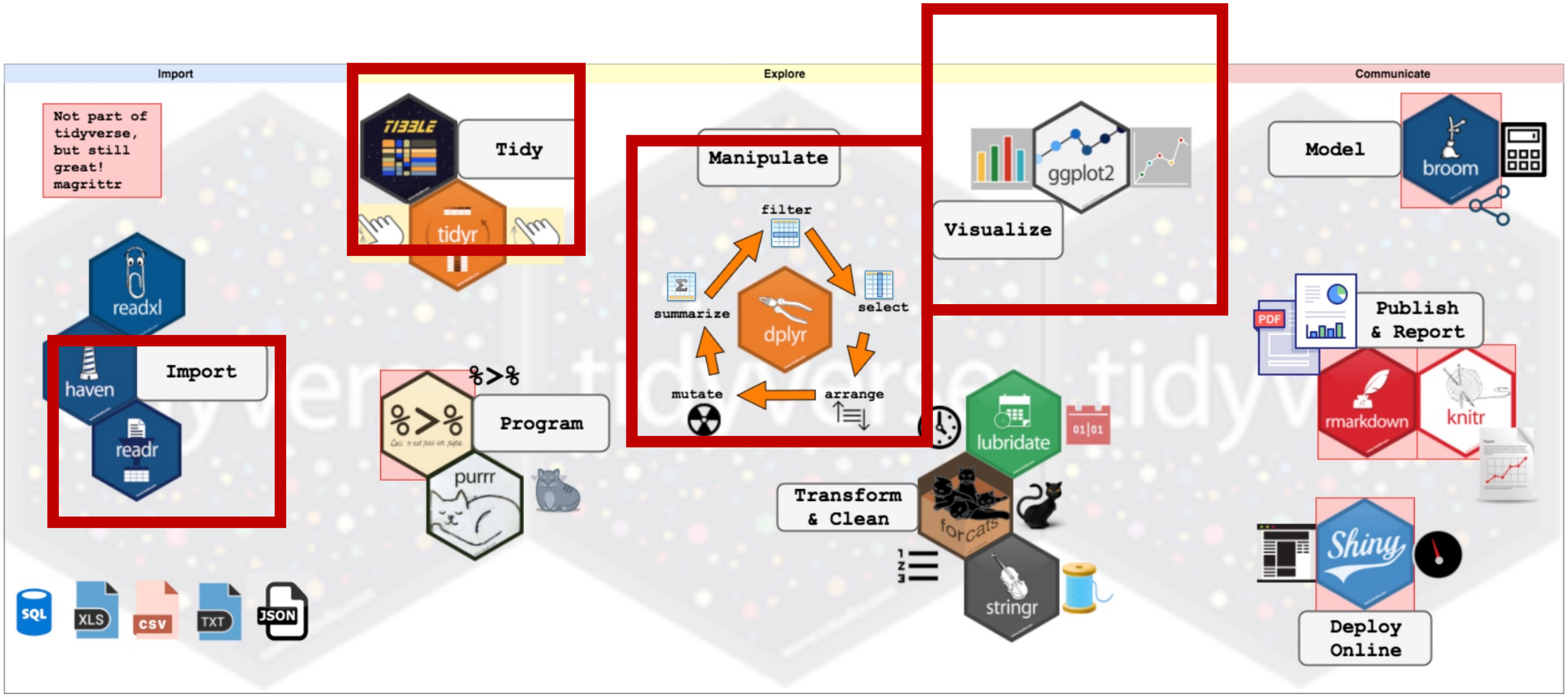
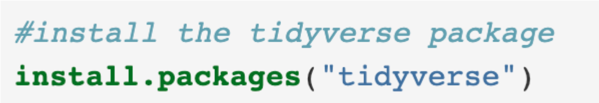
Tidyverse



Read file -> Tidy up the file -> Transform -> visualize

Installing and Loading Tidyverse

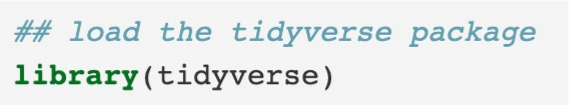
Installing



```r
#install the tidyverse package
install.packages("tidyverse")
```

Do this once

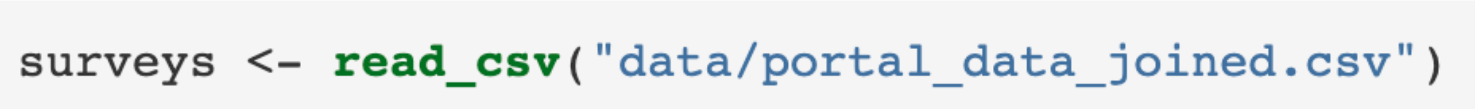
Loading



```r
## load the tidyverse package
library(tidyverse)
```

Do at the top of every* script

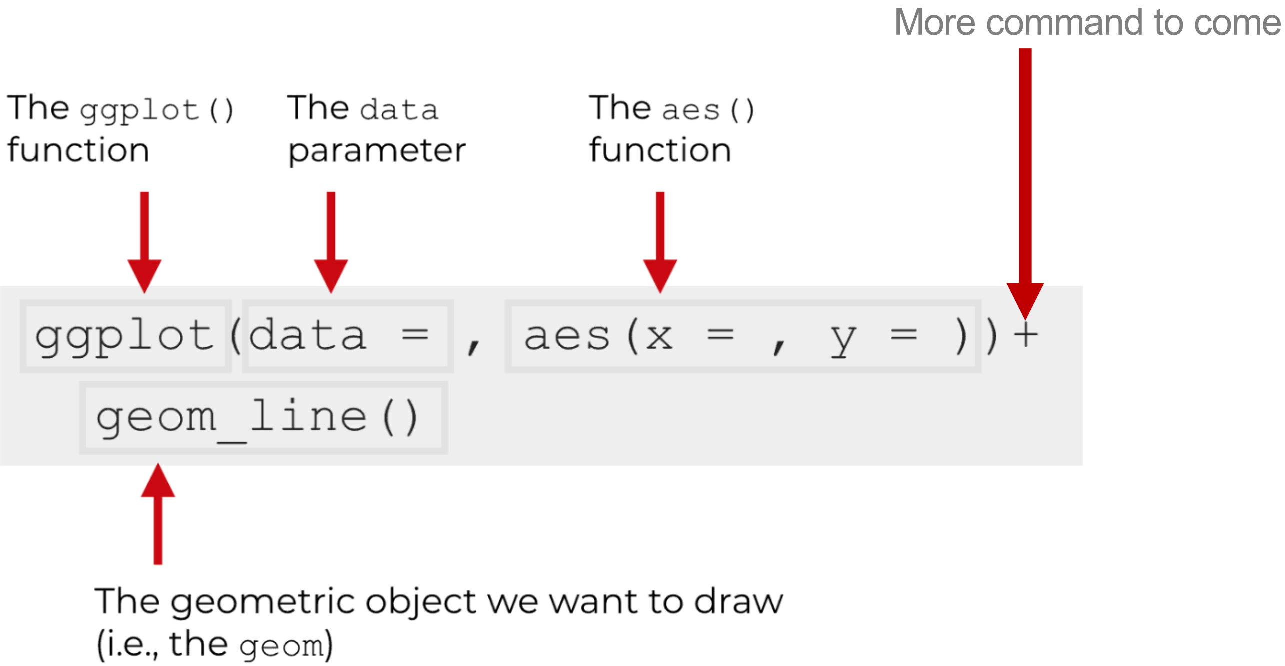
Importing the data with dyplyr



```
surveys <- read_csv("data/portal_data_joined.csv")
```

Notice the underscore!

The basics of ggplot



More command to come

The `ggplot()` function

The `data` parameter

The `aes()` function

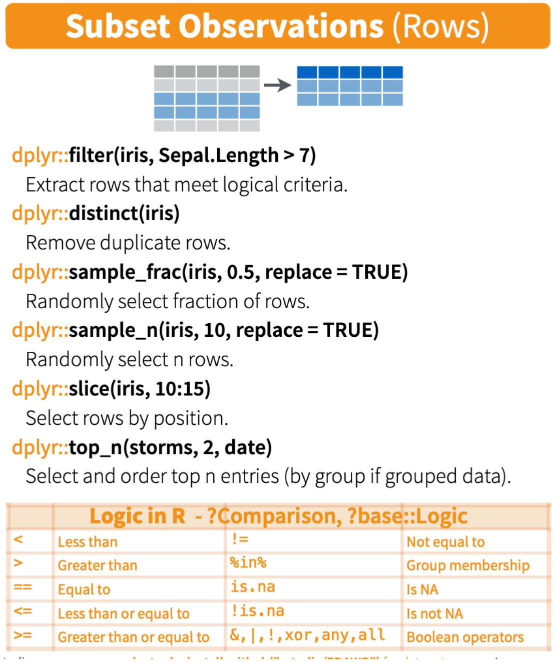
```
ggplot(data = , aes(x = , y = ))+
  geom_line()
```

The geometric object we want to draw
(i.e., the `geom`)

dplyr

- **Select and filter**
- **Mutate**
- **Summarise**
- **Group by**

Subsetting a dplyr data frame



Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

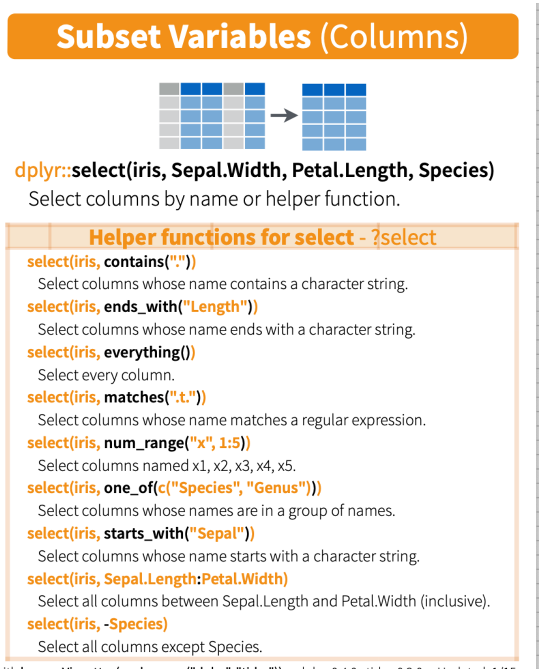
dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators



Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

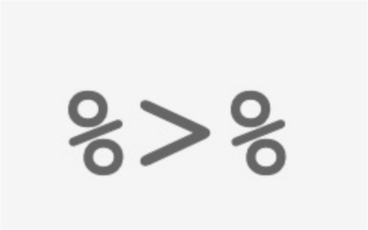
select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, **Sepal.Length:Petal.Width**)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, **-Species**)
Select all columns except Species.

Base R - dataFrame[ROW , COLUMN]

Pipes



```
%>%
```

Use **pipe** for passing **data** to a new command

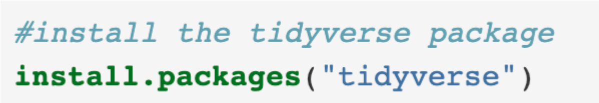
Often confused with the **ggplot +** which is used to add more lines to a plot

What we have learned this morning

- Installing tidyverse once – load every time
- Importing tables with dplyr
- Basic ggplot syntax
- select and filter
- Pipes. %>%

Installing and Loading Tidyverse

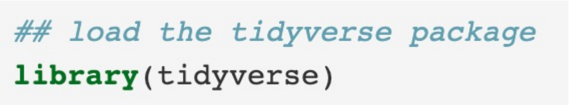
Installing



```r
#install the tidyverse package
install.packages("tidyverse")
```

Do this once

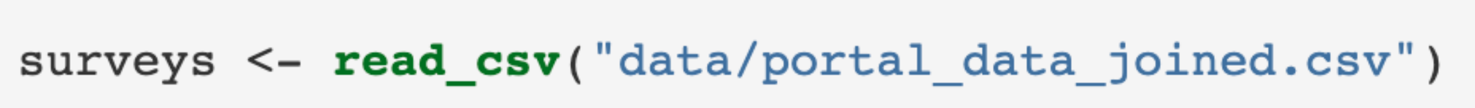
Loading



```r
## load the tidyverse package
library(tidyverse)
```

Do at the top of every* script

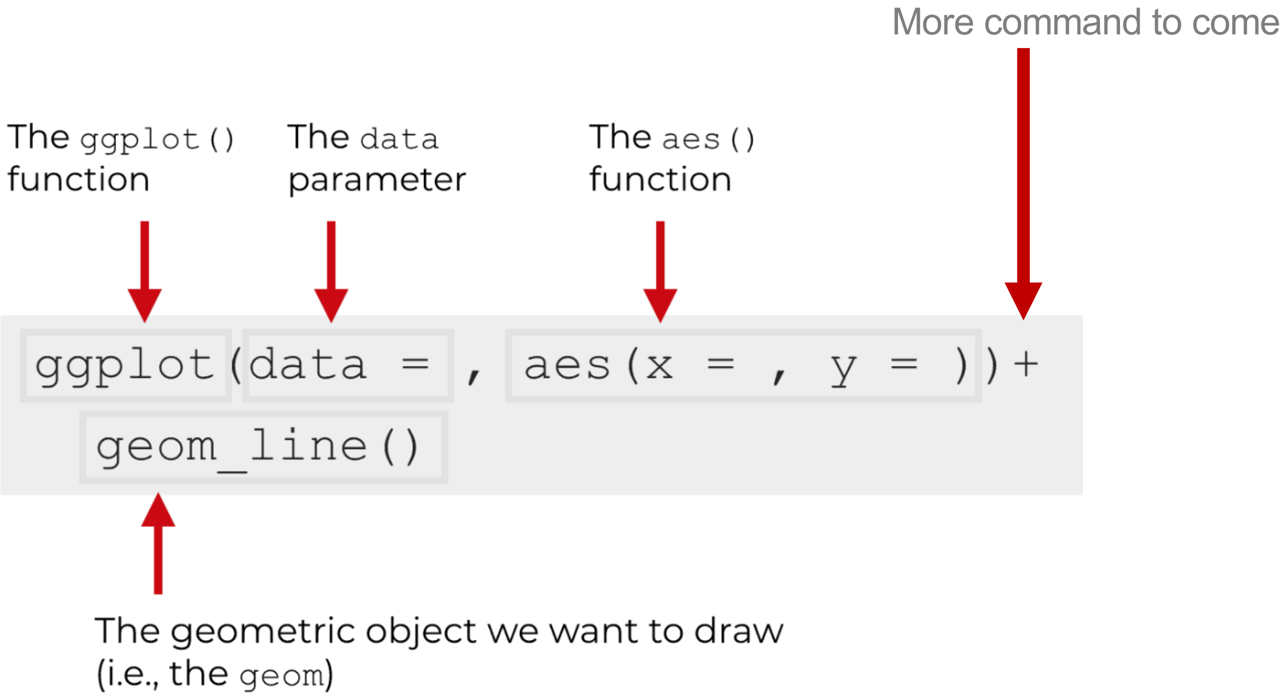
Importing the data with dyplyr



```
surveys <- read_csv("data/portal_data_joined.csv")
```

Notice the underscore!

The basics of ggplot



More command to come

The ggplot() function

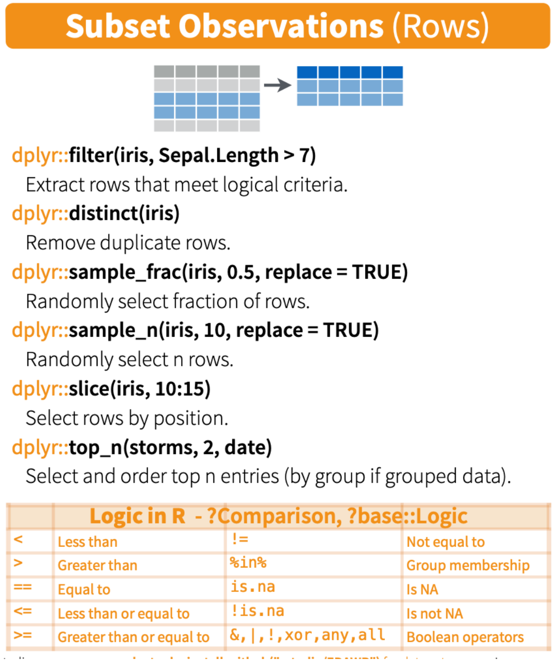
The data parameter

The aes() function

```
ggplot(data = , aes(x = , y = ))+
  geom_line()
```

The geometric object we want to draw (i.e., the geom)

Subsetting a dplyr data frame



Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

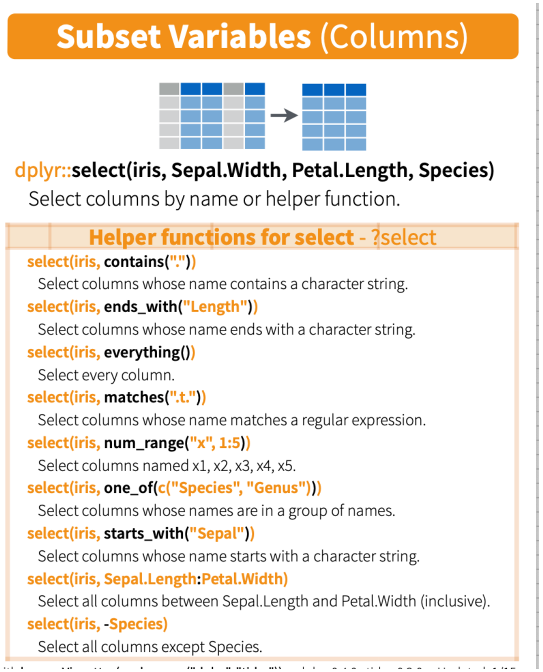
dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators



Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, **Sepal.Length:Petal.Width**)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, **-Species**)
Select all columns except Species.

Base R - dataFrame[ROW , COLUMN]

Pipes

```
%>%
```

Use **pipe** for passing **data** to a new command

Often confused with the **ggplot +** which is used to add more lines to a plot

What we will go though this afternoon

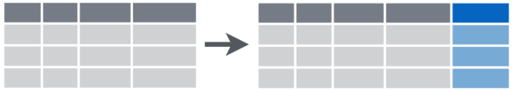
Transforming tables with dplyr

- creating new columns
-
- sorting data tables
- summarizing data tables
- Frequencies
- Grouping data

More ggplot

- Plotting data (lines plots)
- Faceting
- Customising plots
- Exporting tables to file
- Boxplots and histograms

Making new columns – mutate()



dplyr::mutate(iris, sepal = Sepal.Length + Sepal. Width)

Compute and append one or more new columns.

dplyr::mutate_each(iris, funs(min_rank))

Apply window function to each column.

dplyr::transmute(iris, sepal = Sepal.Length + Sepal. Width)

Compute one or more new columns. Drop original columns.

window
function

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

Possible functions include:

dplyr::lead
Copy with values shifted by 1.

dplyr::lag
Copy with values lagged by 1.

dplyr::dense_rank
Ranks with no gaps.

dplyr::min_rank
Ranks. Ties get min rank.

dplyr::percent_rank
Ranks rescaled to [0, 1].

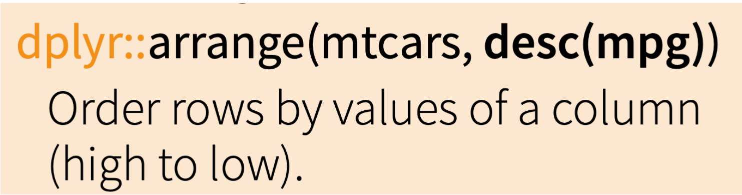
dplyr::row_number
Ranks. Ties got to first value.

dplyr::ntile
Bin vector into n buckets.

dplyr::between
Are values between a and b?

dplyr::cume_dist
Cumulative distribution.

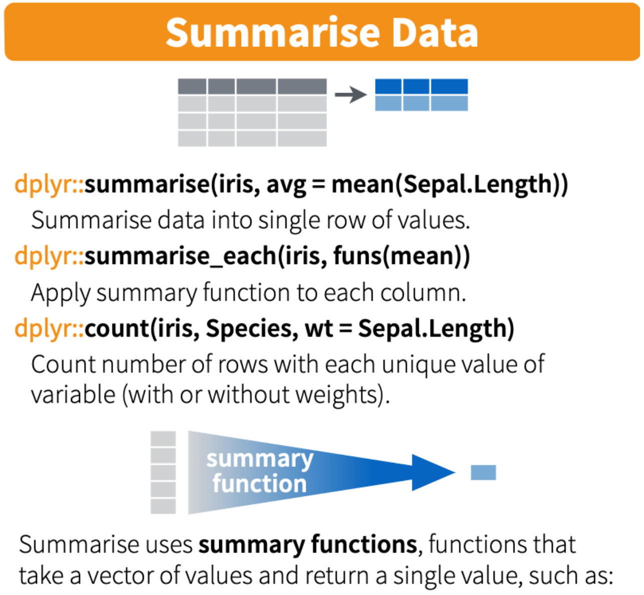
Sorting data – arrange()



```
dplyr::arrange(mtcars, desc(mpg))
```

Order rows by values of a column (high to low).

Summarising data – summarise()



summary
function

Summarise Data

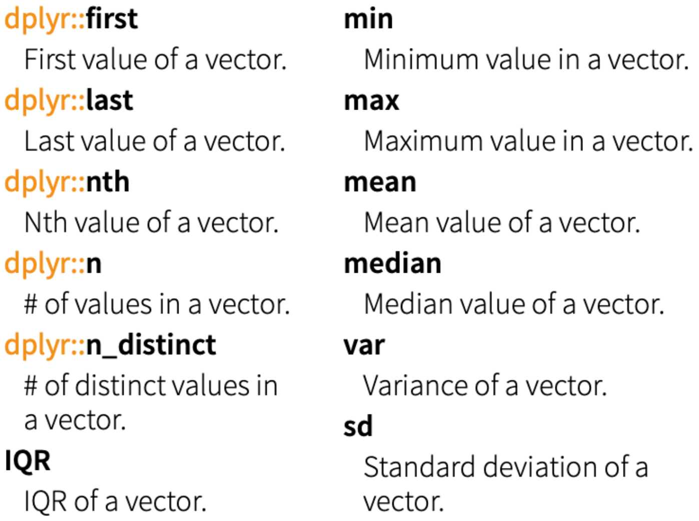
dplyr::summarise(iris, avg = mean(Sepal.Length))
Summarise data into single row of values.

dplyr::summarise_each(iris, funs(mean))
Apply summary function to each column.

dplyr::count(iris, Species, wt = Sepal.Length)
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

Possible functions include:



dplyr::first
First value of a vector.

dplyr::last
Last value of a vector.

dplyr::nth
Nth value of a vector.

dplyr::n
# of values in a vector.

dplyr::n_distinct
# of distinct values in a vector.

IQR
IQR of a vector.

min
Minimum value in a vector.

max
Maximum value in a vector.

mean
Mean value of a vector.

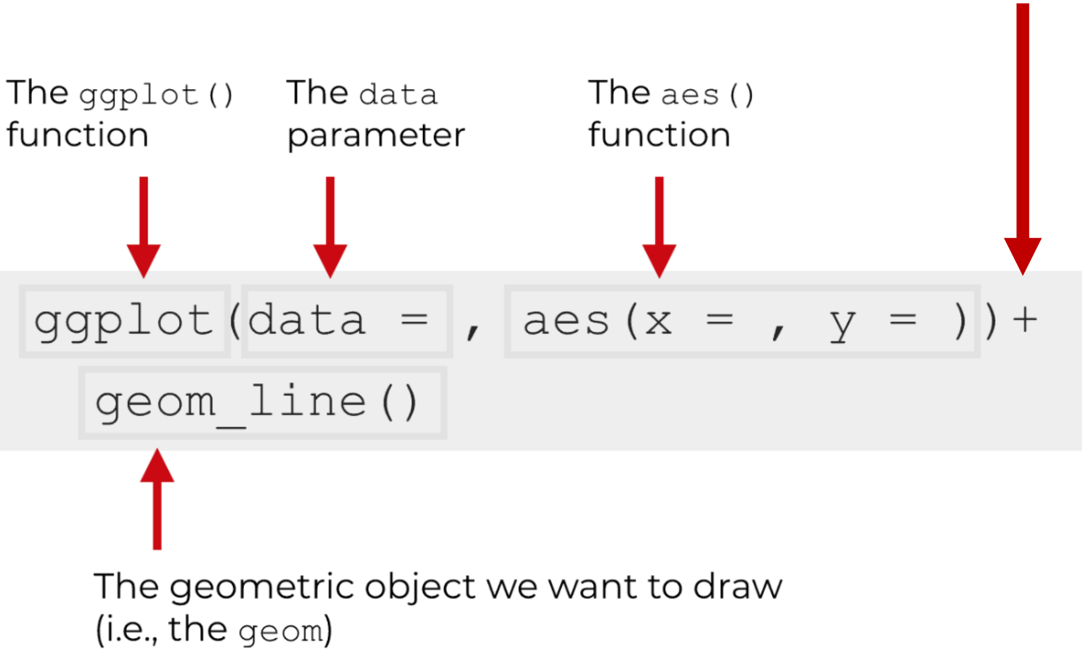
median
Median value of a vector.

var
Variance of a vector.

sd
Standard deviation of a vector.

Plotting time series

More command to come



The ggplot() function

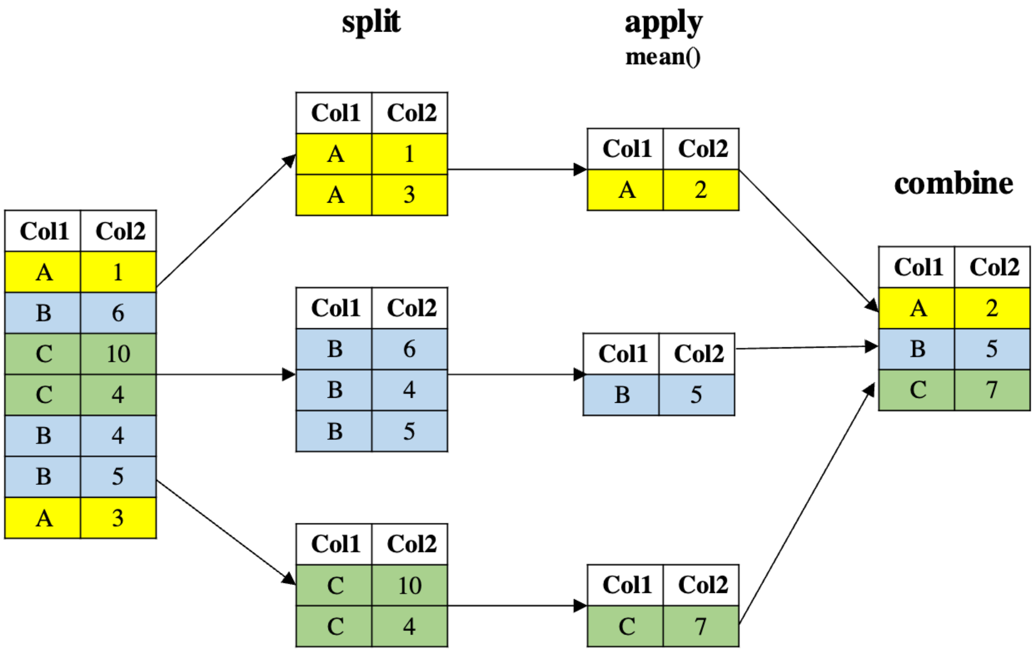
The data parameter

The aes() function

```
ggplot(data = , aes(x = , y = )) +
   geom_line()
```

The geometric object we want to draw
(i.e., the geom)

Performing operations based on a variable column - group _ by()



Col1	Col2
A	1
B	6
C	10
C	4
B	4
B	5
A	3

split

Col1	Col2
A	1
A	3

Col1	Col2
B	6
B	4
B	5

Col1	Col2
C	10
C	4

apply

mean()

Col1	Col2
A	2

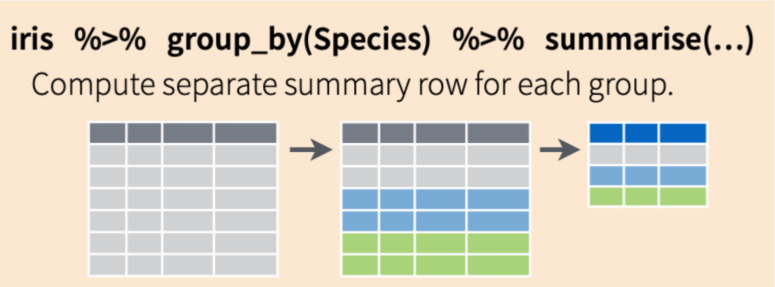
Col1	Col2
B	5

Col1	Col2
C	7

combine

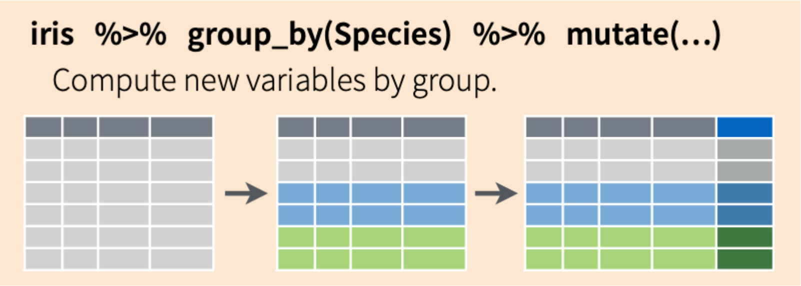
Col1	Col2
A	2
B	5
C	7

Performing operations based on a variable column - group _ by()



```
iris  %>%  group_by(Species)  %>%  summarise(...)
```

Compute separate summary row for each group.

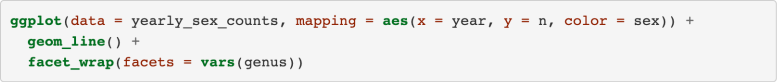


```
iris  %>%  group_by(Species)  %>%  mutate(...)
```

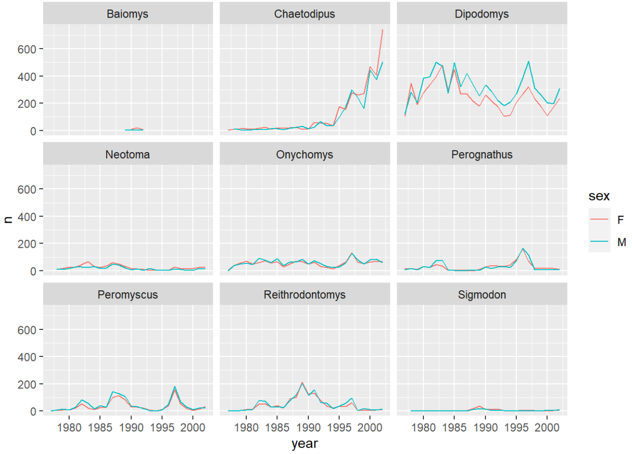
Compute new variables by group.

Facetting – For when you want to plot lots of variables

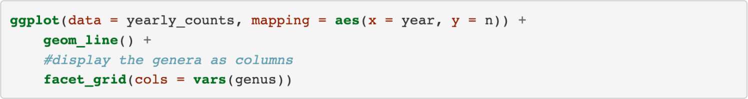
facet_wrap



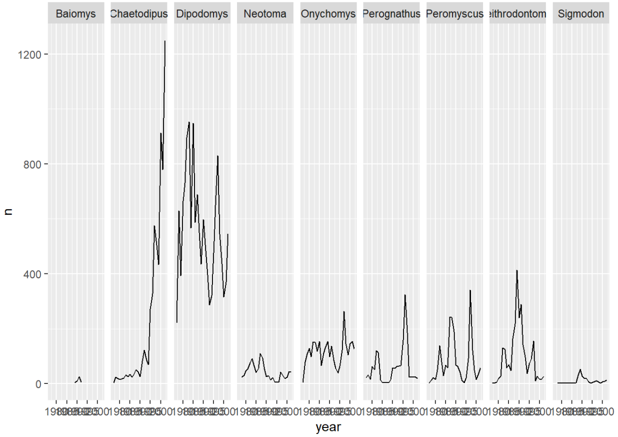
```
ggplot(data = yearly_sex_counts, mapping = aes(x = year, y = n, color = sex)) +
  geom_line() +
  facet_wrap(facets = vars(genus))
```



facet_grid

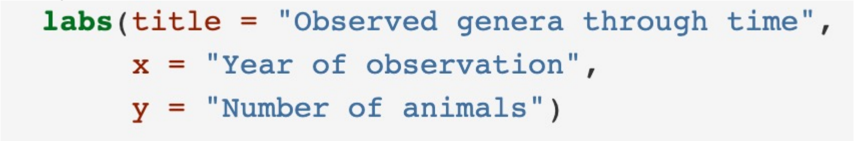


```
ggplot(data = yearly_counts, mapping = aes(x = year, y = n)) +
    geom_line() +
    #display the genera as columns
    facet_grid(cols = vars(genus))
```



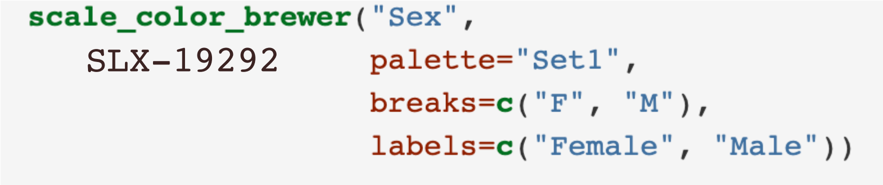
Customisation

Major labels



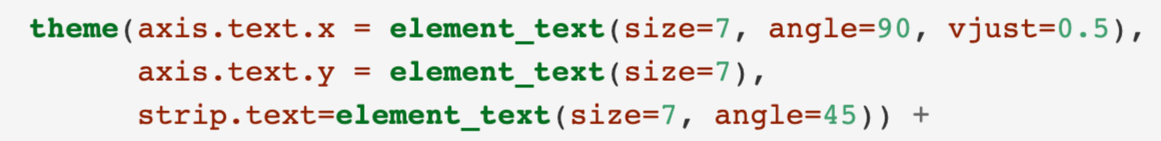
```
labs(title = "Observed genera through time",
     x = "Year of observation",
     y = "Number of animals")
```

Colour and legend



```
scale_color_brewer("Sex",
   SLX-19292     palette="Set1",
                 breaks=c("F", "M"),
                 labels=c("Female", "Male"))
```

Everything else



```
theme(axis.text.x = element_text(size=7, angle=90, vjust=0.5),
      axis.text.y = element_text(size=7),
      strip.text=element_text(size=7, angle=45)) +
```