The grid Package: The grid package in R implements the primitive graphical functions that underlay the ggplot2 plotting system.

The grid package in R provides low-level graphical functions, which allow you to create highly customized and precise graphical outputs. It underlies higher-level plotting systems like ggplot2 and the base R plotting system. With grid, you have control over each aspect of your graphical objects, and you can arrange and layer these objects on a canvas.

Here’s an overview of the basic concepts in the grid package, and how they relate to ggplot2.

**Basic Concepts of grid Package**

1. **Viewports**: A viewport is a rectangular region in the plotting device where drawing occurs. You can define multiple viewports and position them on a canvas.
2. **Grobs (Graphical Objects)**: A "grob" represents a graphical object, such as lines, rectangles, text, or circles. You can manipulate these objects individually.
3. **Units**: The grid package uses a unit system to specify dimensions such as widths, heights, and positions. Units can be relative (npc for normalized parent coordinates) or absolute (e.g., in millimeters or inches).

**Basic Example Using grid**

Let’s walk through a simple example where we use the grid package to draw some graphical objects.

# Load the grid package

library(grid)

# Create a new blank page for drawing

grid.newpage()

# Draw a rectangle in the center of the plot

grid.rect(width = 0.5, height = 0.5, gp = gpar(fill = "lightblue", col = "black"))

# Draw a circle inside the rectangle

grid.circle(x = 0.5, y = 0.5, r = 0.2, gp = gpar(fill = "pink"))

# Add some text inside the circle

grid.text("Hello, Grid!", x = 0.5, y = 0.5, gp = gpar(fontsize = 14))

**Components of the Above Code**

1. **grid.newpage()**: Clears the plotting device and sets up a new page for drawing.
2. **grid.rect()**: Draws a rectangle. The gp argument specifies graphical parameters, such as fill color and line color.
3. **grid.circle()**: Draws a circle. The x and y arguments define the center of the circle, and r is the radius.
4. **grid.text()**: Adds text to the plot. The x and y coordinates specify the location of the text.

**Example: Custom Layout with Viewports**

Using viewports allows for more complex arrangements of graphical objects.

# Load grid package

library(grid)

# Create a new blank page

grid.newpage()

# Create two viewports

vp1 <- viewport(width = 0.5, height = 0.5, x = 0.25, y = 0.75, name = "vp1")

vp2 <- viewport(width = 0.5, height = 0.5, x = 0.75, y = 0.25, name = "vp2")

# Push the first viewport

pushViewport(vp1)

# Draw a rectangle in the first viewport

grid.rect(gp = gpar(fill = "lightgreen"))

# Pop the viewport (return to the main page)

popViewport()

# Push the second viewport

pushViewport(vp2)

# Draw a circle in the second viewport

grid.circle(gp = gpar(fill = "lightblue"))

# Add text inside the second viewport

grid.text("Viewport 2", x = 0.5, y = 0.5, gp = gpar(fontsize = 14))

# Pop the second viewport

popViewport()

**Explanation of Viewports**

* **viewport()**: Defines a new region on the canvas. You can specify the width, height, and location of the viewport using normalized parent coordinates (npc), where 0 represents the left/bottom edge, and 1 represents the right/top edge.
* **pushViewport()**: Pushes the defined viewport onto the current graphics context.
* **popViewport()**: Pops the current viewport, returning to the previous graphics context.

**Example: Overlaying grid and ggplot2**

Since ggplot2 is built on grid, you can overlay grid graphics on top of ggplot2 plots.

# Load ggplot2

library(ggplot2)

library(grid)

# Create a basic ggplot2 plot

p <- ggplot(mtcars, aes(x = wt, y = mpg)) +

geom\_point()

# Draw the ggplot

print(p)

# Add a rectangle using the grid package

grid.rect(width = 0.2, height = 0.2, x = 0.7, y = 0.7, gp = gpar(col = "red", fill = "transparent"))

In this example, after drawing a ggplot2 plot, we overlay a rectangle in the plotting region using grid.rect().

**More on Grobs (Graphical Objects)**

grid allows you to create complex graphical objects (grobs) like lines, polygons, and paths. Here are a few examples:

**Drawing a Line**

grid.newpage()

grid.lines(x = unit(c(0.1, 0.9), "npc"), y = unit(c(0.1, 0.9), "npc"), gp = gpar(col = "blue"))

Drawing a Polygon

grid.newpage()

grid.polygon(x = unit(c(0.2, 0.5, 0.8), "npc"), y = unit(c(0.2, 0.8, 0.2), "npc"), gp = gpar(fill = "orange", col = "black"))

**How grid Relates to ggplot2**

* **ggplot2 is built on top of grid**: All the plots you create in ggplot2 use the functions provided by grid. This is why you can easily combine ggplot2 plots with low-level grid functions to customize your plots further.
* **ggplot2 works with grobs**: Every plot in ggplot2 is a grob (graphical object). You can use ggplotGrob() to convert a ggplot into a grob and manipulate it using grid functions.

**Example: Converting a ggplot Object to a Grob**

# Create a ggplot object

p <- ggplot(mtcars, aes(x = wt, y = mpg)) +

geom\_point()

# Convert the plot to a grob

g <- ggplotGrob(p)

# Use grid functions to display the grob

grid.newpage()

grid.draw(g)

# Overlay additional grid graphics

grid.text("Custom Text", x = 0.5, y = 0.9, gp = gpar(fontsize = 15, col = "red"))

Example 1: Drawing Multiple Shapes with Custom Layout

We can create multiple graphical objects (grobs) and arrange them in custom viewports.

# Load grid package

library(grid)

# Create a new blank page

grid.newpage()

# Define three viewports for layout

vp1 <- viewport(width = 0.3, height = 0.3, x = 0.2, y = 0.8, name = "vp1")

vp2 <- viewport(width = 0.3, height = 0.3, x = 0.8, y = 0.8, name = "vp2")

vp3 <- viewport(width = 0.3, height = 0.3, x = 0.5, y = 0.3, name = "vp3")

# Push and draw in the first viewport

pushViewport(vp1)

grid.rect(gp = gpar(fill = "lightgreen"))

grid.text("Rectangle 1", x = 0.5, y = 0.5, gp = gpar(col = "black", fontsize = 14))

popViewport()

# Push and draw in the second viewport

pushViewport(vp2)

grid.circle(gp = gpar(fill = "lightblue"))

grid.text("Circle", x = 0.5, y = 0.5, gp = gpar(col = "black", fontsize = 14))

popViewport()

# Push and draw in the third viewport

pushViewport(vp3)

grid.polygon(x = unit(c(0.2, 0.8, 0.5), "npc"), y = unit(c(0.2, 0.2, 0.8), "npc"),

gp = gpar(fill = "orange"))

grid.text("Triangle", x = 0.5, y = 0.5, gp = gpar(col = "black", fontsize = 14))

popViewport()

**Explanation:**

* We define three viewports (vp1, vp2, vp3) to place the graphical objects in different areas of the canvas.
* We use grid.rect(), grid.circle(), and grid.polygon() to draw different shapes within these viewports.

**Example 2: Customizing Line Plots**

You can use grid.lines() to create custom line plots.

# Create a new page

grid.newpage()

# Draw a simple line from (0.1, 0.1) to (0.9, 0.9)

grid.lines(x = unit(c(0.1, 0.9), "npc"),

y = unit(c(0.1, 0.9), "npc"),

gp = gpar(col = "blue", lwd = 2))

# Add a second line from (0.9, 0.1) to (0.1, 0.9)

grid.lines(x = unit(c(0.9, 0.1), "npc"),

y = unit(c(0.1, 0.9), "npc"),

gp = gpar(col = "red", lwd = 2))

# Label the intersections with text

grid.text("Intersection", x = 0.5, y = 0.5, gp = gpar(fontsize = 16, col = "black"))

**Example 3: Custom Text Formatting with Viewports**

You can place multiple text objects with various font sizes and styles using grid.text().

# Create a new page

grid.newpage()

# Define custom text formatting using viewports

vp1 <- viewport(x = 0.5, y = 0.8)

vp2 <- viewport(x = 0.5, y = 0.6)

vp3 <- viewport(x = 0.5, y = 0.4)

# Push the first viewport and draw text with bold font

pushViewport(vp1)

grid.text("Bold Text", gp = gpar(fontsize = 18, fontface = "bold"))

popViewport()

# Push the second viewport and draw italic text

pushViewport(vp2)

grid.text("Italic Text", gp = gpar(fontsize = 18, fontface = "italic"))

popViewport()

# Push the third viewport and draw text with custom color and size

pushViewport(vp3)

grid.text("Colored Text", gp = gpar(fontsize = 18, col = "blue"))

popViewport()

**Example 4: Drawing Multiple Shapes in a Grid Layout**

The grid.layout() function allows you to create grid-based layouts for grobs.

# Create a new page

grid.newpage()

# Create a 2x2 grid layout

grid\_layout <- grid.layout(nrow = 2, ncol = 2)

# Define the viewport based on the grid layout

vp <- viewport(layout = grid\_layout)

# Push the layout viewport

pushViewport(vp)

# Draw shapes in each cell of the grid layout

# First cell (top-left)

pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 1))

grid.rect(gp = gpar(fill = "lightblue"))

grid.text("Cell 1", x = 0.5, y = 0.5)

popViewport()

# Second cell (top-right)

pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 2))

grid.circle(gp = gpar(fill = "pink"))

grid.text("Cell 2", x = 0.5, y = 0.5)

popViewport()

# Third cell (bottom-left)

pushViewport(viewport(layout.pos.row = 2, layout.pos.col = 1))

grid.polygon(x = unit(c(0.2, 0.8, 0.5), "npc"), y = unit(c(0.2, 0.2, 0.8), "npc"),

gp = gpar(fill = "green"))

grid.text("Cell 3", x = 0.5, y = 0.5)

popViewport()

# Fourth cell (bottom-right)

pushViewport(viewport(layout.pos.row = 2, layout.pos.col = 2))

grid.lines(x = unit(c(0.2, 0.8), "npc"), y = unit(c(0.2, 0.8), "npc"), gp = gpar(lwd = 3))

grid.text("Cell 4", x = 0.5, y = 0.5)

popViewport()

# Pop the layout viewport

popViewport()

**Explanation:**

* grid.layout() allows you to define a grid structure, and viewport() can be used to access specific rows and columns within that grid.
* Each cell in the grid can contain its own graphical object (rectangle, circle, polygon, etc.).

**Example 5: Grob Manipulation and Reuse**

You can create reusable graphical objects (grobs) and add them to different parts of the plot.

# Create a new page

grid.newpage()

# Create a grob for a circle

circle\_grob <- circleGrob(r = 0.2, gp = gpar(fill = "yellow", col = "black"))

# Create a grob for some text

text\_grob <- textGrob("Reused Grob", gp = gpar(col = "red", fontsize = 14))

# Draw the circle and text in one part of the plot

grid.draw(circle\_grob)

grid.draw(text\_grob)

# Move to a new viewport and reuse the same grobs

pushViewport(viewport(x = 0.7, y = 0.7))

grid.draw(circle\_grob)

grid.draw(text\_grob)

popViewport()

**Example 6: Advanced Polygon Shapes**

You can create complex polygon shapes by defining multiple vertices and connecting them.

# Create a new page

grid.newpage()

# Draw a custom polygon (pentagon)

grid.polygon(x = unit(c(0.5, 0.7, 0.6, 0.4, 0.3), "npc"),

y = unit(c(0.8, 0.6, 0.4, 0.4, 0.6), "npc"),

gp = gpar(fill = "lightblue", col = "black", lwd = 2))

# Add some text inside the polygon

grid.text("Pentagon", x = 0.5, y = 0.55, gp = gpar(fontsize = 16, col = "black"))

**Example 7: Creating a Complex Plot Layout**

Combine multiple viewports, grobs, and graphical elements into a more complex layout.

# Create a new page

grid.newpage()

# Create two main viewports for splitting the page

left\_vp <- viewport(width = 0.5, height = 1, x = 0.25, y = 0.5, name = "left\_vp")

right\_vp <- viewport(width = 0.5, height = 1, x = 0.75, y = 0.5, name = "right\_vp")

# Push and draw on the left viewport

pushViewport(left\_vp)

grid.rect(gp = gpar(fill = "lightgray"))

grid.text("Left Panel", x = 0.5, y = 0