R plotting (ggplot2)

Create amazing plots in R

Víctor Granda (@MalditoBarbudo)

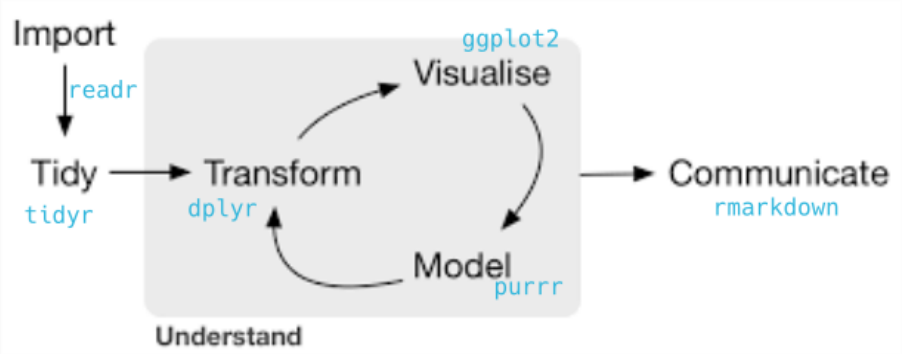
2023-04-25

The tidyverse

The **tidyverse** is a collection of R packages designed for data science, as a suite aimed at easening the data analysis in all its steps.

Created by Hadley Wickham, chief scientist of RStudio, and author of more than 30 R packages (`readr`, `ggplot2`, `plyr`, `devtools`, `roxygen2`, `rmarkdown`...)

All packages share an underlying design philosophy, grammar, and data structures.



`ggplot2`

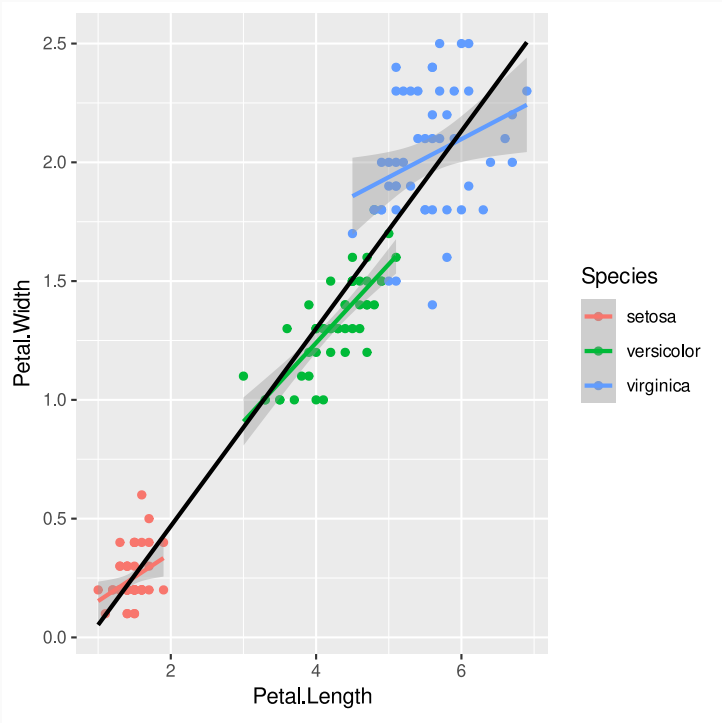
ggplot2

- *ggplot2* package
  - `library(ggplot2)`
- Based on Leland Wilkinson's "The Grammar of Graphs"
- Automatically in charge of plot formatting (text, titles, margins, colors...)
  - It does a lot of thing by default
  - But they can be changed as we want
- *Easy* to use

ggplot2

```r
# Load library
library(ggplot2)
# Draw the plot
plot1 ← ggplot(
  data = iris,
  aes(
    x = Petal.Length,
    y = Petal.Width,
    color = Species
  )
) +
  geom_point() +
  stat_smooth(method = lm) +
  stat_smooth(
    mapping = aes(
      x = Petal.Length,
                  y = Petal.Width
    ),
    data = iris,
    method = lm,
    color = 'black',
    se = FALSE
  )
# see the plot
plot1
```

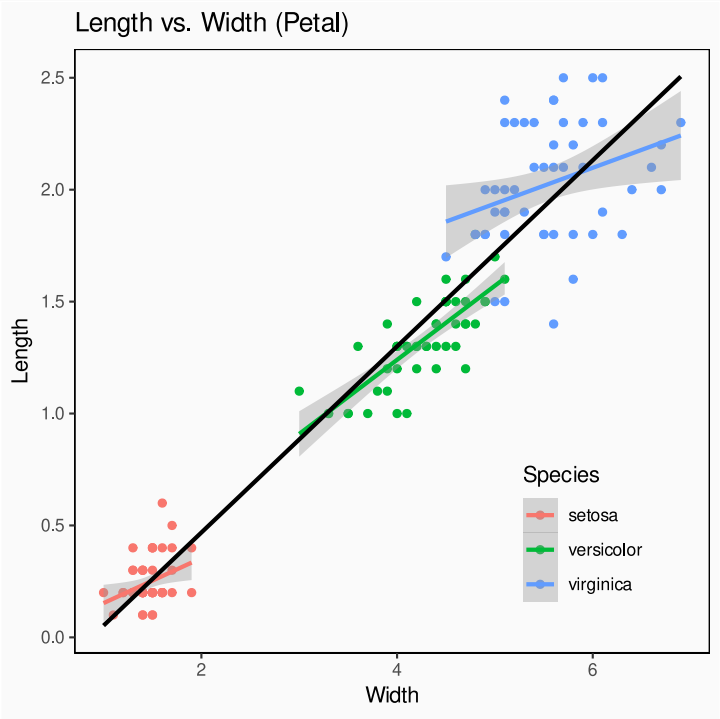
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



ggplot2

```
plot1 +
  theme_bw() +
  labs(
    title = 'Length vs. Width (Petal)',
       x = 'Width', y = 'Length'
  ) +
  theme(plot.background = element_rect(fill =
        panel.background = element_rect(fill =
        legend.background = element_rect(fill
        legend.key = element_rect(fill = 'trar
        panel.grid = element_blank(),
        panel.border = element_rect(colour = '
        legend.position = c(.8,.2))
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



ggplot2

Based on Leland Wilkinson's "The Grammar of Graphs":

> In summary, the grammar of graphs says that an statistical plot consists on **mapping** the data to **aesthetics** attributes (position, colour, shape, size...) of **geometric objects** (points, lines, bars). Plots may also contain statistical transformations (logs, smooths...) and an specific coordinate system (cartesian, polar...)

ggplot2

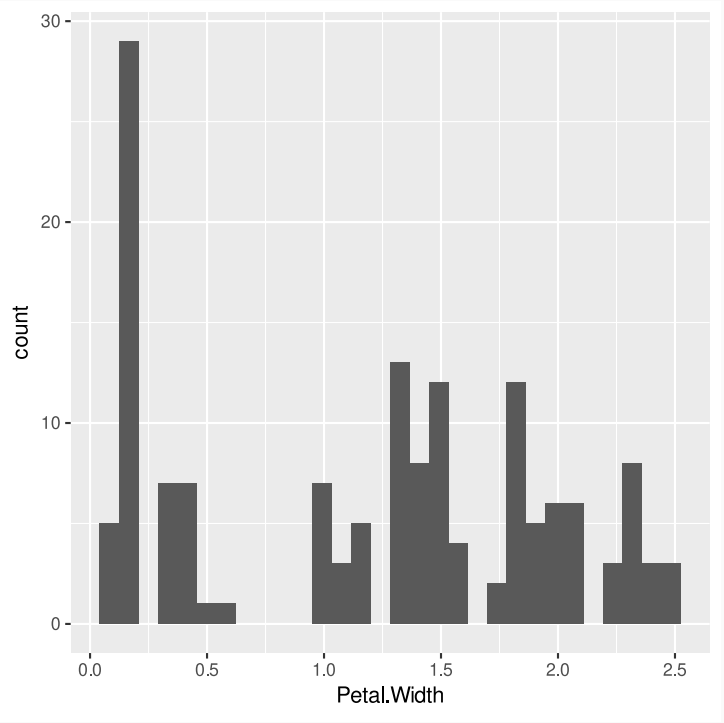
- **Main components of a ggplot plot**
  - Data: a data.frame
  - *aesthetics*: How and to where mapping the data
  - *geom*: geometric objects mapped
  - *facets*: conditional panels
  - *stats*: statistical transformations (ablines, histograms...)
  - *scales*: mapping scales (color, sizes, axes)
  - coordinate system
  - *themes*: predefined and custom themes and modifications

Plot examples

Histogram

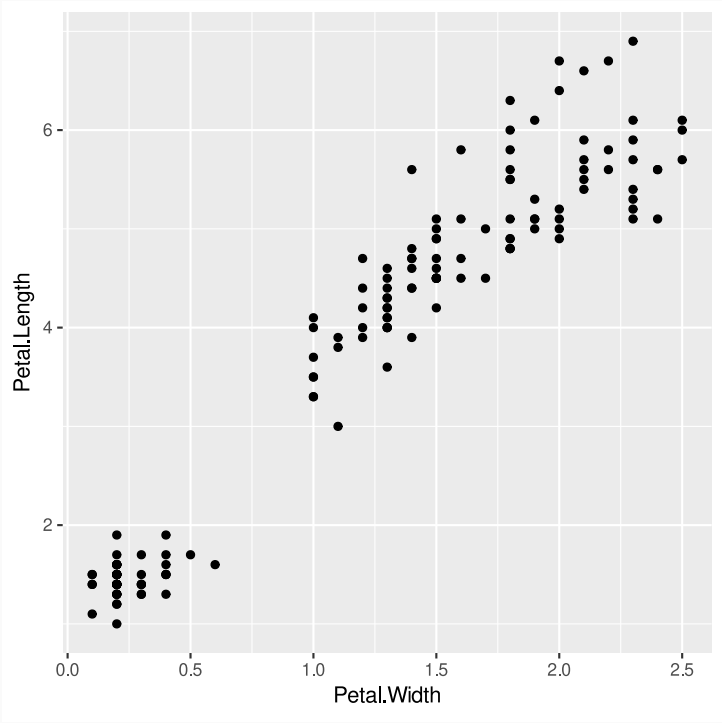
```r
# Load the library and the data
# install.packages('ggplot2', dep=TRUE)
library(ggplot2)

# Histogram of petal width
iris_histogram ←
  # data and aesthetics layers
  ggplot(data = iris, aes(x = Petal.Width)) +
  # geometry layer
  geom_histogram()

# Call the plot to see it
iris_histogram
```

```
## `stat_bin()` using `bins = 30`. Pick better
```



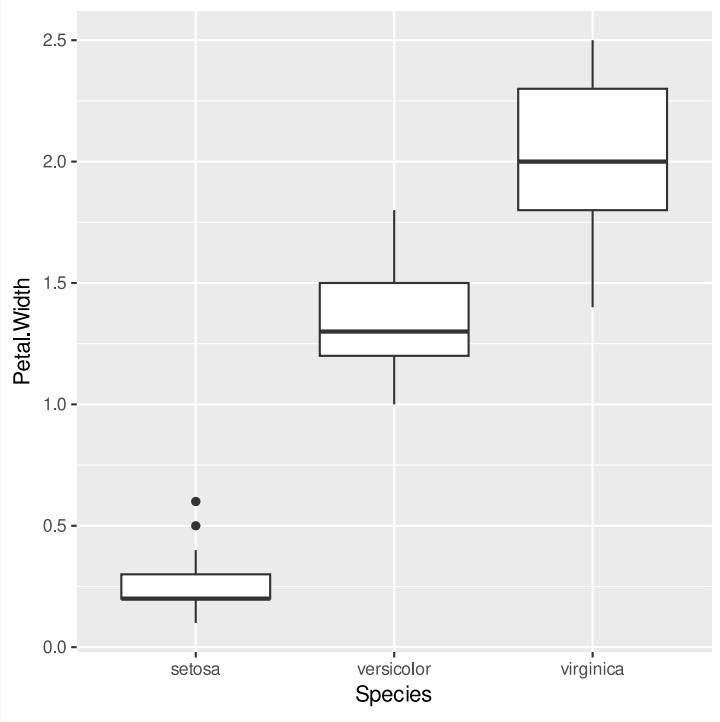
Points

```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(x = Petal.Width, y = Petal.Length)
  ) +
  # geom layer
  geom_point()

# Call the plot to see it
iris_points
```



Box-plots

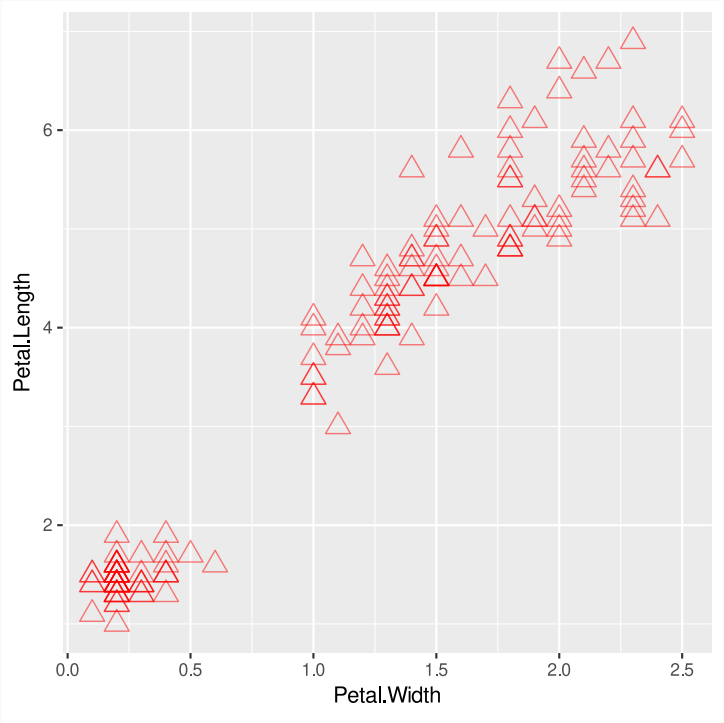
```
iris_boxplot ←
  ggplot(
    data = iris,
    aes(x = Species, y = Petal.Width)
  ) +
  geom_boxplot()

# Call the plot to see it
iris_boxplot
```



More aesthetics

Colors, sizes, transparency and shapes can be **fixed** in the geometry:

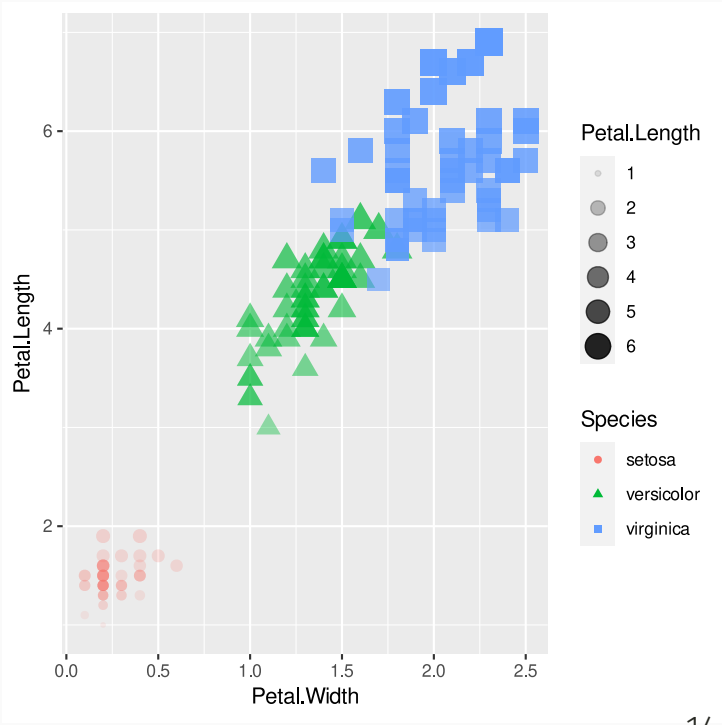
```
ggplot(
  data = iris,
  aes(x = Petal.Width, y = Petal.Length)
) +
  geom_point(
    color = 'red', size = 4,
    alpha = .5, shape = 2
  )
```



More aesthetics

Colors, sizes, transparency and shapes can be fixed in the geometry, or they can be **mapped** to data variables:

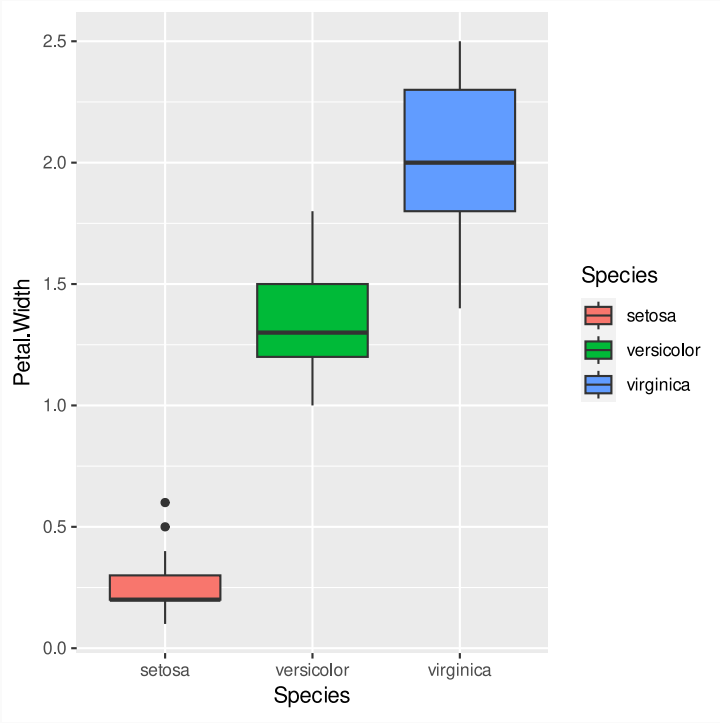
```
ggplot(
  data = iris,
  aes(x = Petal.Width, y = Petal.Length)
) +
  geom_point(
    aes(
      colour = Species, size = Petal.Length,
      alpha = Petal.Length, shape = Species
    )
  )
```



More aesthetics

For some geometries, instead of `colour` we need to use `fill`:

```
iris_boxplot ←
  ggplot(
    data = iris,
    aes(x = Species, y = Petal.Width)
  ) +
  geom_boxplot(aes(fill = Species))

# Call the plot to see it
iris_boxplot
```



More aesthetics

Where to set the *aesthetics*?

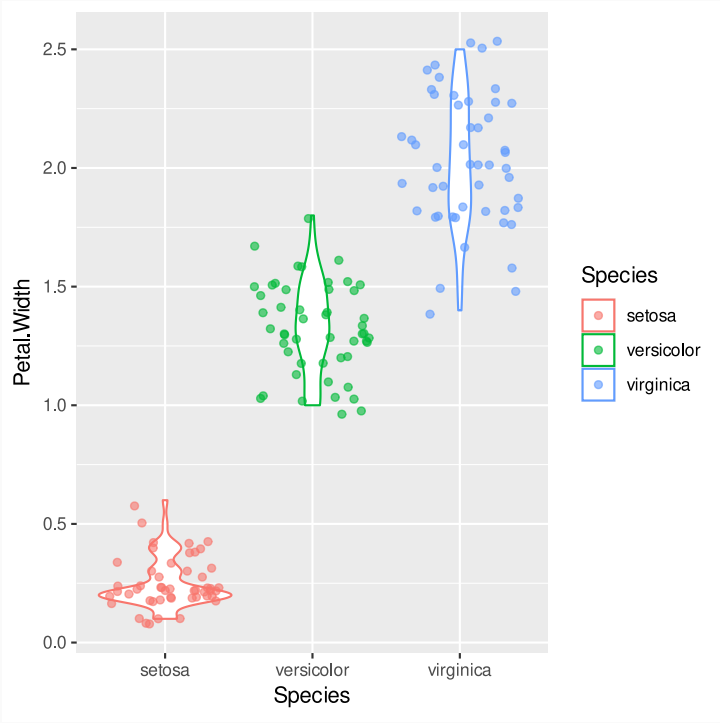
```
iris_boxplot ←
  ggplot(
    data = iris,
    aes(x = Species, y = Petal.Width, fill = S
  ) +
  geom_boxplot()

# Call the plot to see it
iris_boxplot
```

```
iris_boxplot_2 ←
  ggplot(data = iris) +
  geom_boxplot(aes(
    x = Species, y = Petal.Width, fill = Speci
  ))

# Call the plot to see it
iris_boxplot_2
```

More aesthetics

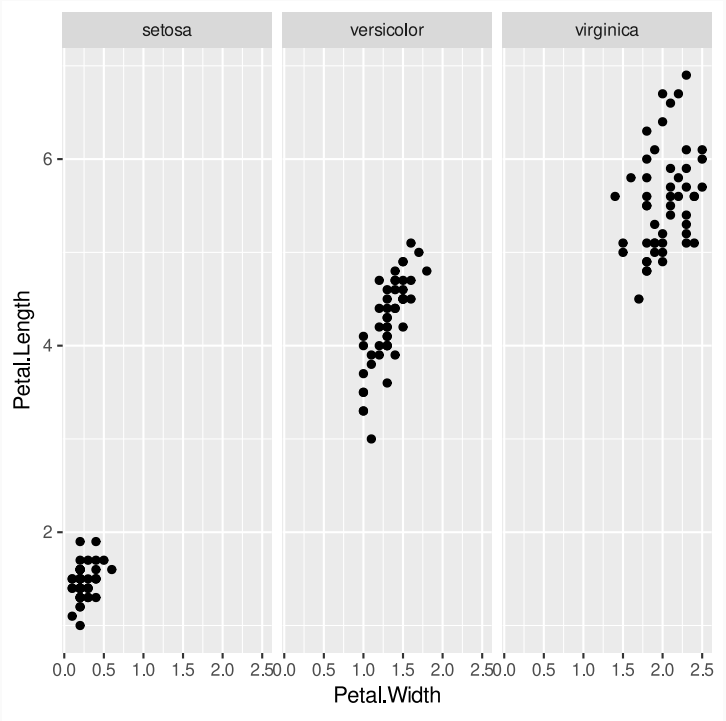
Where to set the *aesthetics*?

```
iris_boxplot ←
  ggplot(
    data = iris,
    aes(x = Species, y = Petal.Width, colour =
  ) +
  geom_violin() +
  geom_point(position = 'jitter', alpha = 0.6)

# Call the plot to see it
iris_boxplot
```



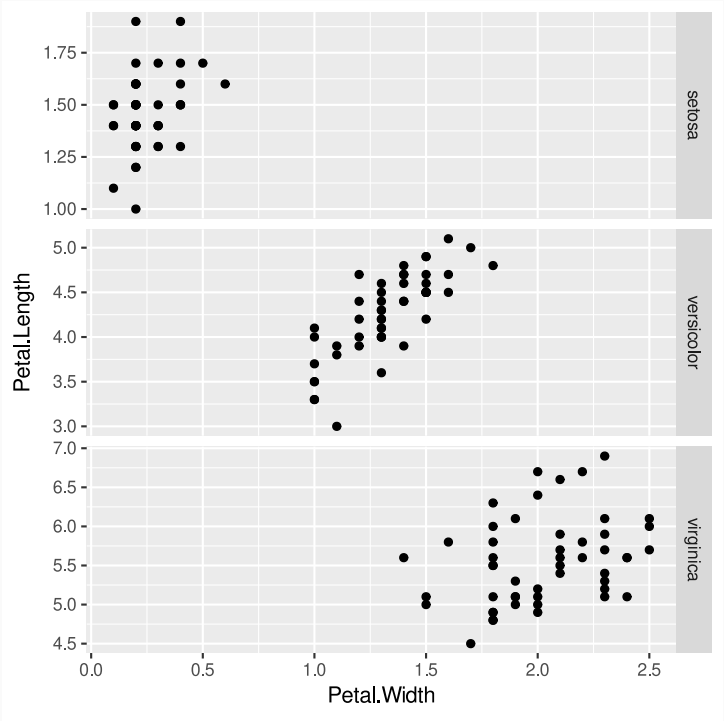
Facets

```
iris_facets ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(x = Petal.Width, y = Petal.Length)
  ) +
  # geom layer
  geom_point() +
  # facets layer
  facet_grid(cols = vars(Species))

# Call the plot to see it
iris_facets
```



Facets

```
iris_facets ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(x = Petal.Width, y = Petal.Length)
  ) +
  # geom layer
  geom_point() +
  # facets layer
  facet_grid(
    rows = vars(Species),
    scales = 'free'
  )

# Call the plot to see it
iris_facets
```

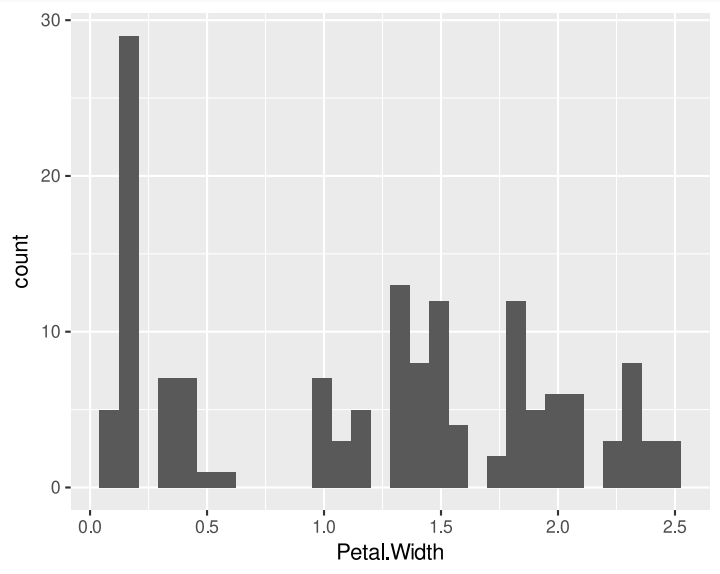


Statistical transformations

Automatic ones

```
# Load the library and the data
# install.packages('ggplot2', dep=TRUE)
library(ggplot2)

# Histogram of petal width
iris_histogram ←
  # data and aesthetics layers
  ggplot(data = iris, aes(x = Petal.Width)) +
  # geometry layer
  geom_histogram()

# Call the plot to see it
iris_histogram
```

```
## `stat_bin()` using `bins = 30`. Pick better
```

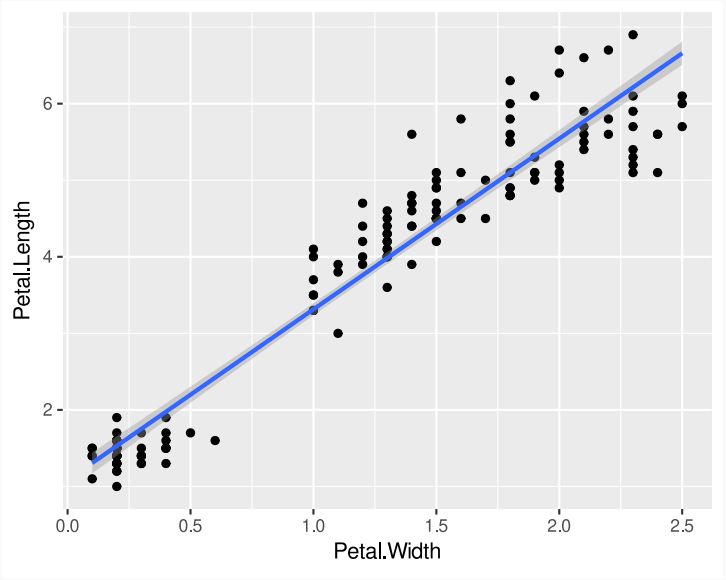


Statistical transformations

Declarative ones:

```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(x = Petal.Width, y = Petal.Length)
  ) +
  # geom layer
  geom_point() +
  # stats layer
  stat_smooth(method = 'lm')

# Call the plot to see it
iris_points
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

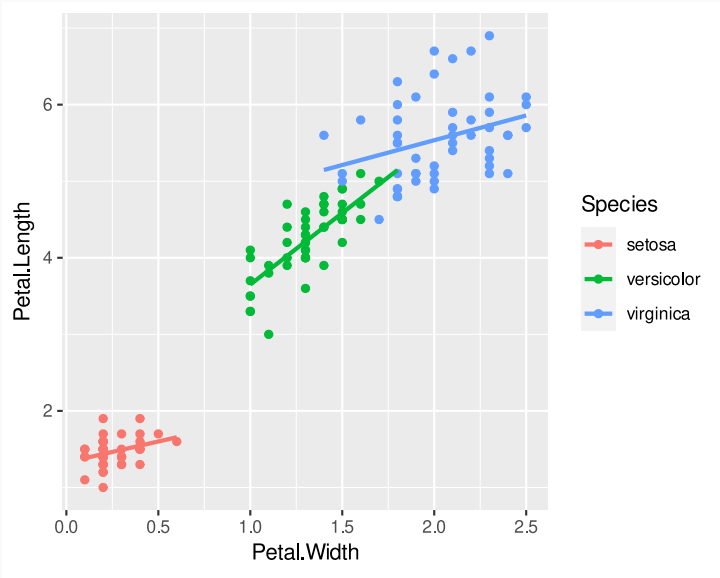


Statistical transformations

Declarative ones:

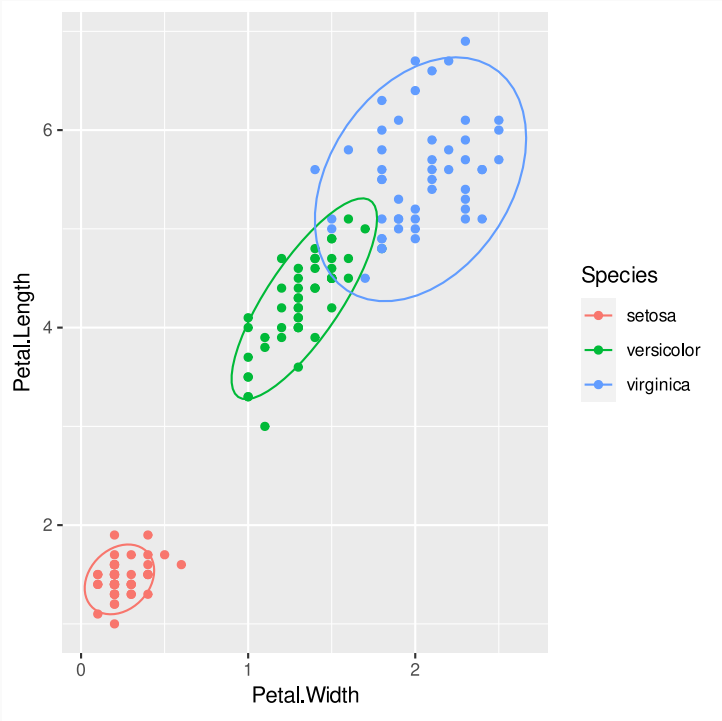
```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # stats layer
  stat_smooth(method = 'lm', se = FALSE)

# Call the plot to see it
iris_points
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Other stats

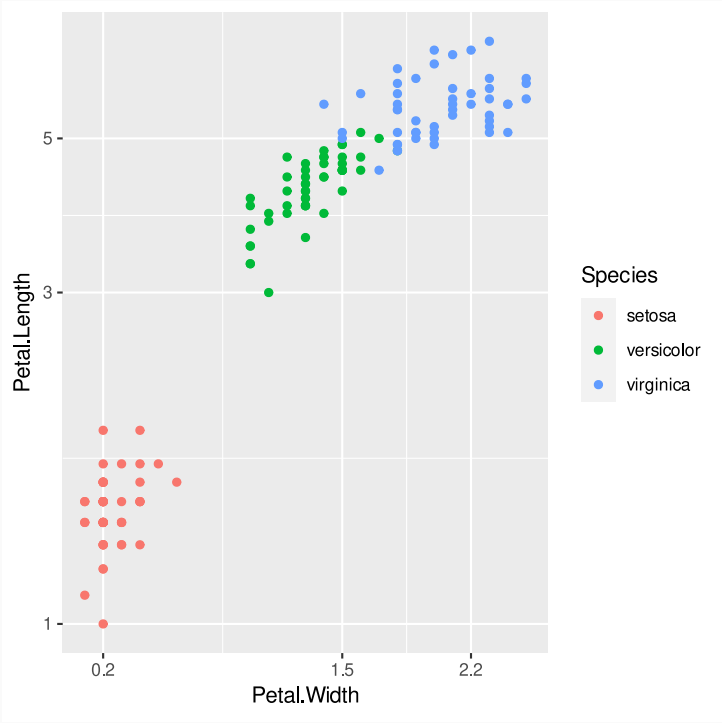
```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # stats layer
  stat_ellipse()

# Call the plot to see it
iris_points
```



Scales

Scales refers not only to axis scales, but all *aesthetics* (color/fill scales, shape scales...)

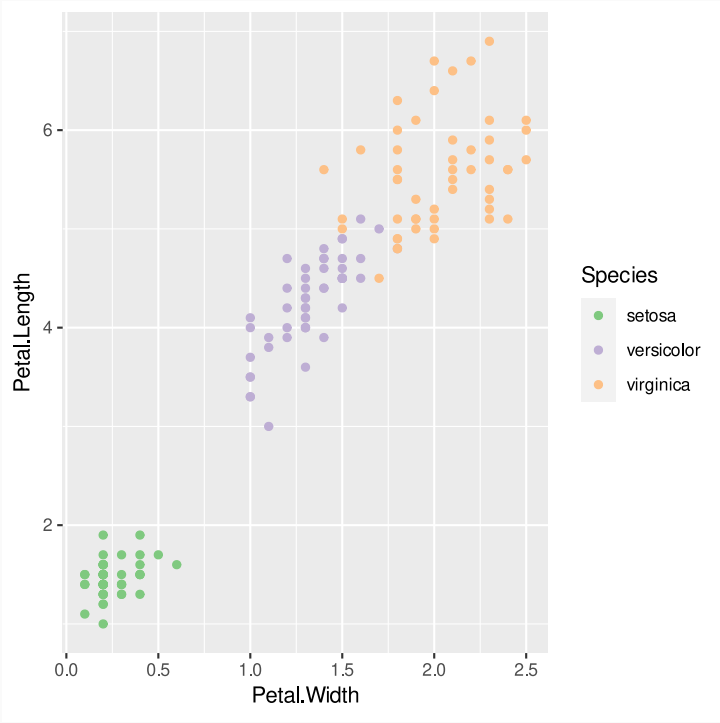
```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # scales layers
  scale_x_continuous(breaks = c(0.2, 1.5, 2.2)
  scale_y_continuous(trans = 'log10')

# Call the plot to see it
iris_points
```



Scales

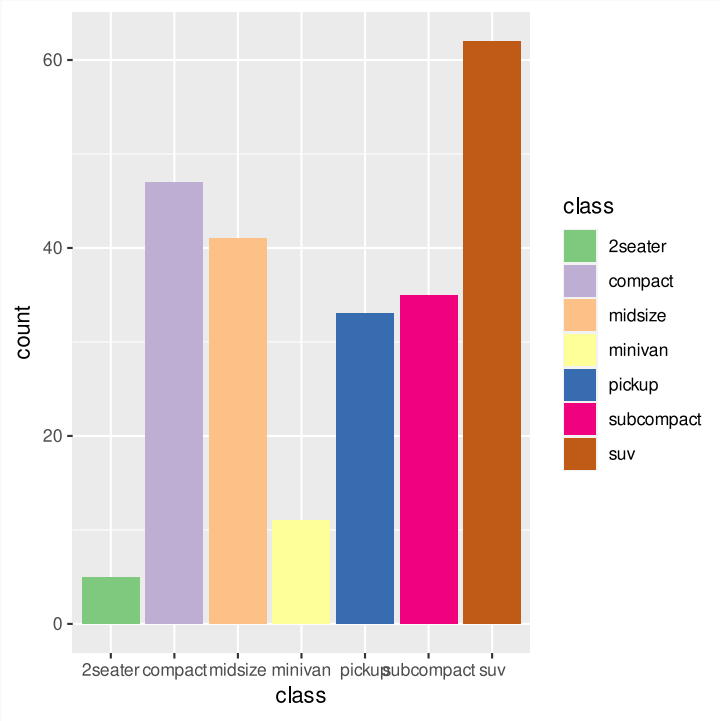
Scales refers not only to axis scales, but all *aesthetics* (color/fill scales, shape scales...)

```
iris_points ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # scales layers
  scale_colour_brewer(type = 'qual')

# Call the plot to see it
iris_points
```



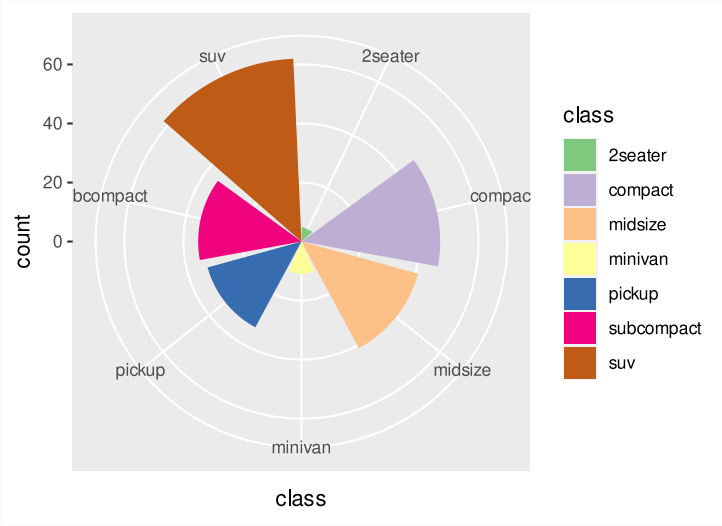
Coordinates

```
mpg_pie ←
  # data and aesthetics mapping layer
  ggplot(
    data = mpg,
    aes(x = class, fill = class)
  ) +
  # geom layer
  geom_bar() +
  # scales layers
  scale_fill_brewer(type = 'qual')

# Call the plot to see it
mpg_pie
```



Coordinates

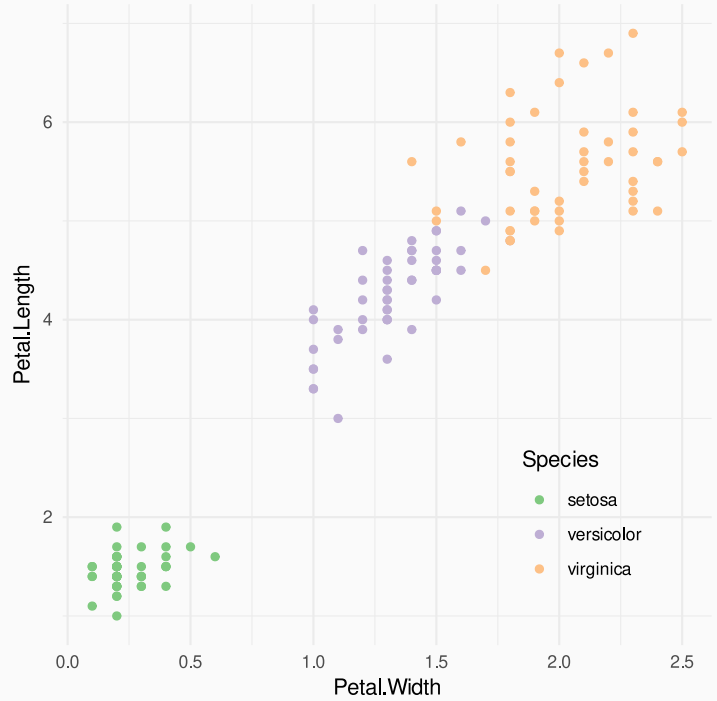
```
mpg_pie ←
  # data and aesthetics mapping layer
  ggplot(
    data = mpg,
    aes(x = class, fill = class)
  ) +
  # geom layer
  geom_bar() +
  # scales layers
  scale_fill_brewer(type = 'qual') +
  # coordinates layer
  coord_polar()

# Call the plot to see it
mpg_pie
```



Themes

Themes allows tuning plot elements not depending on mapping or aesthetics (like plot background, grid lines...)

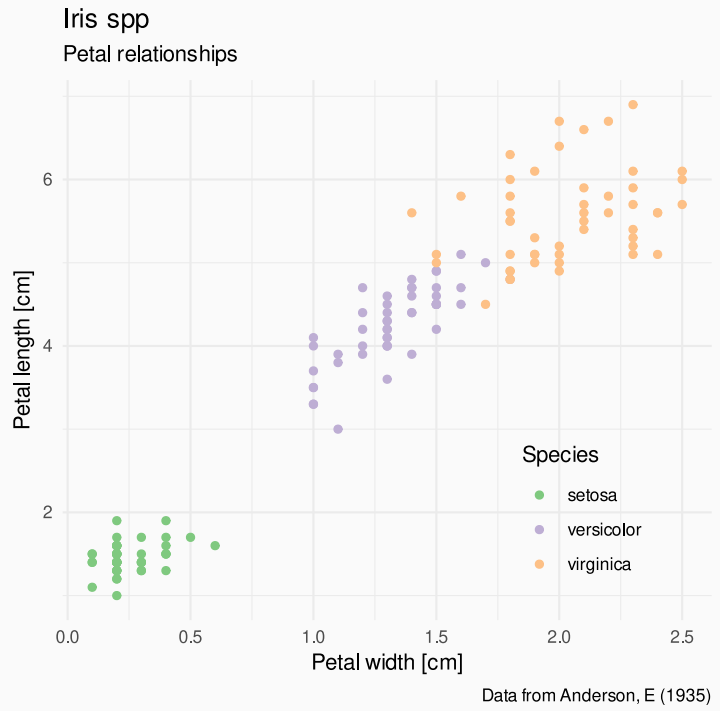
```
iris_points_themed ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # scales layers
  scale_colour_brewer(type = 'qual') +
  # theme
  theme_minimal() +
  theme(legend.position = c(.8, .2))

# Call the plot to see it
iris_points_themed
```



Annotation

Themes allows tuning plot elements not depending on mapping or aesthetics (like plot background, grid lines...)

```
iris_points_annotated ←
  # data and aesthetics mapping layer
  ggplot(
    data = iris,
    aes(
      x = Petal.Width, y = Petal.Length,
      colour = Species
    )
  ) +
  # geom layer
  geom_point() +
  # scales layers
  scale_colour_brewer(type = 'qual') +
  # theme
  theme_minimal() +
  theme(legend.position = c(.8, .2)) +
  labs(
    x = "Petal width [cm]",
    y = "Petal length [cm]",
    title = "Iris spp",
    subtitle = 'Petal relationships',
    caption = "Data from Anderson, E (1935)"
  )

# Call the plot to see it
iris_points_annotated
```



`gpplot2` ecosystem

A lot of satellite packages around ggplot offer *extras* to improve plots and add functionalities:

- Text annotations and labels:
  - `ggforce`
  - `ggrepel`
- Animation
  - `gganimate`
- Plot composition
  - `patchwork`
  - `cowplot`
- And so more (https://exts.ggplot2.tidyverse.org/gallery/)

patchwork

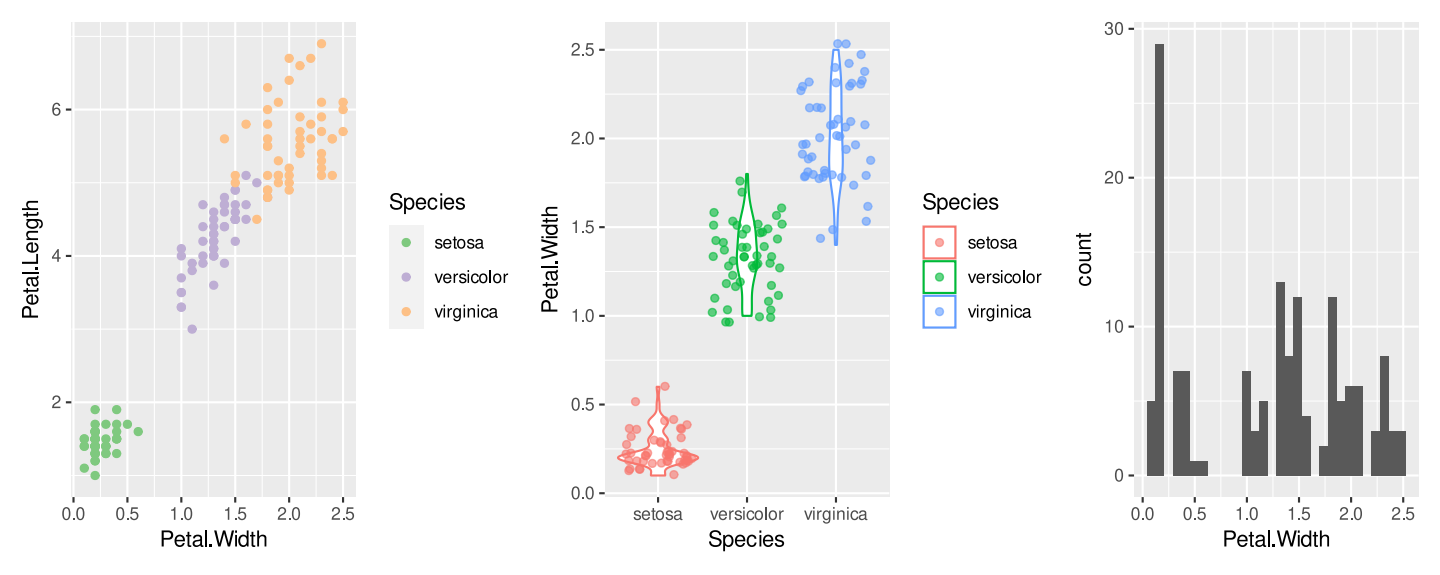
Composition of plots with labels in a easy way

```
# install.packages('patchwork')
library(patchwork)

iris_points + iris_boxplot + iris_histogram
```

patchwork

Composition of plots with labels in a easy way

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



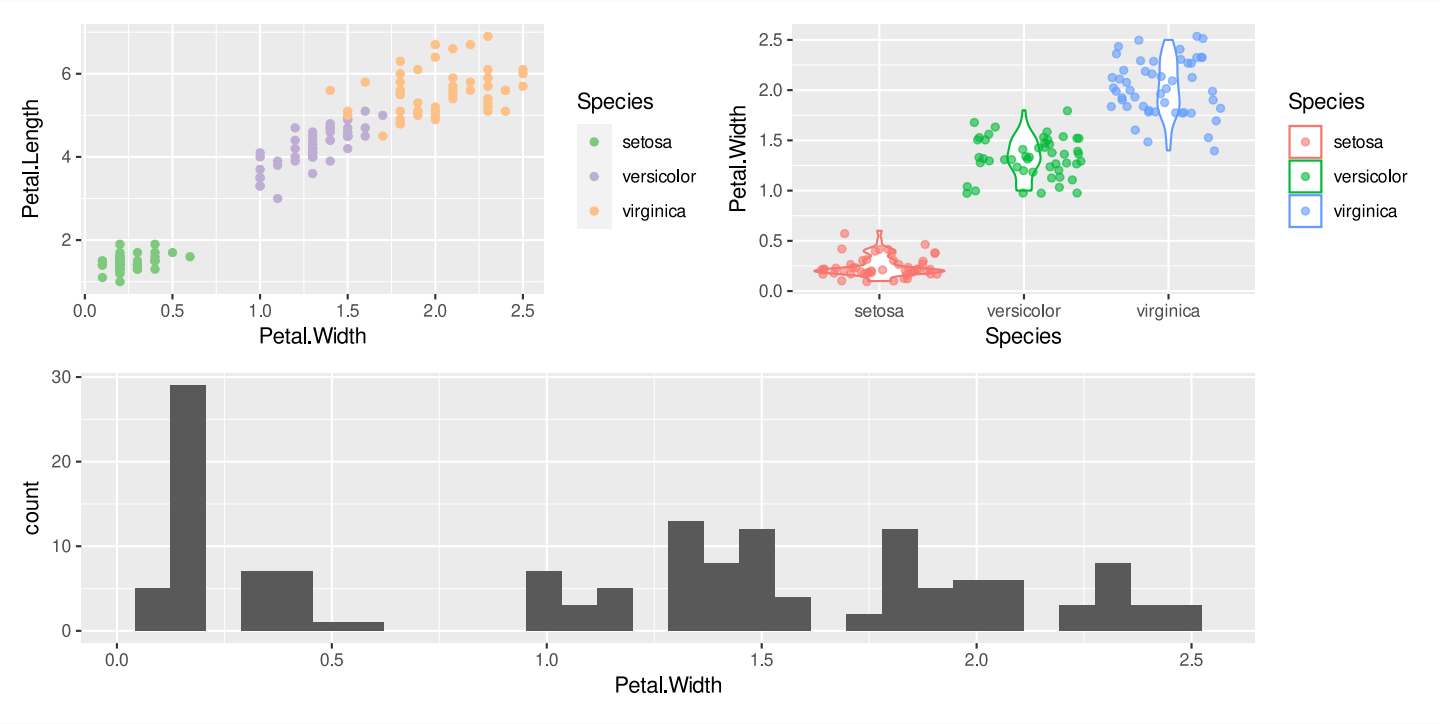
patchwork

Composition of plots with labels in a easy way

```
# install.packages('patchwork')
library(patchwork)

(iris_points | iris_boxplot) / iris_histogram
```

patchwork

Composition of plots with labels in a easy way



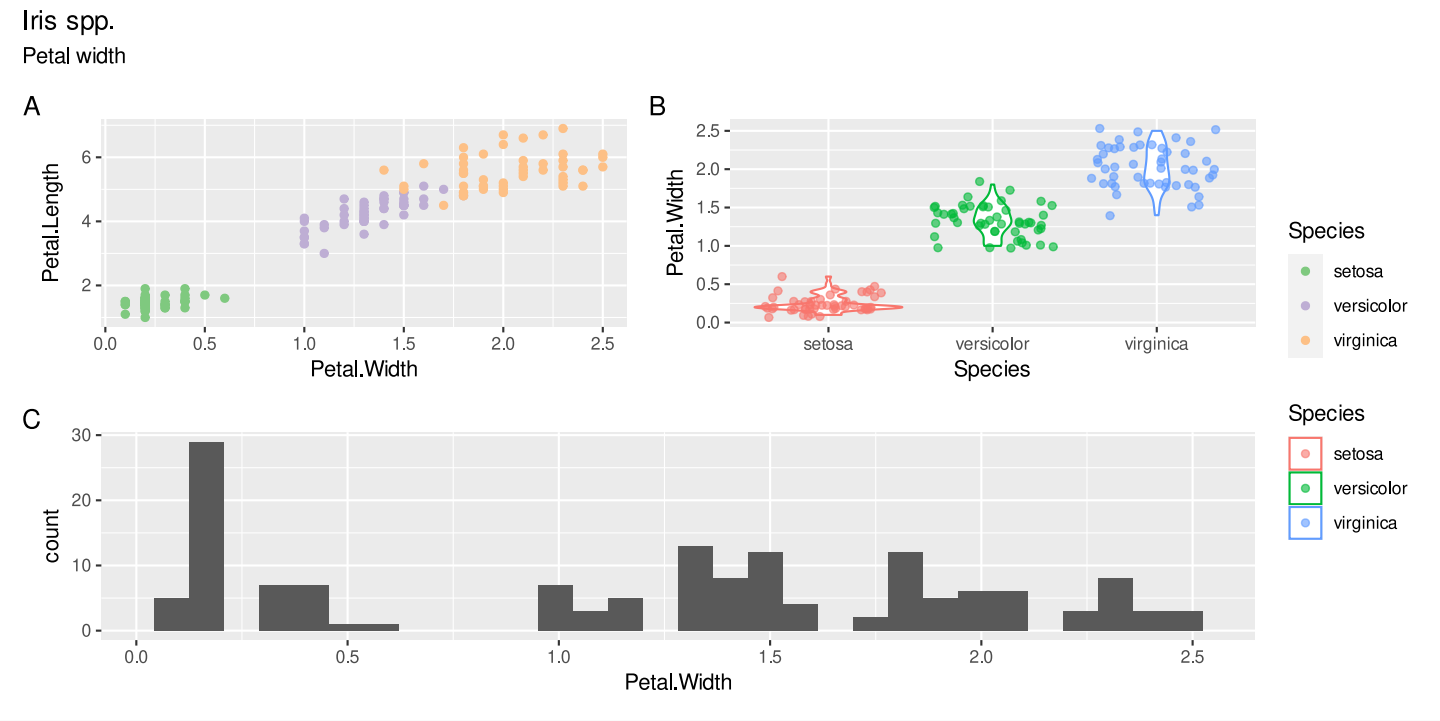
patchwork

Composition of plots with labels in a easy way

```
# install.packages('patchwork')
library(patchwork)

composed_plot ← (iris_points | iris_boxplot) / iris_histogram

composed_plot +
  plot_layout(guides = 'collect') +
  plot_annotation(
    title = 'Iris spp.',
    subtitle = 'Petal width',
    tag_levels = 'A'
  )
```

patchwork

Composition of plots with labels in a easy way



Saving plots


```
ggsave(
  "composed_plot.svg", composed_plot,
  device = "svg", width = 800, height = 600, units = "px", dpi = 70
)
```

Exercise

As we saw yesterday, we are gonna check the health of the Star Wars characters, but graphically, still using the the `starwars` dataset:

1. Represent height (x axis) versus mass (y axis), with different symbols/color for each gender, and adjusted linear model for each gender. If you find any outlier (thinking of you Jabba!!), remove it from the data set before plotting the data

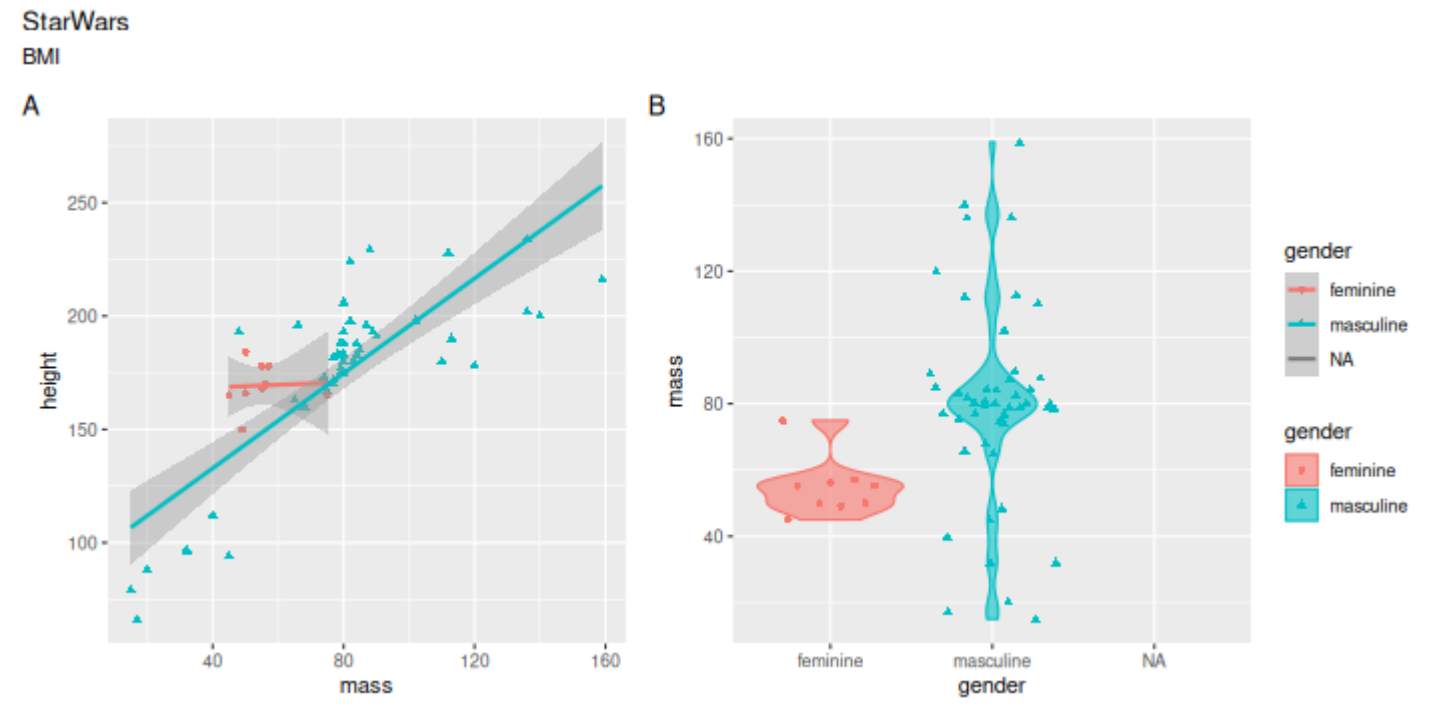
2. weight violin plot, with points also represented, for each gender.

3. repeat both plots, but faceted by gender.

4. combine the first two plots in one with patchwork, adding title, and letters indicating each plot.

Exercise

```
## `geom_smooth()` using formula = 'y ~ x'
```



Thank you!

https://github.com/MalditoBarbudo/
v.granda@creaf.uab.cat