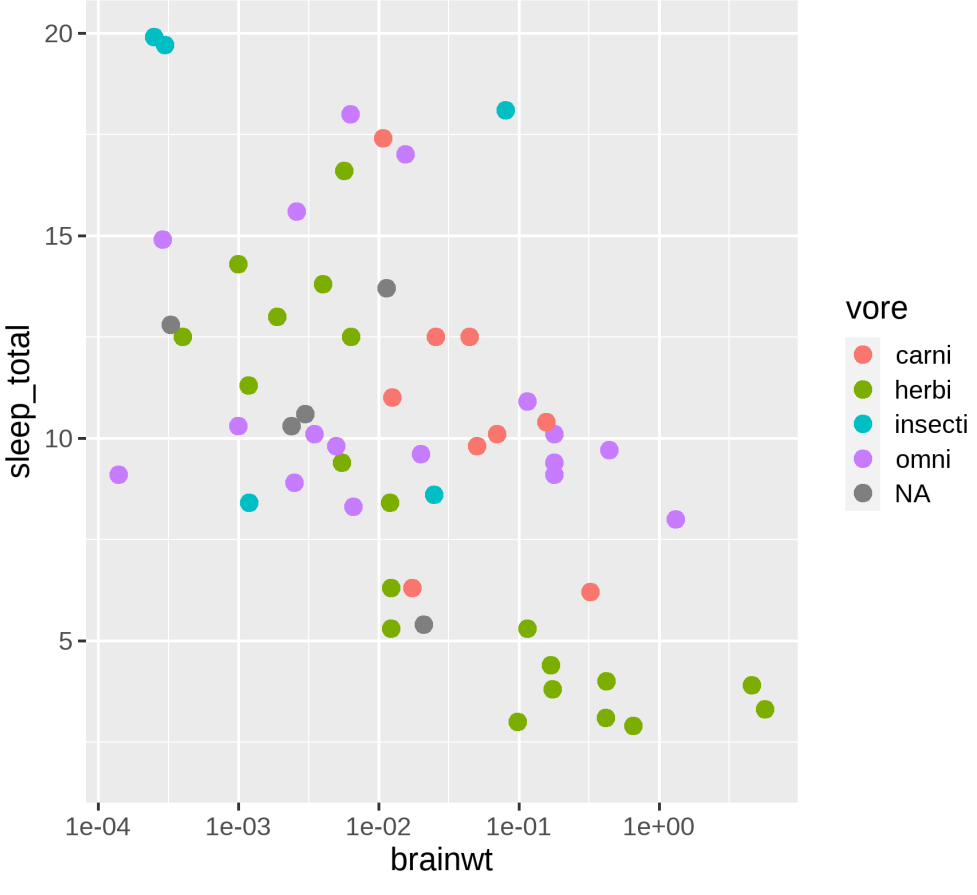
- Have a look here for the point shape codes:



`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
```

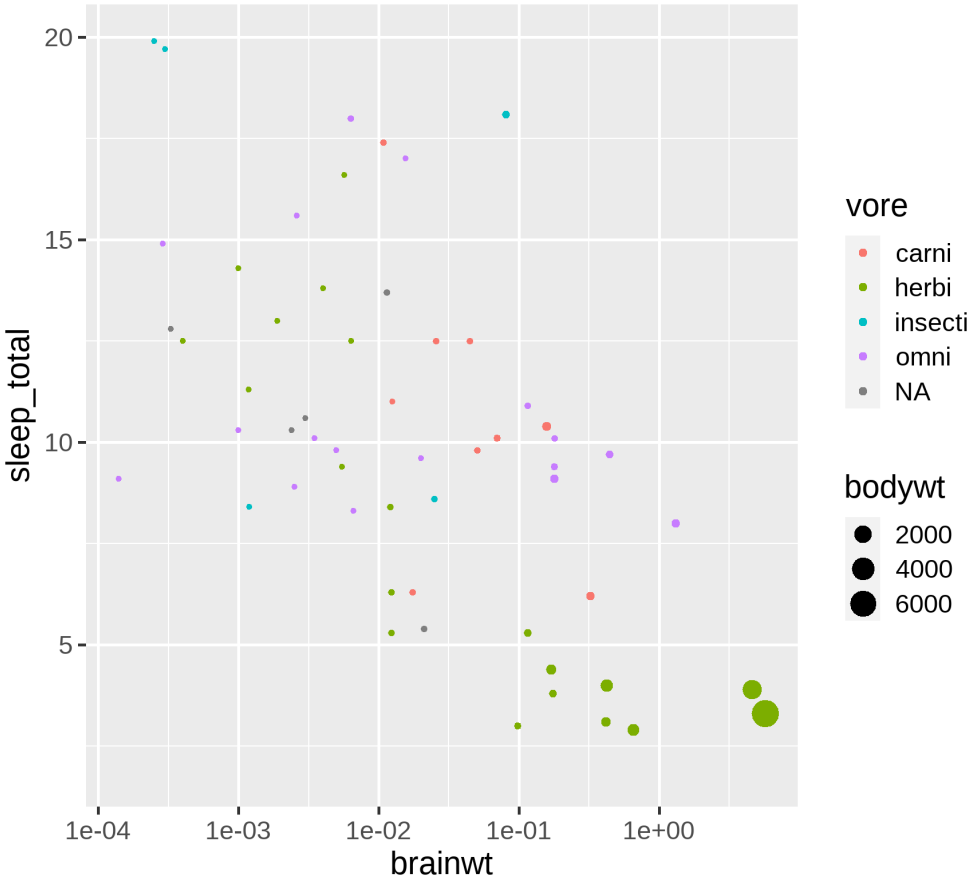


- Map the `vore` variable to the color aesthetic of the plot
- Plots can be saved in variables

`aes(size)`: mapping size to a variable

We can do the same with size:

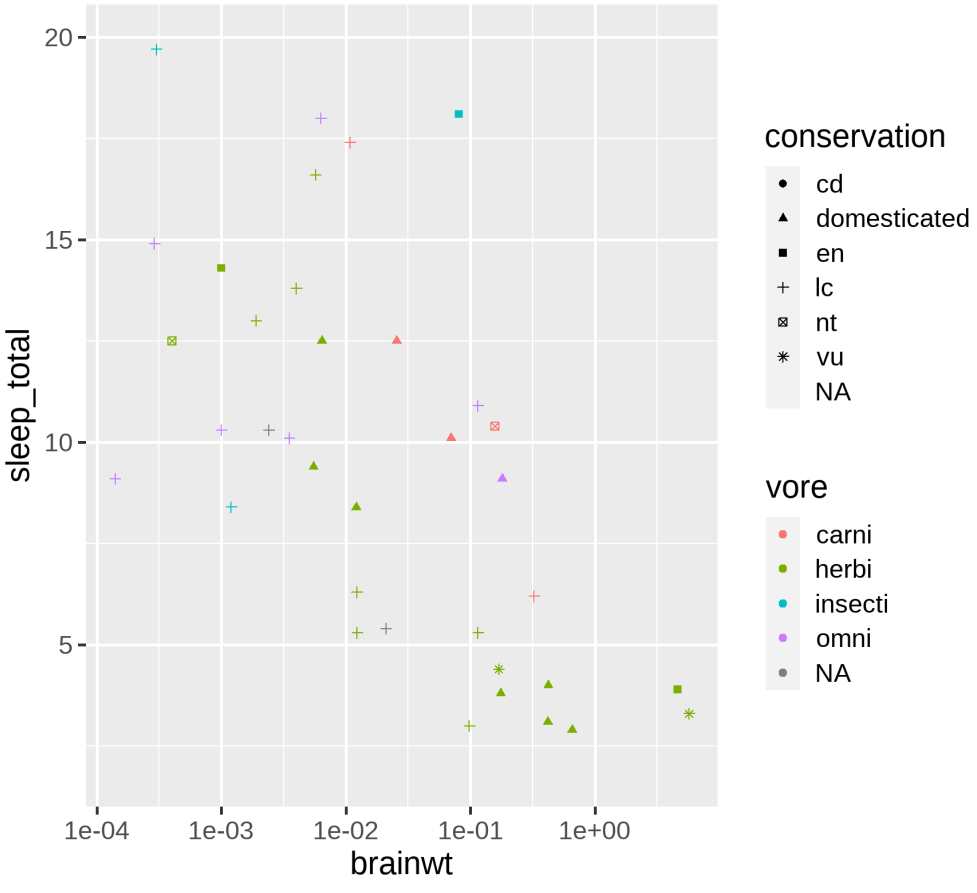
```
ggplot(
  data = msleep,
  aes(
    x = brainwt,
    y = sleep_total,
    color = vore,
    size = bodywt
  )
) +
  geom_point() +
  scale_x_log10()
```



`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()
```

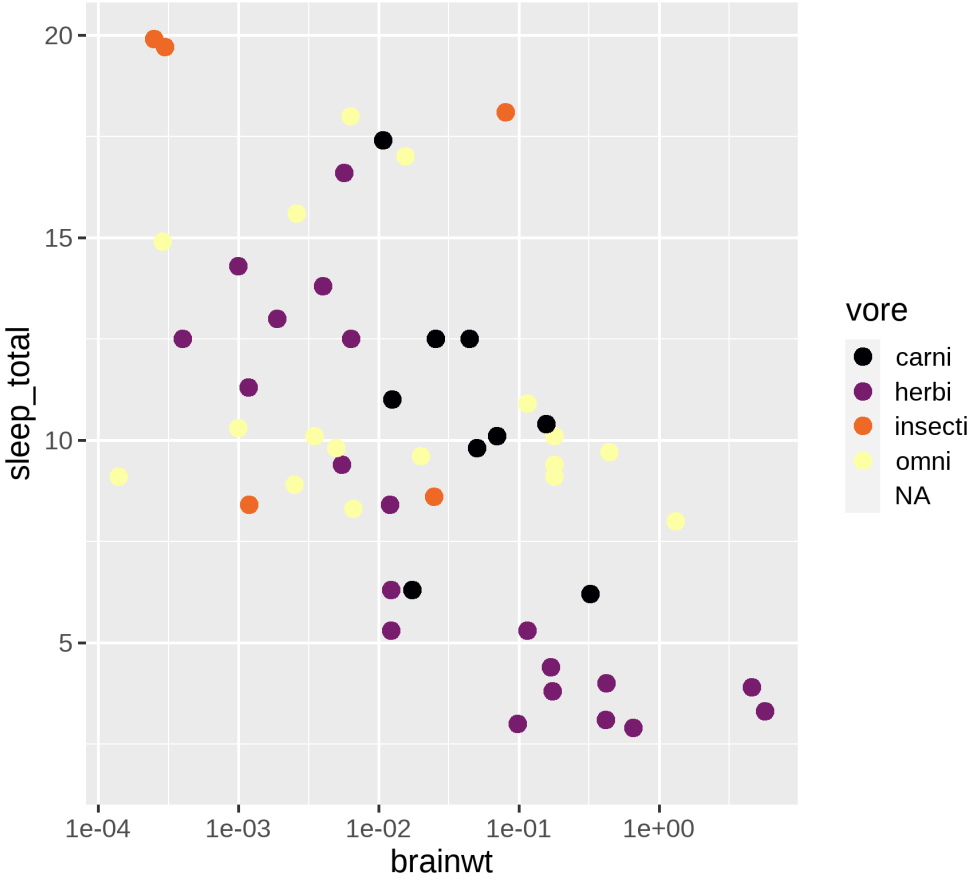


scale_color_viridis_d()

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 are:
  - `"magma"`, `"inferno"`, `"plasma"`, `"viridis"`, `"cividis"`

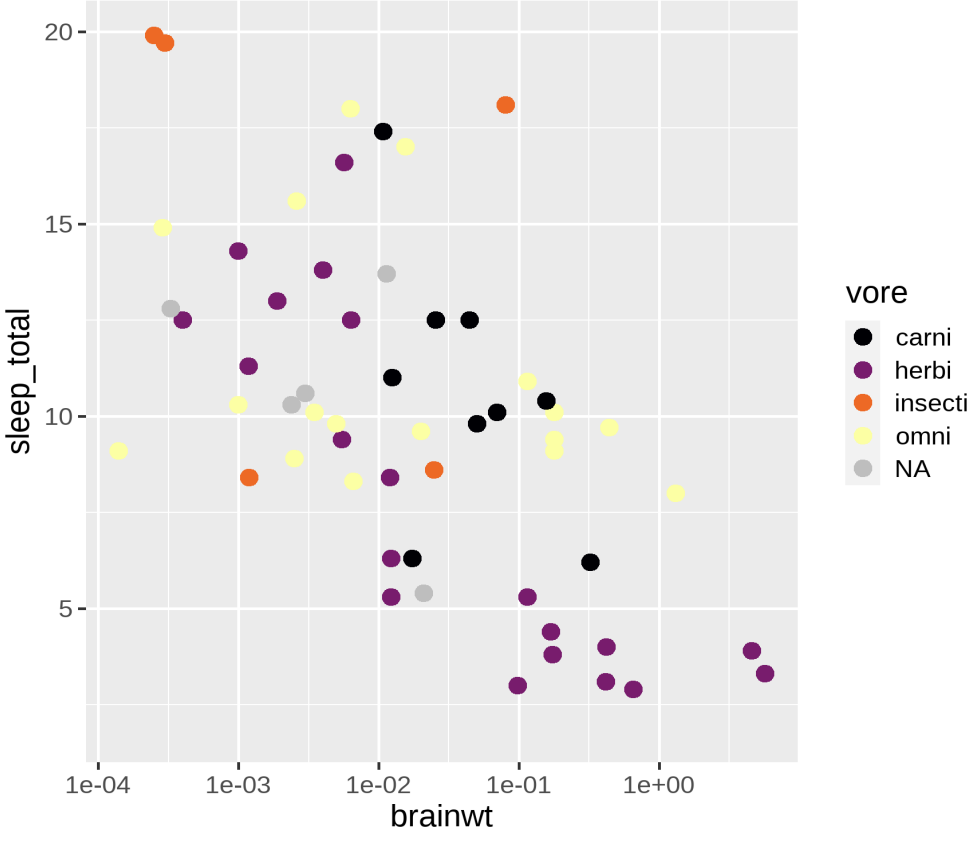


scale_color_viridis_d()

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)

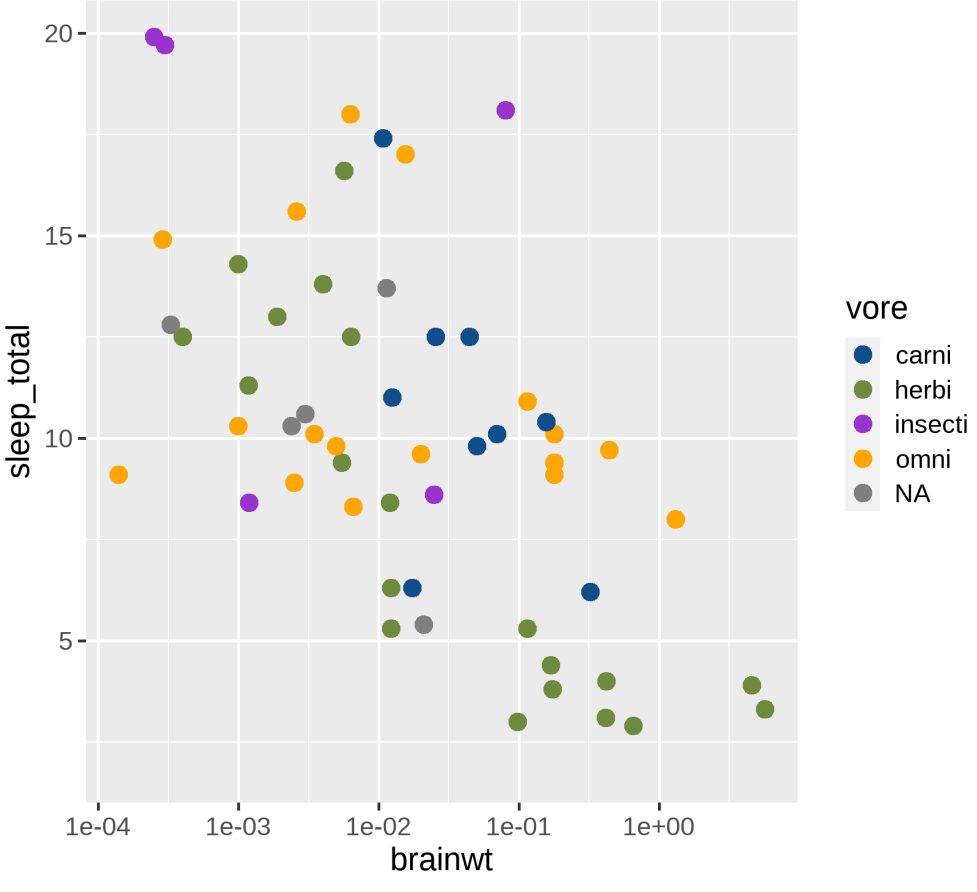


scale_color_manual()

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 (use websites to generate your own color palettes, e.g. here)



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
```

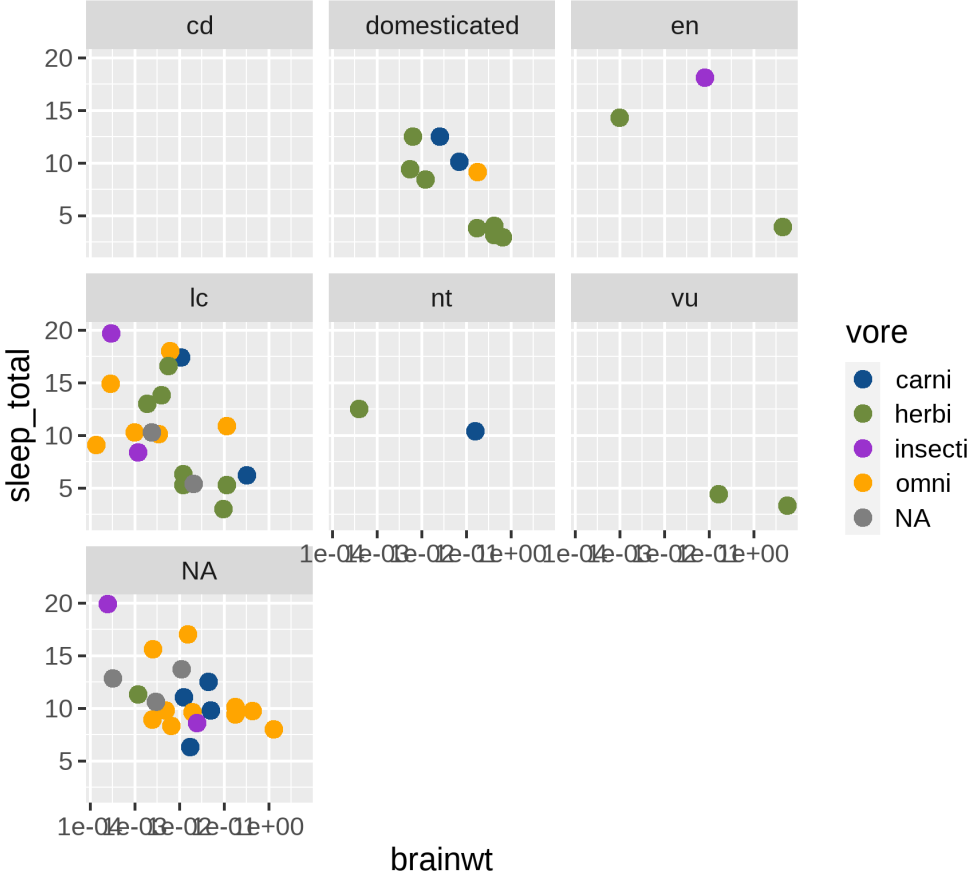
For a comprehensive list of color palettes available, check out this repository by Emil Hvitfeldt.

`facet_wrap()` and `facet_grid()`

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

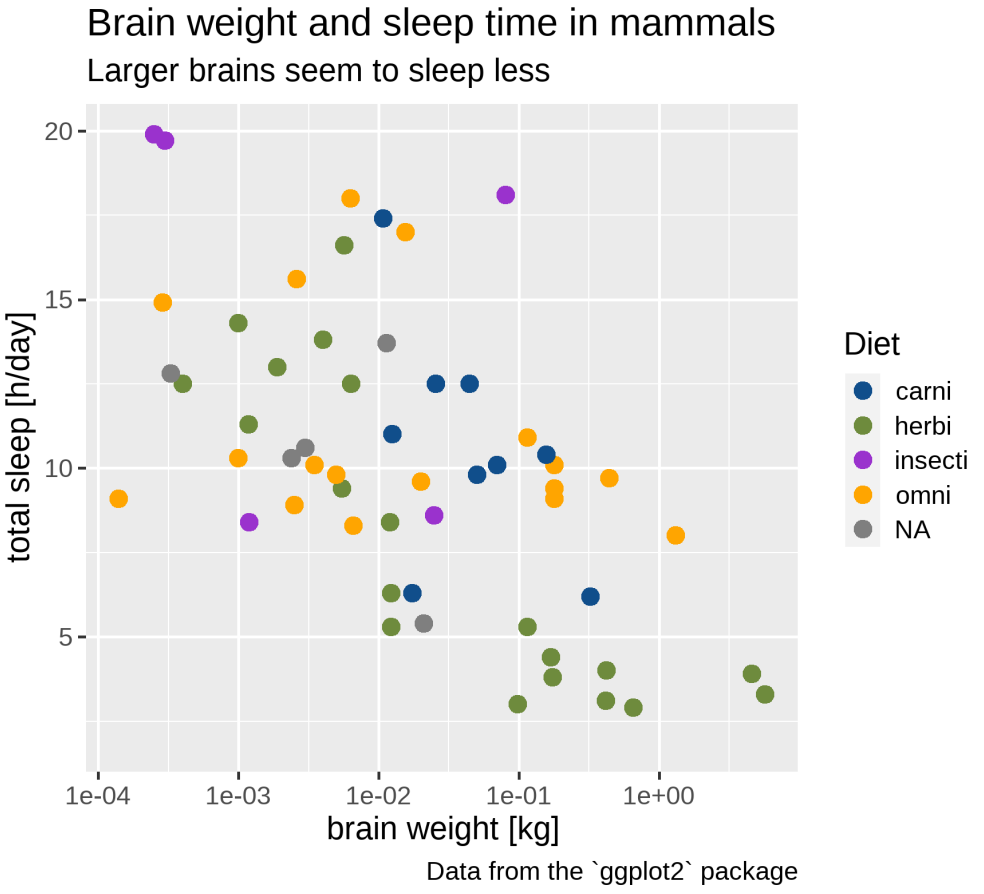
```
g +
  facet_wrap(~conservation)
```

- 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

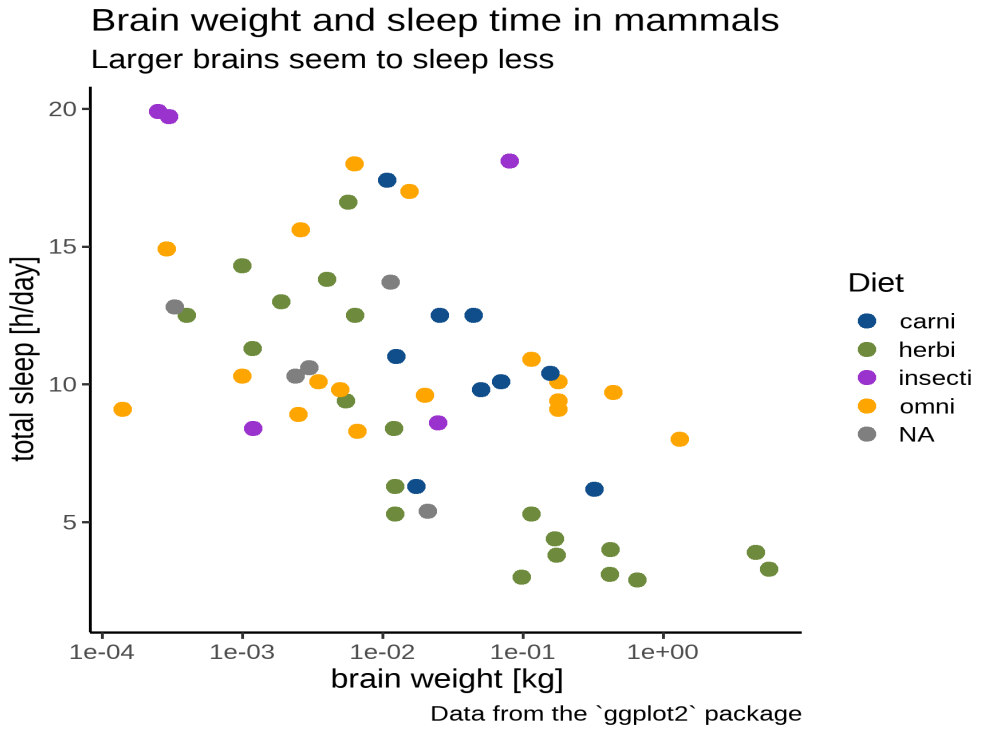
```
g <- g +
  labs(
    x = "brain weight [kg]",
    y = "total sleep [h/day]",
    color = "Diet",
    title = "Brain weight and sleep time in
mammals",
    subtitle = "Larger brains seem to sleep
less",
    caption = "Data from the ggplot2 package")
g
```



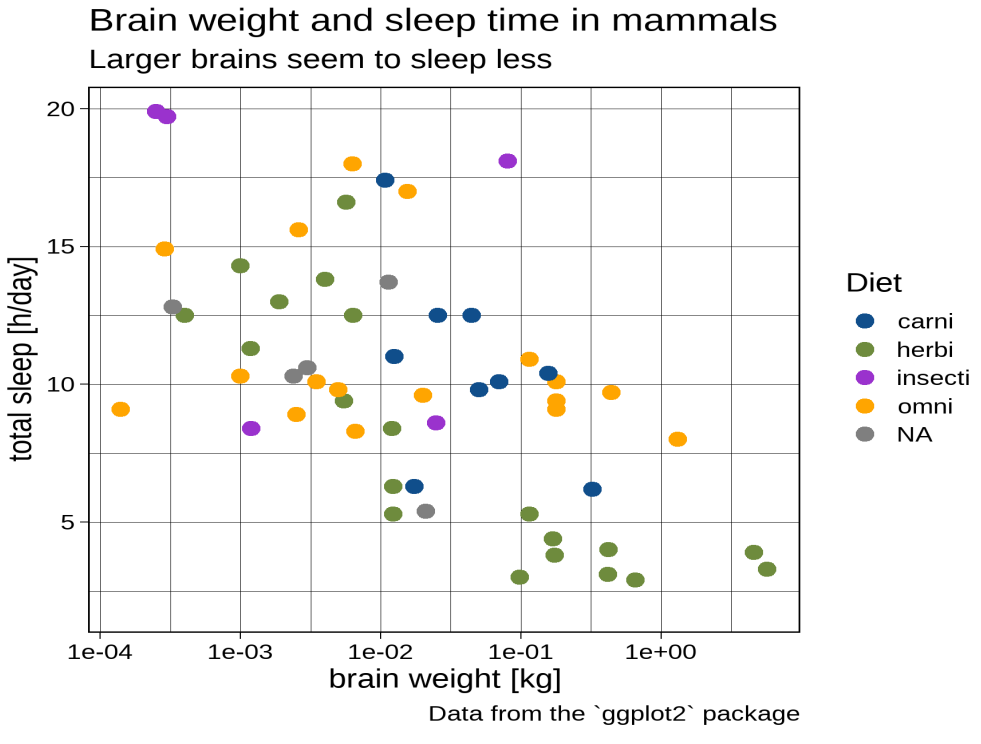
`theme_*()`: change appearance

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

```
g +
  theme_classic()
```



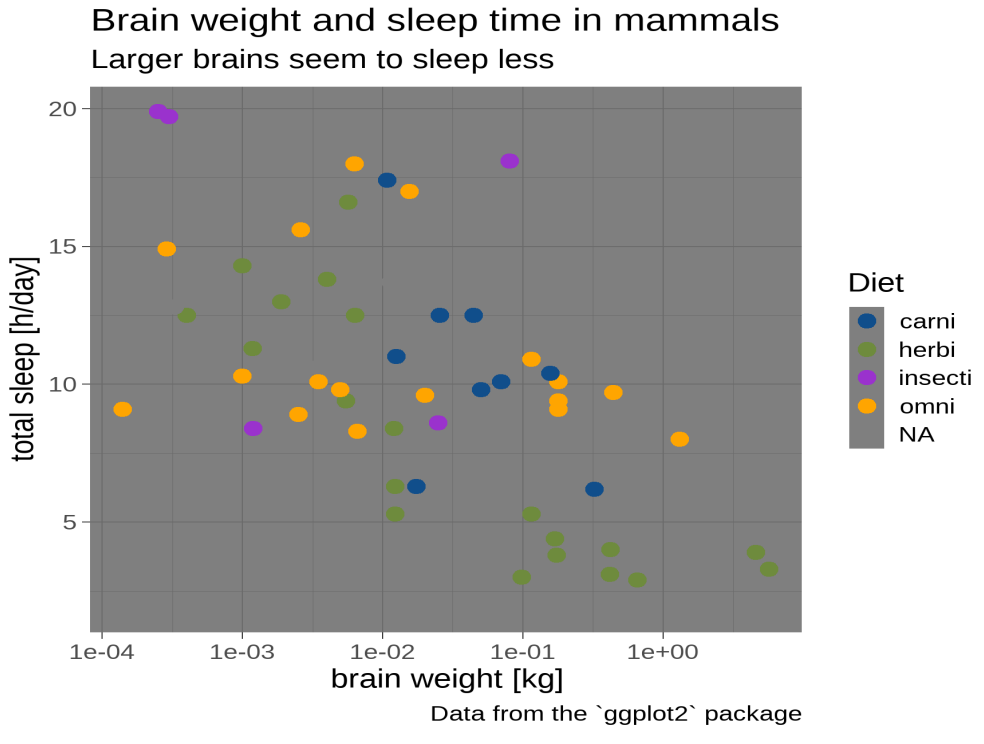

```
g +
  theme_linedraw()
```



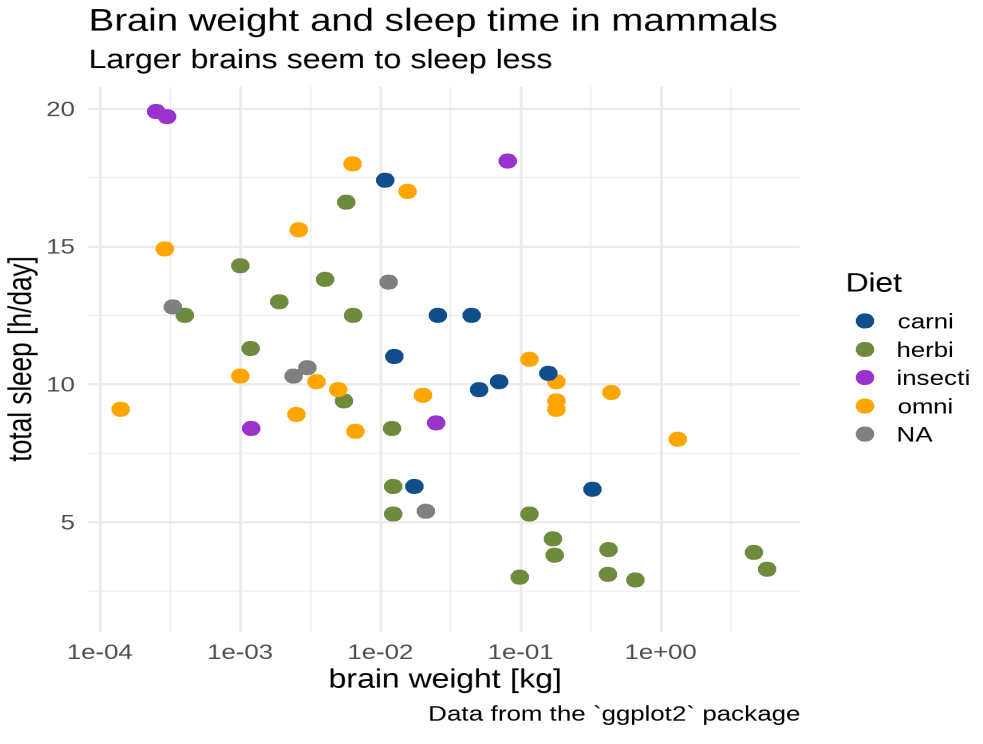
`theme_*()`: change appearance

`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

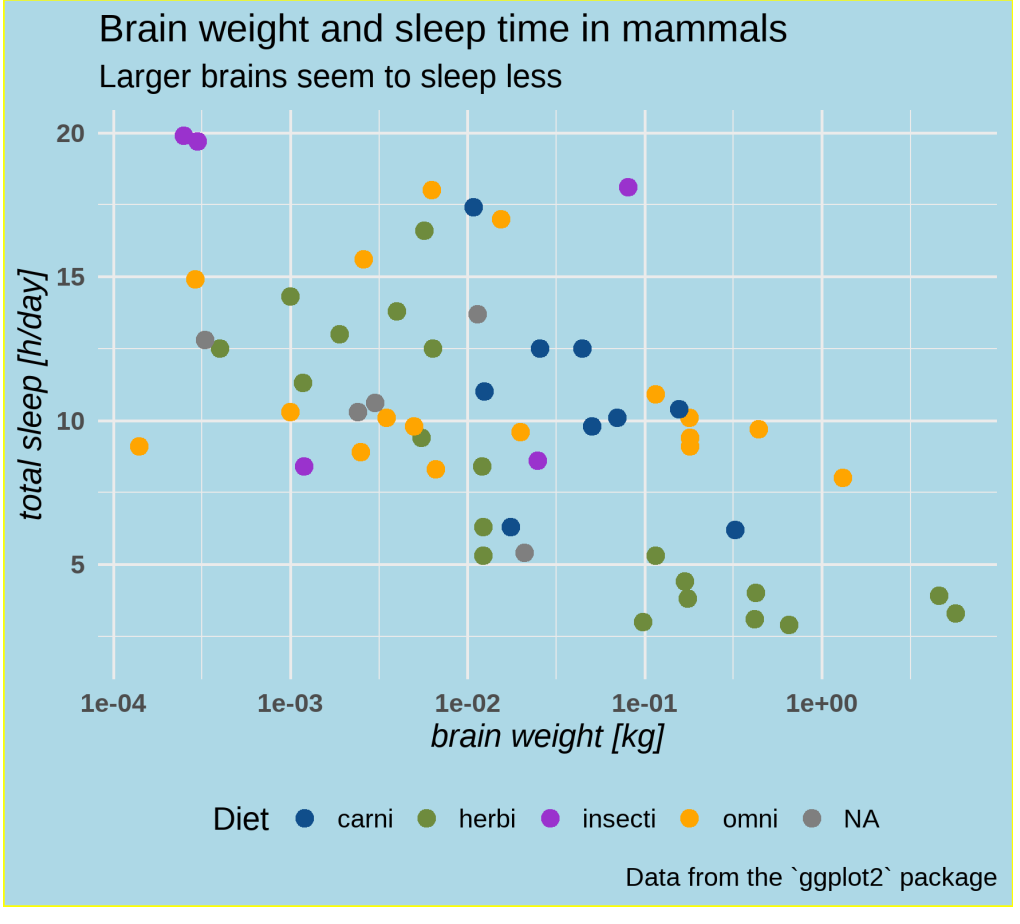
To edit a theme, just add another `theme()` layer to your plot.

```
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")
  )
```

- The basic functioning is:

```
theme(
  element_name = element_function()
)
```

- Look here for an overview of the elemtens that you can change and the corresponding



`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:

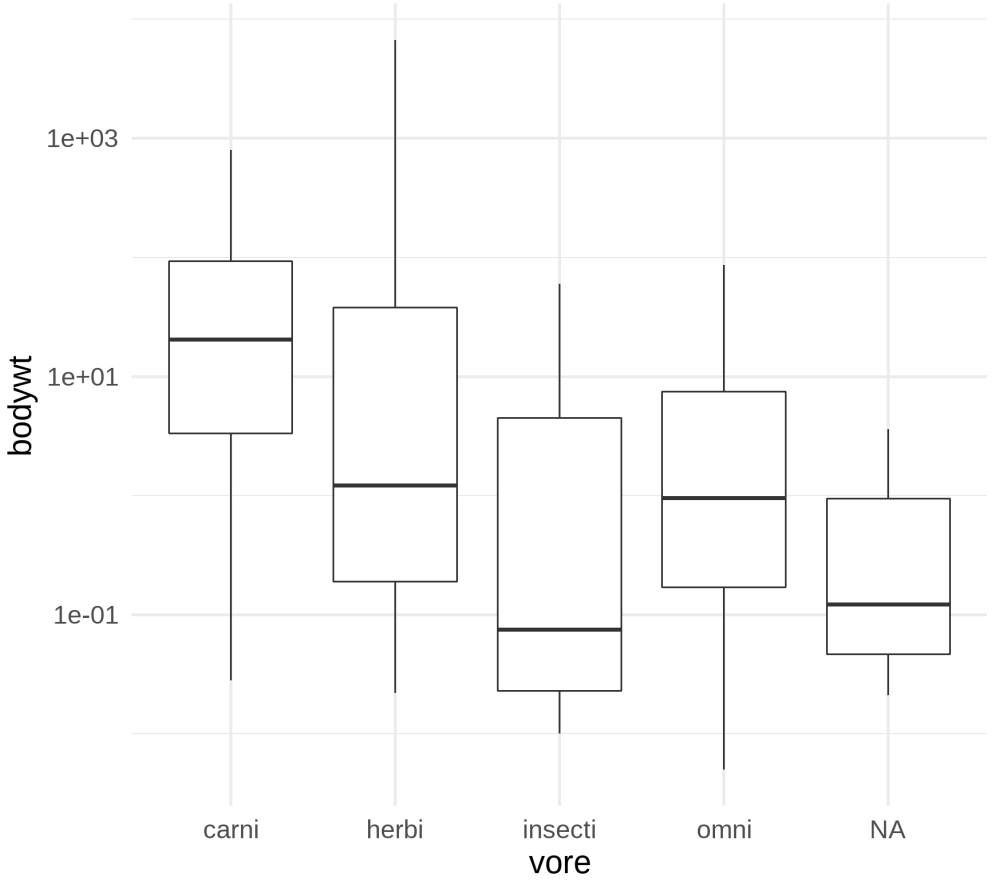
```
theme_set(theme_minimal(base_size = 16))
```

This is very practical if you want to achieve a consistent look, e.g. for a scientific journal.

geom_boxplot()

A boxplot can be created with `geom_boxplot`:

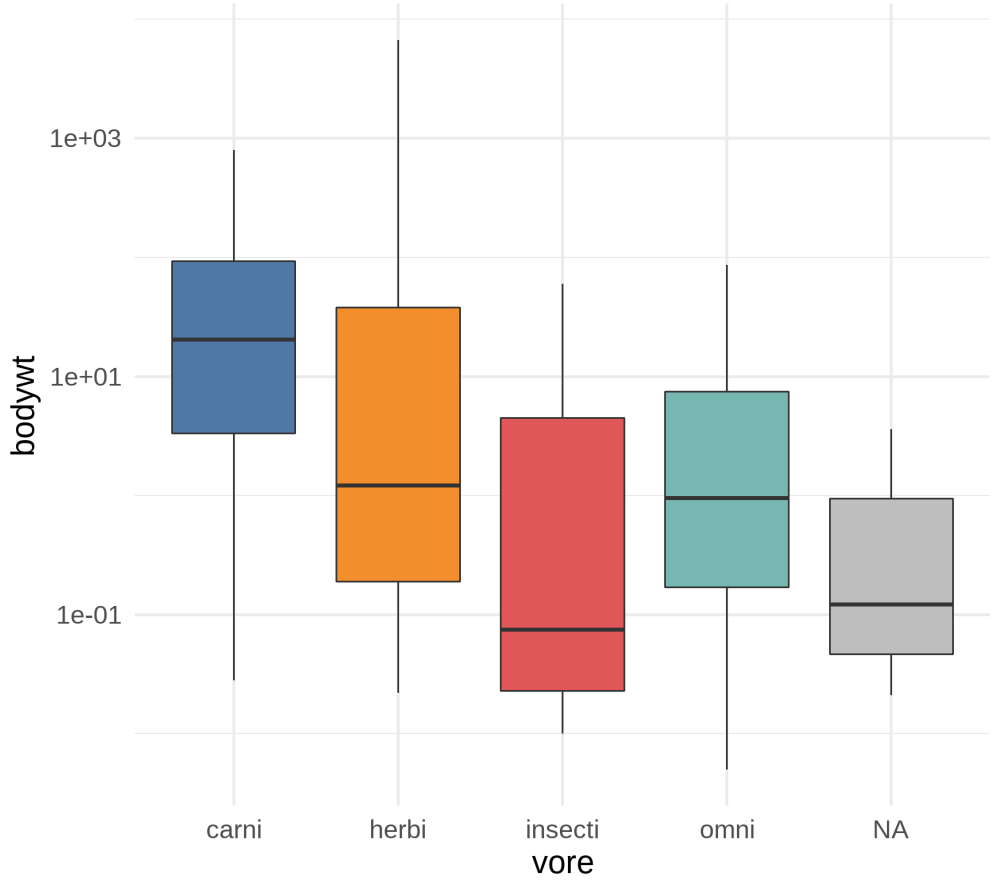
```
ggplot(msleep, aes(x = vore, y = bodywt)) +
  geom_boxplot() +
  scale_y_log10()
```



`aes(fill)` and `scale_fill_*()`

Pay attention to fill vs. color aesthetic

```
ggplot(msleep, aes(x = vore, y = bodywt)) +
  geom_boxplot(
    aes(fill = vore)
  ) +
  scale_y_log10() +
ggthemes::scale_fill_tableau(
    na.value = "gray",
    guide = "none")
```

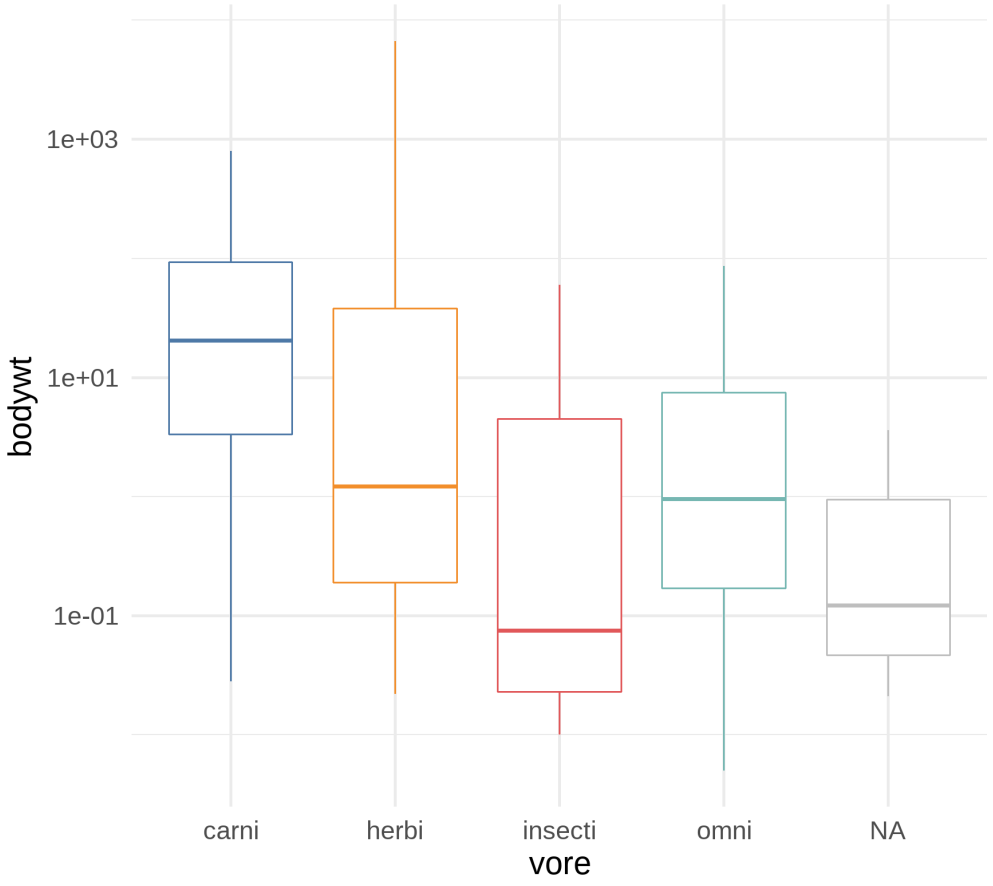


`aes(color)` and `scale_color_*()`

Pay attention to fill vs. color aesthetic

```
ggplot(msleep, aes(x = vore, y = bodywt)) +
  geom_boxplot(
    aes(color = vore)
  ) +
  scale_y_log10() +
  ggthemes::scale_color_tableau(
    na.value = "gray",
    guide = "none")
```

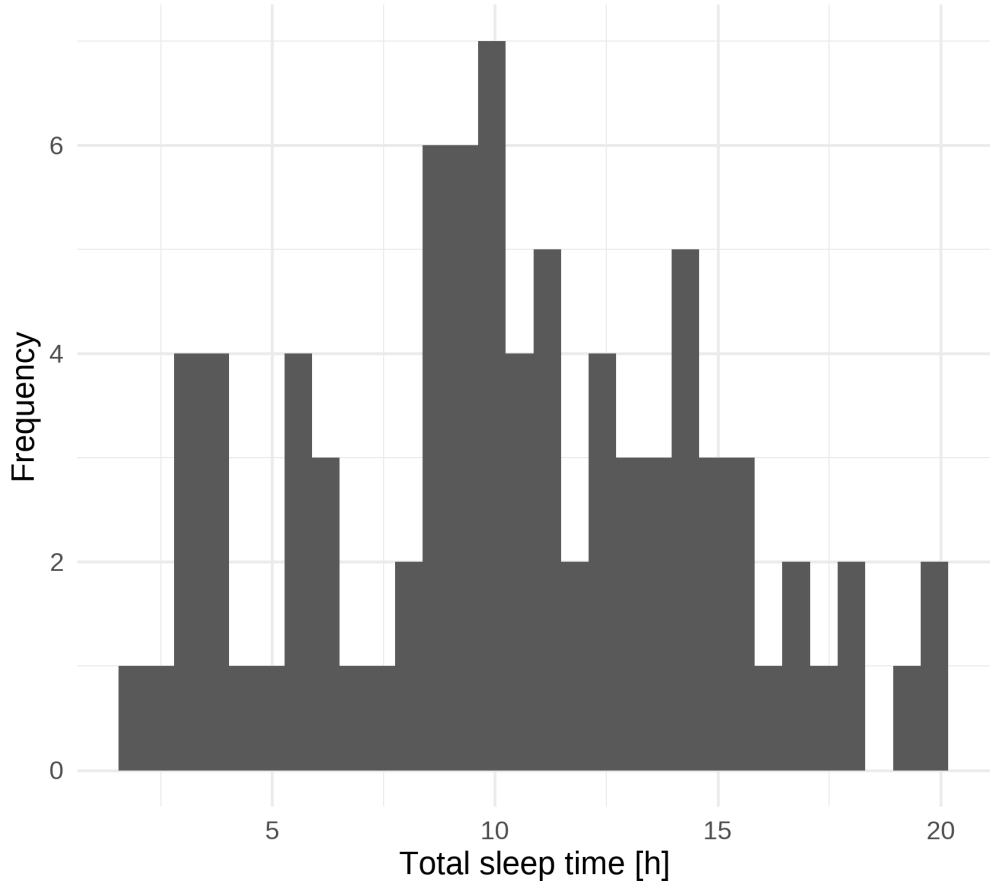
- `color` means the border color or the color of points and lines
- `fill` means the fill color of boxes, bars, ...



Histograms: `geom_histogram()`

Basic histogram of total sleep time:

```
ggplot(msleep, aes(x=sleep_total)) +
  geom_histogram() +
  labs(x = "Total sleep time [h]",
       y = "Frequency")
```



Discrete vs. continous variables

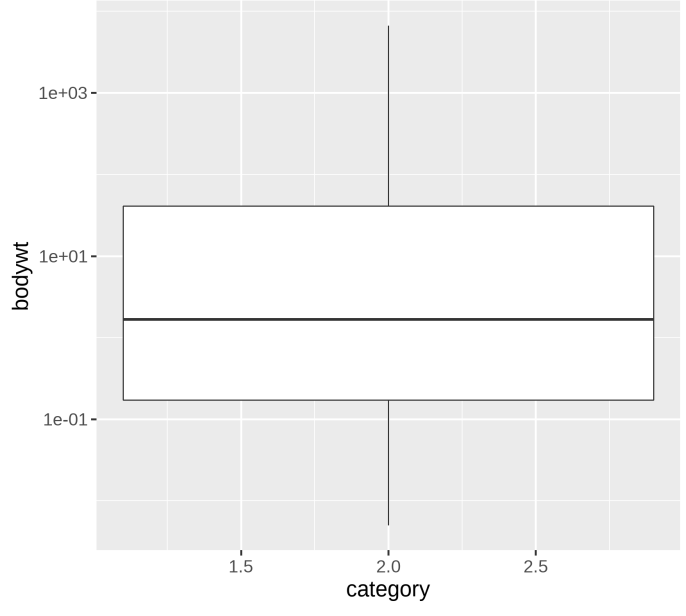
Let's add a sleep category to the data set depending on the hours of sleep an animal gets in total:

```
msleep$category <- ifelse(
  msleep$sleep_total < 5,
  1,
  ifelse(msleep$sleep_total > 5 & msleep$sleep_total < 10,
         2,
         3)
)
str(msleep)
## tibble [83 x 12] (S3: tbl_df/tbl/data.frame)
##  $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed
shrew" ...
##  $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
##  $ vore        : chr [1:83] "carni" "omni" "herbi" "omni" ...
##  $ order       : chr [1:83] "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
##  $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
##  $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
##  $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
##  $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
##  $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
##  $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
##  $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
##  $ category    : num [1:83] 3 3 3 3 1 3 2 2 3 1 ...
```

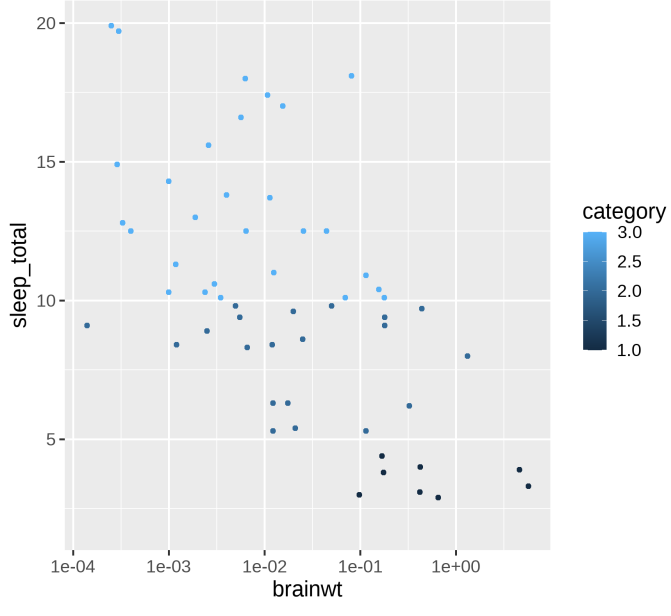
Discrete vs. continous variables

What's happening here?

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




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



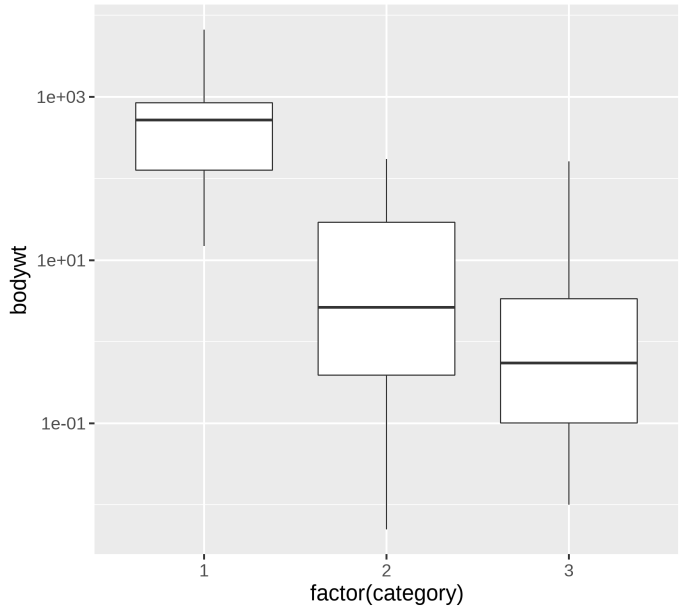
Discrete vs. continous variables

Artwork by Allison Horst

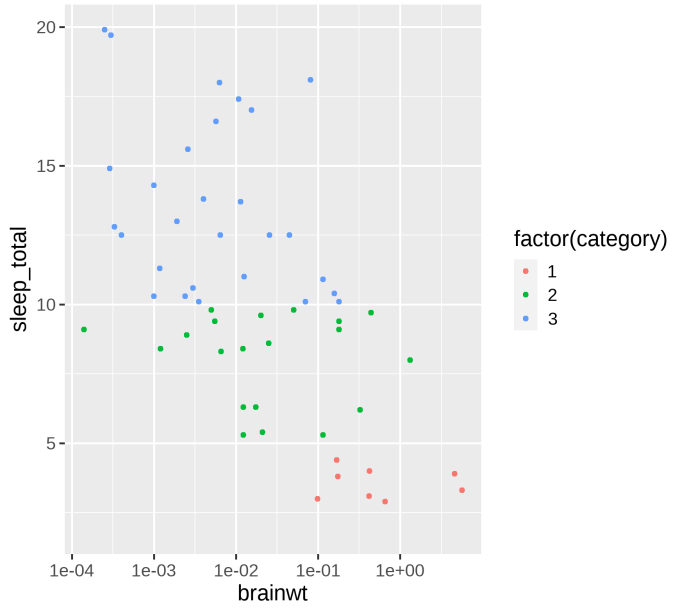
Discrete vs. continous variables

Solution: Turn continuous variable into a factor before plotting

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



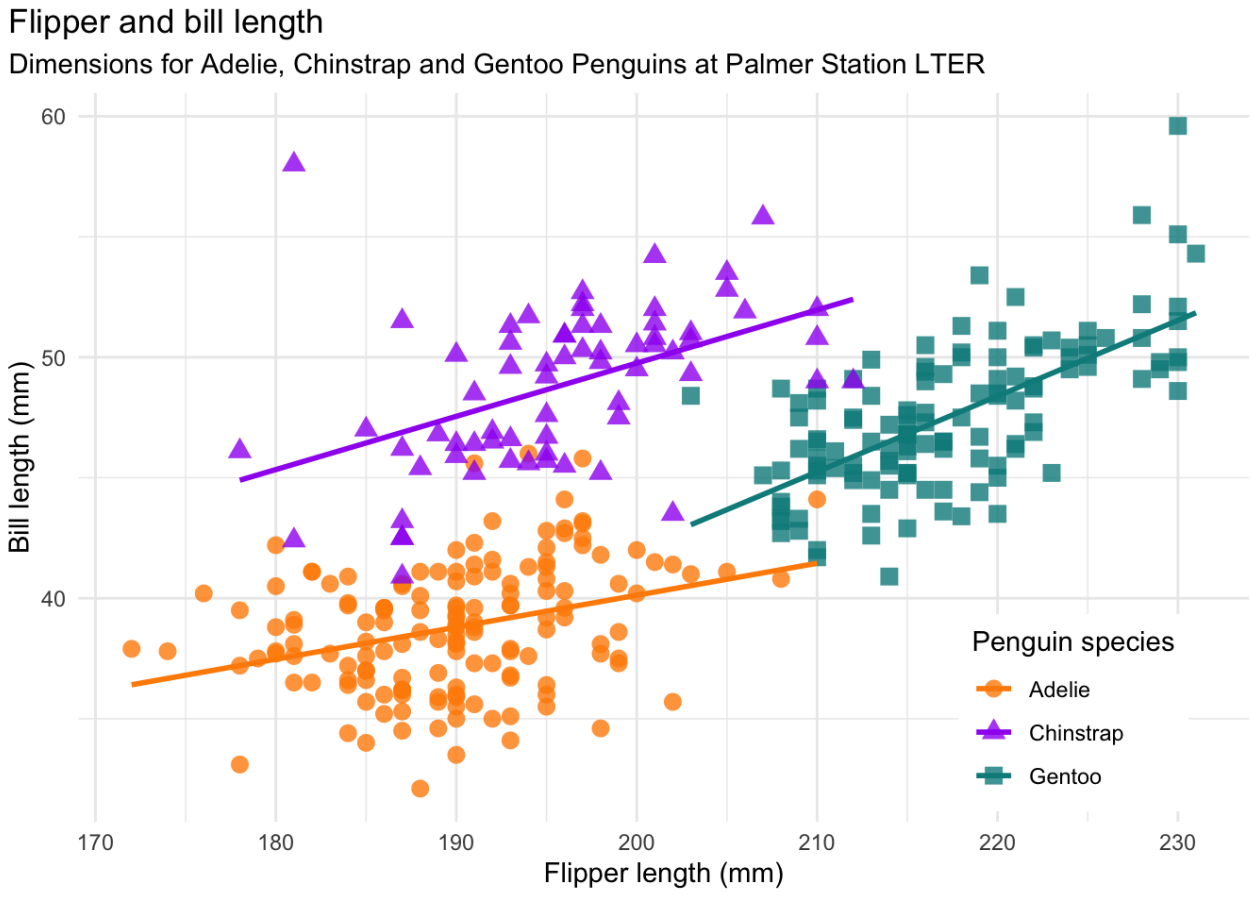

```
ggplot(msleep,
       aes(x = brainwt,
           y = sleep_total,
           color = factor(category))) +
  geom_point() + scale_x_log10()
```



Let's look at an example

What are the layers in this plot?

- data
- aesthetic mapping
- geoms
- themes



Visualization from the palmerpenguins package

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,
       heigth = 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`, `tidyr`, `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 1: Create your own penguin ggplots (90 min)

Find the task description here