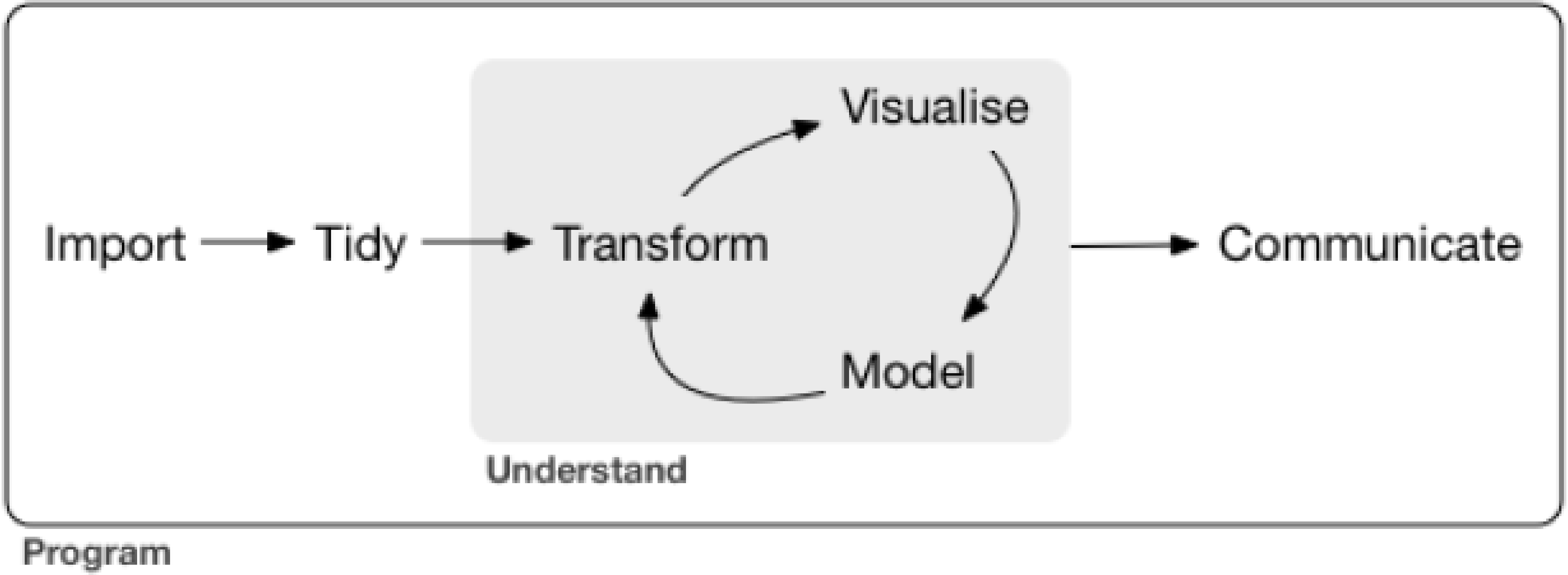
Data preprocessing and visualization in R

Yoon-Ho Hong

Work flow


Import
Tidy
Transform
Visualise
Model
Communicate
Understand
Program

Data import

5 types

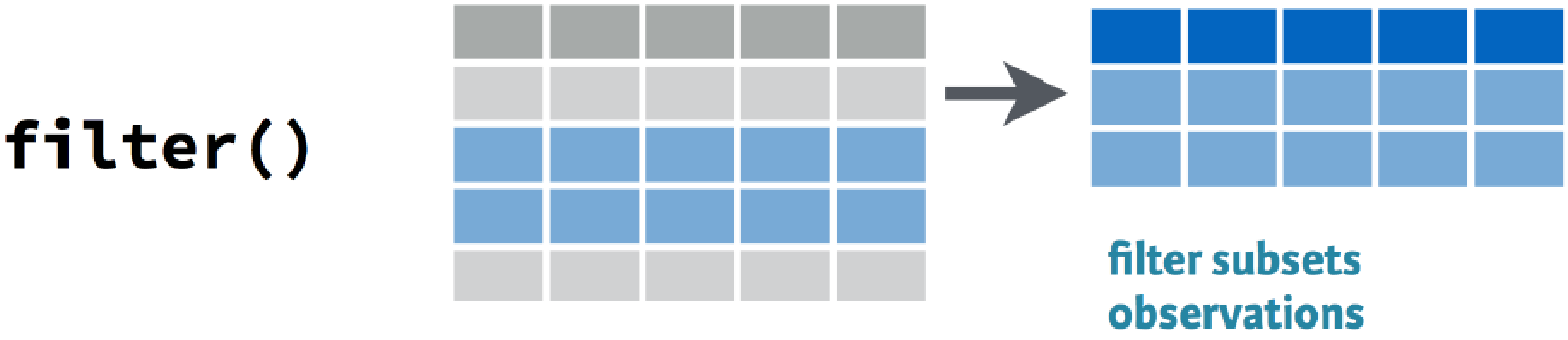
- Flat files
- Data from Excel
- Databases
- Web
- Statistical software

X
PostgreSQL
MySQL
SPSS
sas
STATA
R

Wrapping in utils and readr

utils	readr
read.table()	read_delim()
read.csv()	read_csv()
read.delim()	read_tsv()

Dplyr::filter

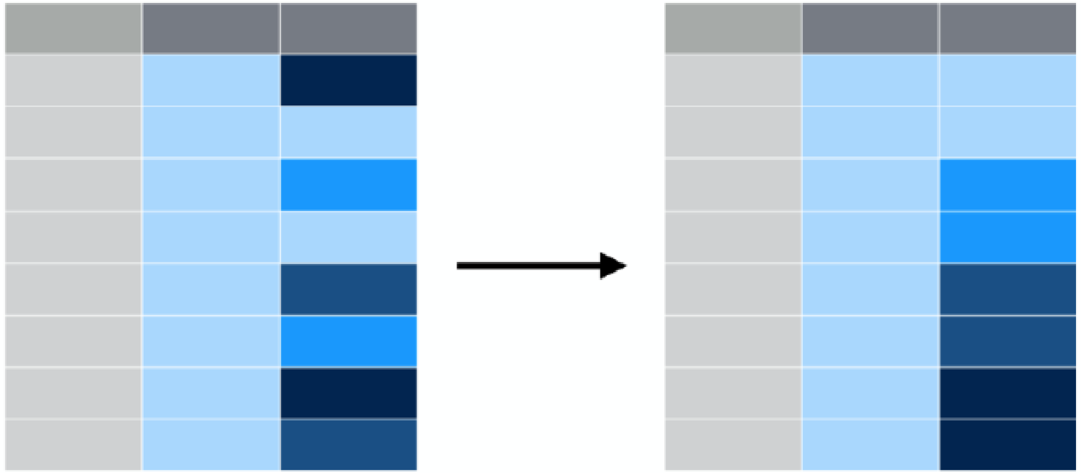


filter()

filter subsets observations

```r
gapminder %>%
  filter(year == 2007, country == "United States")
```

Arrange

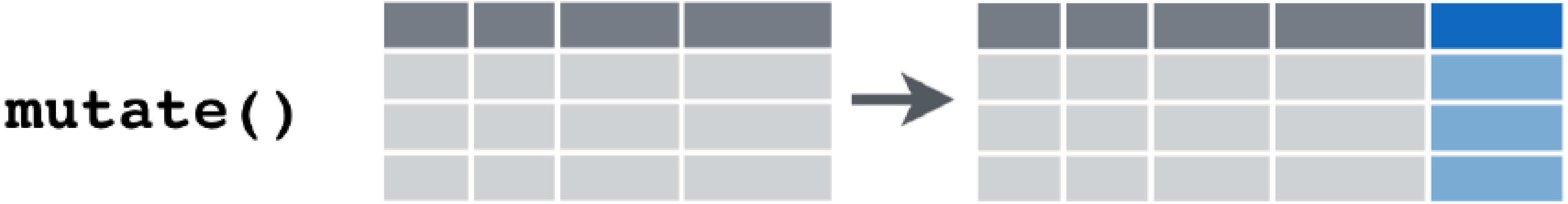


`arrange()` sorts a table based on a variable

```
gapminder %>%
  arrange(gdpPercap)
```

```
gapminder %>%
  arrange(desc(gdpPercap))
```

Mutate



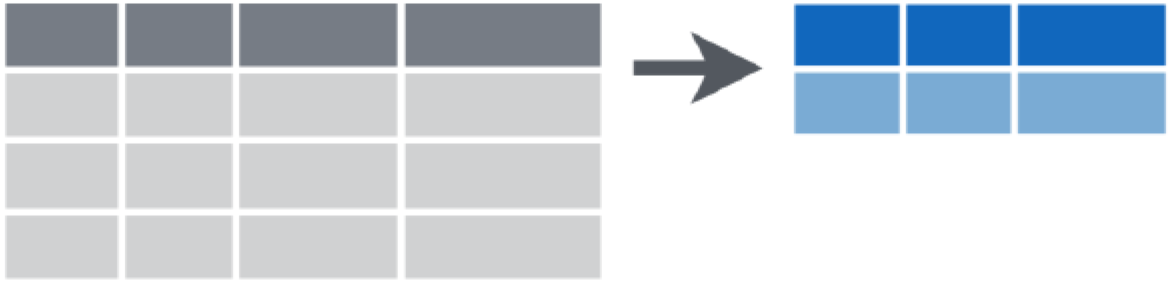
mutate()

mutate changes or adds variables

```
gapminder %>%
  mutate(gdp = gdpPercap * pop)
```

Summarize

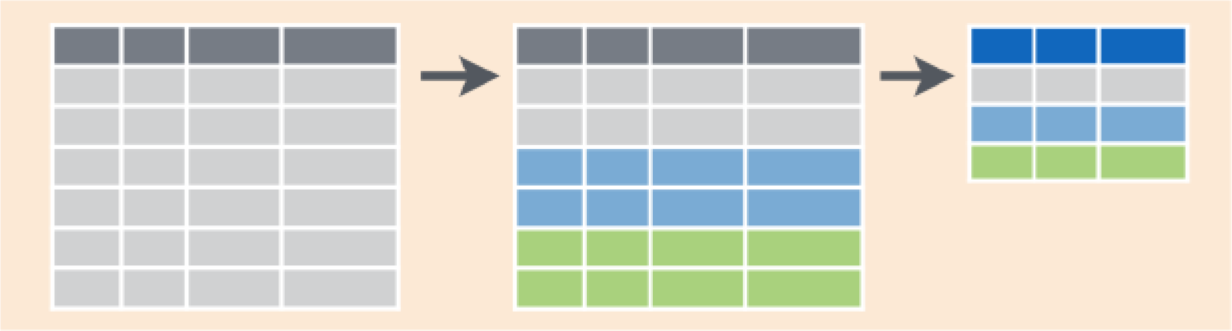
`summarize()` turns many rows into one



```
gapminder %>%
  summarize(meanLifeExp = mean(lifeExp))
```

Group_by

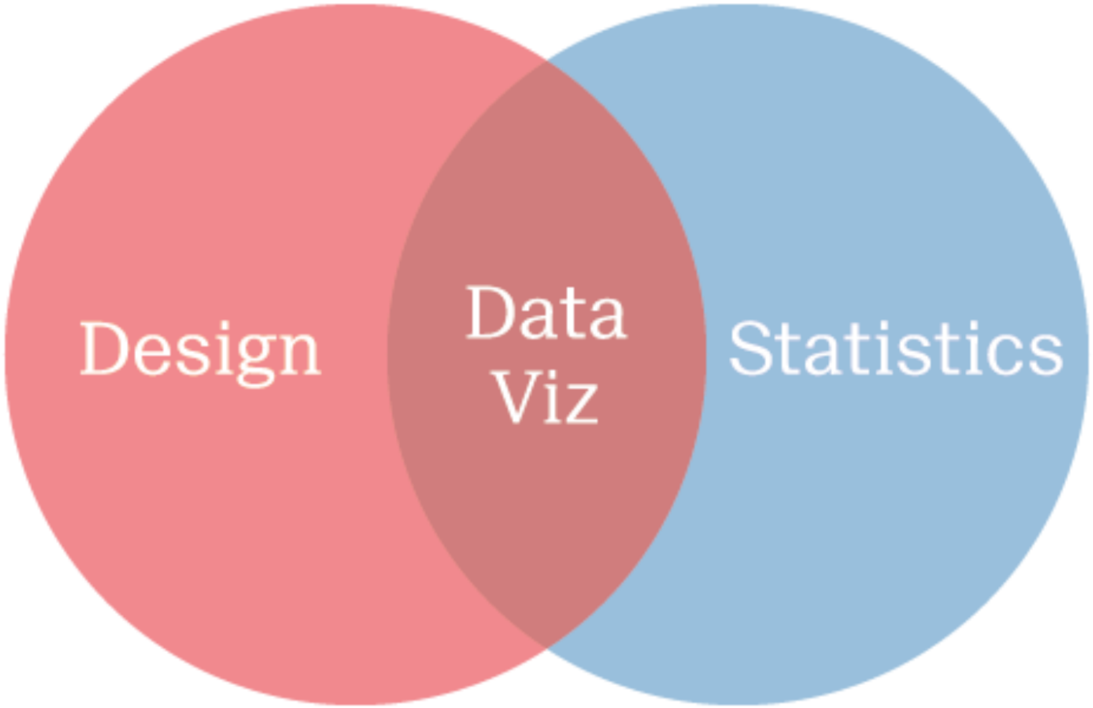
group_by() before summarize() turns groups into one row each



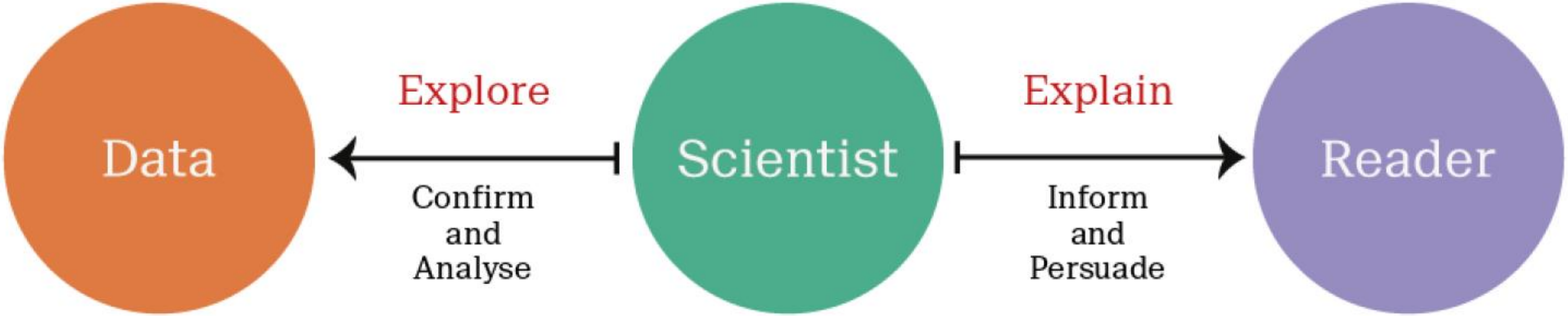
```
gapminder %>%
  group_by(year) %>%
  summarize(meanLifeExp = mean(lifeExp),
            totalPop = sum(pop))
```

Data visualization & data science

- A core skill in Data Science.

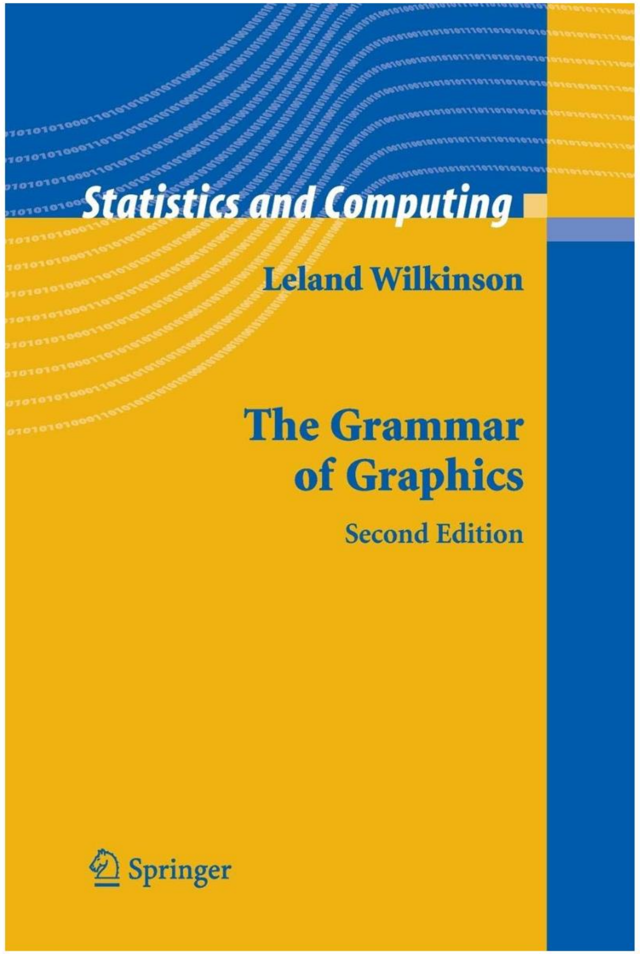


Exploratory vs. Explanatory



Grammar of graphics

- Plotting framework
- Leland Wilkinson, Grammar of Graphics, 1999
- 2 principles
    - Graphics = distinct layers of grammatical elements
    - Meaningful plots through aesthetic mappings



Statistics and Computing

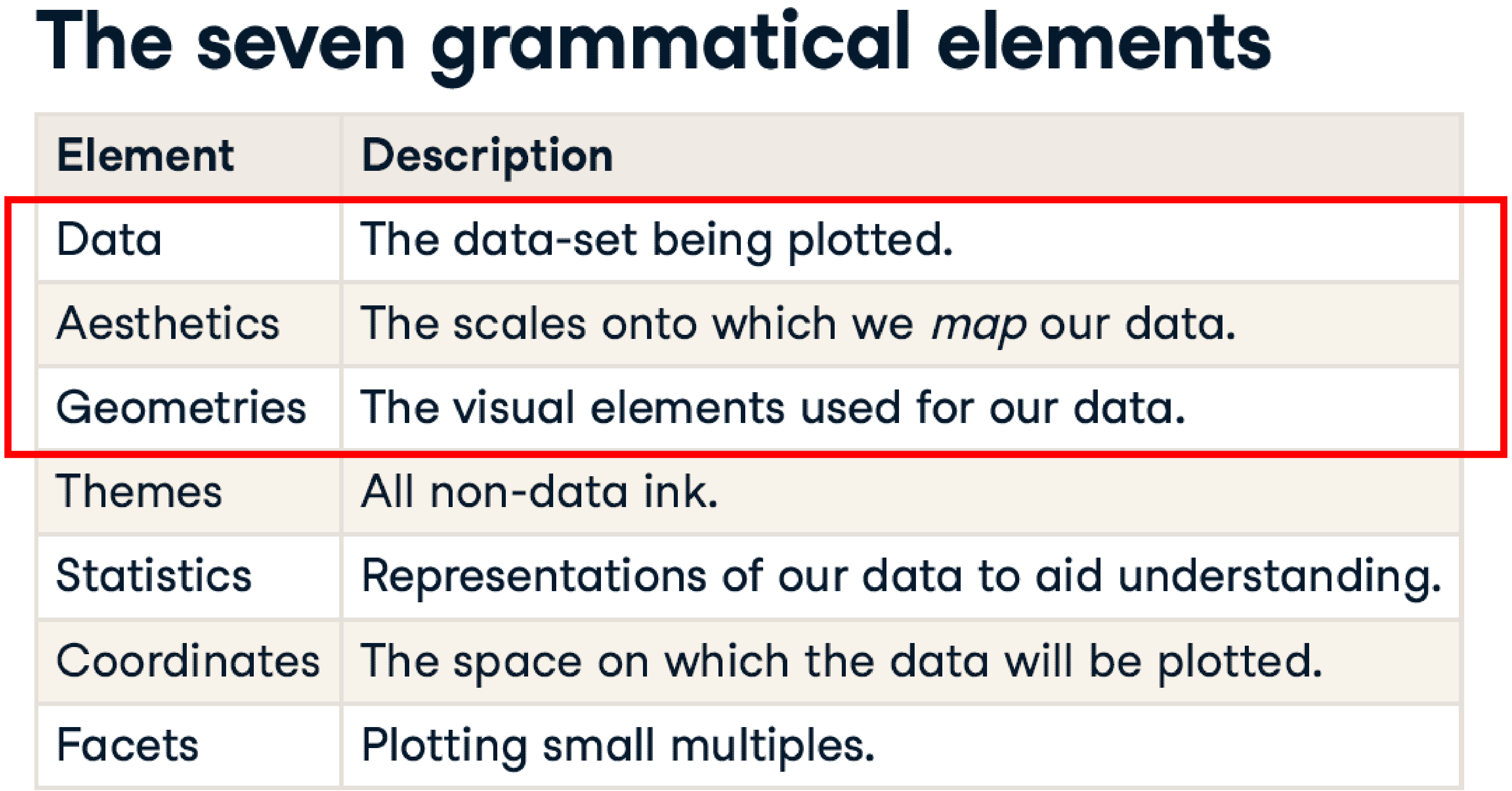
Leland Wilkinson

The Grammar of Graphics

Second Edition

Springer

Grammar of Graphics



The seven grammatical elements

Element	Description
Data	The data-set being plotted.
Aesthetics	The scales onto which we *map* our data.
Geometries	The visual elements used for our data.
Themes	All non-data ink.
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Facets	Plotting small multiples.

Three elememts

Data	*{variables of interest}*				
Aesthetics	*x-axis* *y-axis*	*colour* *fill*	*size* *labels*	*alpha* *shape*	*line width* *line type*
Geometries	*point*	*line*	*histogram*	*bar*	*boxplot*
Themes	*non-data ink*				
Statistics	*binning*	*smoothing*	*descriptive*	*inferential*	
Coordinates	*cartesian*	*fixed*	*polar*	*limits*	
Facets	*columns*	*rows*			

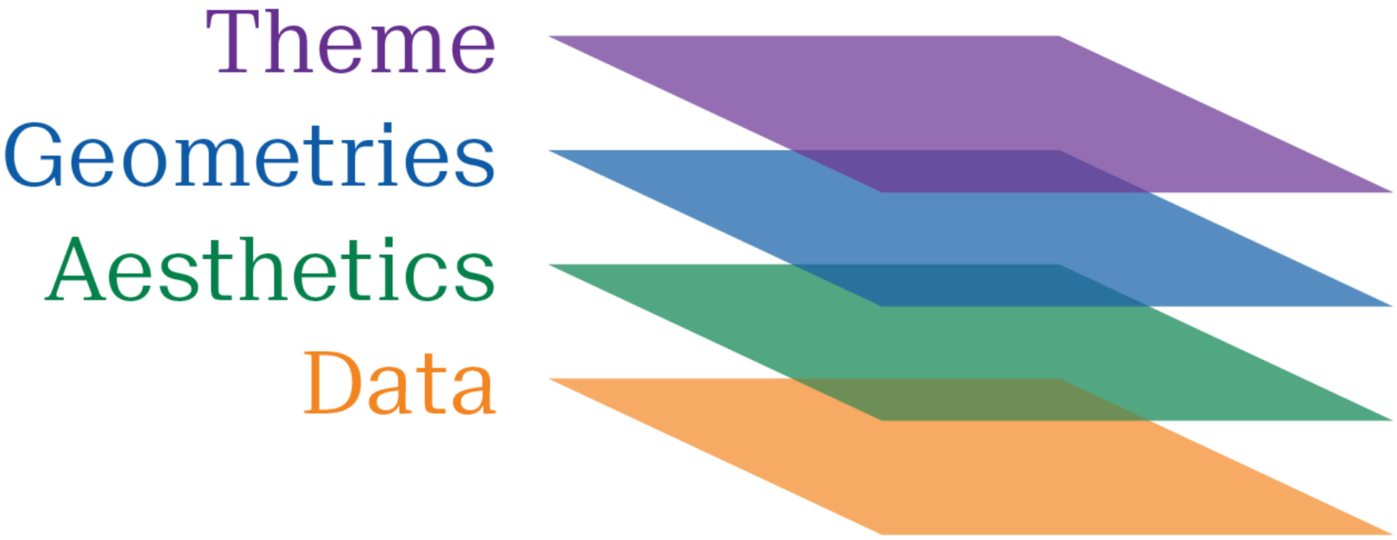
Typical visible aesthetics

Aesthetic	Description
x	X axis position
y	Y axis position
fill	Fill color
color	Color of points, outlines of other geoms
size	Area or radius of points, thickness of lines

Aesthetic	Description
alpha	Transparency
linetype	line dash pattern
labels	Text on a plot or axes
shape	Shape

ggplot2 package

- The grammar of graphics implemented in R
- Two key concepts:
  1. Layer grammatical elements
  2. Aesthetic mappings



Aesthetics? Attributes!

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width)) +
  geom_point(color = "red")
```

position = "jitter"

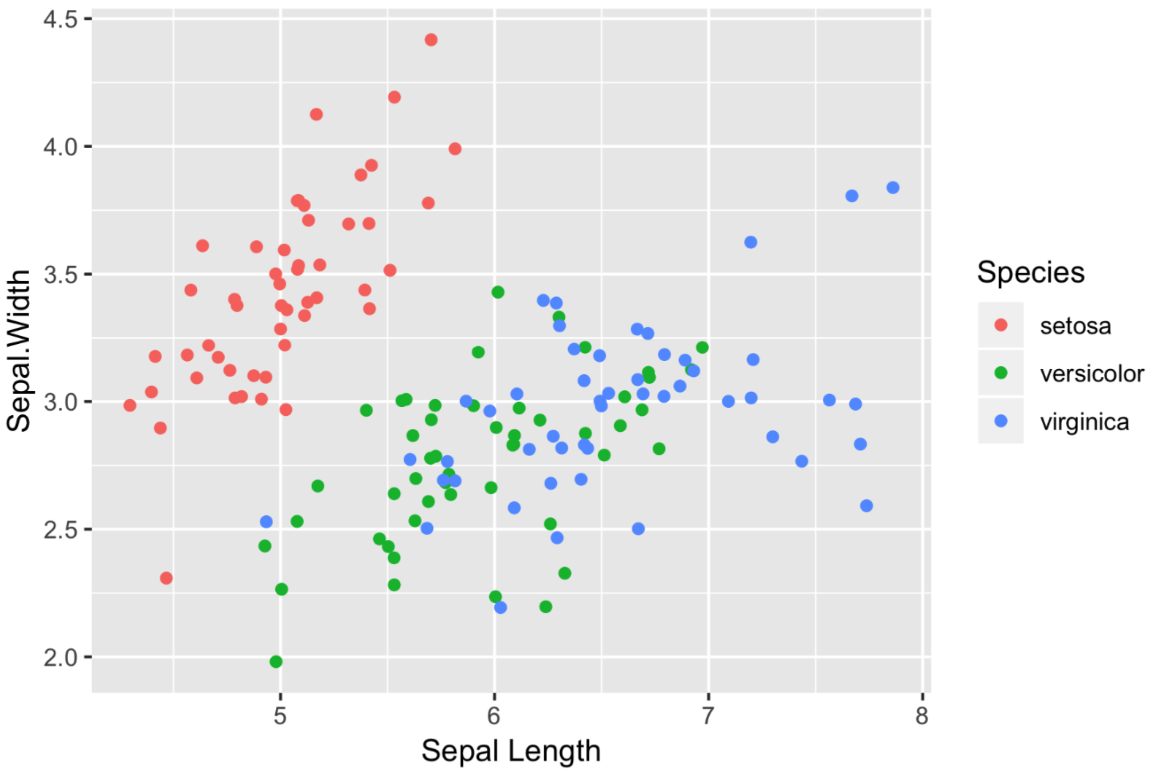
```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point(position = "jitter")
```

Scale functions

- `scale_x_continuous()`
- `scale_y_*()`
- `scale_color_discrete()`
  - Alternatively, `scale_colour_*()`
- `scale_fill_*()`
- `scale_shape_*()`
- `scale_linetype_*()`
- `scale_size_*()`

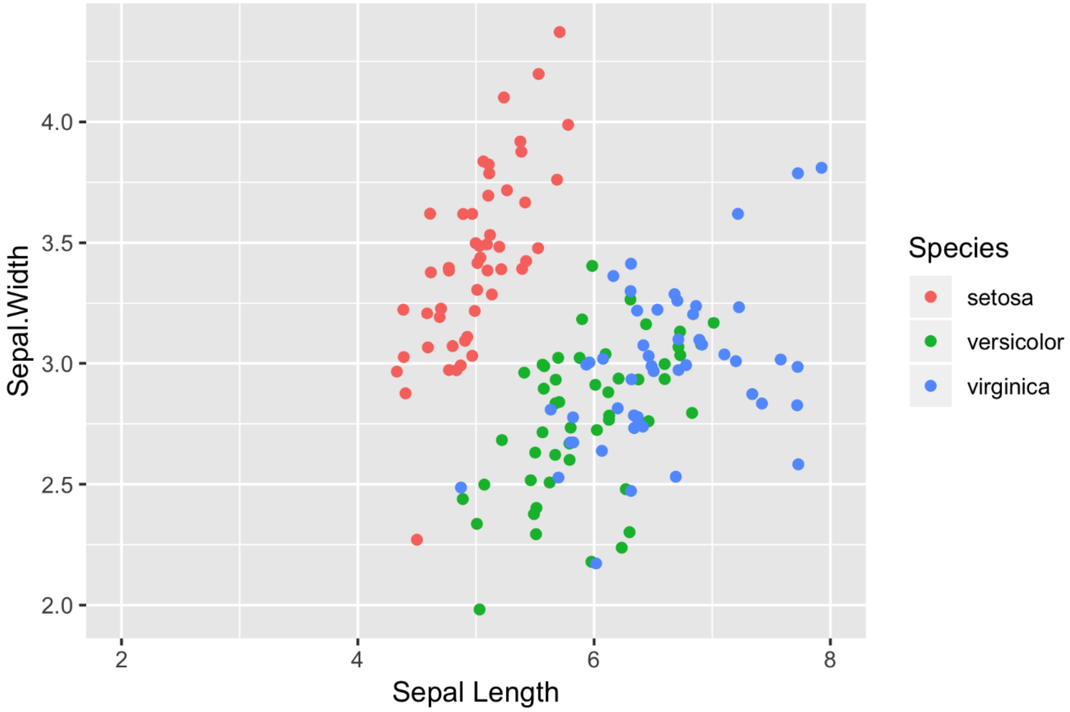
scale_*_*()

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point(position = "jitter") +
  scale_x_continuous("Sepal Length") +
  scale_color_discrete("Species")
```



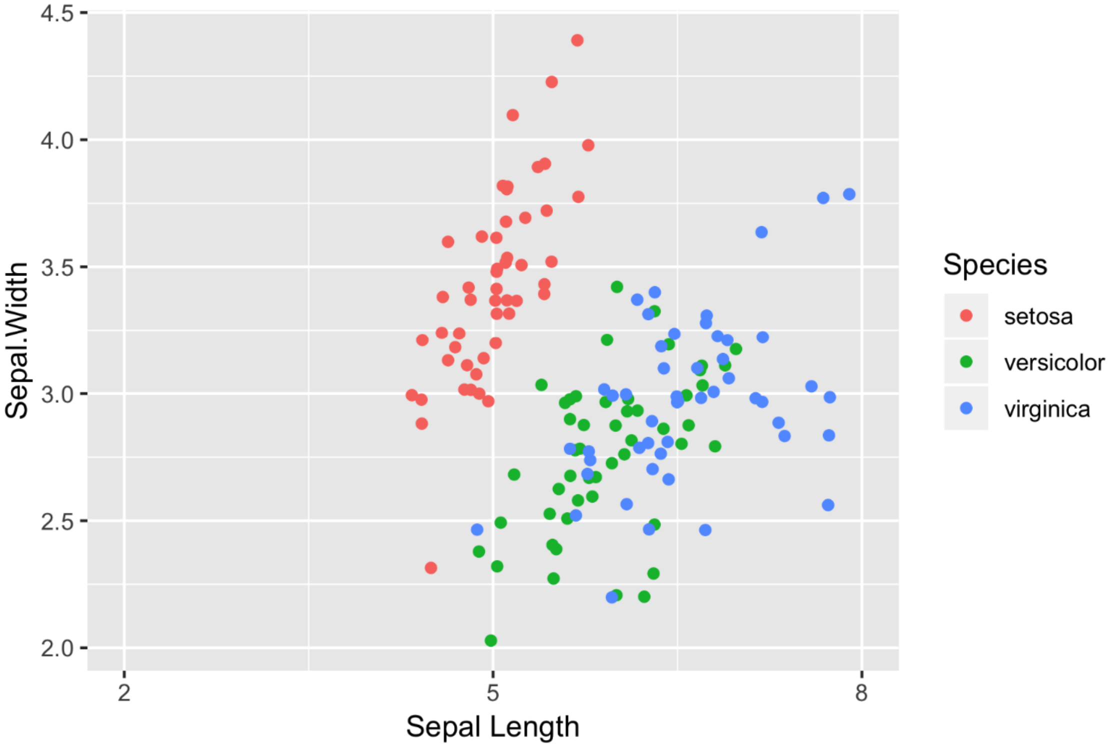
The limits argument

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point(position = "jitter") +
  scale_x_continuous("Sepal Length",
                     limits = c(2,8)) +
  scale_color_discrete("Species")
```



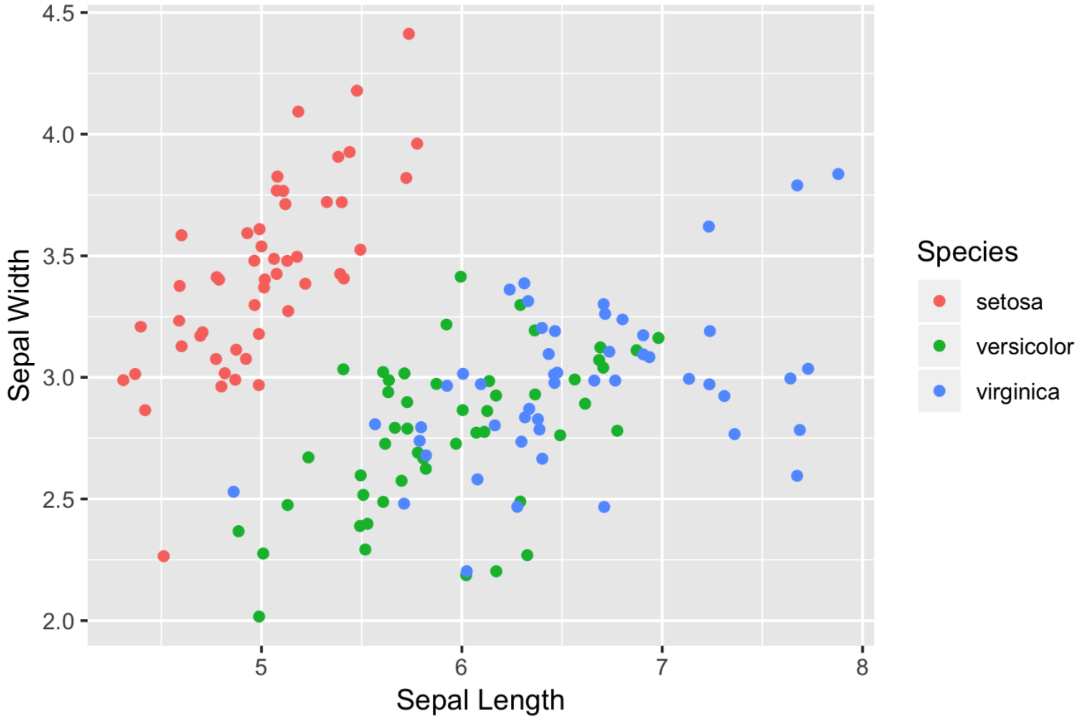
The breaks argument

```
ggplot(iris, aes(x = Sepal.Length,
                  y = Sepal.Width,
                 color = Species)) +
  geom_point(position = "jitter") +
  scale_x_continuous("Sepal Length",
                     limits = c(2, 8),
                     breaks = seq(2, 8, 3)) +
  scale_color_discrete("Species")
```



labs()

```r
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point(position = "jitter") +
  labs(x = "Sepal Length",
       y = "Sepal Width",
       color = "Species")
```

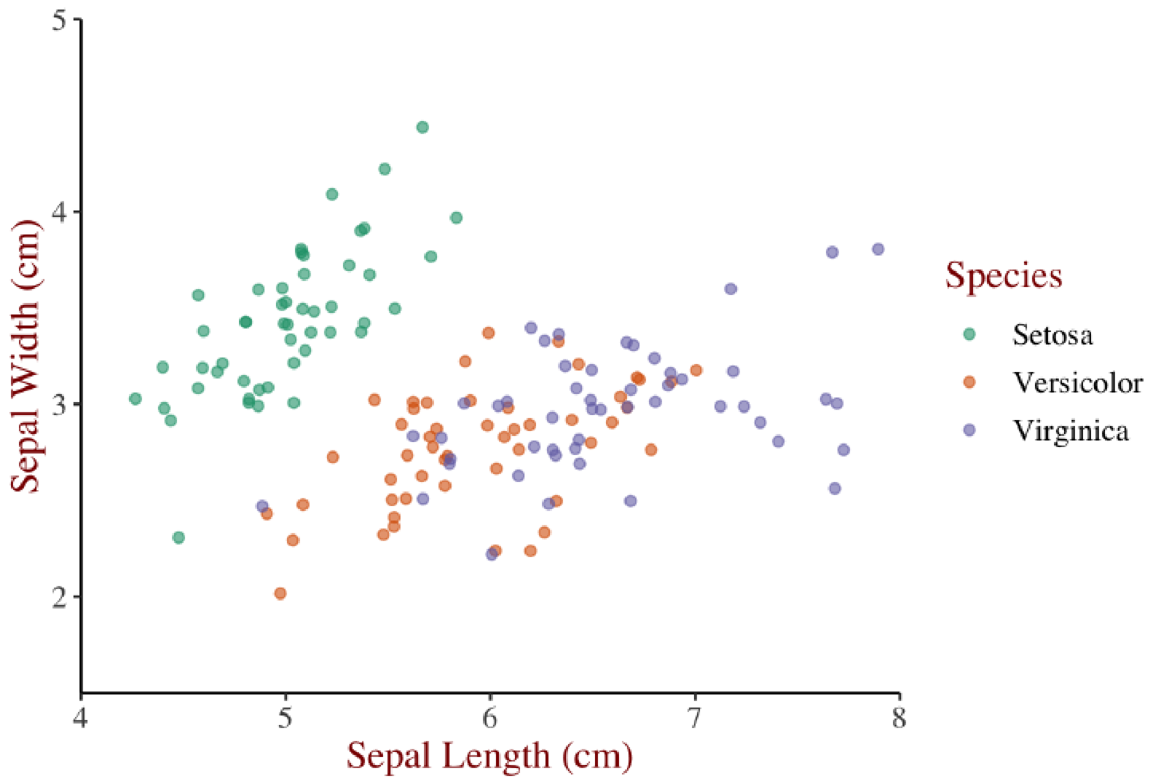


The themes layer

- All non-data ink
- Visual elements not part of the data

Three types

type	modified using
text	element_text()
line	element_line()
rectangle	element_rect()

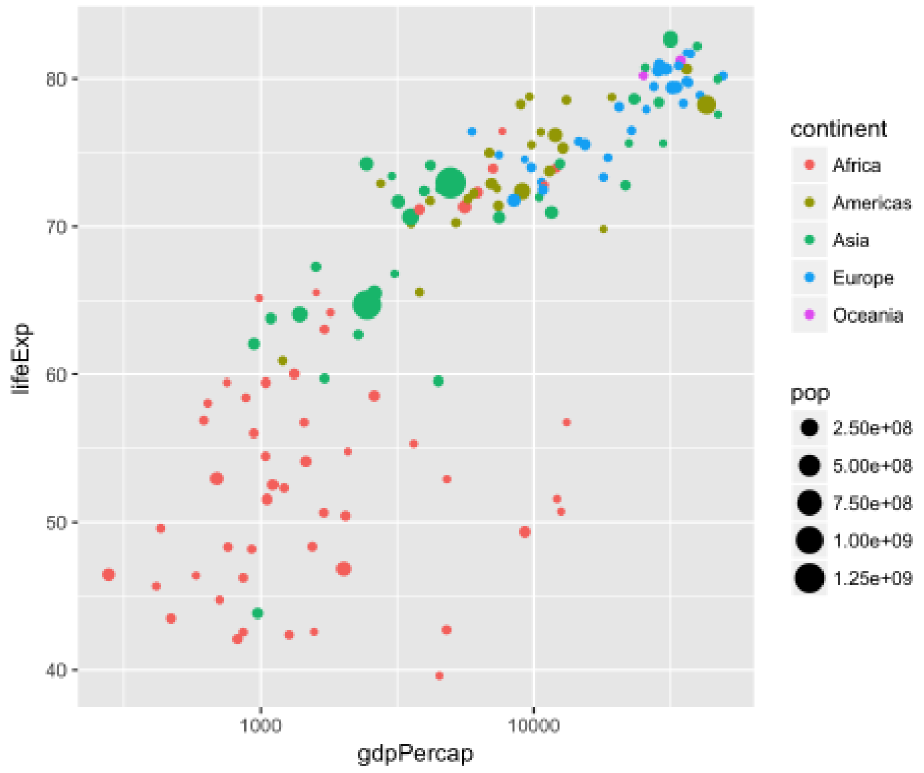


Defining theme objects

```
theme_iris <- theme(text = element_text(family = "serif", size = 14),
          rect = element_blank(),
          panel.grid = element_blank(),
          title = element_text(color = "#8b0000"),
          axis.line = element_line(color = "black"))
```

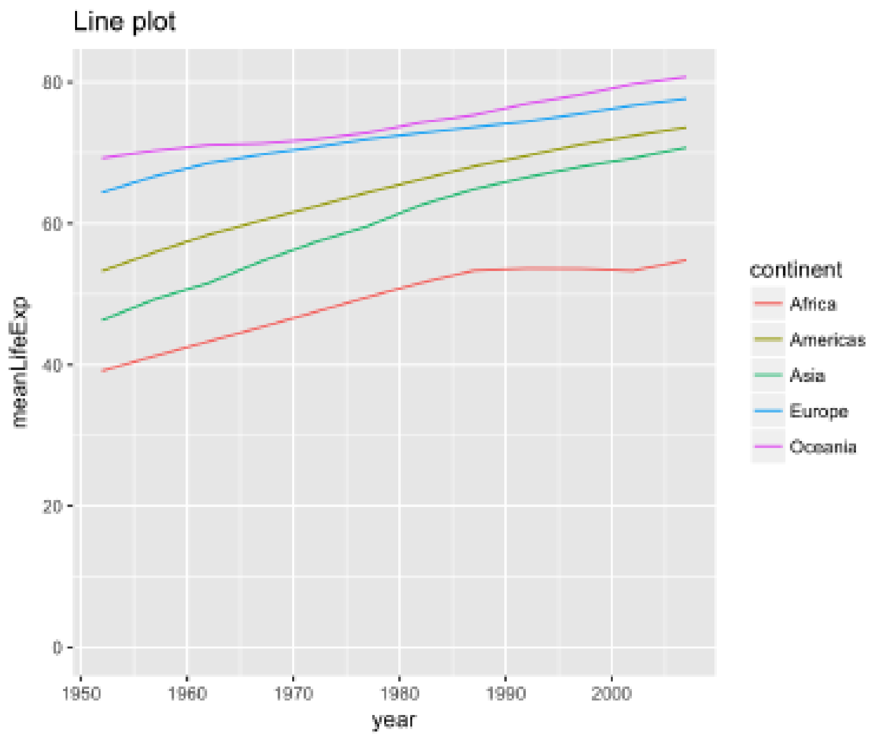
Data visualization: ggplot2 (Scatter plot)

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp, color = continent,
                           size = pop)) +
  geom_point() +
  scale_x_log10()
```



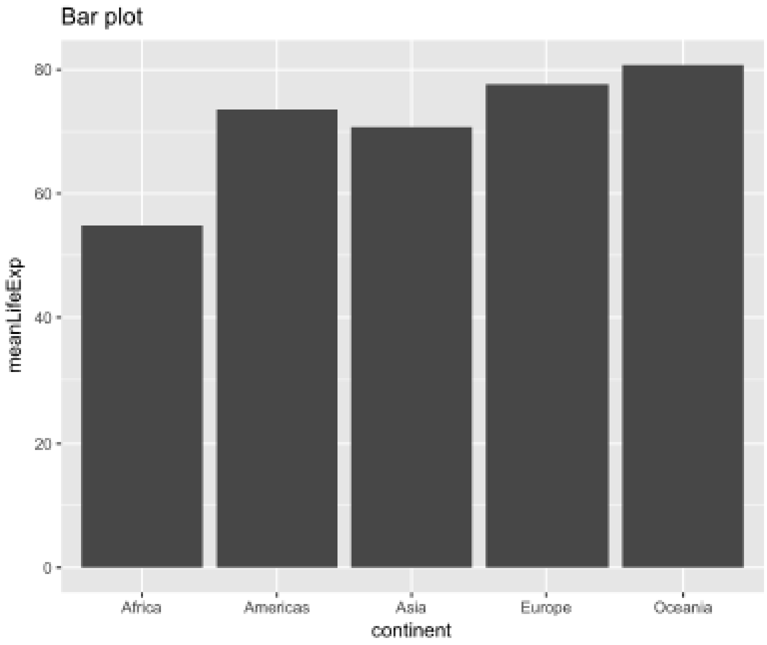
Line plot

```
ggplot(year_continent, aes(x = year, y = meanLifeExp, color = continent)) +
  geom_line() +
  expand_limits(y = 0)
```



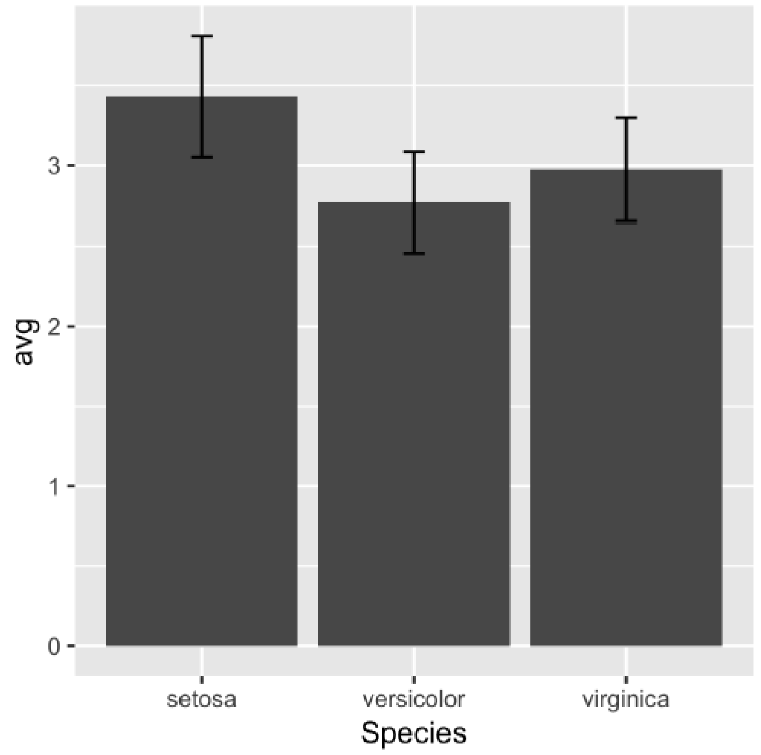
Bar plot

```
ggplot(by_continent, aes(x = continent, y = meanLifeExp)) +
  geom_col()
```



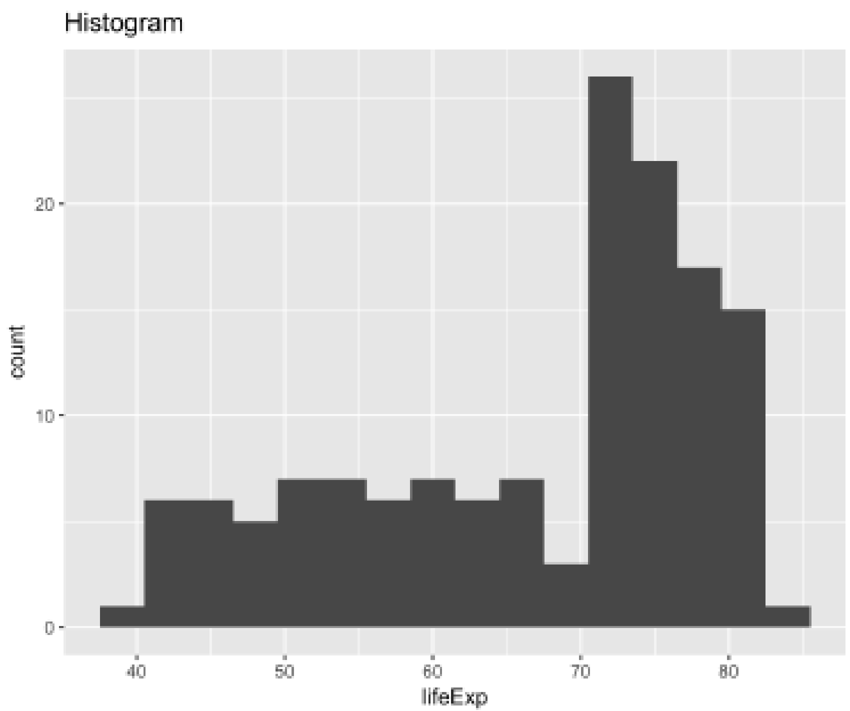
Bar plot w/ error bars

```
ggplot(iris_summ_long, aes(x = Species,
                           y = avg)) +
  geom_col() +
  geom_errorbar(aes(ymin = avg - stdev,
                    ymax = avg + stdev),
                width = 0.1)
```



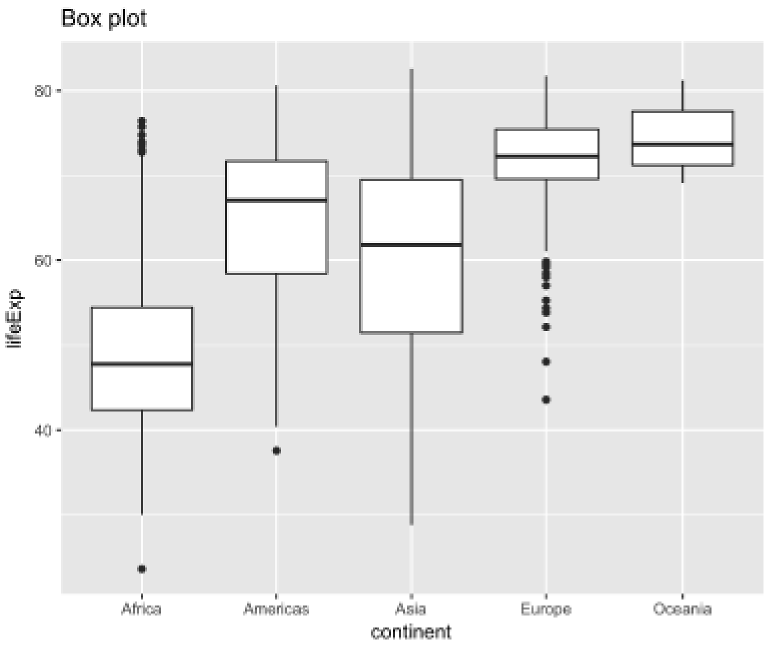
Histogram

```
ggplot(gapminder_2007, aes(x = lifeExp)) +
  geom_histogram()
```



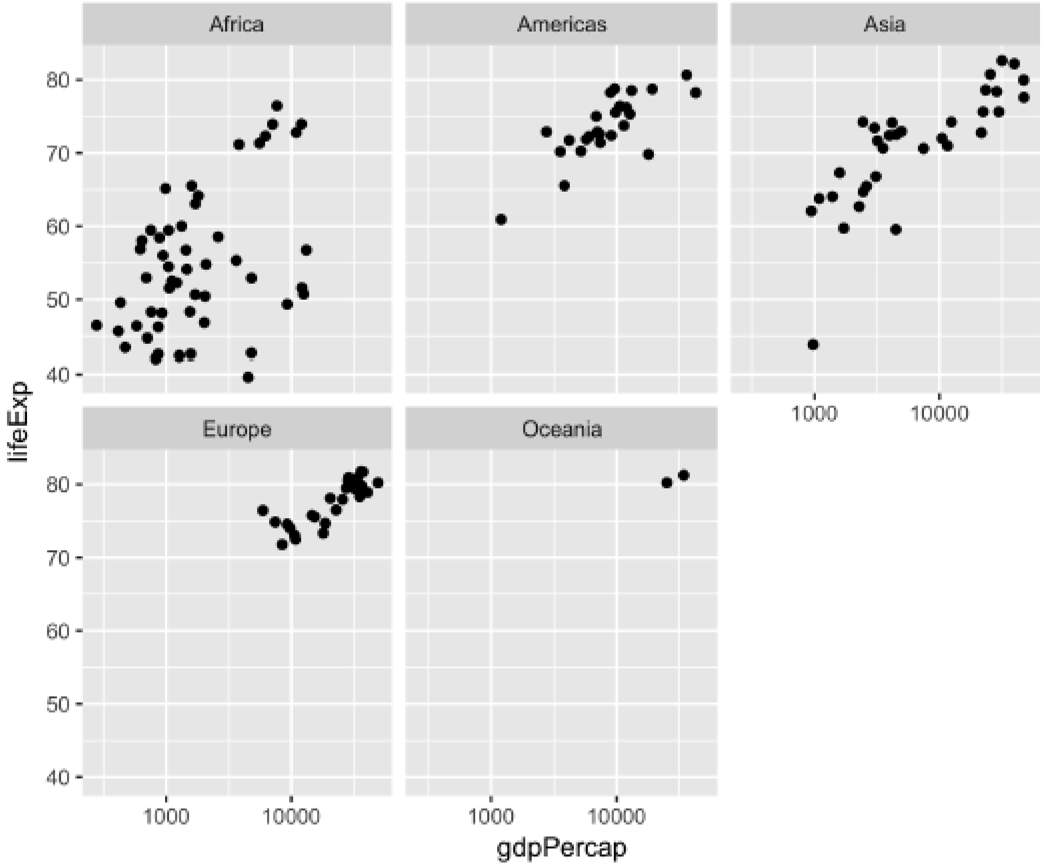
Box plot

```
ggplot(gapminder_2007, aes(x = continent, y = lifeExp)) +
  geom_boxplot()
```

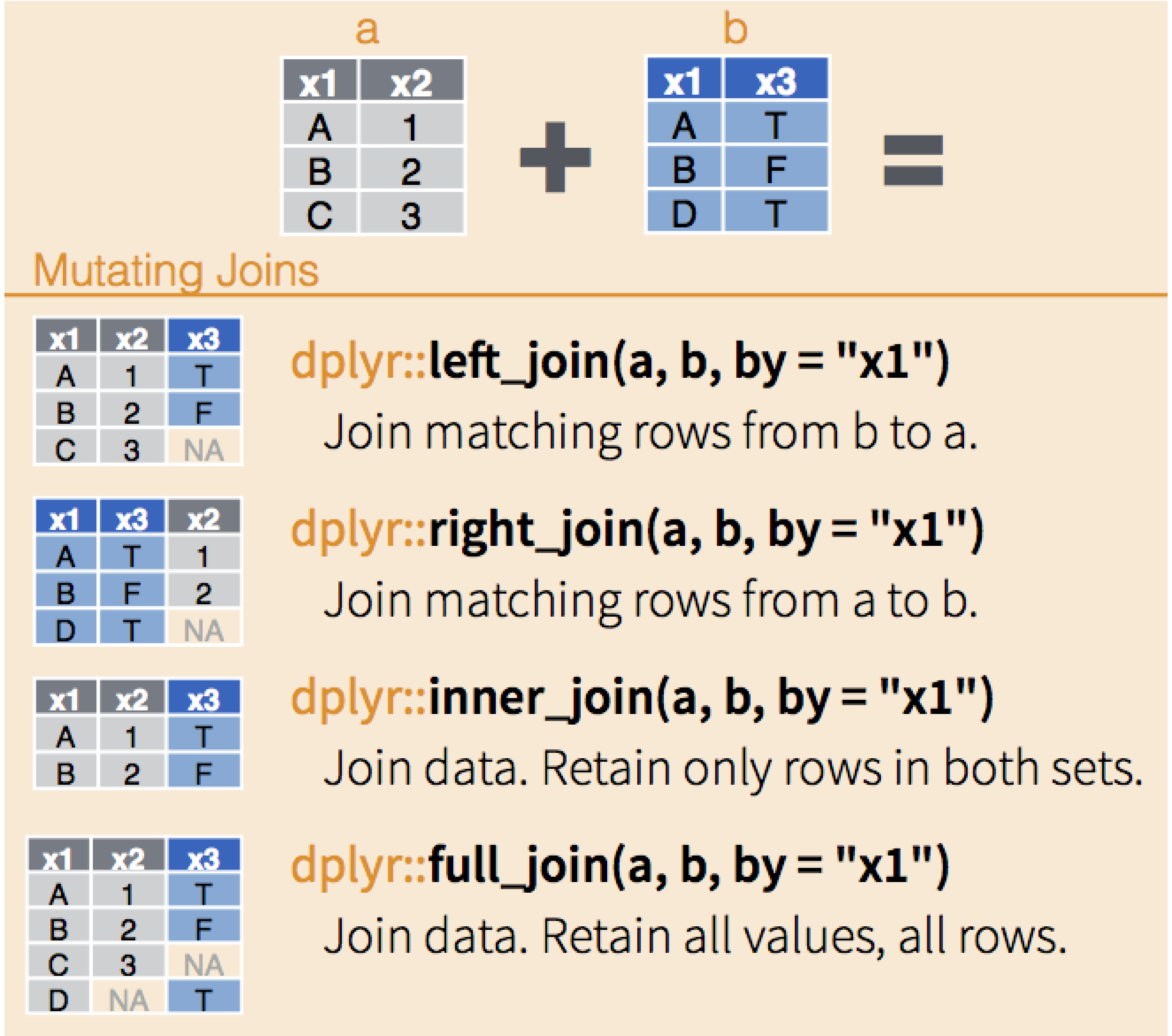


Facet

```
ggplot(gapminder_2007, aes(x = gdpPercap, y = lifeExp)) +
  geom_point() +
  scale_x_log10() +
  facet_wrap(~ continent)
```



Join



a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

=

Mutating Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NA

`dplyr::left_join(a, b, by = "x1")`

Join matching rows from b to a.

x1	x3	x2
A	T	1
B	F	2
D	T	NA

`dplyr::right_join(a, b, by = "x1")`

Join matching rows from a to b.

x1	x2	x3
A	1	T
B	2	F

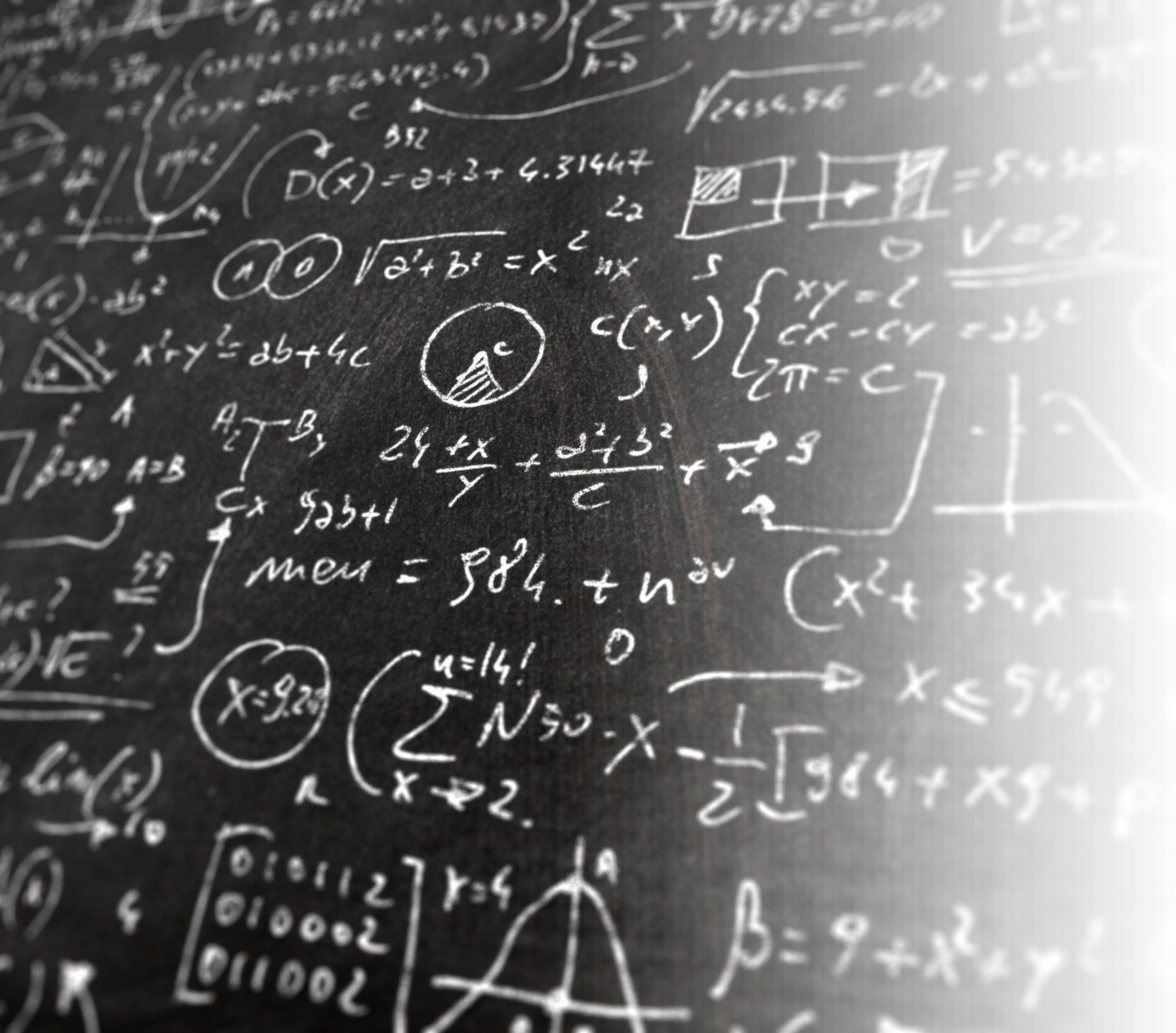
`dplyr::inner_join(a, b, by = "x1")`

Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

`dplyr::full_join(a, b, by = "x1")`

Join data. Retain all values, all rows.



Programming

조건문

if statement

```
if(condition) {
  expr
}
```

```
x <- -3
```

```
if(x < 0) {
  print("x is a negative number")
}
```

```
"x is a negative number"
```

조건문

if, else if, else

```
x <- 6
```

```
if(x %% 2 == 0) {
  print("divisible by 2")
} else if(x %% 3 == 0) {
  print("divisible by 3")
} else {
  print("not divisible by 2 nor by 3...")
}
```

```
"divisible by 2"
```

반복문

for loop

```
for(var in seq) {
  expr
}
```

```
cities <- c("New York", "Paris",
            "London", "Tokyo",
            "Rio de Janeiro", "Cape Town")

for(city in cities) {
  print(city)
}
```

반복문

while loop

```
ctr <- 1
while(ctr <= 7) {
  print(paste("ctr is set to", ctr))
  ctr <- ctr + 1
}
```

```
"ctr is set to 1"
"ctr is set to 2"
...
"ctr is set to 7"
```

```
ctr
```

```
8
```

함수

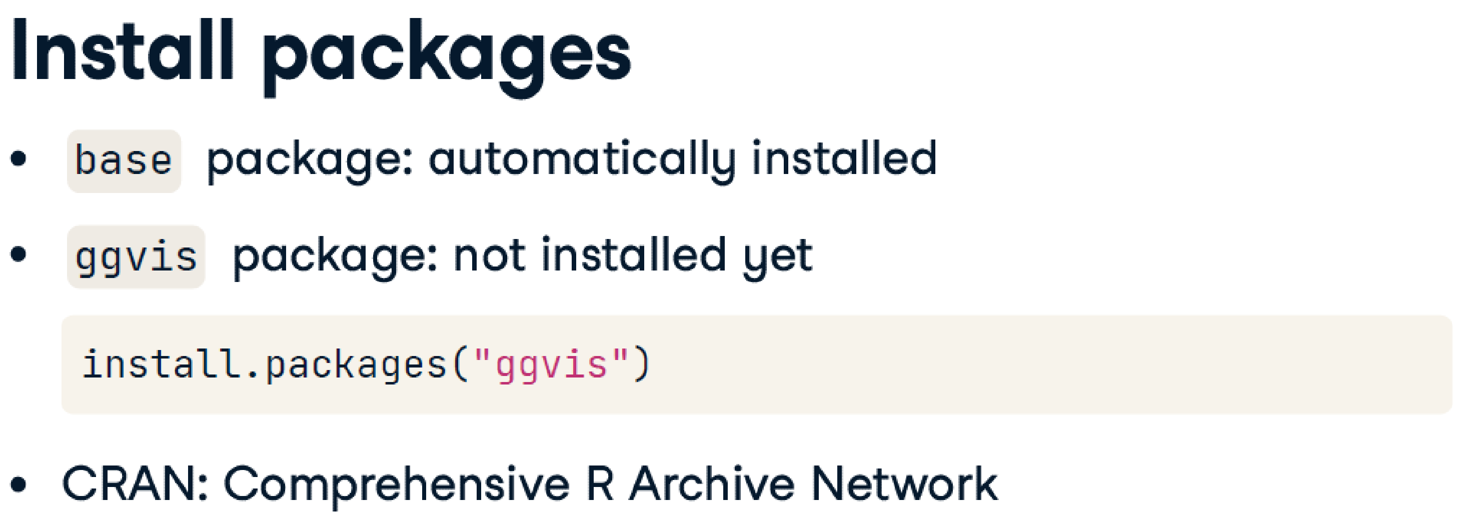
```
math_magic <- function(a, b) {
  a*b + a/b
}
```

```
math_magic(4, 2)
```

```
10
```

- Argument matching: by position or by name
- Function arguments can have defaults

Packages

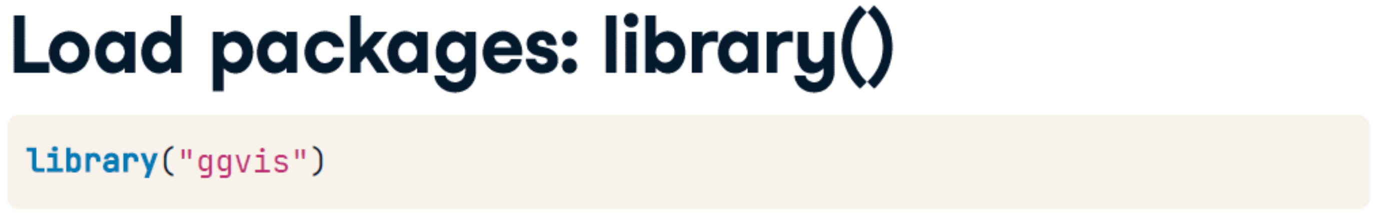


Install packages

- `base` package: automatically installed
- `ggvis` package: not installed yet

```
install.packages("ggvis")
```

- CRAN: Comprehensive R Archive Network



Load packages: library()

```
library("ggvis")
```