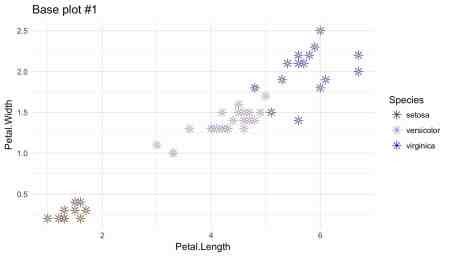
All plots have coordinate systems. Perhaps because they are such an integral element of plots, they are easily overlooked. However, in ggplot2, there are several very useful options to customize the coordinate systems of plots, which we will not overlook but explore in this blog post.

Since it is spring, we will use a random subset of the famous iris data set. When we plot the petal length against the petal width, and map species onto color and play around a little with the shape, color and sizes of aesthetics, one obtains this vernal plot:



# Base plot

plot\_base <- ggplot(data = df\_iris) +

geom\_point(aes(x = Petal.Length, y = Petal.Width, color = Species),

size = 3, alpha = 0.9, shape = 8) +

geom\_point(aes(x = Petal.Length, y = Petal.Width),

color = "yellow", size = 0.4) +

scale\_color\_manual(values = c("#693FE9", "#A089F8", "#0000FF")) +

theme\_minimal()

**Cartesian coordinate system**

**Zooming in and out**

The coordinate system can be manipulated by adding one of ggplot’s different coordinate systems. When you are imagining a coordinate system, you are most likely thinking of a Cartesian one. The Cartesian coordinate system combines x and y dimension orthogonally and is ggplots default (*coord\_cartesian*).

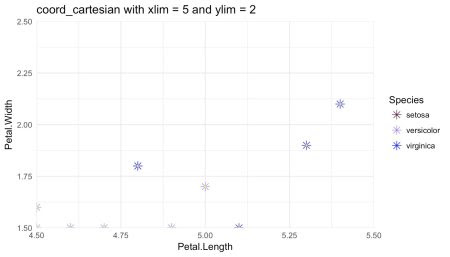
There also are several varaitions of the familiar Cartesian coordinate system in ggplot, namely *coord\_fixed*, *coord\_flip* and *coord\_trans*. For all of them, the displayed section of the data can be specified by defining the maximal value depicted on the x (*xlim =*) and y (*ylim =*) axis. This allows to “zoom in” or “zoom out” of a plot. It is a great advantage, that all manipulations of the coordinate system only alter the depiction of the data but not the data itself.

# Zooming in with xlim/ylim

plot\_base +

coord\_cartesian(xlim = 5, ylim = 2) +

ggtitle("coord\_cartesian with xlim = 5 and ylim = 2")



**Specifying the “aspect ratio” of the axes**

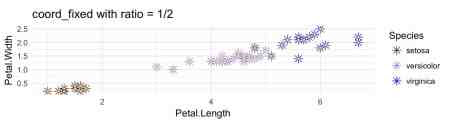
Via *coord\_fixed* one can specify the exact ratio of the length of a y unit relative to the length of a x unit within the final visualization.

# Setting the "aspect ratio" of y vs. x units

plot\_base +

coord\_fixed(ratio = 1/2) +

ggtitle("coord\_fixed with ratio = 1/2")



**Transforming the scales of the axes**

This helps to emphasize the exact insight one wants to communicate. Another way to do so is *coord\_trans*, which allows several transformations of the x and y variable (see table below, taken from Wickham 2016 page 97). Let me stress this again, very conveniently such transformations only pertain to the depicted – not the actual – scale of the data. This also is the reason why, regardless of the conducted transformation, the original values are used as axis labels.

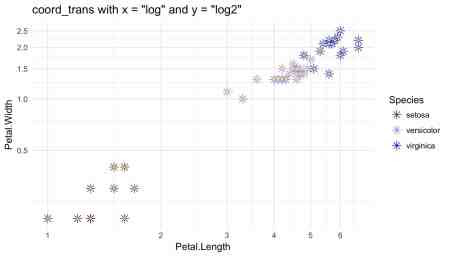
| **Name** | **Funktion f(x)** | **Inverse f^{ -1 } (y)** |
| --- | --- | --- |
| asn | tanh^{-1}(x) | tanh^{-1}(y) |
| exp | e^x | log(y) |
| identity | x | y |
| log | log(x) | e^y |
| log10 | log_{10}(x) | 10^y |
| log2 | log_{2}(x) | 2^y |
| logit | log(\frac{ x }{ 1-x }) | log(\frac{ x }{ 1+e(y) }) |
| pow10 | 10^x | log_{10}(y) |
| probit | \Phi(x) | \Phi^{-1}(y) |
| recip | x^{-1} | y^{-1} |
| reverse | -x | -y |
| sqrt | x^{1/2} | y^2 |

# Transforming the axes

plot\_base +

coord\_trans(x = "log", y = "log2") +

ggtitle("coord\_trans with x = \"log\" and y = \"log2\"")



**Swapping the axes**

The last of the Cartesian options, *cood\_flip*, swaps x and y axis. For example, this option can be useful, when we intend to change the orientation of univariate plots as histograms or plot types – like box plots – that visualize the distribution of a continuous variable over the categories of another variable. Nonetheless, *coord\_flip* also works with all other plots. This multiplies the overall possibilities for designing plots, especially since all Cartesian coordinate systems can be combined.

# Swapping axes

# base plot #2

p1 <- ggplot(data = df\_iris) +

geom\_bar(aes(x = Species, fill = Species), alpha = 0.6) +

scale\_fill\_manual(values = c("#693FE9", "#A089F8", "#4f5fb7")) +

theme\_minimal()

# base plot & coord\_flip()

p2 <- ggplot(data = df\_iris) +

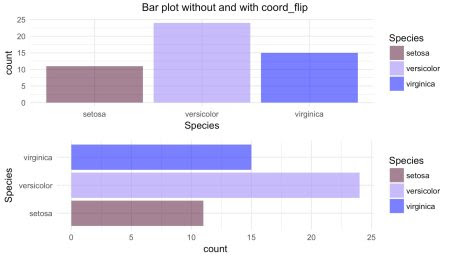
geom\_bar(aes(x = Species, fill = Species), alpha = 0.6) +

scale\_fill\_manual(values = c("#693FE9", "#A089F8", "#4f5fb7")) +

theme\_minimal() +

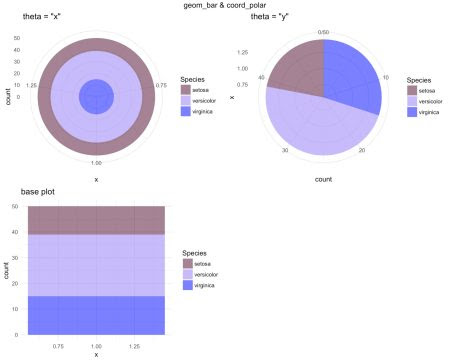
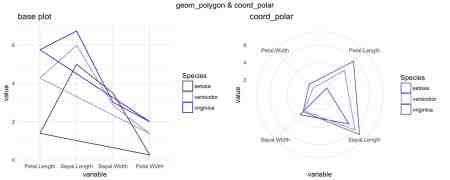
coord\_flip()

gridExtra::grid.arrange(p1, p2, top = "Bar plot without and with coord\_flip")



**Polar coordinate system**

The customization of Cartesian coordinate systems allows for the fine tuning of plots. However, *coord\_polar*, the final coordinate system discussed here, changes the whole character of a plot. By using *coord\_polar*, bar geoms are transformed to pie charts or “bullseye” plots, while line geoms are transformed to radar charts. This is done by mapping x and y to the angle and radius of the resulting plot. By default, the x variable is mapped to the angle but by setting the *theta* augment in *coord\_polar* to *“y”* this can be changed.

While such plots might shine with respect to novelty and looks, their perceptual properties are intricate, and their correct interpretation may be quite hard and rather unintuitive.

# Base plot 2 (long format, x = 1 is summed up to generate count)

plot\_base\_2 <- df\_iris %>%

dplyr::mutate(x = 1) %>%

ggplot(.) +

geom\_bar(aes(x = x, fill = Species), alpha = 0.6) +

theme(axis.text = element\_blank(),

axis.ticks = element\_blank(),

axis.title = element\_blank()) +

scale\_fill\_manual(values = c("#693FE9", "#A089F8", "#4f5fb7")) +

theme\_minimal() +

ggtitle("base plot")

# Bullseye plot

# geom\_bar & coord\_polar(theta = "x")

p2 <- plot\_base\_2 +

coord\_polar(theta = "x") +

ggtitle("theta = \"x\"")

# Pie chart

# geom\_bar & coord\_polar(theta = "y")

p3 <- plot\_base\_2 +

coord\_polar(theta = "y") +

ggtitle("theta = \"y\"")

gridExtra::grid.arrange(p2, p3, plot\_base\_2, top = "geom\_bar & coord\_polar", ncol = 2)

# Base plot 3 (long format, mean width/length of sepals/petals calculated)

plot\_base\_3 <- iris %>%

dplyr::group\_by(Species) %>%

dplyr::summarise(Petal.Length = mean(Petal.Length),

Sepal.Length = mean(Sepal.Length),

Sepal.Width = mean(Sepal.Width),

Petal.Width = mean(Petal.Width)) %>%

reshape2::melt() %>%

ggplot() +

geom\_polygon(aes(group = Species, color = Species, y = value, x = variable),

fill = NA) +

scale\_color\_manual(values = c("#693FE9", "#A089F8", "#4f5fb7")) +

theme\_minimal() +

ggtitle("base plot")

# Radar plot

# geom\_polygon & coord\_polar

p2 <- plot\_base\_3 +

theme\_minimal() +

coord\_polar() +

ggtitle("coord\_polar")

gridExtra::grid.arrange(plot\_base\_3, p2, top = "geom\_polygon & coord\_polar", ncol = 2)

**References**

* Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. Springer.