

Journal Club Bioinfo

The Tidyverse: A collection of R packages designed for data science.

ROCHER Vincent

28 mars 2019

R for data science

O'REILLY



Hadley Wickham &
Garrett Grolemund

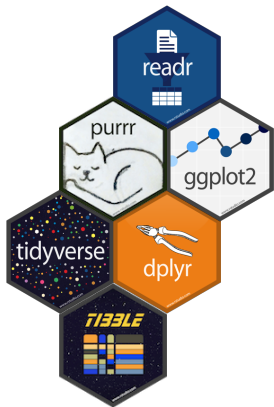
- **R for data science** : The best place to start learning the tidyverse by Hadley Wickham and Garrett Grolemund^a.
- **ggplot2 : elegant graphics for data science** by Hadley Wickham. Goes into greater depth into the ggplot2 visualisation system.

a. available online : <https://r4ds.had.co.nz>.

- 1 Tidyverse & tidy data
- 2 Pipe and tibble
- 3 Tidying dataset
- 4 Manipulate a dataset using dplyr
- 5 Data visualisation using ggplot2
- 6 Sources

Tidyverse & tidy data

Tidyverse



R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

<https://www.tidyverse.org/>

```
install.packages("tidyverse")
```

Tidyverse

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ----- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0      v purrr  0.2.5
```

```
## v tibble 2.0.1      v dplyr 0.7.8
```

```
## v tidyr      0.8.2      v stringr 1.4.0
```

```
## v readr 1.3.1 v forcats 0.3.0
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()      masks stats::lag()
```

Tidy data

Tidying : structuring datasets to facilitate analysis.

A tidy dataset :

country	year	cases	population
Afghanistan	1999	15	187071
Afghanistan	2000	666	2095360
Brazil	1999	3737	17206362
Brazil	2000	8488	17404898
China	1999	21258	127215272
China	2000	21766	128078583

variables

country	year	cases	population
china	1999	710	1200000000
china	2000	2200	1200000000
china	2001	6700	1200000000
china	2002	20000	1200000000
china	2003	270000	1200000000
china	2004	270000	1200000000

observations

country	year	cases	population
Afghanistan	2000	723	19,800,071
Afghanistan	2001	655	20,000,360
Brazil	2000	5747	173,000,362
Brazil	2001	6449	174,000,898
China	2000	21,025	127,000,272
China	2001	21,675	128,000,583

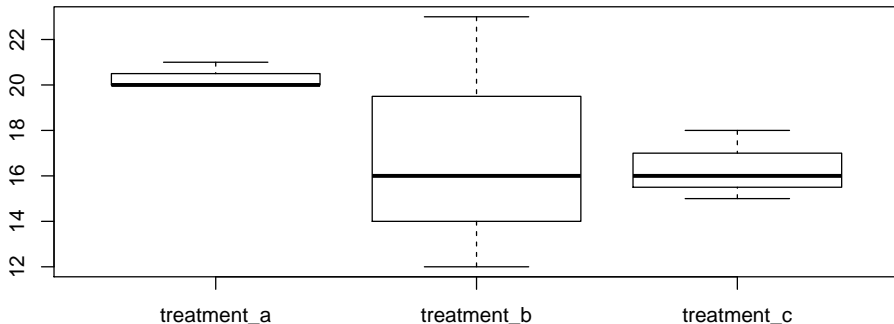
values

Why tidying a dataset ?

TABLE 1 – Typical presentation dataset

	treatment_a	treatment_b	treatment_c
John Smith	20	23	18
Jane Doe	20	12	15
Mary Johnson	21	16	16

```
boxplot(treatments)
```



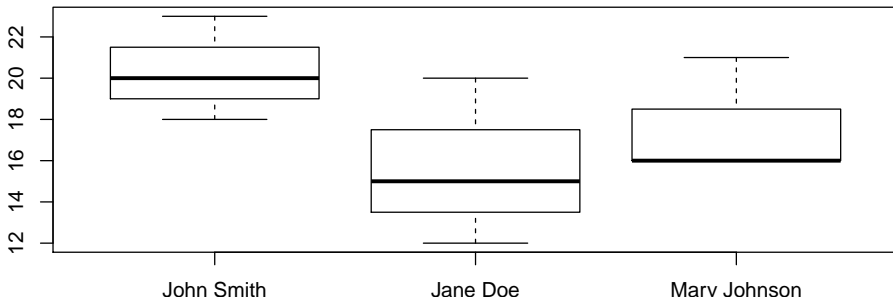
Why tidying a dataset ?

```
treatments.2 <- t(treatments)
```

TABLE 2 – The same data but structured differently

	John Smith	Jane Doe	Mary Johnson
treatment_a	20	20	21
treatment_b	23	12	16
treatment_c	18	15	16

```
boxplot(treatments.2)
```

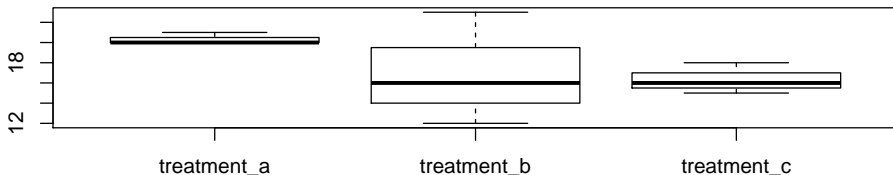


Why tidying a dataset ?

TABLE 3 – An exemple of a tidy dataset

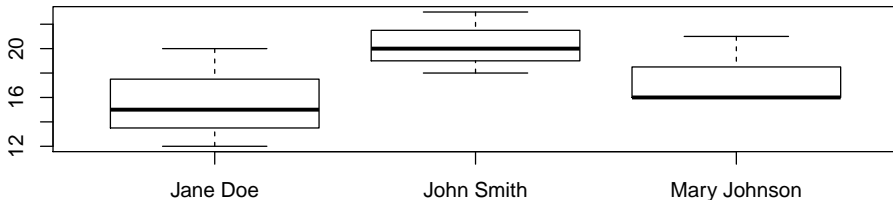
person	treatment	result
John Smith	treatment_a	20
Jane Doe	treatment_a	20
Mary Johnson	treatment_a	21
John Smith	treatment_b	23
Jane Doe	treatment_b	12
Mary Johnson	treatment_b	16
John Smith	treatment_c	18
Jane Doe	treatment_c	15
Mary Johnson	treatment_c	16

```
boxplot(result~treatment,data=treatments.3)
```

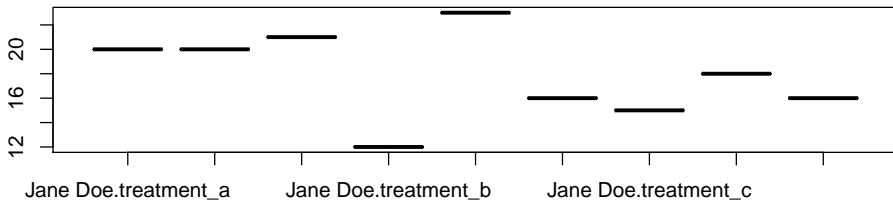


Why tidying a dataset ?

```
boxplot(result~person,data=treatments.3)
```



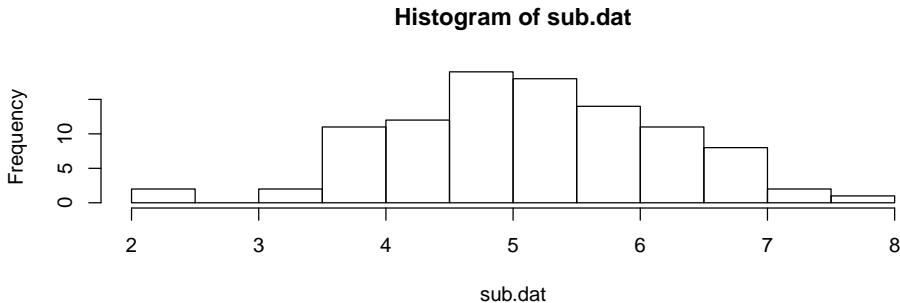
```
boxplot(result~person+treatment,data=treatments.3)
```



Pipe and tibble

Pipes vs no pipes : without pipe

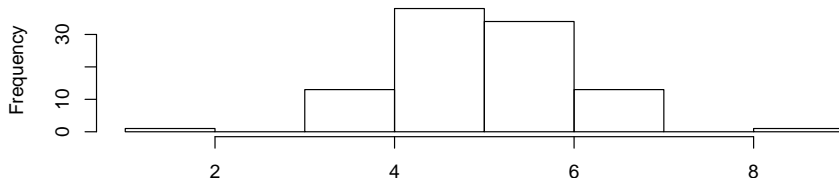
```
ex.dat <- rnorm(n = 1000, mean = 5, sd = 1)
sub.dat <- sample(ex.dat, size = 100, replace = F)
hist(sub.dat)
```



Pipes vs no pipes : without pipe

```
hist(sample(rnorm(n = 1000, mean = 5, sd = 1), size = 100, replace = F))
```

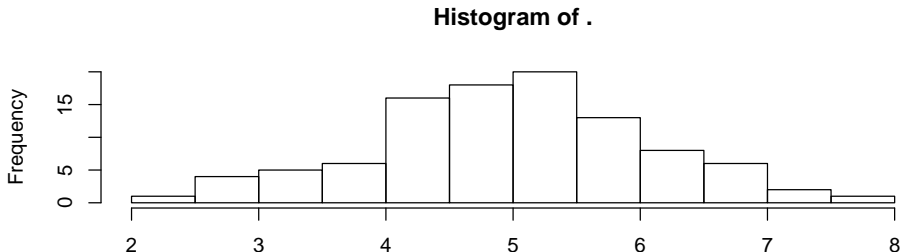
Histogram of `sample(rnorm(n = 1000, mean = 5, sd = 1), size = 100, replace = F)`



`sample(rnorm(n = 1000, mean = 5, sd = 1), size = 100, replace = F)`

Pipes vs no pipes : with pipe

```
rnorm(n = 1000, mean = 5, sd = 1) %>%  
  sample(size = 100, replace = F) %>%  
  hist()
```



Tibbles : tibble::tibble

```
my.tibble <- tibble(
  person = c("John Smith","Jane Doe","Mary Johnson"),
  treatment_a = sample(10:25,size = 3,replace = T),
  treatment_b = sample(10:25,size = 3,replace = T),
  treatment_c = sample(10:25,size = 3,replace = T)
)
```

```
## # A tibble: 3 x 4
```

```
##   person      treatment_a treatment_b treatment_c
##   <chr>          <int>         <int>         <int>
## 1 John Smith           18            12            15
## 2 Jane Doe             17            15            17
## 3 Mary Johnson         14            14            15
```

Tibbles : `tibble::tibble()`

Pros :

- Cells can contain list and data.frame.
- It never changes an input's type (i.e., no more `stringsAsFactors = FALSE!`).
- Can use not valid R variable names (`:`) as column names.
- Refined print method that shows only the first 10 rows.

Cons :

- It never uses `row.names()`.
 - use `tibble::rownames_to_column()`.

```
treatments %>% tibble::rownames_to_column("person") %>%
  as_tibble() %>%
  rename(`:` = person)
```

```
## # A tibble: 3 x 4
##   `:` treatment_a treatment_b treatment_c
##   <chr>         <int>         <int>         <int>
## 1 John Smith      20             23             18
## 2 Jane Doe        20             12             15
## 3 Mary Johnson    21             16             16
```

Split a dataset into multiples dataset using `tidyr::nest()`

```
my.tibble.split <- my.tibble %>% group_by(person) %>% nest(.key = Data)
my.tibble.split
```

```
## # A tibble: 3 x 2
##   person      Data
##   <chr>      <list>
## 1 John Smith <tibble [1 x 3]>
## 2 Jane Doe   <tibble [1 x 3]>
## 3 Mary Johnson <tibble [1 x 3]>
```

```
my.tibble.split[["Data"]][[1]]
```

```
## # A tibble: 1 x 3
##   treatment_a treatment_b treatment_c
##   <int>      <int>      <int>
## 1         18         12         15
```

Read/write tibbles : readr::read_* / readr::write_*

Read/write tibbles : :

■ readr::read_* :

- read_csv : comma delimited files.
- read_csv2 : semi-colon delimited files.
- read_tsv : tab delimited files.
- read_delim : any delimiter.

■ readr::write_* :

- write_csv : comma delimited files.
- write_csv2 : semi-colon delimited files.
- write_tsv : tab delimited files.
- write_delim : any delimiter.

Read/write tibbles : readr::read_* / readr::write_*

```
read_tsv("iris.tsv",
  col_types = cols(
    Sepal.Length = col_double(), Sepal.Width = col_double(),
    Petal.Length = col_double(), Petal.Width = col_double(),
    Species = col_character()
  )
)
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <chr>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
## 5         5         3.6         1.4         0.2 setosa
## 6         5.4         3.9         1.7         0.4 setosa
## 7         4.6         3.4         1.4         0.3 setosa
## 8         5         3.4         1.5         0.2 setosa
## 9         4.4         2.9         1.4         0.2 setosa
## 10        4.9         3.1         1.5         0.1 setosa
## # ... with 140 more rows
```

Tidying dataset

Using tidyr to reshape data

Use `tidyr::gather()` and `tidyr::spread()` to reorganize the value.

```
treatments %>% tibble::rownames_to_column("person") %>%
  tidyr::gather(key="treatment",value = "result",-person)
```

```
##           person  treatment result
## 1   John Smith treatment_a     20
## 2   Jane Doe  treatment_a     20
## 3 Mary Johnson treatment_a     21
## 4   John Smith treatment_b     23
## 5   Jane Doe  treatment_b     12
## 6 Mary Johnson treatment_b     16
## 7   John Smith treatment_c     18
## 8   Jane Doe  treatment_c     15
## 9 Mary Johnson treatment_c     16
```

```
treatment.3 %>% spread(key = "treatment",value = "result")
```

```
##           person treatment_a treatment_b treatment_c
## 1   Jane Doe           20           12           15
## 2   John Smith          20           23           18
## 3 Mary Johnson          21           16           16
```

Using tidyr to reshape data

```
treatment.3 %>% separate(person,into = c("First Name","Last Name"),sep=" ")
```

##	First Name	Last Name	treatment	result
## 1	John	Smith	treatment_a	20
## 2	Jane	Doe	treatment_a	20
## 3	Mary	Johnson	treatment_a	21
## 4	John	Smith	treatment_b	23
## 5	Jane	Doe	treatment_b	12
## 6	Mary	Johnson	treatment_b	16
## 7	John	Smith	treatment_c	18
## 8	Jane	Doe	treatment_c	15
## 9	Mary	Johnson	treatment_c	16

Manipulate a dataset using dplyr

dplyr :

- Manipulate variables.
- Manipulate observations.
- Group / summarise observations.

TABLE 4 – iris dataset

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

iris dataset

The famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*. (from ?iris)

Manipulate **variables** using `dplyr::select()`

```
iris.tbl %>% select(1:3)
iris.tbl %>% select(Sepal.Length, Sepal.Width, Petal.Length)
iris.tbl %>% select(Sepal.Length:Petal.Length)
iris.tbl %>% select(-Petal.Width, -Species)
```

```
## # A tibble: 150 x 3
##   Sepal.Length Sepal.Width Petal.Length
##   <dbl>         <dbl>         <dbl>
## 1         5.1         3.5         1.4
## 2         4.9         3         1.4
## 3         4.7         3.2         1.3
## 4         4.6         3.1         1.5
## 5         5         3.6         1.4
## 6         5.4         3.9         1.7
## 7         4.6         3.4         1.4
## 8         5         3.4         1.5
## 9         4.4         2.9         1.4
## 10        4.9         3.1         1.5
## # ... with 140 more rows
```

Manipulate **variables** using `dplyr::select()`

```
iris.tbl %>% select(starts_with("Petal"))
```

```
## # A tibble: 150 x 2
##   Petal.Length Petal.Width
##   <dbl>         <dbl>
## 1         1.4         0.2
## 2         1.4         0.2
## 3         1.3         0.2
## 4         1.5         0.2
## 5         1.4         0.2
## 6         1.7         0.4
## 7         1.4         0.3
## 8         1.5         0.2
## 9         1.4         0.2
## 10        1.5         0.1
## # ... with 140 more rows
```

Manipulate **variables** using `dplyr::pull()`

```
iris.tbl %>% select(Sepal.Length)
```

```
## # A tibble: 150 x 1
##   Sepal.Length
##         <dbl>
## 1         5.1
## 2         4.9
## 3         4.7
## 4         4.6
## 5          5
## # ... with 145 more rows
```

```
iris.tbl %>% pull(Sepal.Length)
```

```
## [1] 5.1 4.9 4.7 4.6 5.0
```

Modify **variables** using `dplyr::mutate()`

```
iris.tbl %>%
  mutate(Petal.Length =
    scales::percent(Petal.Length/max(Petal.Length))
  )
```

A tibble: 150 x 5

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
##	<dbl>	<dbl>	<chr>	<dbl>	<fct>
## 1	5.1	3.5	20.3%	0.2	setosa
## 2	4.9	3	20.3%	0.2	setosa
## 3	4.7	3.2	18.8%	0.2	setosa
## 4	4.6	3.1	21.7%	0.2	setosa
## 5	5	3.6	20.3%	0.2	setosa
## 6	5.4	3.9	24.6%	0.4	setosa
## 7	4.6	3.4	20.3%	0.3	setosa
## 8	5	3.4	21.7%	0.2	setosa
## 9	4.4	2.9	20.3%	0.2	setosa
## 10	4.9	3.1	21.7%	0.1	setosa

... with 140 more rows

Create **variables** using `dplyr::mutate()`

```
iris.tbl %>%
  #Case 1
  mutate(Size = dplyr::case_when(
    Petal.Length >= 4 & Petal.Width >= 1.3 ~ "Big",
    Petal.Length < 4 & Petal.Width < 1.3 ~ "Small"
  )
) %>%
#Case 2
mutate(Ratio_Petal_Sepal_width = Petal.Width/Sepal.Width)
```

```
## # A tibble: 3 x 7
##   Ratio_Petal_Sep~ Size   Sepal.Length Sepal.Width Petal.Length Petal.Width
##         <dbl> <chr>         <dbl>         <dbl>         <dbl>         <dbl>
## 1      0.0571 Small           5.1           3.5           1.4           0.2
## 2      0.0667 Small           4.9           3            1.4           0.2
## 3      0.0625 Small           4.7           3.2           1.3           0.2
## # ... with 1 more variable: Species <fct>
```

other functions :

- `dplyr::mutate_all()` : Apply a function to every columns.
- `dplyr::mutate_at()` : Apply a function to specific column.

Arrange dataset using `dplyr::arrange()`

```
iris.tbl %>% arrange(Petal.Length) %>% slice(1:5)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         4.6         3.6           1           0.2 setosa
## 2         4.3         3             1.1         0.1 setosa
## 3         5.8         4             1.2         0.2 setosa
## 4         5           3.2           1.2         0.2 setosa
## 5         4.7         3.2           1.3         0.2 setosa
```

```
iris.tbl %>% filter(min_rank(Petal.Length) <= 5)
iris.tbl %>% top_n(-5,Petal.Length)
```

Use `desc()` to order by high to low.

Manipulate **observations** using `dplyr::filter()`

```
iris.tbl %>% filter(Sepal.Length < 7) %>% slice(1:5)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
## 5         5         3.6         1.4         0.2 setosa
```

```
iris.tbl %>% filter(Species == "setosa") %>% slice(1:5)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
## 5         5         3.6         1.4         0.2 setosa
```

Manipulate **observations** using `dplyr::filter()`

```
iris.tbl %>%
  filter(Sepal.Length < 7) %>%
  filter(Species == "setosa") %>% slice(1:5)
```

```
## # A tibble: 5 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         5.1         3.5         1.4         0.2 setosa
## 2         4.9         3         1.4         0.2 setosa
## 3         4.7         3.2         1.3         0.2 setosa
## 4         4.6         3.1         1.5         0.2 setosa
## 5         5         3.6         1.4         0.2 setosa
```

```
iris.tbl %>%
  filter(Sepal.Length < 7 & Species == "setosa") %>% slice(1:5)

iris.tbl %>%
  filter(Sepal.Length < 7, Species == "setosa") %>% slice(1:5)
```

Summarise observations using `dplyr::summarise()`

```
iris.tbl %>% summarise(mean = mean(Petal.Length), sd = sd(Petal.Length))
```

```
## # A tibble: 1 x 2
##   mean    sd
##   <dbl> <dbl>
## 1  3.76  1.77
```

```
iris.tbl %>% select(-Species) %>% summarise_all(mean)
```

```
## # A tibble: 1 x 4
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
##           <dbl>         <dbl>         <dbl>         <dbl>
## 1      5.84      3.06      3.76      1.20
```

```
iris.tbl %>% summarise(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   150
```

Summarise observations using `dplyr::group_by()`

```
iris.tbl %>% group_by(Species) %>% summarise(mean = mean(Petal.Length), sd = sd(Petal.Length))
```

```
## # A tibble: 3 x 3
##   Species      mean      sd
##   <fct>      <dbl> <dbl>
## 1 setosa      1.46  0.174
## 2 versicolor  4.26  0.470
## 3 virginica   5.55  0.552
```

```
iris.tbl %>% group_by(Species) %>% summarise(n = n())
```

```
## # A tibble: 3 x 2
##   Species      n
##   <fct>    <int>
## 1 setosa     50
## 2 versicolor 50
## 3 virginica  50
```

```
iris.tbl %>% group_by(Species) %>% tally()
iris.tbl %>% count(Species)
```

Summarise observations using `dplyr::group_by()`

```
iris.tbl %>% group_by(Species) %>% filter(Petal.Length >= max(Petal.Length))
```

```
## # A tibble: 4 x 5
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         4.8         3.4         1.9         0.2 setosa
## 2         5.1         3.8         1.9         0.4 setosa
## 3          6         2.7         5.1         1.6 versicolor
## 4         7.7         2.6         6.9         2.3 virginica
```

```
iris.tbl %>% group_by(Species) %>% top_n(1,Petal.Length)
```

```
## # A tibble: 4 x 5
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         4.8         3.4         1.9         0.2 setosa
## 2         5.1         3.8         1.9         0.4 setosa
## 3          6         2.7         5.1         1.6 versicolor
## 4         7.7         2.6         6.9         2.3 virginica
```

Mutating joins using `dplyr::*_join()`

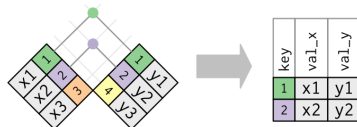
```
iris.meta <- tibble(  
  Species = factor(c("setosa","versicolor","virginica")),  
  Colony = c("A","A","B") ,  
  Ploidy = c("diploid","hexaploid","tetraploid"),  
  `Common name` = c("Beachhead iris","Harlequin blueflag","Virginia iris")  
)
```

TABLE 5 – Metadata for iris dataset

Species	Colony	Ploidy	Common name
setosa	A	diploid	Beachhead iris
versicolor	A	hexaploid	Harlequin blueflag
virginica	B	tetraploid	Virginia iris

Inner join

Inner join : return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.



```
iris.meta.nosetosa <- iris.meta %>% filter(Species != "setosa")
iris.tbl %>% inner_join(iris.meta.nosetosa, by = "Species") %>% print(n=5)
```

```
## # A tibble: 100 x 8
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Colony Ploidy
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>   <chr> <chr>
## 1           7         3.2         4.7         1.4 versic~ A    hexap~
## 2          6.4         3.2         4.5         1.5 versic~ A    hexap~
## 3          6.9         3.1         4.9         1.5 versic~ A    hexap~
## 4          5.5         2.3         4          1.3 versic~ A    hexap~
## 5          6.5         2.8         4.6         1.5 versic~ A    hexap~
## # ... with 95 more rows, and 1 more variable: `Common name` <chr>
```

Outer joins

- A **left join** keeps all observations in *x*.
- A **right join** keeps all observations in *y*.
- A **full join** keeps all observations in *x* and *y*.

Left



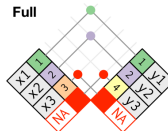
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Right



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Full



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

Left join

TABLE 6 – Metadata for iris dataset

Species	Colony	Ploidy	Common name
setosa	A	diploid	Beachhead iris
versicolor	A	hexaploid	Harlequin blueflag
virginica	B	tetraploid	Virginia iris

left join :

```
iris.meta.nosetosa <- iris.meta %>% filter(Species != "setosa")  
iris.tbl %>% left_join(iris.meta.nosetosa, by = "Species") %>% print(n=5)
```

```
## # A tibble: 150 x 8  
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Colony Ploidy  
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>   <chr>  <chr>  
## 1         5.1           3.5           1.4           0.2 setosa <NA>  <NA>  
## 2         4.9           3             1.4           0.2 setosa <NA>  <NA>  
## 3         4.7           3.2           1.3           0.2 setosa <NA>  <NA>  
## 4         4.6           3.1           1.5           0.2 setosa <NA>  <NA>  
## 5         5             3.6           1.4           0.2 setosa <NA>  <NA>  
## # ... with 145 more rows, and 1 more variable: `Common name` <chr>
```

Right join

TABLE 7 – iris dataset

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica

right join :

```
iris.tbl.nosetosa <- iris.tbl %>% filter(Species != "setosa")
iris.tbl.nosetosa %>% right_join(iris.meta, by = "Species") %>% print(n=5)
```

```
## # A tibble: 101 x 8
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Colony Ploidy
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>   <chr>   <chr>
## 1         NA         NA         NA         NA   setosa   A       diplo~
## 2         7         3.2         4.7         1.4 versic~ A       hexap~
## 3         6.4        3.2         4.5         1.5 versic~ A       hexap~
## 4         6.9        3.1         4.9         1.5 versic~ A       hexap~
## 5         5.5        2.3         4         1.3 versic~ A       hexap~
## # ... with 96 more rows, and 1 more variable: `Common name` <chr>
```

Outer joins

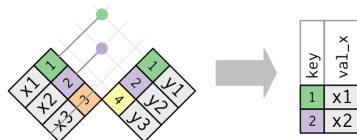
full join :

```
iris.meta.nosetosa <- iris.meta %>% filter(Species != "setosa")
iris.tbl %>% filter(Species != "virginica") %>%
  full_join(iris.meta.nosetosa, by = "Species")
```

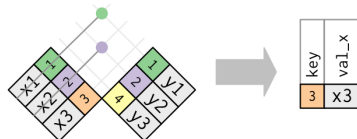
```
## # A tibble: 11 x 8
## # Groups:   Species [3]
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Colony Ploidy
##   <dbl>         <dbl>         <dbl>         <dbl> <fct>   <chr> <chr>
## 1         5.1         3.5         1.4         0.2 setosa  <NA>  <NA>
## 2         4.9         3         1.4         0.2 setosa  <NA>  <NA>
## 3         4.7         3.2         1.3         0.2 setosa  <NA>  <NA>
## 4         4.6         3.1         1.5         0.2 setosa  <NA>  <NA>
## 5          5         3.6         1.4         0.2 setosa  <NA>  <NA>
## 6          7         3.2         4.7         1.4 versic~ A      hexap~
## 7         6.4         3.2         4.5         1.5 versic~ A      hexap~
## 8         6.9         3.1         4.9         1.5 versic~ A      hexap~
## 9         5.5         2.3          4         1.3 versic~ A      hexap~
## 10        6.5         2.8         4.6         1.5 versic~ A      hexap~
## 11          NA          NA          NA          NA  virgin~ B      tetra~
## # ... with 1 more variable: `Common name` <chr>
```

Filtering joins

- `semi_join(x, y)` keeps all observations in `x` that have a match in `y`.



- `anti_join(x, y)` drops all observations in `x` that have a match in `y`.



Filtering joins

TABLE 8 – Patients files

First_Name	Last_Name	age	sex	adress
John	Smith	46	M	221B Baker Street
Jane	Doe	33	F	57 Rue de Varenne

```
treatments.3 %>%  
  separate(person,into = c("First_Name","Last_Name"),sep=" ") %>%  
  semi_join(treatment.meta,by = c("First_Name","Last_Name"))
```

```
##   First_Name Last_Name  treatment result  
## 1      John      Smith treatment_a     20  
## 2      Jane       Doe treatment_a     20  
## 3      John      Smith treatment_b     23  
## 4      Jane       Doe treatment_b     12  
## 5      John      Smith treatment_c     18  
## 6      Jane       Doe treatment_c     15
```

Use multiple command together

```
iris.tbl
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##         <dbl>         <dbl>         <dbl>         <dbl> <fct>
## 1         5.1           3.5           1.4           0.2 setosa
## 2         4.9           3             1.4           0.2 setosa
## 3         4.7           3.2           1.3           0.2 setosa
## 4         4.6           3.1           1.5           0.2 setosa
## 5         5             3.6           1.4           0.2 setosa
## 6         5.4           3.9           1.7           0.4 setosa
## 7         4.6           3.4           1.4           0.3 setosa
## 8         5             3.4           1.5           0.2 setosa
## 9         4.4           2.9           1.4           0.2 setosa
## 10        4.9           3.1           1.5           0.1 setosa
## # ... with 140 more rows
```

Use multiple command together

```
iris.tbl %>%
  gather(key = "Type",value = "obs",-Species) %>%
  separate(Type,into = c("Type_1","Type_2"),se="\\.")
```

```
## # A tibble: 600 x 4
##   Species Type_1 Type_2   obs
##   <fct>   <chr> <chr> <dbl>
## 1 setosa  Sepal  Length 5.1
## 2 setosa  Sepal  Length 4.9
## 3 setosa  Sepal  Length 4.7
## 4 setosa  Sepal  Length 4.6
## 5 setosa  Sepal  Length 5
## 6 setosa  Sepal  Length 5.4
## 7 setosa  Sepal  Length 4.6
## 8 setosa  Sepal  Length 5
## 9 setosa  Sepal  Length 4.4
## 10 setosa Sepal  Length 4.9
## # ... with 590 more rows
```

Use multiple command together

```
iris.tbl %>%  
  gather(key = "Type",value = "obs",-Species) %>%  
  separate(Type,into = c("Type_1","Type_2"),se="\\.") %>%  
  group_by(Species,Type_1,Type_2) %>%  
  mutate(med = median(obs))
```

```
## # A tibble: 600 x 5  
## # Groups:   Species, Type_1, Type_2 [12]  
##   Species Type_1 Type_2   obs   med  
##   <fct>   <chr> <chr> <dbl> <dbl>  
## 1 setosa Sepal Length  5.1     5  
## 2 setosa Sepal Length  4.9     5  
## 3 setosa Sepal Length  4.7     5  
## 4 setosa Sepal Length  4.6     5  
## 5 setosa Sepal Length  5       5  
## 6 setosa Sepal Length  5.4     5  
## 7 setosa Sepal Length  4.6     5  
## 8 setosa Sepal Length  5       5  
## 9 setosa Sepal Length  4.4     5  
## 10 setosa Sepal Length  4.9     5  
## # ... with 590 more rows
```


Use multiple command together

```
iris.tbl %>%
  gather(key = "Type",value = "obs",-Species) %>%
  separate(Type,into = c("Type_1","Type_2"),se="\\.") %>%
  group_by(Species,Type_1,Type_2) %>%
  mutate(med = median(obs)) %>%
  mutate(Size = ifelse(obs < med,"Small","Big")) %>%
  left_join(iris.meta,by = "Species")
```

```
## # A tibble: 600 x 9
```

```
## # Groups:   Species, Type_1, Type_2 [?]
```

```
##   Species Type_1 Type_2  obs  med Size Colony Ploidy `Common name`
##   <fct>   <chr>  <chr>  <dbl> <dbl> <chr> <chr>   <chr>   <chr>
## 1 setosa  Sepal  Length  5.1    5 Big   A      diploid Beachhead iris
## 2 setosa  Sepal  Length  4.9    5 Small A      diploid Beachhead iris
## 3 setosa  Sepal  Length  4.7    5 Small A      diploid Beachhead iris
## 4 setosa  Sepal  Length  4.6    5 Small A      diploid Beachhead iris
## 5 setosa  Sepal  Length  5      5 Big   A      diploid Beachhead iris
## 6 setosa  Sepal  Length  5.4    5 Big   A      diploid Beachhead iris
## 7 setosa  Sepal  Length  4.6    5 Small A      diploid Beachhead iris
## 8 setosa  Sepal  Length  5      5 Big   A      diploid Beachhead iris
## 9 setosa  Sepal  Length  4.4    5 Small A      diploid Beachhead iris
## 10 setosa Sepal  Length  4.9    5 Small A      diploid Beachhead iris
## # ... with 590 more rows
```

Use multiple command together

```
iris.tbl %>%
  gather(key = "Type", value = "obs", -Species) %>%
  separate(Type, into = c("Type_1", "Type_2"), se = "\\.") %>%
  group_by(Species, Type_1, Type_2) %>%
  mutate(med = median(obs)) %>%
  mutate(Size = ifelse(obs < med, "Small", "Big")) %>%
  left_join(iris.meta, by = "Species") %>%
  arrange(desc(obs)) %>% slice(1)
```

```
## # A tibble: 12 x 9
```

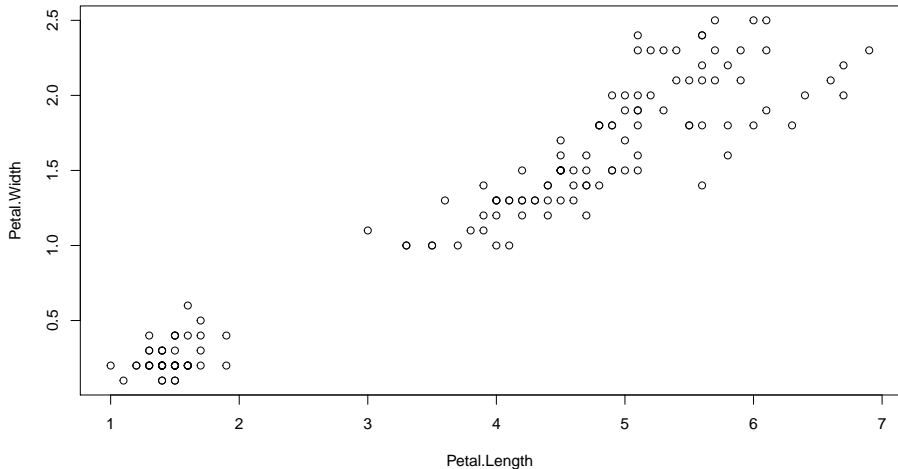
```
## # Groups:   Species, Type_1, Type_2 [12]
```

	Species	Type_1	Type_2	obs	med	Size	Colony	Ploidy	Common name
	<chr>	<chr>	<chr>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>
## 1	setosa	Petal	Length	1.9	1.5	Big	A	diploid	Beachhead iris
## 2	setosa	Petal	Width	0.6	0.2	Big	A	diploid	Beachhead iris
## 3	setosa	Sepal	Length	5.8	5	Big	A	diploid	Beachhead iris
## 4	setosa	Sepal	Width	4.4	3.4	Big	A	diploid	Beachhead iris
## 5	versicol~	Petal	Length	5.1	4.35	Big	A	hexapl~	Harlequin blue~
## 6	versicol~	Petal	Width	1.8	1.3	Big	A	hexapl~	Harlequin blue~
## 7	versicol~	Sepal	Length	7	5.9	Big	A	hexapl~	Harlequin blue~
## 8	versicol~	Sepal	Width	3.4	2.8	Big	A	hexapl~	Harlequin blue~
## 9	virginica	Petal	Length	6.9	5.55	Big	B	tetrap~	Virginia iris
## 10	virginica	Petal	Width	2.5	2	Big	B	tetrap~	Virginia iris
## 11	virginica	Sepal	Length	7.9	6.5	Big	B	tetrap~	Virginia iris
## 12	virginica	Sepal	Width	3.8	3	Big	B	tetrap~	Virginia iris

Data visualisation using ggplot2

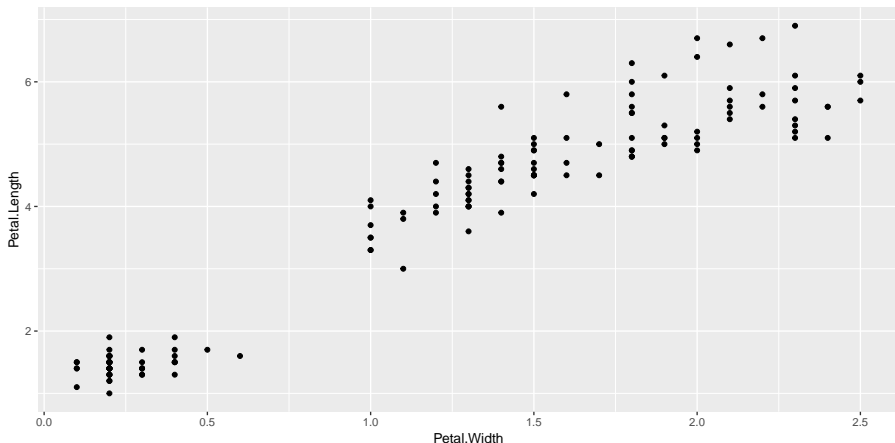
A base plot

```
plot(Petal.Width~Petal.Length,data=iris.tbl)
```



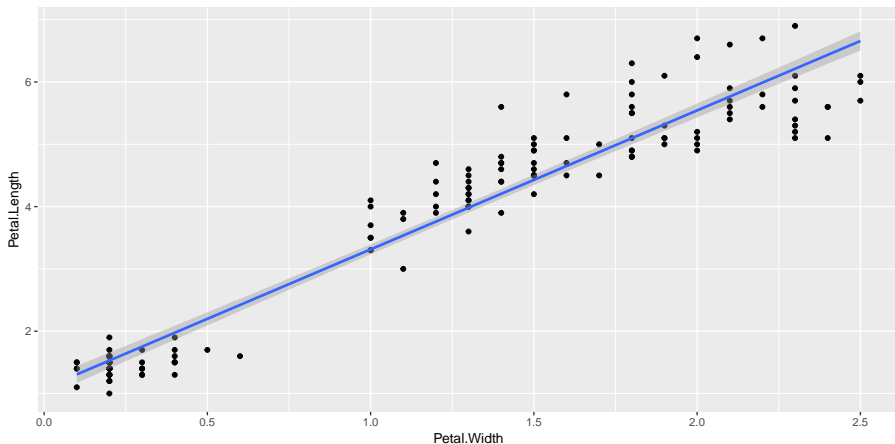
A ggplot2 plot

```
iris.tbl %>% ggplot(aes(x=Petal.Width,y=Petal.Length)) +  
  geom_point()
```



A ggplot2 plot

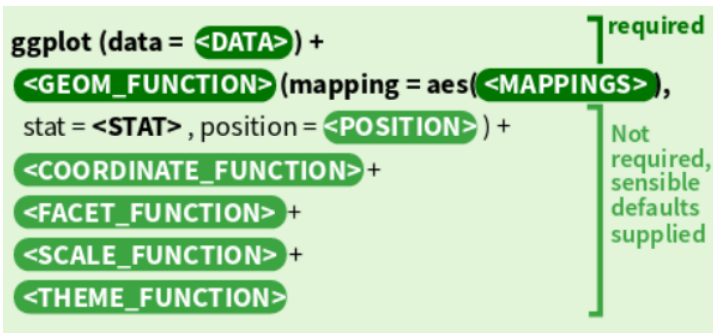
```
iris.tbl %>% ggplot(aes(x=Petal.Width,y=Petal.Length)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```



The three key components of every ggplot2

The three components of every graphics :

- **data** : the dataset who's need to be plotted.
- **aesthetics** : aesthetics mapping between variables in the data (x,y,visuals properties, ...).
- **geoms** : one or more layers to render each observations.



The three key components of every ggplot2

The three components of every graphics :

- **data** : the dataset who's need to be plotted.
- **aesthetics** : aesthetics mapping between variables in the data (x,y,visuals properties, ...).
- **geoms** : one or more layers to render each observations.

```
p <- iris.tbl %>%# Data  
  ggplot(aes(x=Petal.Width,y=Petal.Length)) +# aesthetics  
  geom_point()# Layer: points
```

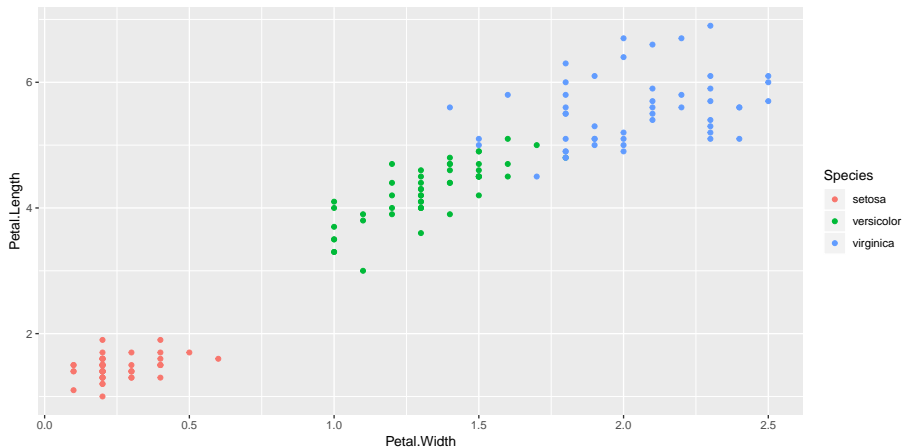
Add layer : smooth with linear model

```
p + geom_smooth(method = "lm")
```

More you add components with +, more you'll build sophisticated plots.

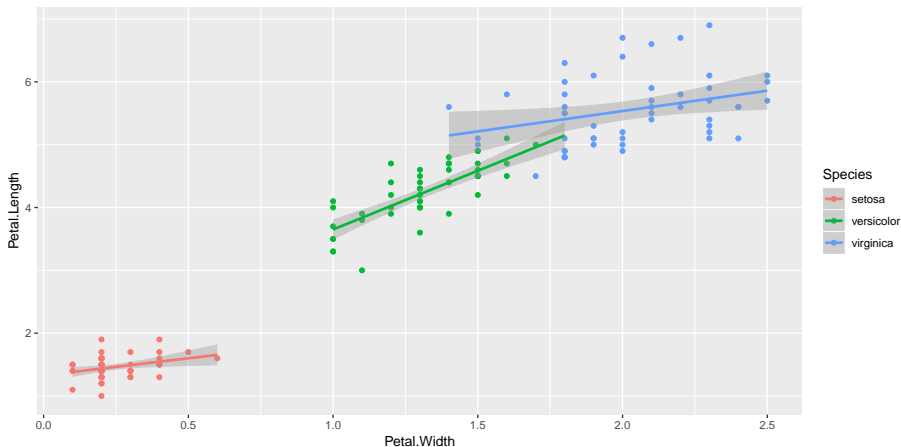
Add more aesthetics

```
p <- iris.tbl %>%# Data
  ggplot(aes(x=Petal.Width,y=Petal.Length,col=Species)) +# aesthetics
  geom_point()# Layer: points
print(p)
```



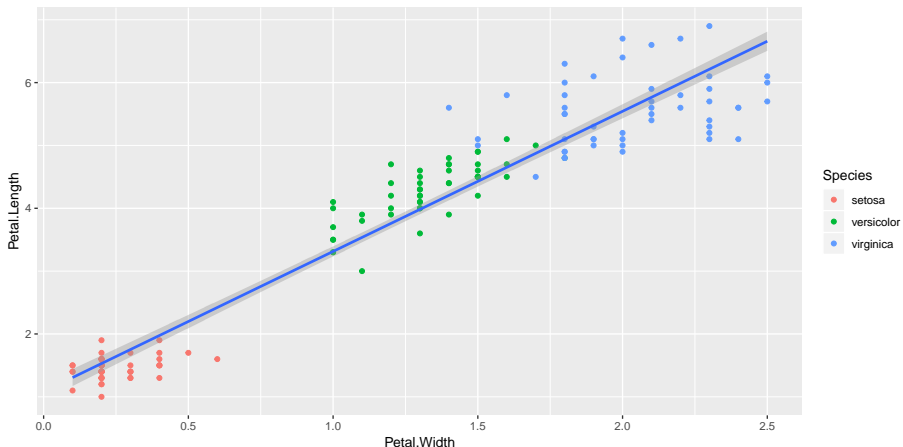
Add more aesthetics

```
p + geom_smooth(method = "lm")
```



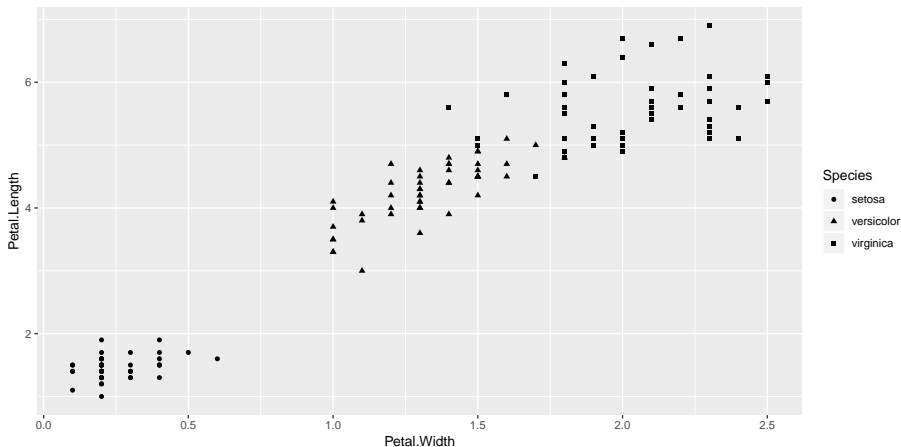
Add more aesthetics

```
p <- iris.tbl %>% # Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) + # aesthetics
  geom_point(aes(col=Species)) + # Layer: points
  geom_smooth(method = "lm")
print(p)
```



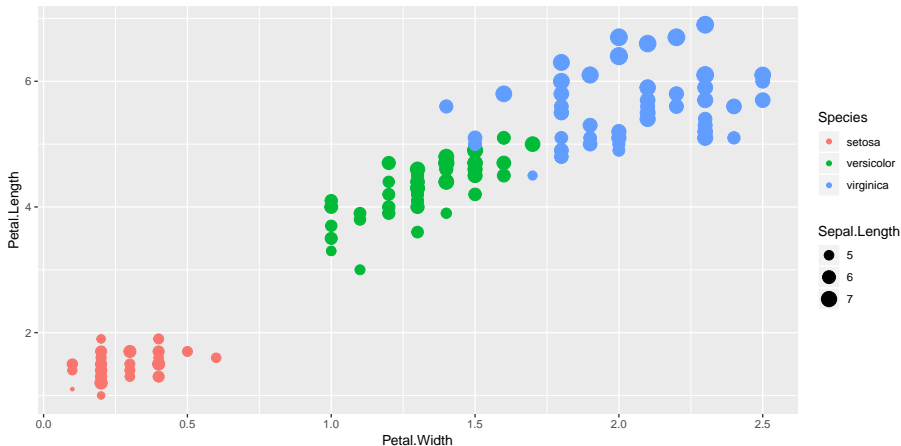
Other aesthetics attributes

```
p <- iris.tbl %>%# Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) +# aesthetics
  geom_point(aes(shape=Species))# Layer: points
print(p)
```



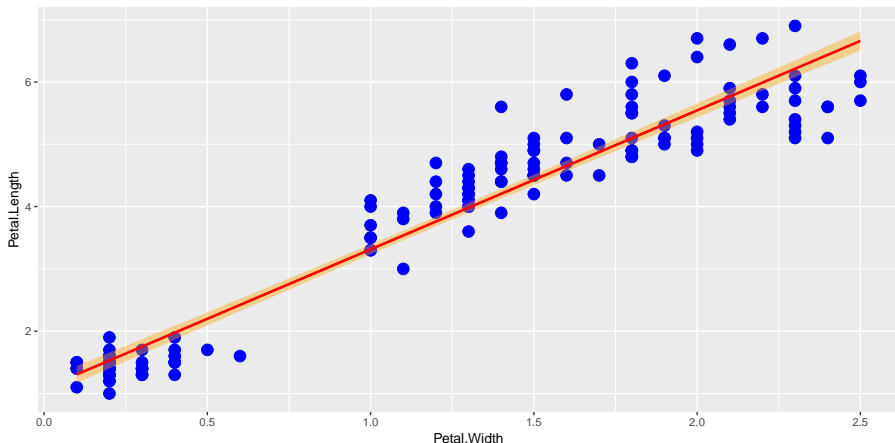
Other aesthetics attributes

```
p <- iris.tbl %>%# Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) +# aesthetics
  geom_point(aes(col=Species,size = Sepal.Length))# Layer: points
print(p)
```



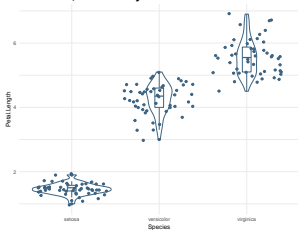
Use fixed value

```
p <- iris.tbl %>% # Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) + # aesthetics
  geom_point(col="blue",size=4) + # Layer: points
  geom_smooth(method = "lm",col="red",fill="orange")
print(p)
```

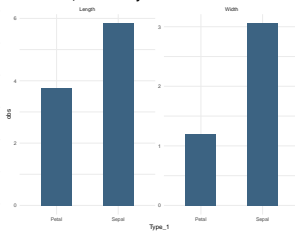


Plot geoms

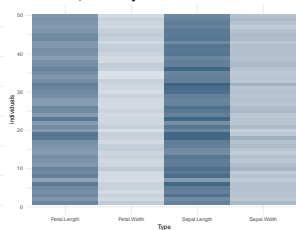
discrete x, continous y



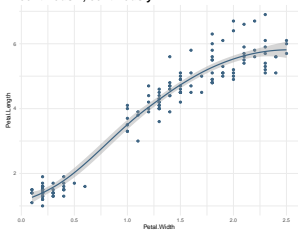
discrete x, continous y



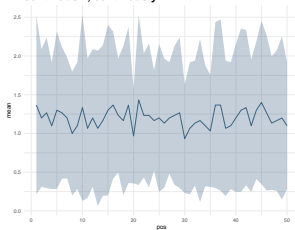
discrete x, discrete y



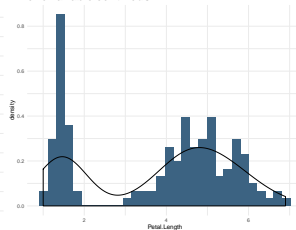
continous x, continous y



continous x, continous y



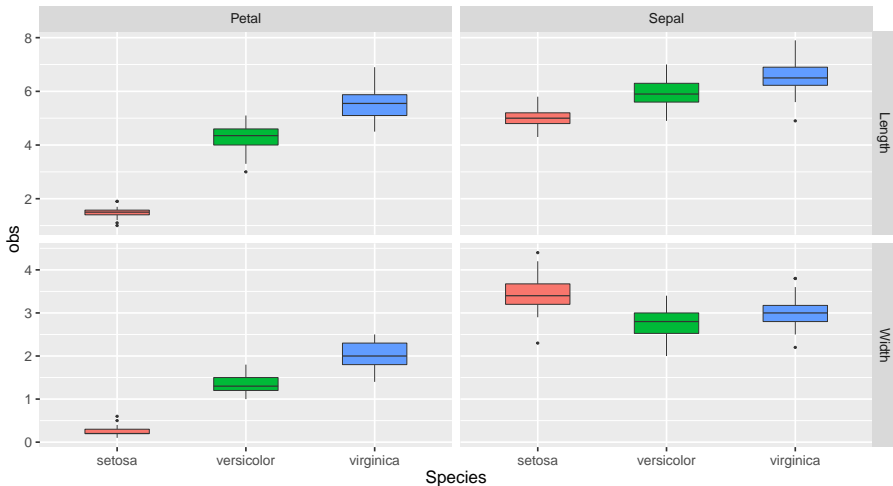
one variable continous



Facetting

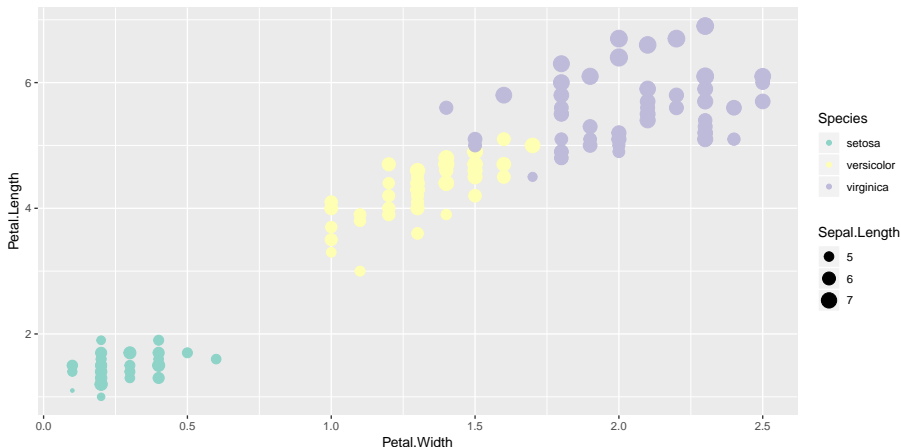
Facetting :

- `facet_grid(variable~foliaces)` : Display all possibility even if some plots are **empty**.
- `facet_wrap(variable~foliaces)` : Display only the plots having actual values.



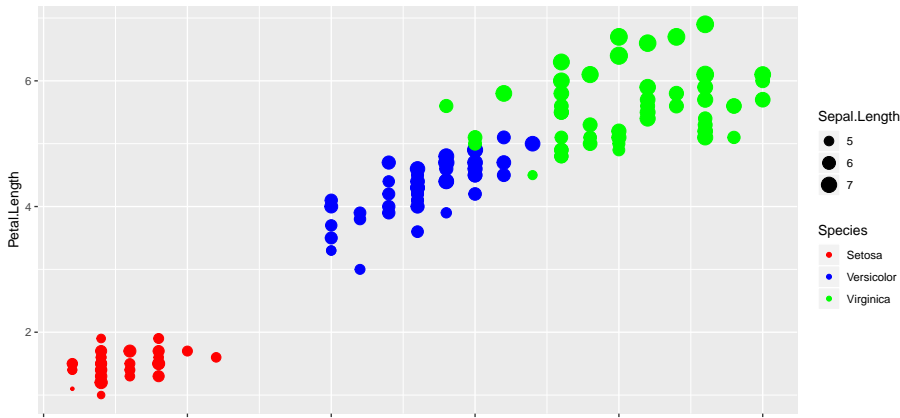
Graphical options

```
p <- iris.tbl %>%# Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) + # aesthetics
  geom_point(aes(col=Species,size = Sepal.Length)) + # Layer: points
  scale_color_brewer(palette="Set3") #A new palette
print(p)
```

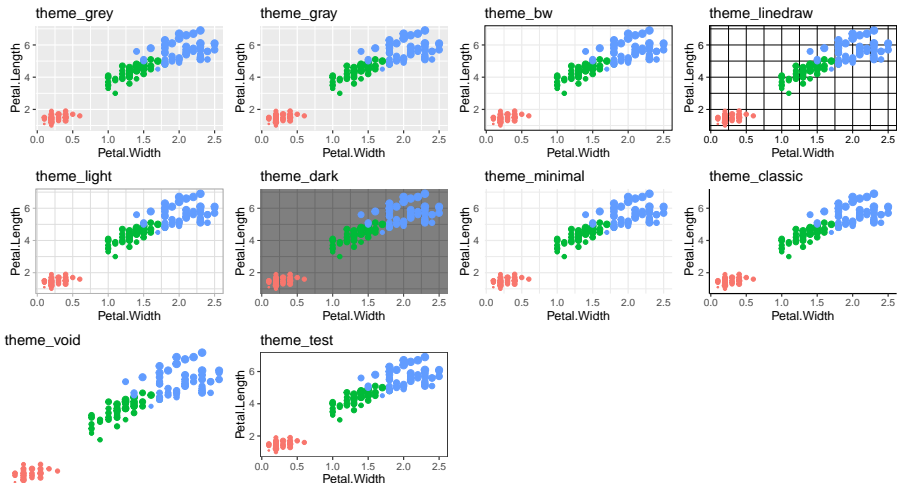


Graphical options

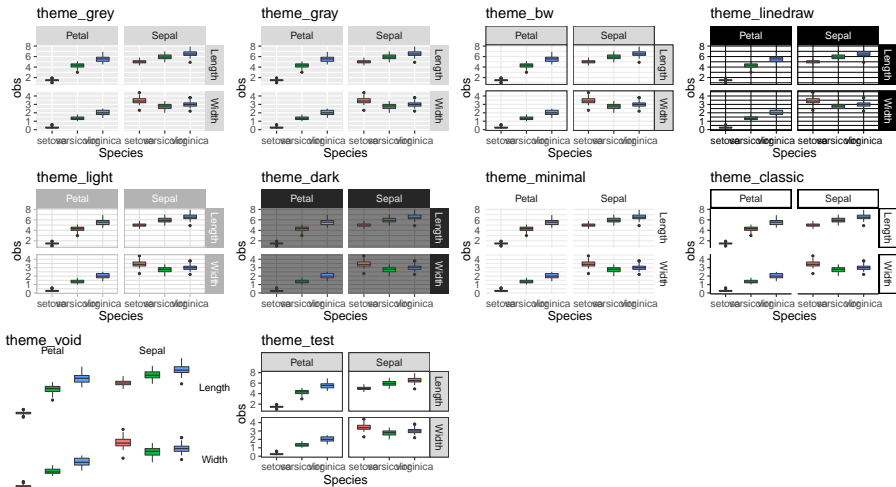
```
p <- iris.tbl %>% # Data
  ggplot(aes(x=Petal.Width,y=Petal.Length)) + # aesthetics
  geom_point(aes(col=Species,size = Sepal.Length)) + # Layer: points
  scale_color_manual(values = c("red","blue","green"), #A new palette
                    labels = c("Setosa","Versicolor","Virginica"))
print(p)
```



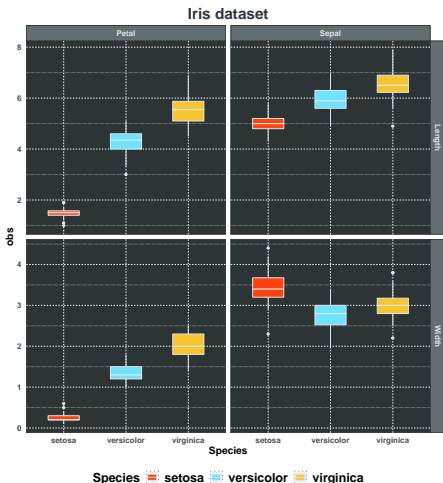
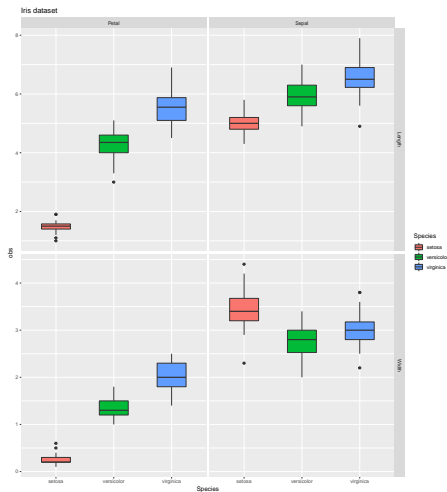
Themes



Themes



Themes



Extensions

- **ggplot2** extensions : This site tracks and lists ggplot2 extensions developed by R users in the community.
<https://www.ggplot2-exts.org/>.
- **cowplot** : Arranging graphs into a grid and improve plot design.
- **ggpubr & ggstatsplot** : Publication ready plots & add statistical tests.
- **ggrepel** : Automatic label placement.
- **esquisse** : Explore and Visualize Your Data Interactively with ggplot2.
- **ggvis** : Create rich interactive graphics with a syntax similar in spirit to ggplot2.

Why I use ggplot2

- The “default” output is much nicer than with base graphics.
- Automatic legend, color with mapping to the variables.
- Easy to combine multiple plot.
- Easy facetting.
- A lot of themes.
- A lot of extensions ...

Sources

Sources :

- **PDF and sources :**
https://github.com/rochevin/tidyverse_journal_club
- **Web version :** https://rochevin.github.io/tidyverse_journal_club