ggplot2

Maria Novosolov

28-04-2020

A quick introduction to R projects and github

(adopted from the GLOBE R working group)

What is Git and Github

- Git is a version control system that lives on your computer. Think "track changes" for files.
- Github is "Dropbox" for git-based projects on the internet.

What github is used for

- share/store analyses and functions
- browse past versions of code
- browse source code (CRAN, tidyverse, rOpenSci)
- use packages not on CRAN (`devtools::install_github("account/repo")`)
- host web pages

You get unlimited private GitHub repositories for *free* (normally $7/month) while you are a student! Sign up here.

GitHub terminology

- **repository** or repo - project/folder
- **local** - files stored on your computer
- **remote** - files on github.com

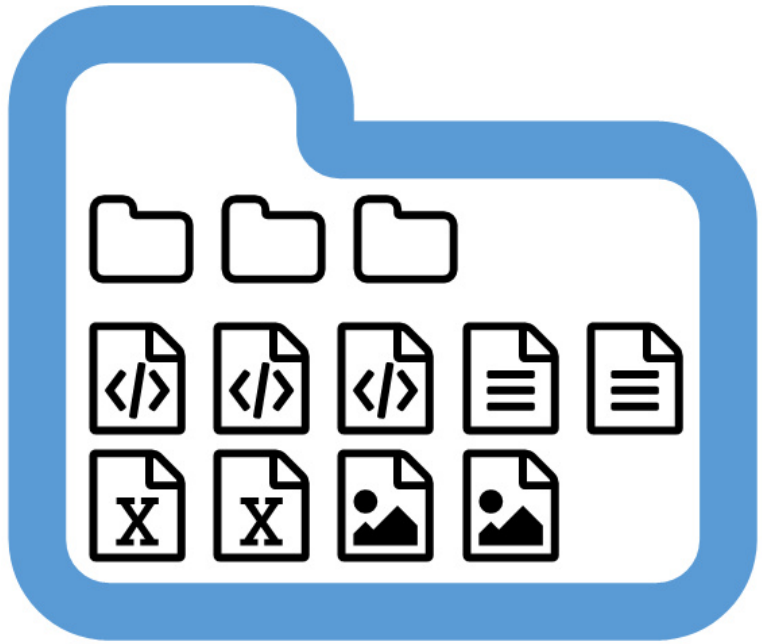
R projects

Comfortable but hard to reproduce

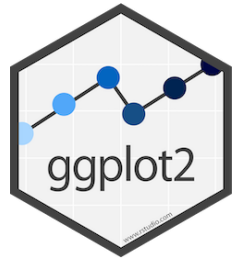
Portable, organized, minimalistic

What's the deal with projects?

Each project is a self-contained set of files. Projects make it **easy to transfer** files to another computer because its **boundaries are clearly defined**



Why use projects?

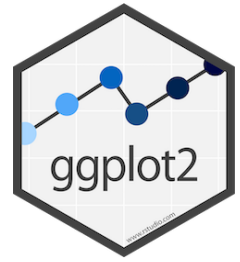


Projects help to maintain good workflow habits:

- **Fresh R processes** - each project comes with its own environment
- **Portability** - projects have clearly defined base directories; makes it easy to use relative paths

Extra reading: basic care and feeding of data in R and project management

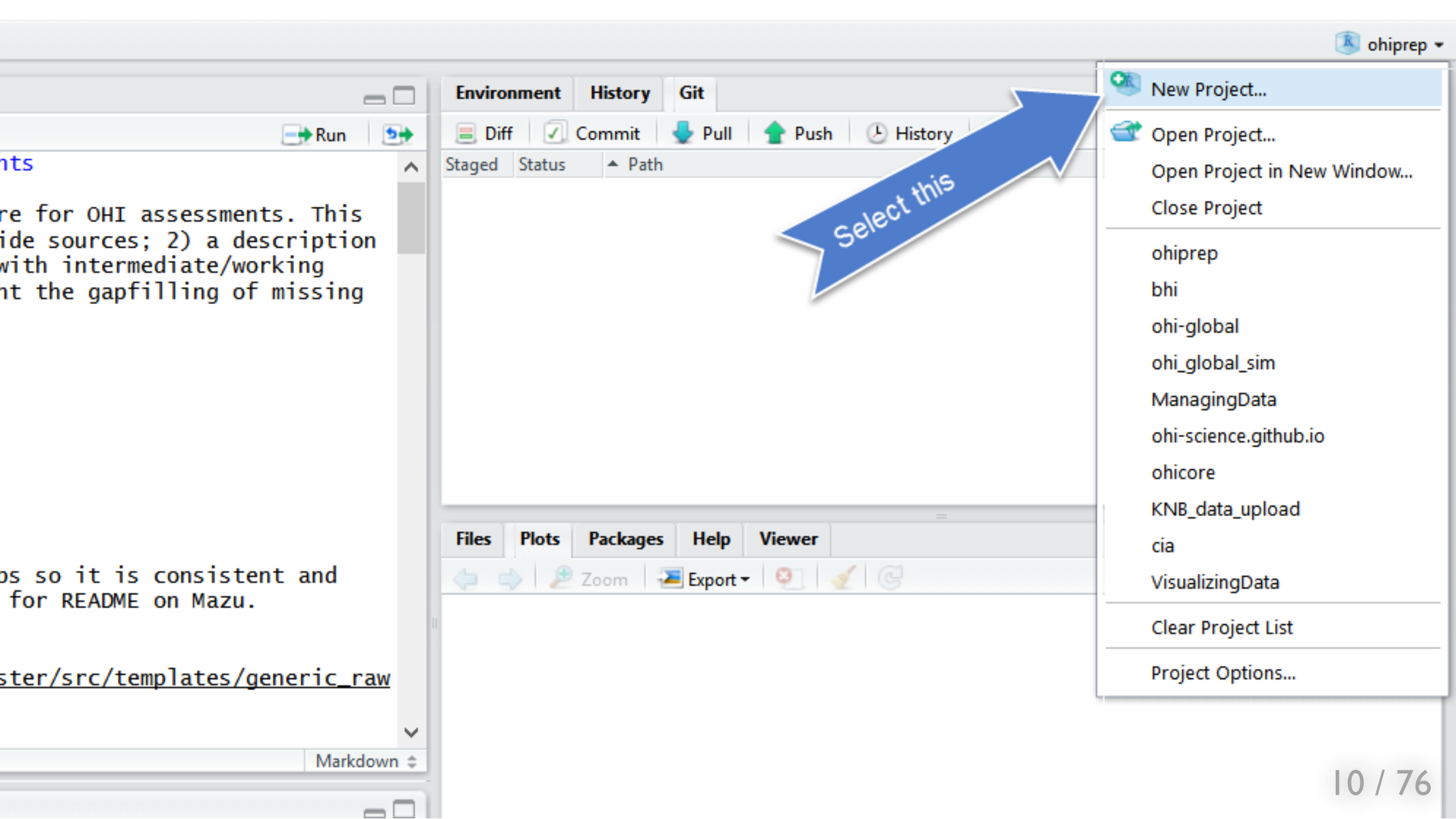
Use relative paths inside projects



- looks cleaner
- works for other people/computers

```
read.csv("data/eilat_survey_2017.csv")
#rather than
read.csv("C:/חזי/phd/thesis/reef_surveys/data/eilat_survey_2017.csv")
# or
setwd("C:/חזי/phd/thesis/reef_surveys/data")
read.csv("/eilat_survey_2017.csv")
```

You never have to use `setwd()` again!


ohiprep
Run
nts
re for OHI assessments. This
ide sources; 2) a description
with intermediate/working
nt the gapfilling of missing
os so it is consistent and
for README on Mazu.
ster/src/templates/generic_raw
Markdown
Environment
History
Git
Diff
Commit
Pull
Push
History
Staged
Status
Path
Select this
New Project...
Open Project...
Open Project in New Window...
Close Project
ohiprep
bhi
ohi-global
ohi_global_sim
ManagingData
ohi-science.github.io
ohicore
KNB_data_upload
cia
VisualizingData
Clear Project List
Project Options...
Files
Plots
Packages
Help
Viewer
Zoom
Export


TAR
WARS

If you're not convinced

Everytime you start working with R remember to set your working directory to the folder where your data is in:

```
setwd("path/to/your/files")
```

Or write the full path for each file you load

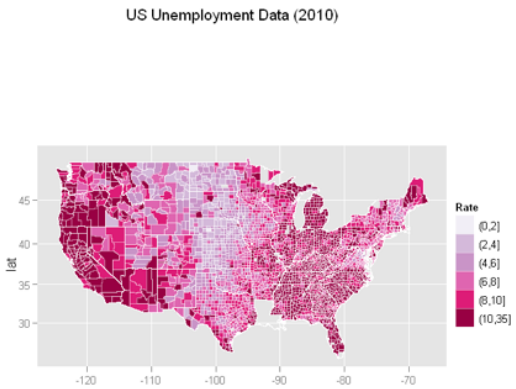
```
data← read.csv("path/to/your/data")
```

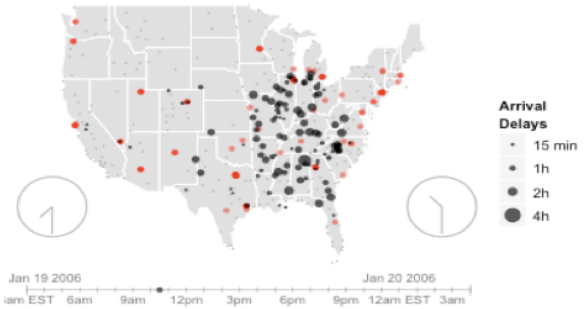
Let's start with `ggplot2`

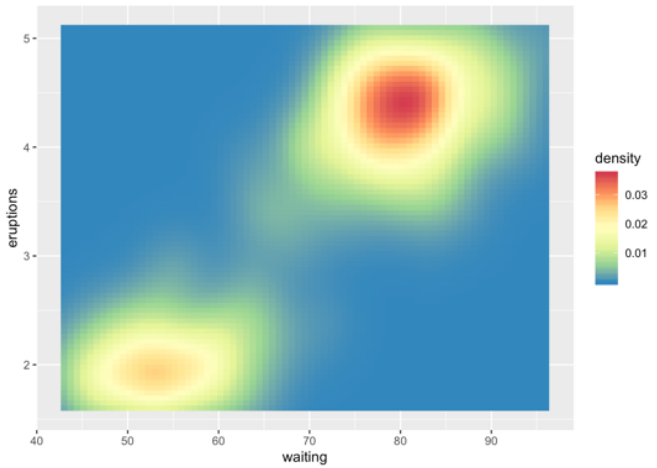
The idea behind `ggplot2`

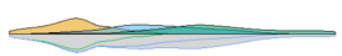
Combining all the good and leaving out all the bad of all base R packages for plotting

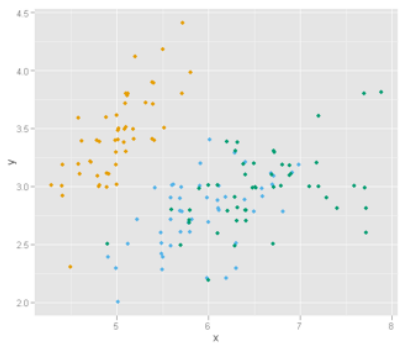
Some examples

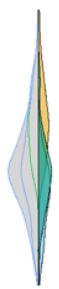


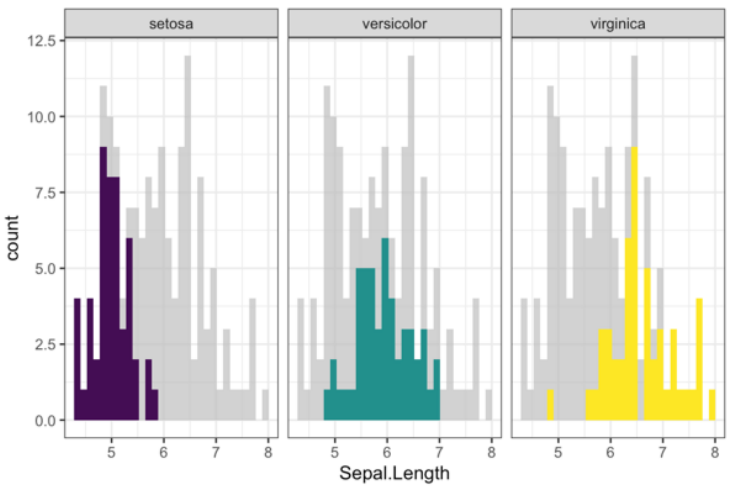


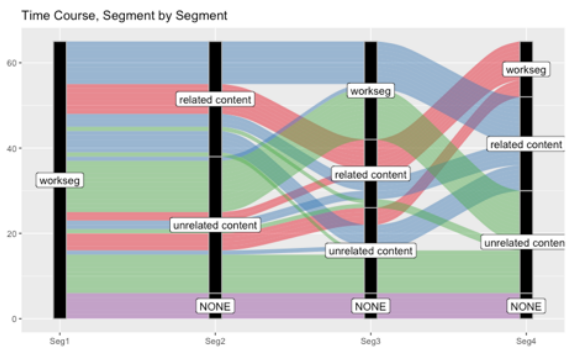












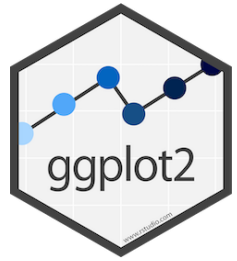
What `ggplot2` can do?

Almost anything you want it

What `ggplot2` can't do?

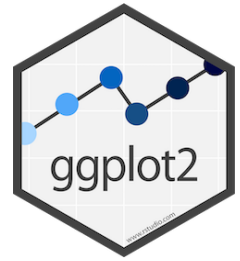
It can't choose for you which graphics represent your data best

The basics of `ggplot2`



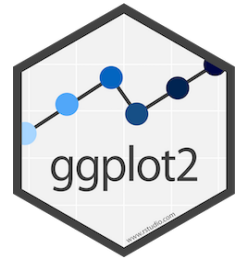
- Based on the Grammar of Graphics (Wilkinson, 2005)
- Based on a layer system
- Flexible set of components for creating any type of graphics

Specification



Components of a graphic:

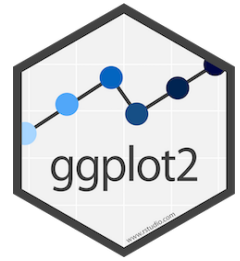
- **DATA** - the dataset that will be used for the other components
- **TRANS** - variable transformation
- **SCALE** - scale transformation
- **ELEMENT** - graphs and their aesthetic attributes
- **COORD** - a coordinate system
- **GUIDE** - one or more guides



Install the package using `install.packages("ggplot2")`

load the package with `library(ggplot2)`

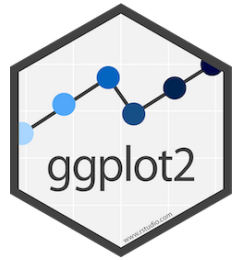
Two options of working with the package



`qplot()` - quick and dirty plotting. Limited options

`ggplot()` - base function for more versatile plotting

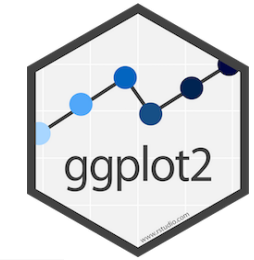
qplot()



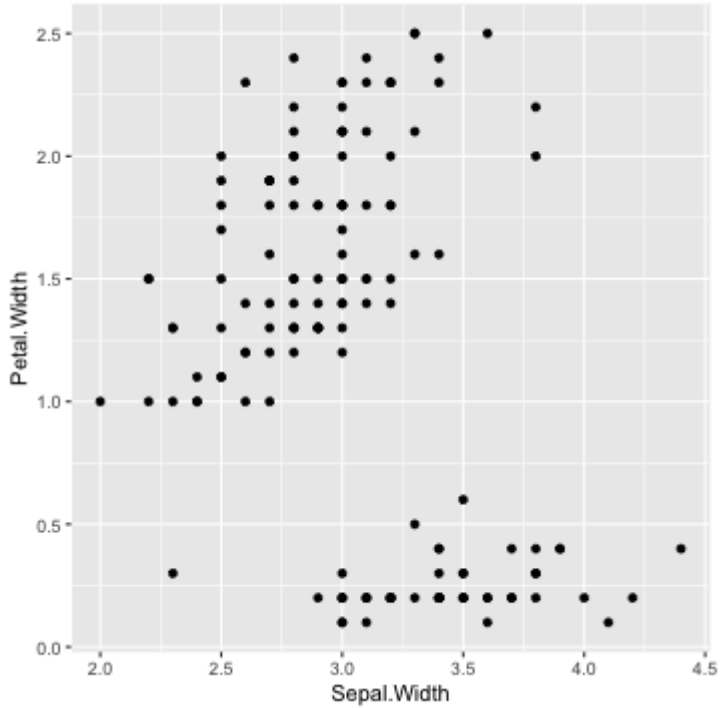
- Syntax similar the the `base` R `plot()`
- **x** == predictor
- **y** == response
- **data** == the data frame that holds the variables to plot

```
qplot(x,y,data = mydata)
```

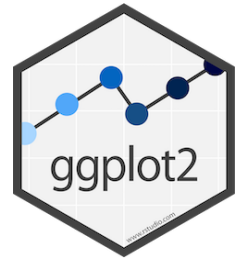
Example using the `iris` dataset




```
qplot(x = Sepal.Width,y = Petal.Width,data = iris)
```



The building blocks of `ggplot()`



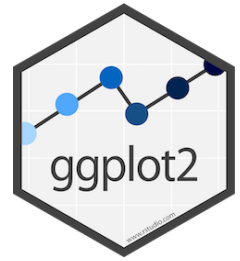
A **plot** is made up of multiple layers.

A **layer** consists of data, a set of mappings between variables and **aes**thetics, a **geom**etric object and a **stat**istical transformation.

Scales control the details of the mapping.

All components are independent and reusable.

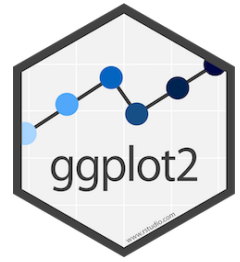
Advantages of the layers system



- Change a single feature at a time
- Create new types of plots on the fly
- Cure against immobility
- Developers can easily develop new layers without affecting other layers

Lets dive into the syntax

Anatomy of the plot



1. **Data** – the dataset name that you want to present
2. **Aesthetics** – the variables you want to use in the plot
3. **Geom** – the type of plot you want do
4. **Themes** – control presentation of non-data elements
5. **Stat** – statistical transformations
6. **Scale** – define parameters of the plot (exp: x-axis, color, etc.)

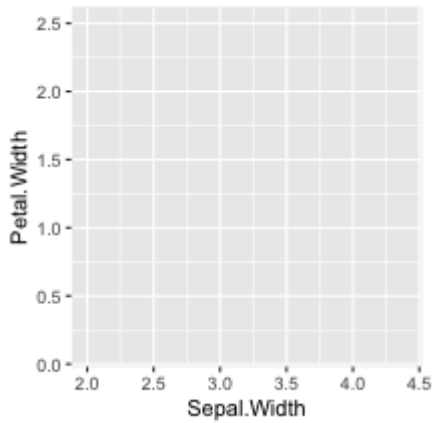
*The green variables are mandatory to create a plot

1. Data

In `ggplot()` you always have to specify what data you are using for the plot.

Thus, you always work with data.frames/tibbles

```
ggplot(data = iris,aes(x = Sepal.Width,y = Petal.Width))
```



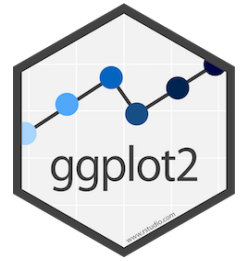
*R function are placement based which means it anticipates the parameters type based on location. So you don't have to write "data", "x", "y" if you place them in the right place

2. Aesthetic mapping

In `ggplot()` land it means "something you can see" and it includes:

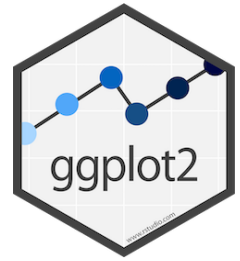
- Position (x and y)
- Color (a name of a color or a vector that will be used for color)
- Fill (a name of a color or a vector that will be used for color)
- Shape (a name of a shape or a vector that will be used for shape)
- Line type
- Size

3. Geometric objects aka `geom`



- A geom can only display certain aesthetics
- A plot must have at least one geom but there is no upper limit

Lets look at some examples using **PanTHERIA** dataset



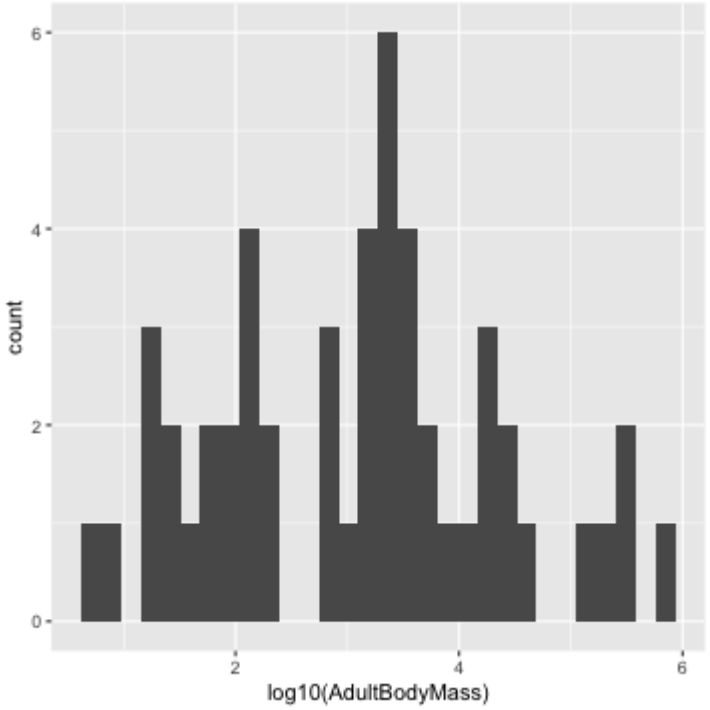
Load the data

```
pantheria<- read_csv("data/sub_PanTHERIA.csv") %>%
  na.omit() %>% mutate(HabitatBreadth = as.character(HabitatBreadth))
pantheria
```

```
## # A tibble: 51 x 18
##    Order Family Genus Species Binomial ActivityCycle AdultBodyMass GestationLen
##    <chr> <chr>  <chr> <chr>   <chr>    <chr>                <dbl>        <dbl>
##  1 Sori… Soric… Sorex minutus Sorex m… both                  4.32         23.8
##  2 Sori… Soric… Sorex corona… Sorex c… both                  9.32         24.0
##  3 Sori… Soric… Neom… fodiens Neomys … both                 15.3          21.4
##  4 Rode… Crice… Pero… leucop… Peromys… nocturnal            18.1          23.2
##  5 Sori… Soric… Blar… brevic… Blarina… nocturnal            18.6          20.6
##  6 Rode… Crice… Micr… pineto… Microtu… both                 26.0          24.0
##  7 Dasy… Dasyu… Ante… stuart… Antechi… both                 29.0          28.2
##  8 Rode… Bathy… Hete… glaber  Heteroc… both                 39.4          70.5
```

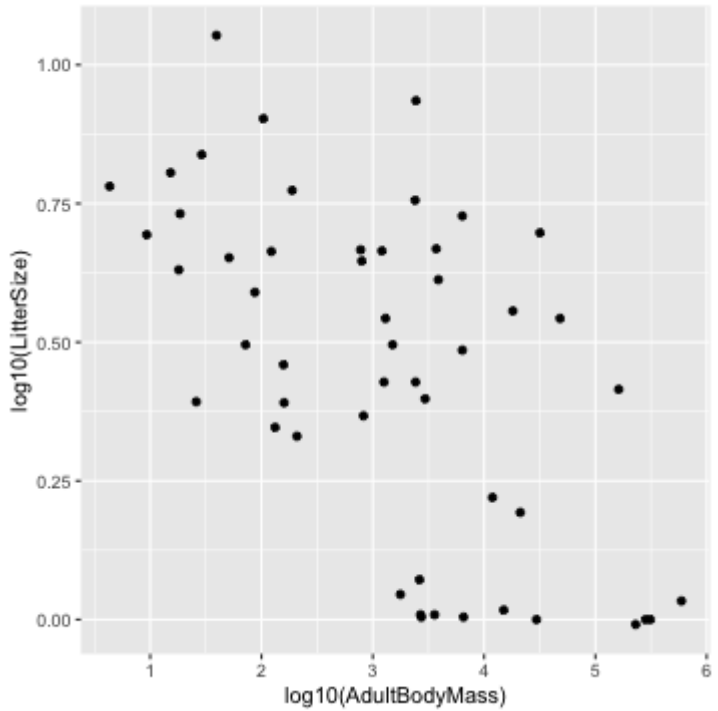
geom_histogram()


```
ggplot(pantheria,aes(x = log10(AdultBodyMass)))+
  geom_histogram()
```



geom_point()


```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize)))+
  geom_point()
```



Lets practice!

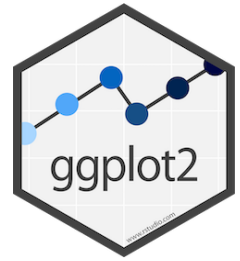
First plot:

Plot a histogram for age at first birth and litter per year

Second plot:

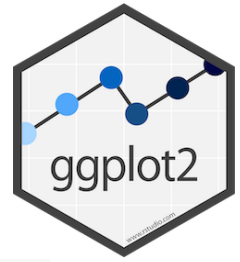
Plot a scatter plot for litter per year as a function of gestation length

4. Statistical Transformations



- Each geom has a default statistic, but these can be changed
- plots, such as boxplots, barplot, prediction lines etc. require statistical transformations

Example



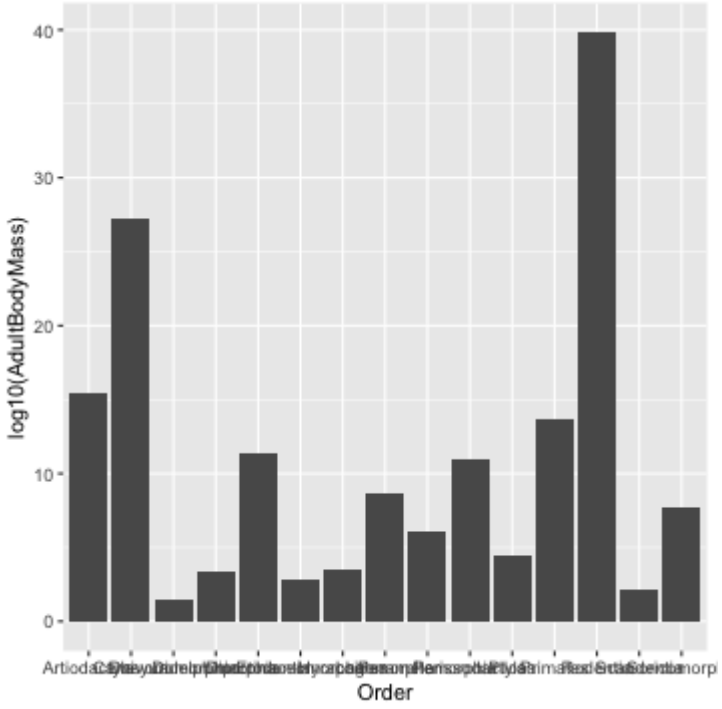

```
ggplot(pantheria,aes(Order,log10(AdultBodyMass)))+
  geom_bar()
```

```
## Error: stat_count() can only have an x or y aesthetic.
```

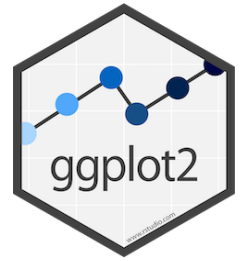
- The default statistics for `geom_bar()` is a histogram count.

The fix


```
ggplot(pantheria,aes(Order,log10(AdultBodyMass)))+
  geom_bar(stat = "identity")
```

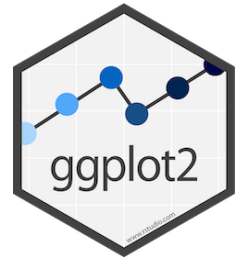


5. Scales



- Control mapping from data to aesthetic attributes
- One scale per aesthetic
- general syntax: `scale_<aesthetic>_<type>`

Examples



Plotting using default colors

Plotting the litter size as a function of body mass grouped for activity cycle

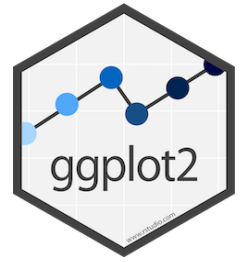
```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle))+
  geom_point()
```

Plotting with `scale` option

Scale color using `RColorBrewer`

```
library(RColorBrewer)
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")
```

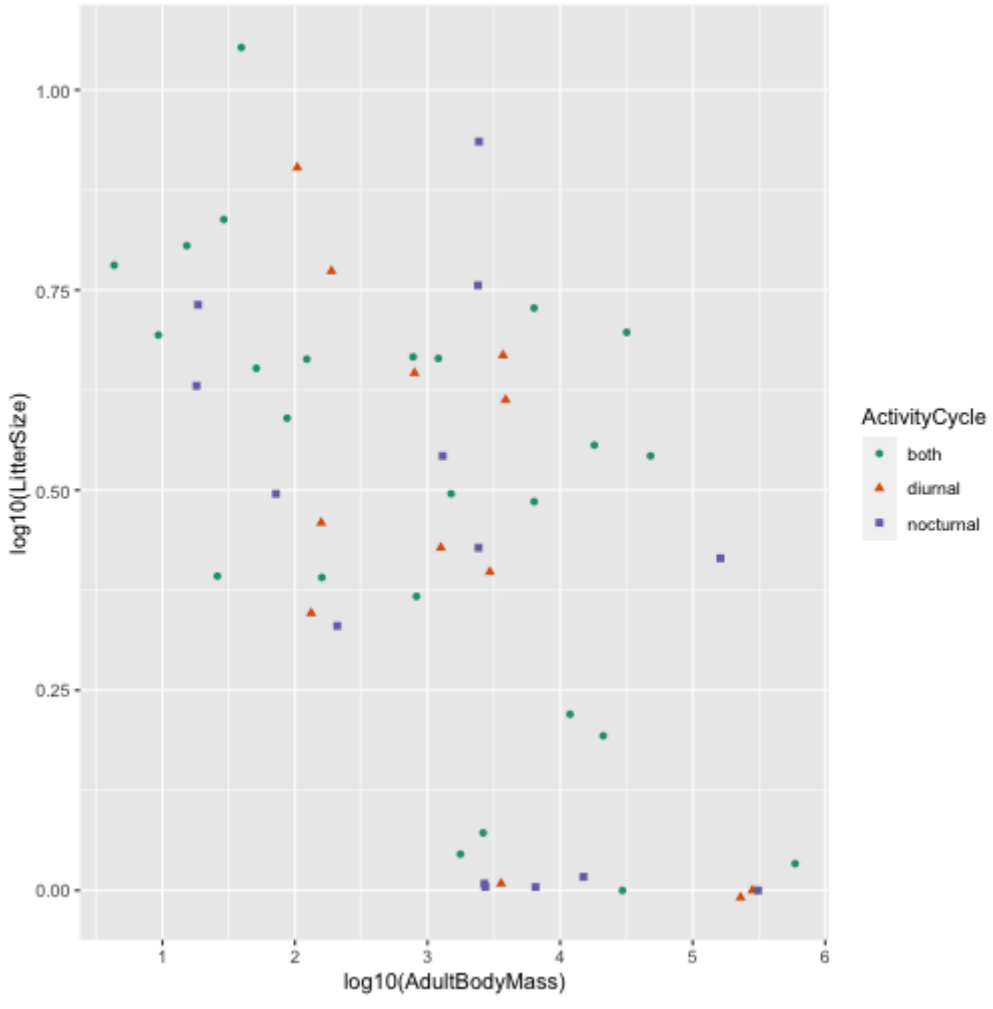
Scaling a parameter that is not defined does nothing. Example with trying to scale the shape




```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape(solid = F)
```

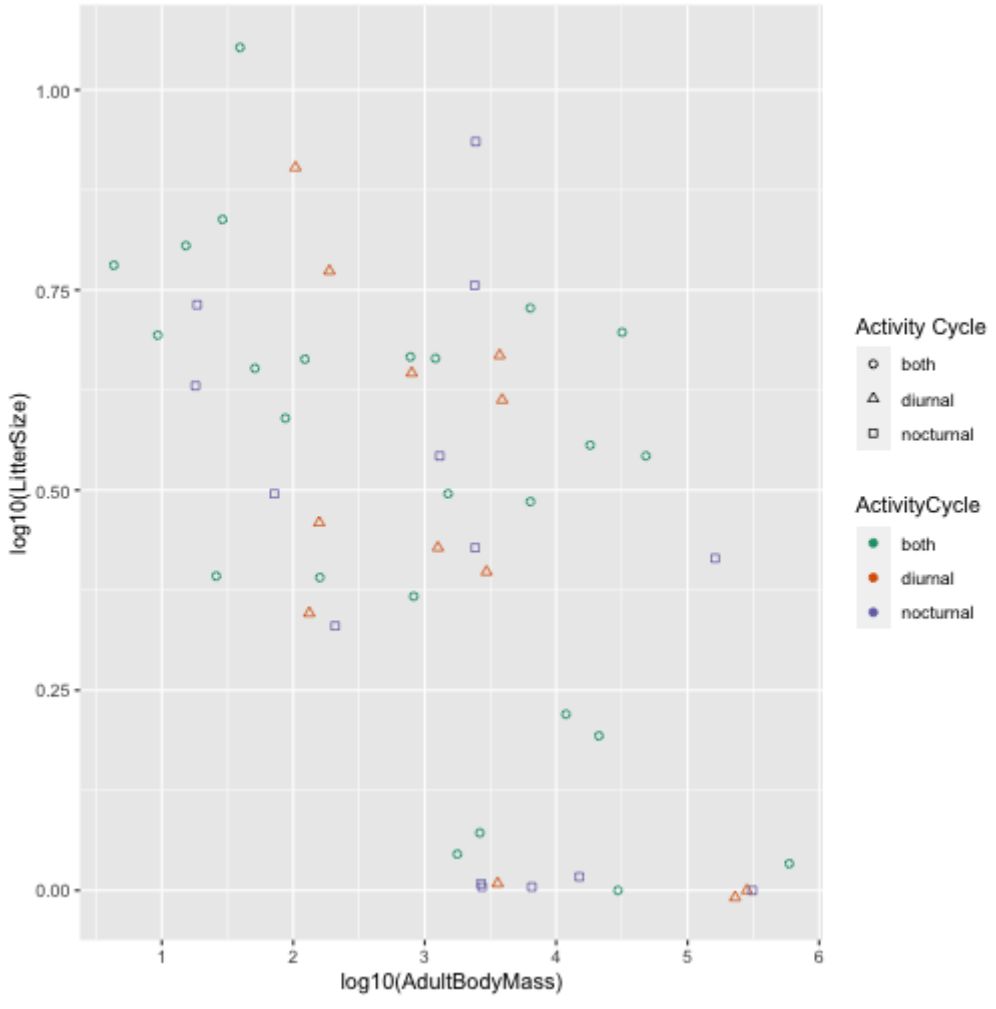
So how to add shape?


```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")
```


1.00
0.75
0.50
0.25
0.00
log10(LitterSize)
1
2
3
4
5
6
log10(AdultBodyMass)
ActivityCycle
both
diurnal
nocturnal

Scale shape parameters:

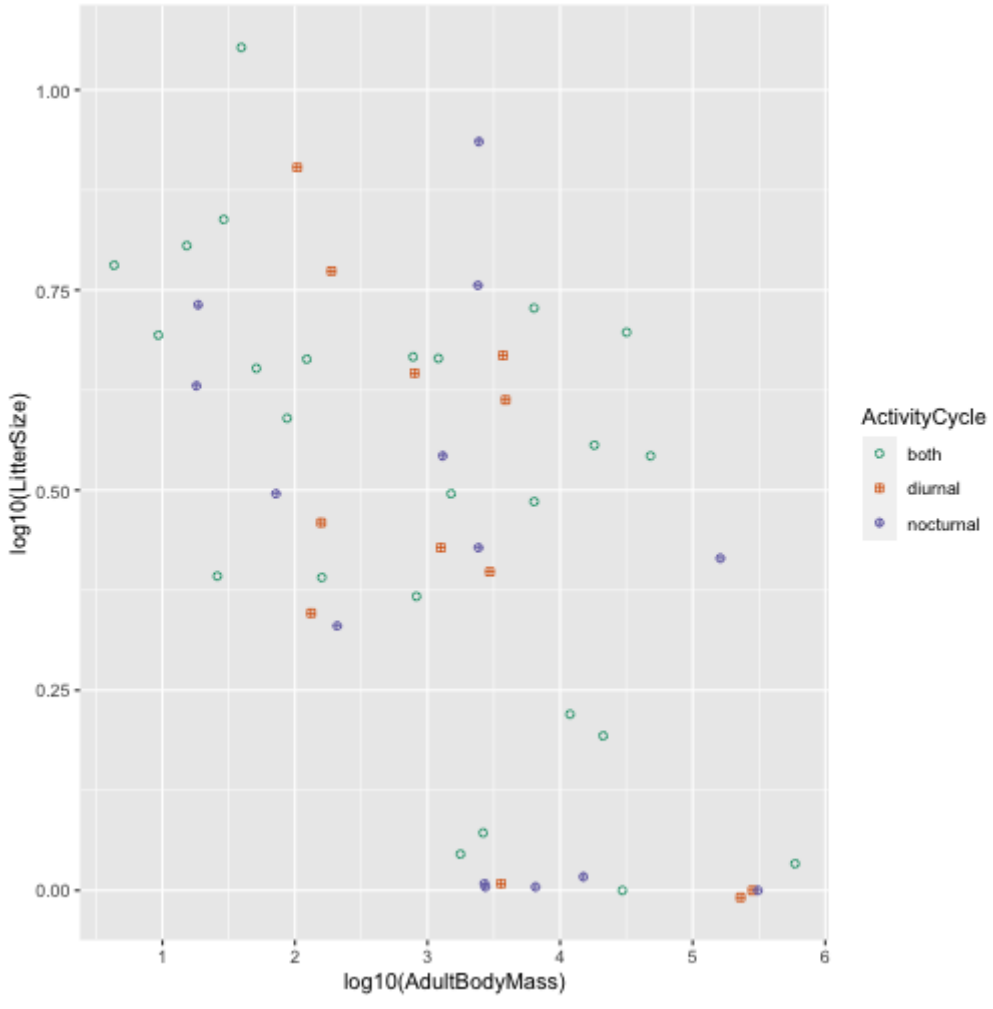

```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape(solid = F,name = "Activity Cycle")
```


1.00
0.75
0.50
0.25
0.00
log10(LitterSize)
1
2
3
4
5
6
log10(AdultBodyMass)
Activity Cycle
both
diurnal
nocturnal
ActivityCycle
both
diurnal
nocturnal

Change the shapes manually

Change the shapes manually


```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape_manual(values = c(1,12,10))
```

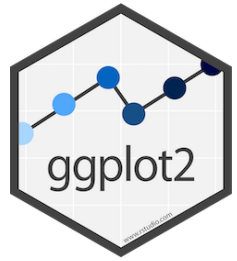

1.00
0.75
0.50
0.25
0.00
log10(LitterSize)
1
2
3
4
5
6
log10(AdultBodyMass)
ActivityCycle
both
diurnal
nocturnal

Lets practice!

Plot the Home range as a function of body mass, with diet breadth as the color and activity cycle as the shape.

Assign the shapes manually and use the Set2 palette

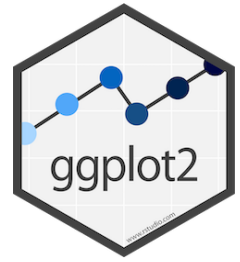
6. Faceting



- Lay out multiple plots on a page
- Plot subsets into different panels

faceting exampels

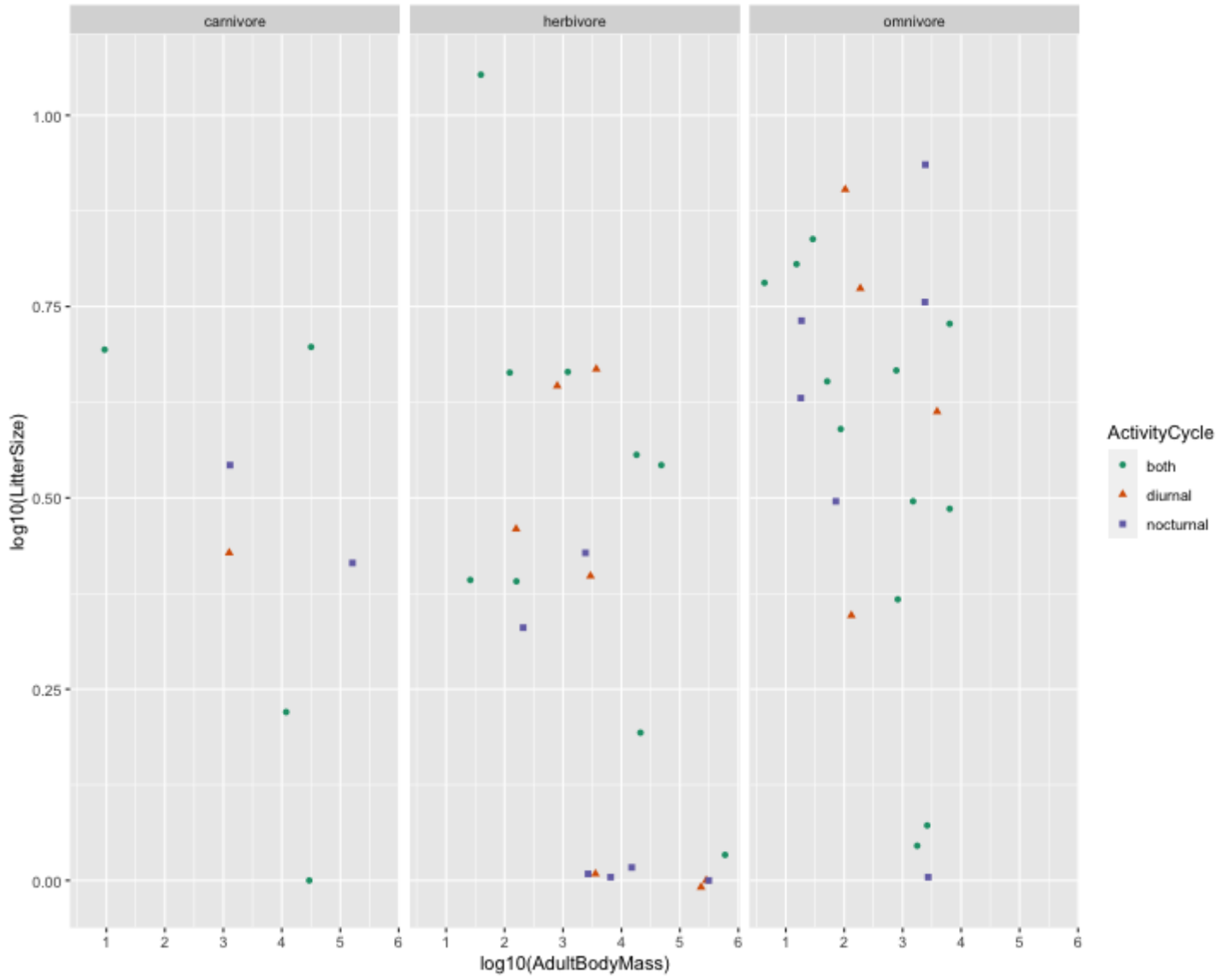
Example



Plot the litter size as a function of body mass for each trophic level separately while keeping the shape-color scaling for activity cycle

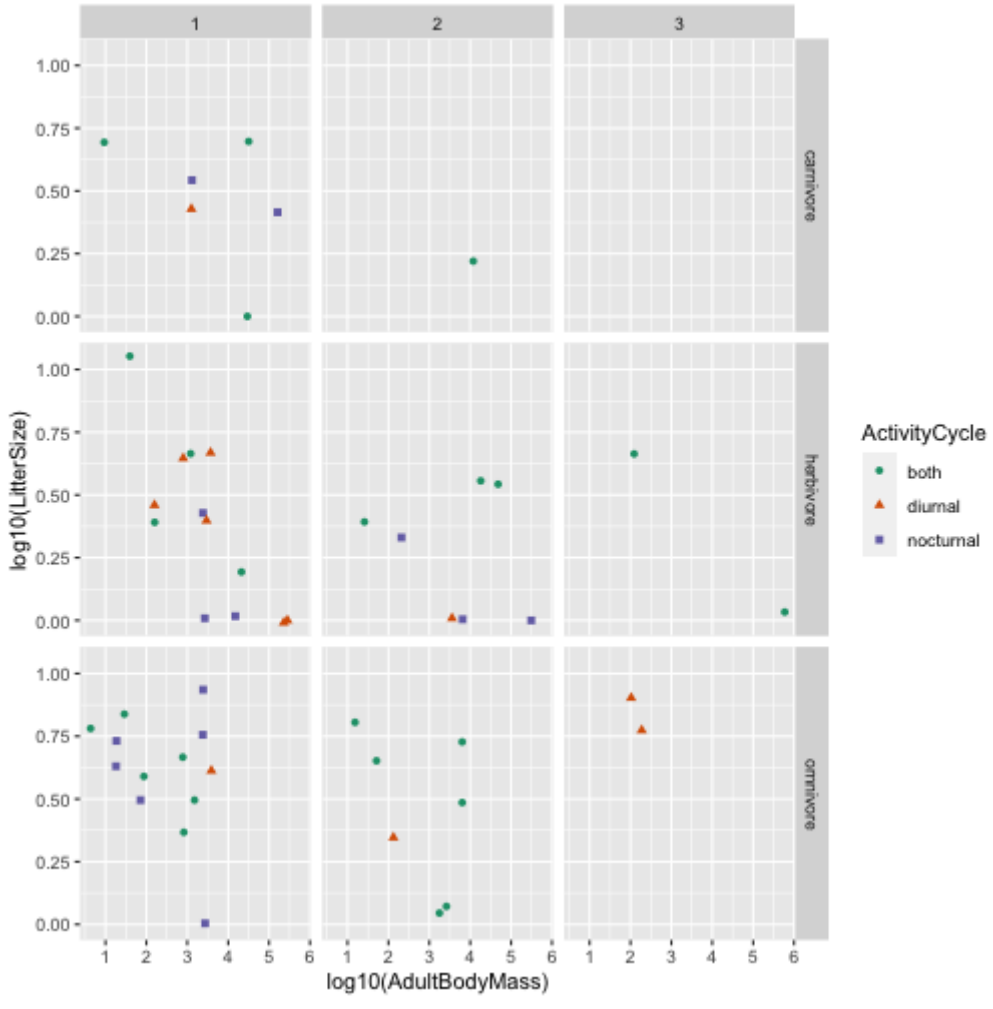
`facet_wrap()`: define subsets as the levels of a single grouping variable

```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape()+
  facet_wrap(TrophicLevel~.)
```


carnivore
herbivore
omnivore
log10(LitterSize)
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
log10(AdultBodyMass)
ActivityCycle
both
diurnal
nocturnal

`facet_grid()`: define subsets as the crossing of two grouping variables

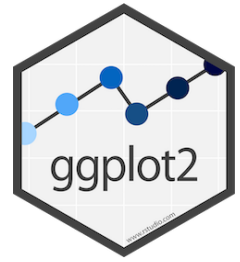

```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape()+
  facet_grid(TrophicLevel~HabitatBreadth)
```


1
2
3
carnivore
herbivore
omnivore
log10(LitterSize)
log10(AdultBodyMass)
1.00
0.75
0.50
0.25
0.00
1
2
3
4
5
6
ActivityCycle
both
diurnal
nocturnal

Lets practice!

Plot the Home range as a function of body mass, with a single plot for each habitat breadth

Scale axis in facet



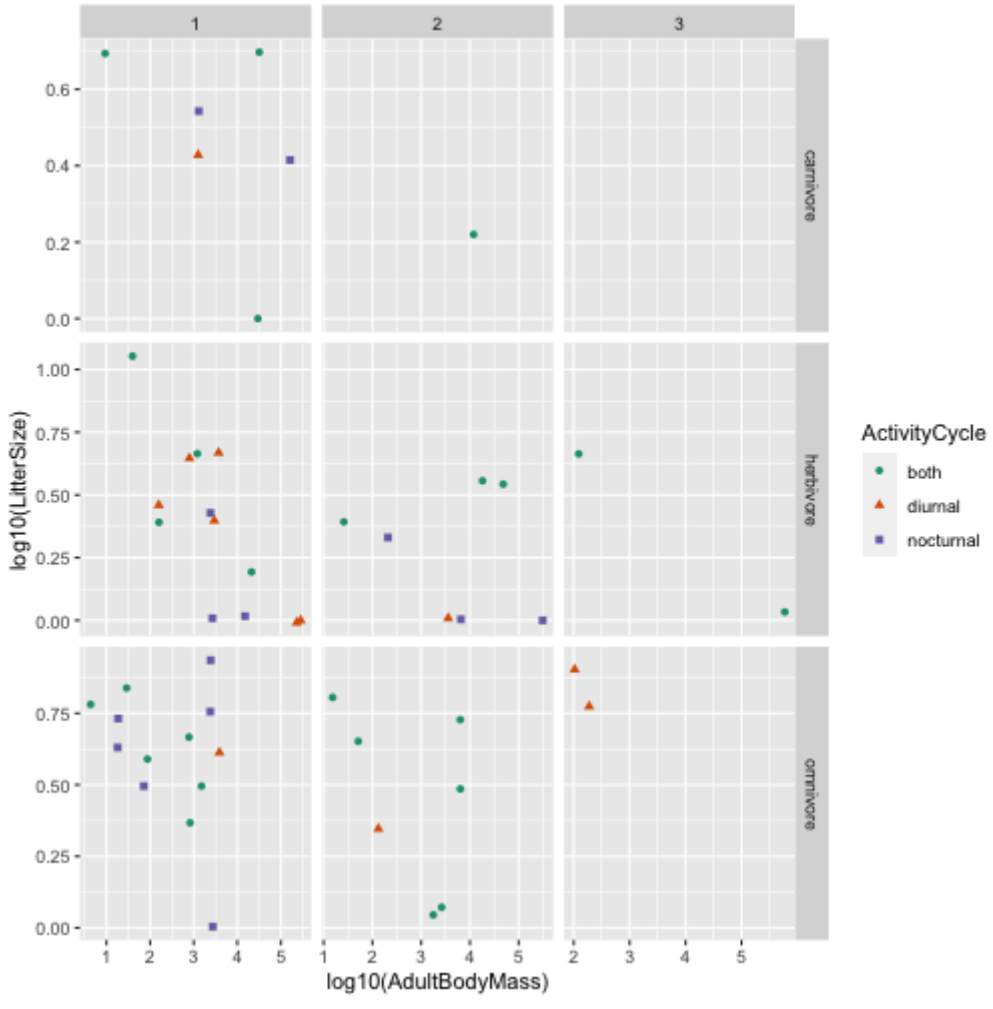
When you have different values for the x or y axis in your data you need to scale it when faceting

scales value	free
fixed	-
free	x, y
free_x	x
free_y	y

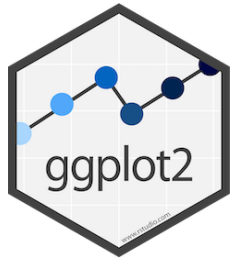
- The axis are fixed by default

Example


```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape()+
  facet_grid(TrophicLevel~HabitatBreadth,scales = "free")
```

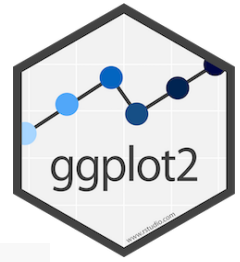

1
2
3
carnivore
herbivore
omnivore
log10(LitterSize)
log10(AdultBodyMass)
ActivityCycle
both
diurnal
nocturnal

7. Themes

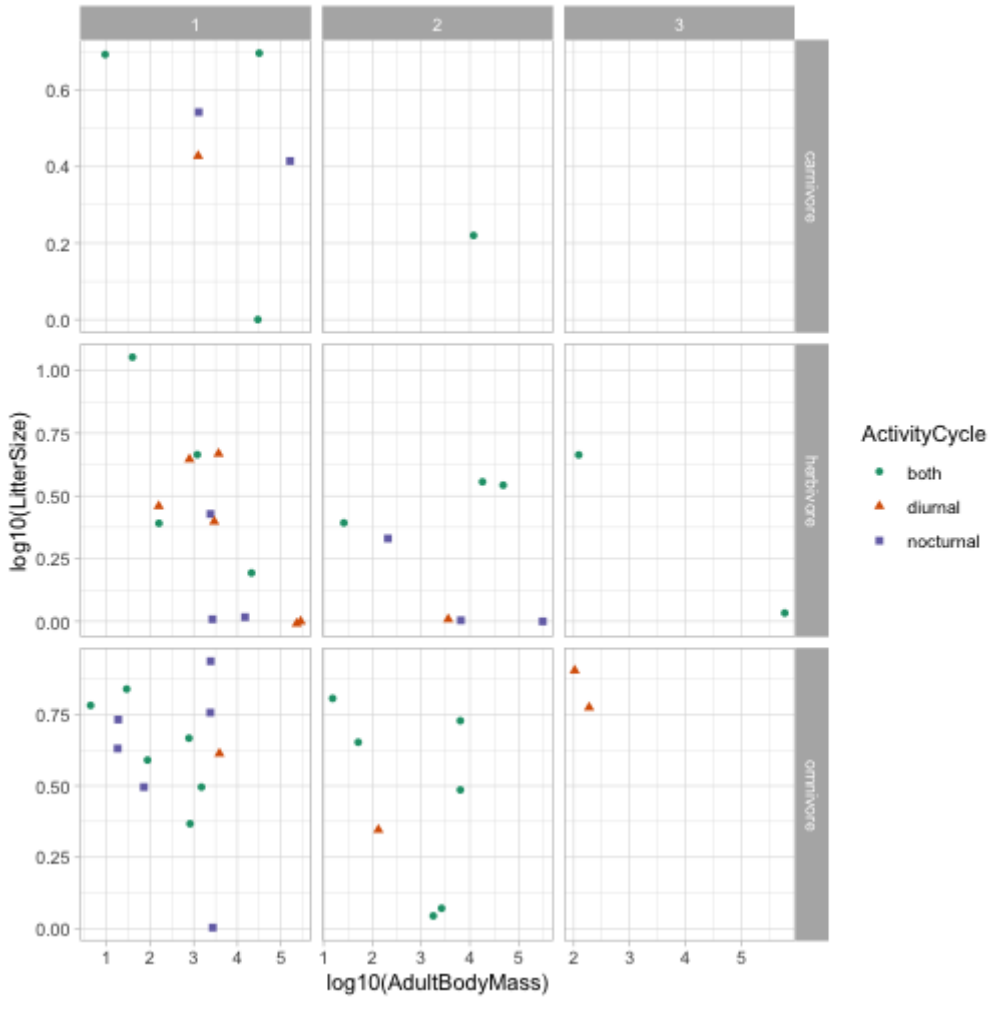


- handles non-data plot elements such as:
  - Axis labels
  - Plot background
  - Facet label background
  - Legend appearance

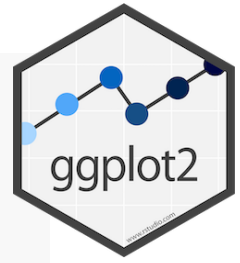
Example

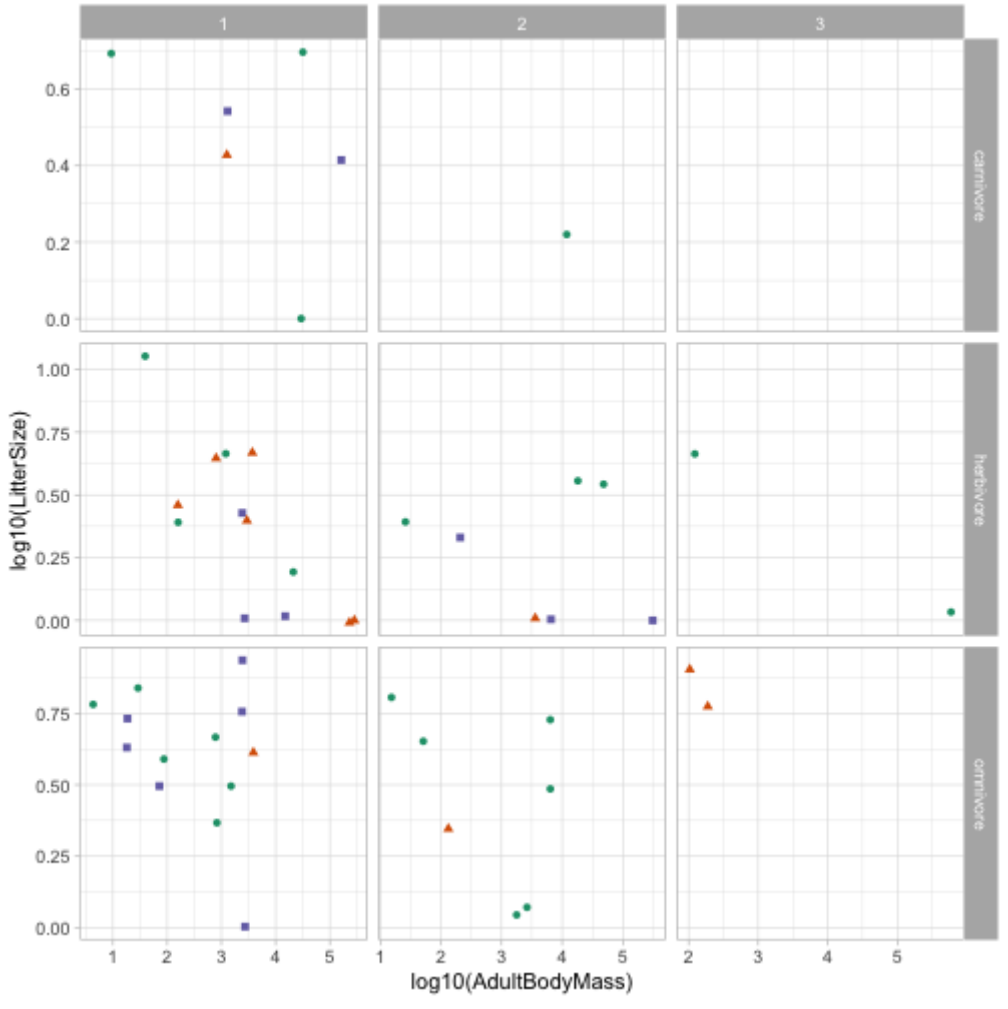



```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape()+
  facet_grid(TrophicLevel~HabitatBreadth,scales = "free") +
  theme_light()
```


1
2
3
carnivore
herbivore
omnivore
log10(LitterSize)
log10(AdultBodyMass)
ActivityCycle
both
diurnal
nocturnal

```
ggplot(pantheria,aes(log10(AdultBodyMass),log10(LitterSize),
                     color = ActivityCycle,shape = ActivityCycle))+
  geom_point()+
  scale_color_brewer(palette = "Dark2")+
  scale_shape()+
  facet_grid(TrophicLevel~HabitatBreadth,scales = "free") +
  theme_light()+
  theme(legend.position = "none")
```



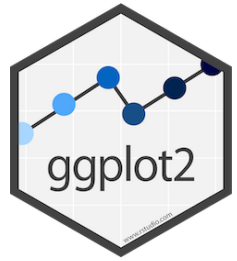

1
2
3
carnivore
herbivore
omnivore
log10(LitterSize)
log10(AdultBodyMass)
0.0
0.2
0.4
0.6
0.00
0.25
0.50
0.75
1.00
1
2
3
4
5

Lets practice!

Use the plot from the previous practice and remove the grey background using the `theme()` function

Hint: if you're having trouble try to use Google ☺

8. Position adjustments

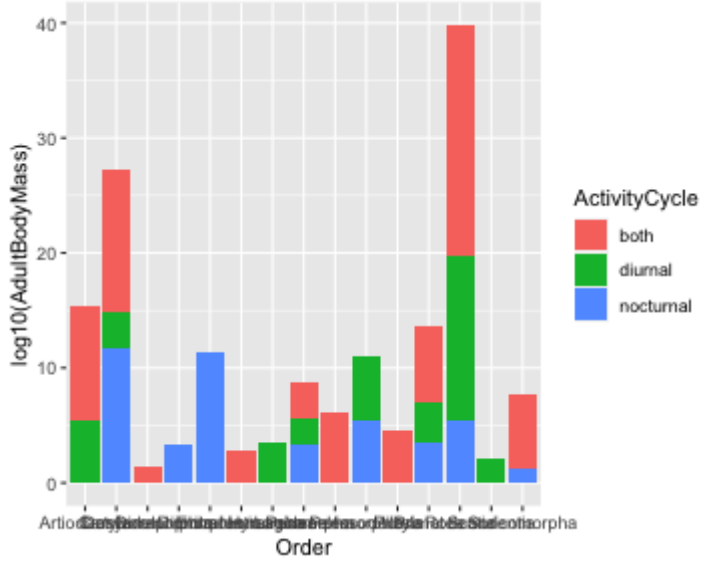


- Tweak positioning of geometric objects
- Avoid overlaps

Example

If we add fill to the barplot from before without position adjustment we see that the parameters are stacked one on top of each other

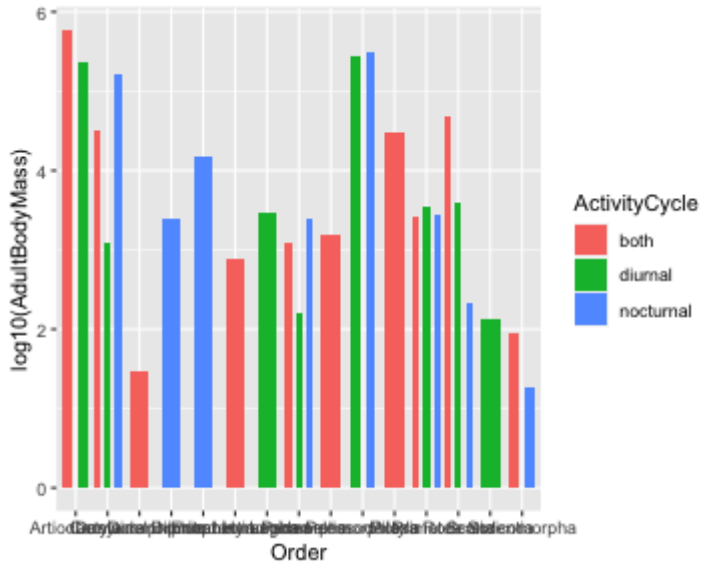
```
ggplot(pantheria,aes(Order,log10(AdultBodyMass),fill = ActivityCycle))+
  geom_bar(stat = "identity")
```



position_dodge()

**To put them one next to the other

```
ggplot(pantheria,aes(Order,log10(AdultBodyMass),fill = ActivityCycle))+
  geom_bar(stat = "identity",width = 0.6,
           position = position_dodge(width = 1))
```

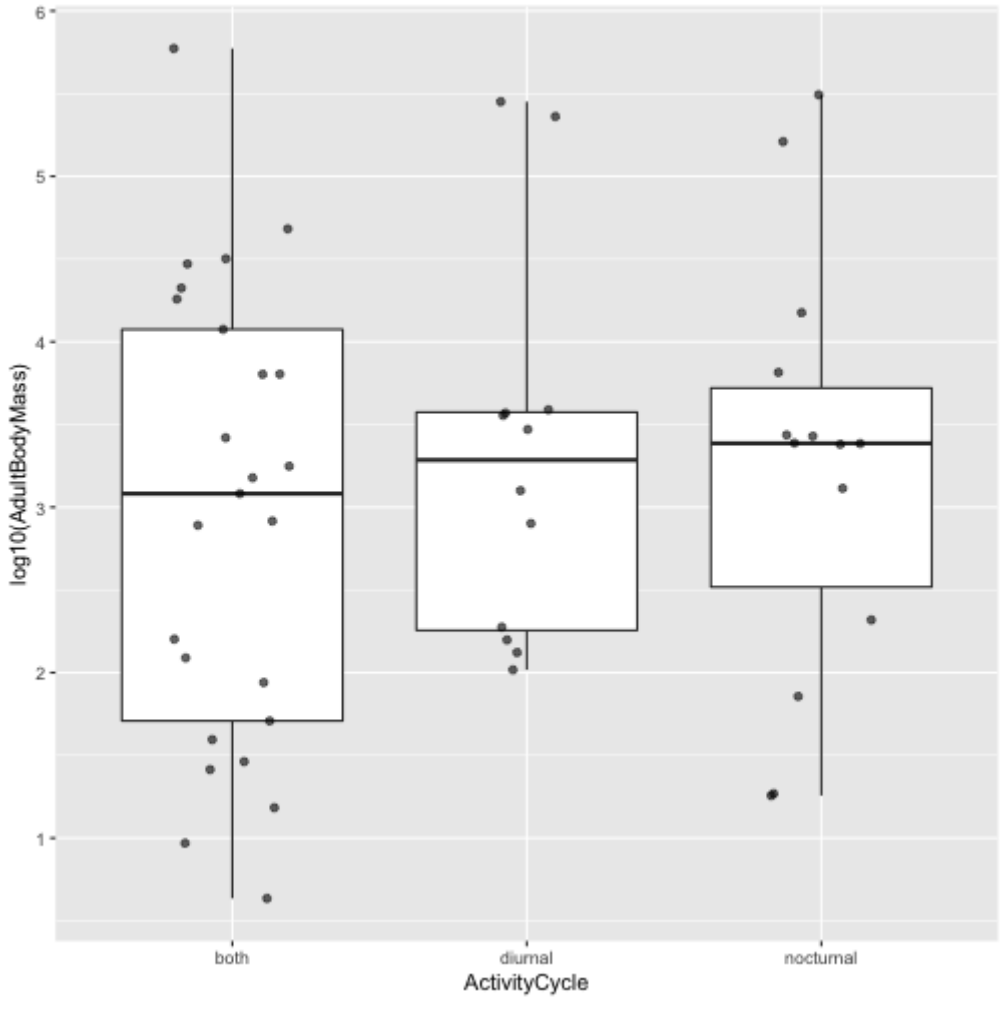


position_jitter()

Avoid over-plotting by jittering points

This is useful for **boxplot** and **violin** plot

```
ggplot(pantheria,aes(ActivityCycle,log10(AdultBodyMass)))+
  geom_boxplot()+
  geom_jitter(alpha = 0.6,width = 0.2)
```


6
5
4
3
2
1
log10(AdultBodyMass)
both
diurnal
nocturnal
ActivityCycle

Summary

- Allows creating high quality plots
- Many options
- Many blogs and webpages explaining how to do different plots in ggplot
- Disadvantage – takes time to learn the grammar

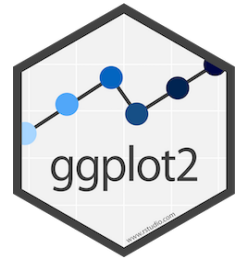
The book:

https://ggplot2-book.org/

Useful links

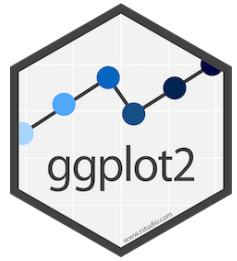
- https://bradleyboehmke.github.io/tutorials/barchart
- http://www.sthda.com/english/wiki/ggplot2-barplots-quick-start-guide-r-software-and-data-visualization
- color palette generator: https://coolors.co/

Swirl



- A package to learn R in R.
- Saves your progress as long as you use the same name
- If you make a mistake you get a hint
- To install: `install.packages("swirl")`
- To load `library(swirl)`

Homework

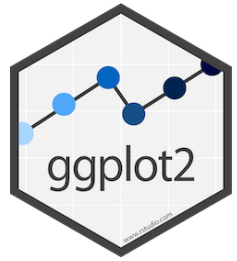


- After installing swirl download homework using the following command:

```
install_course_github("marianovosolov","exercise_REcoStat2020",multi = F)
```

- There will also be the homework for the tidyverse session ignore that for now
- Start the swirl working environment with `swirl()` and follow the istructions
- Due date: 04/05/2020 23:59

Tips to navigate within `swirl`



| When you are at the R prompt (>):

| -- Typing skip() allows you to skip the current question.

| -- Typing play() lets you experiment with R on your own; swirl will ignore what you do...

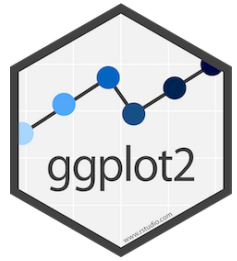
| -- UNTIL you type nxt() which will regain swirl's attention.

| -- Typing bye() causes swirl to exit. Your progress will be saved.

| -- Typing main() returns you to swirl's main menu.

| -- Typing info() displays these options again.

Good luck!!



Use #ggplot slack channel for any questions and feedback