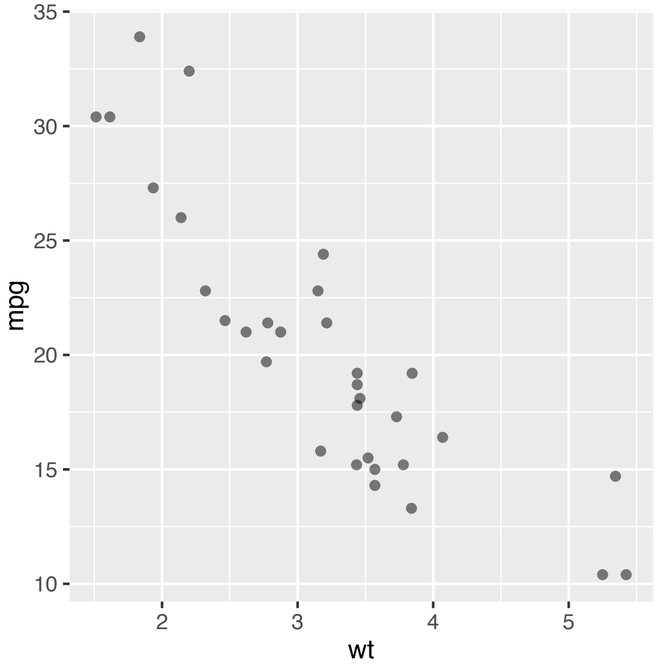
DATA VISUALIZATION WITH GGPLOT2

Grid Graphics

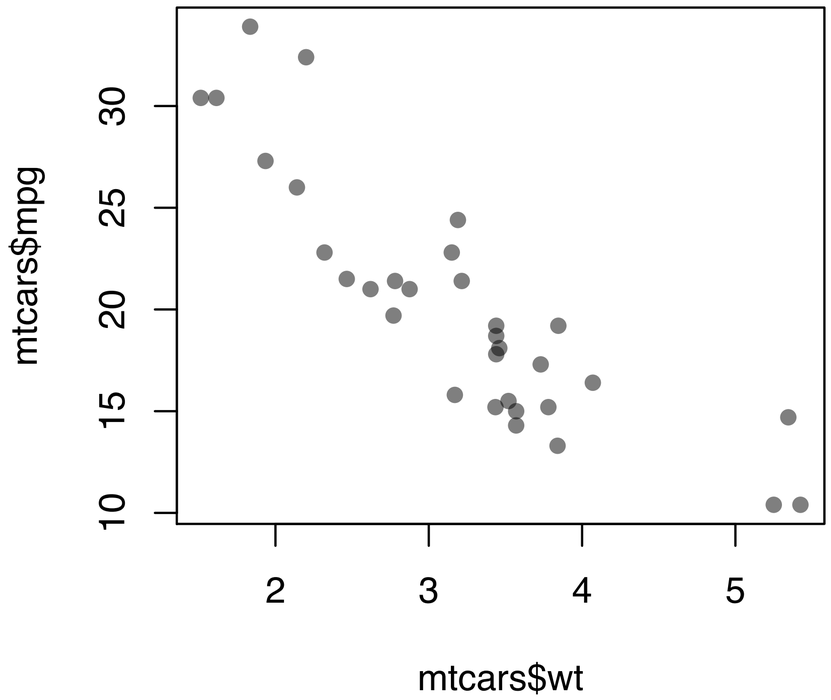
ggplot2 internals

- Explore grid graphics
- Elements of ggplot2 plot
- How do graphics work in R?
- 2 plotting systems
  - base package
  - grid graphics



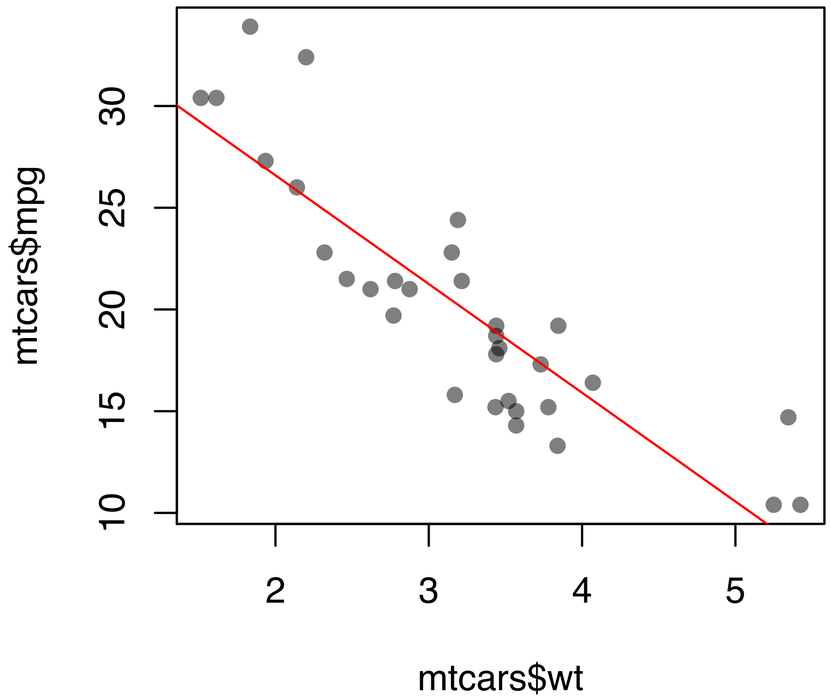
base package

```
> plot(mtcars$wt, mtcars$mpg, pch = 16, col = "#00000080")
```



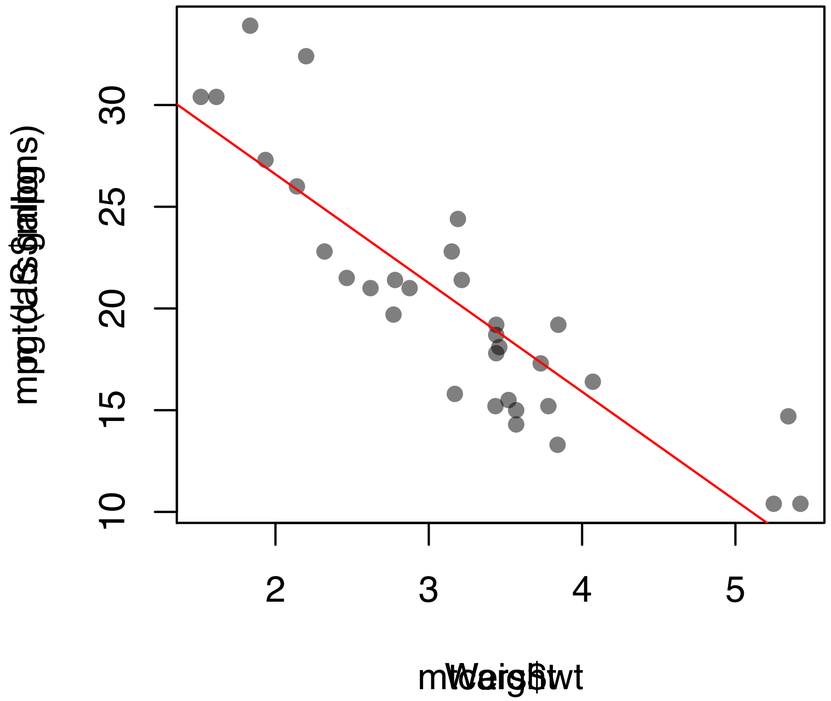
base package

```
> plot(mtcars$wt, mtcars$mpg, pch = 16, col = "#00000080")
> abline(lm(mpg ~ wt, data = mtcars), col = "red")
```



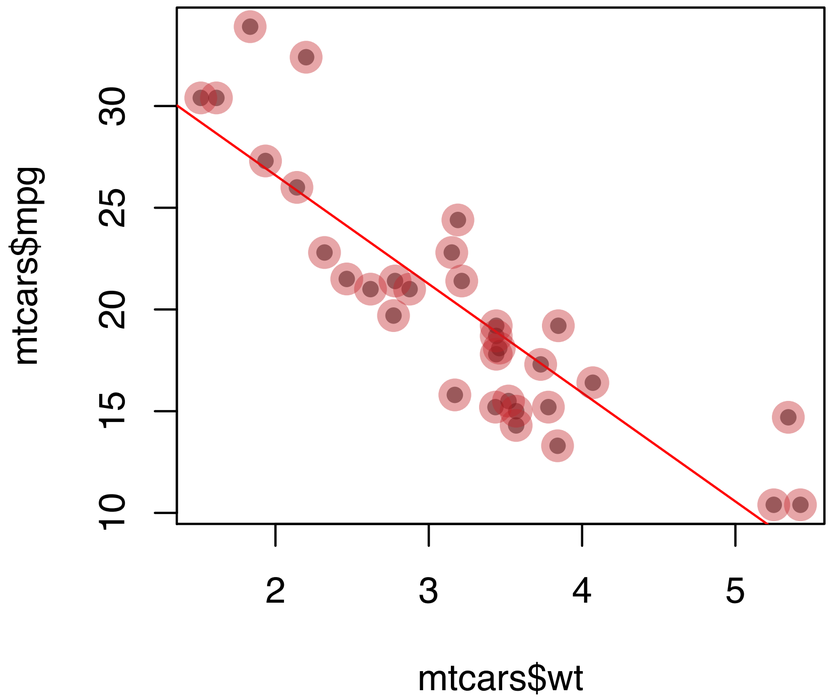
base package - change labels

```
> plot(mtcars$wt, mtcars$mpg, pch = 16, col = "#00000080")
> abline(lm(mpg ~ wt, data = mtcars), col = "red")
> mtext("Weight", 1, 3)
> mtext("mpg (US gallons)", 2, 3)
```



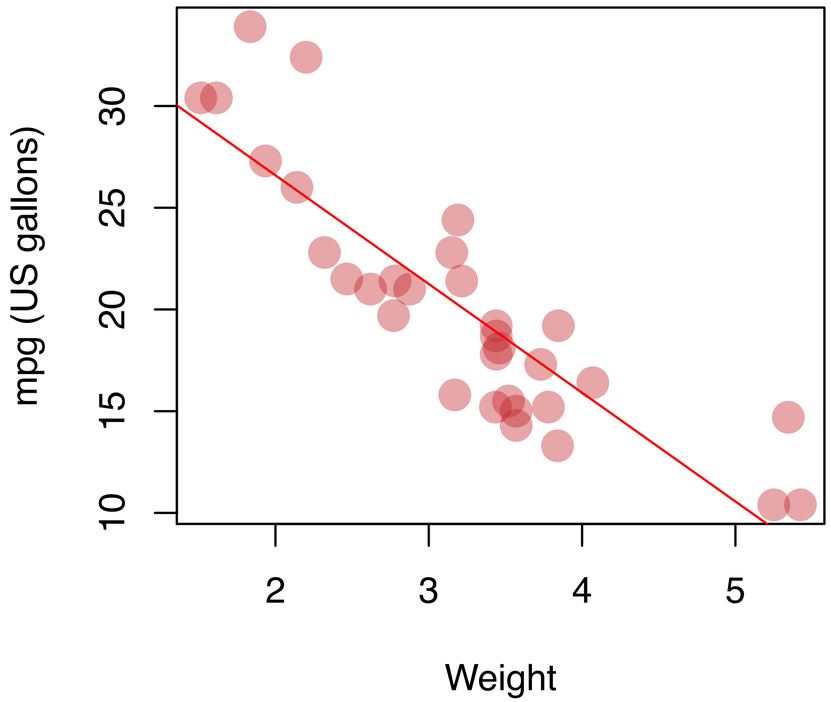
base package - change dots

```
> plot(mtcars$wt, mtcars$mpg, pch = 16, col = "#00000080")
> abline(lm(mpg ~ wt, data = mtcars), col = "red")
> points(mtcars$wt, mtcars$mpg, pch = 16,
         col = "#C3212766", cex = 2)
```



base package - restart

```
> plot(mtcars$wt, mtcars$mpg, pch = 16, col = "#C3212766",
       cex = 2, xlab = "Weight", ylab = "mpg (US gallons)")
> abline(lm(mpg ~ wt, data = mtcars), col = "red")
```

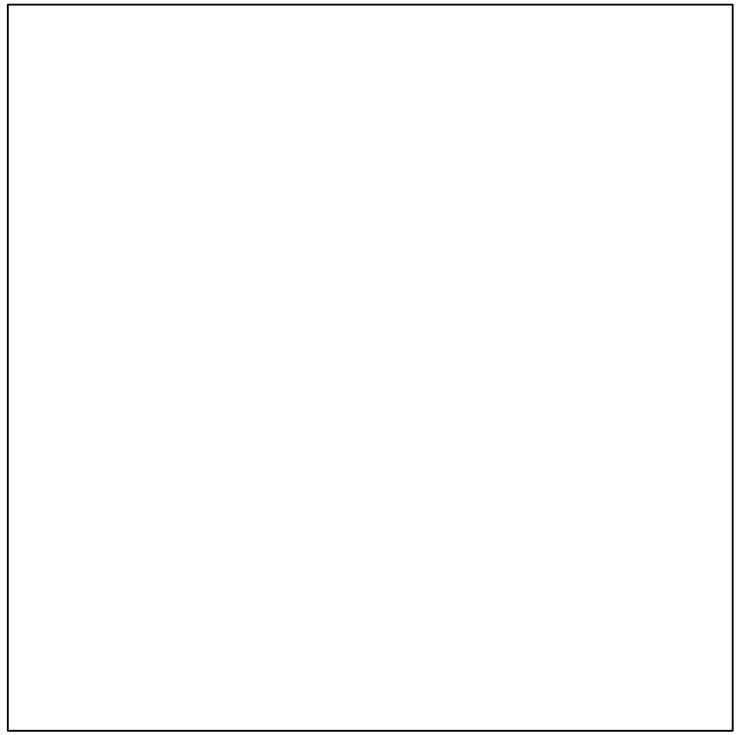


grid package

- Paul Murell
- Low-level graphic functions
- Assemble yourself
- ggplot2 built on top of grid
- Two components
  - Create graphic outputs
  - Layer and position outputs with viewports

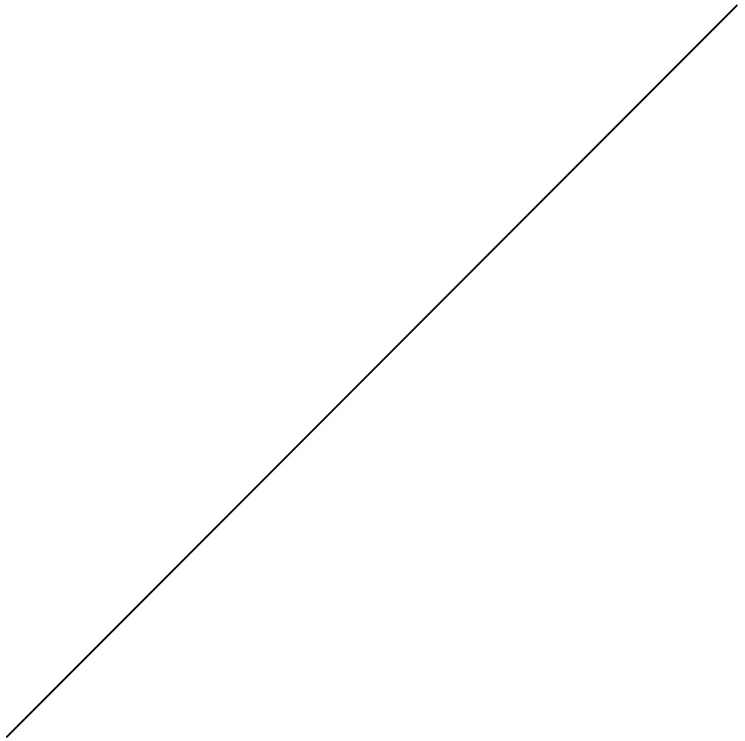
Graphic output

```
> # Rectangle
> grid.rect()
```



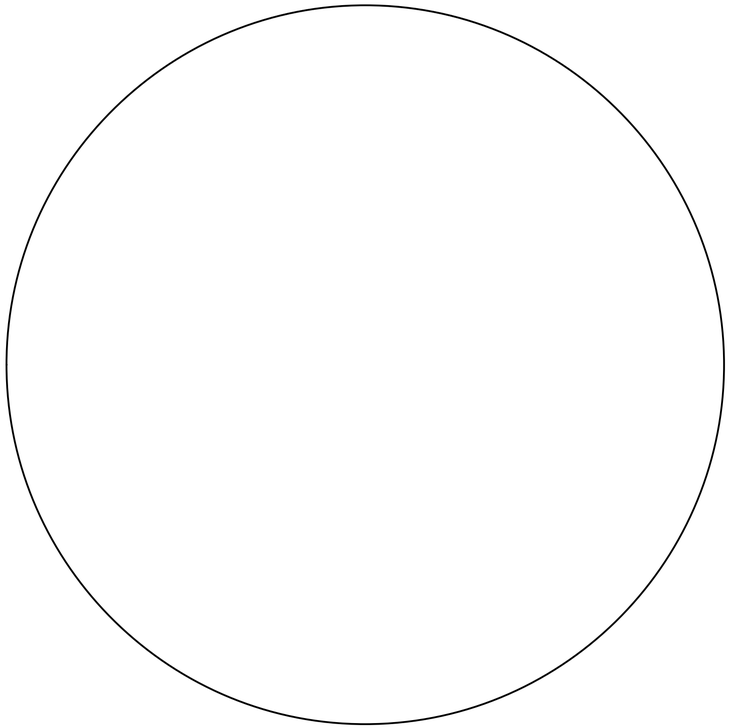
Graphic output

```
> # Rectangle
> grid.rect()

> # Line
> grid.lines()
```



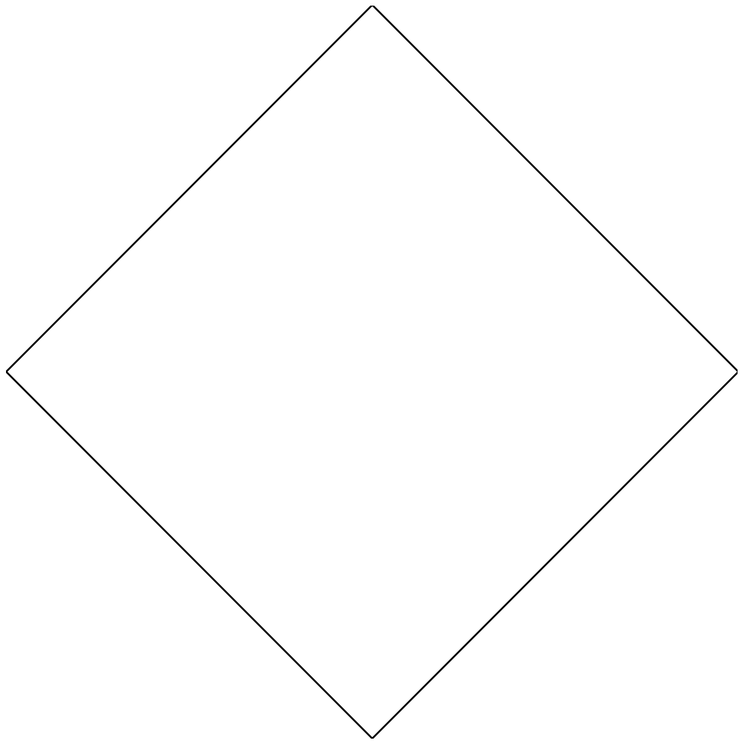
Graphic output

```
> # Rectangle
> grid.rect()

> # Line
> grid.lines()

> # Circle
> grid.circle()
```



Graphic output

```
> # Rectangle
> grid.rect()

> # Line
> grid.lines()

> # Circle
> grid.circle()

> # Grid polygon
> grid.polygon()
```

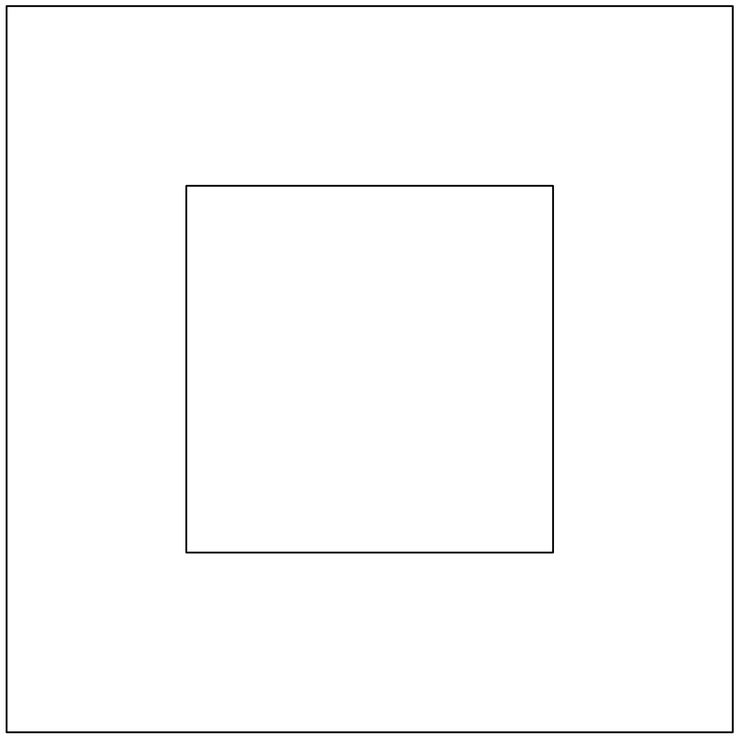


Graphic output

```
> # Rectangle
> grid.rect()

> # Line
> grid.lines()

> # Circle
> grid.circle()

> # Grid polygon
> grid.polygon()

> # Text
> grid.text("hello")
```

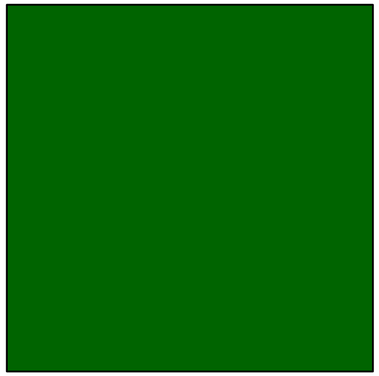
Graphic output - adjust

```
> # Rectangle
> grid.rect()

> # Modified rectangle
> grid.rect(x = 0.5, y = 0.5,
            width = 0.5, height = 0.5,
            just = "center")
```



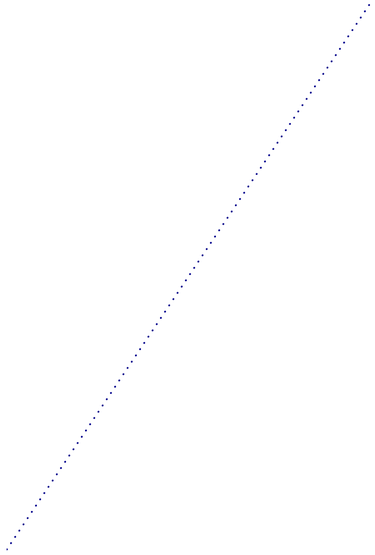
Graphic output - gpar()

```
> # Rectangle
> grid.rect(x = 0.5, y = 0.5,
            width = 0.5, height = 0.5,
            just = "center",
            gp = gpar(fill = "darkgreen"))
```



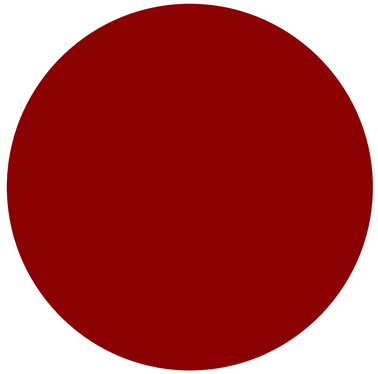
Graphic output - gpar()

```
> # Rectangle
> grid.rect(x = 0.5, y = 0.5,
            width = 0.5, height = 0.5,
            just = "center",
            gp = gpar(fill = "darkgreen"))

> # Line
> grid.lines(x = c(0, 0.5), y = c(0.25, 1),
           gp = gpar(lty = 3,
                     col = "darkblue"))
```



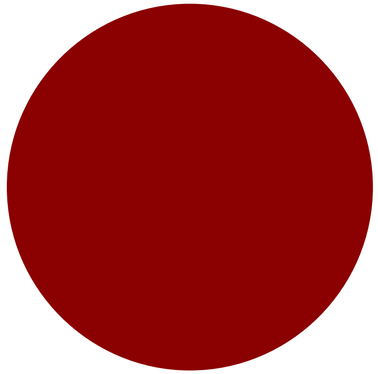
Graphic output - gpar()

```
> # Rectangle
> grid.rect(x = 0.5, y = 0.5,
            width = 0.5, height = 0.5,
            just = "center",
            gp = gpar(fill = "darkgreen"))

> # Line
> grid.lines(x = c(0, 0.5), y = c(0.25, 1),
           gp = gpar(lty = 3,
                     col = "darkblue"))

> # Circle
> grid.circle(x = 0.5, y = 0.5, r = 0.25,
              gp = gpar(fill = "darkred",
                        col = NA))
```



Naming graphic output

```
> # Circle
> grid.circle(x = 0.5, y = 0.5, r = 0.25,
              gp = gpar(fill = "darkred",
                        col = NA),
              name = "myCircle")
```



Naming graphic output

```
> # Circle
> grid.circle(x = 0.5, y = 0.5, r = 0.25,
              gp = gpar(fill = "darkred",
                        col = NA),
              name = "myCircle")

> grid.edit("myCircle",
            gp = gpar(fill = "pink"))
```

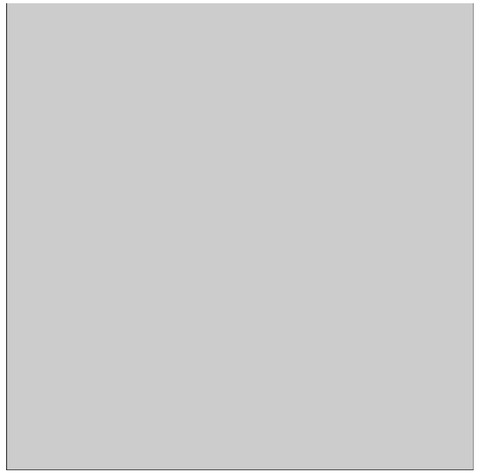
Naming graphic output

```
> # Circle
> grid.circle(x = 0.5, y = 0.5, r = 0.25,
              gp = gpar(fill = "darkred",
                        col = NA),
              name = "myCircle")

> grid.edit("myCircle",
            gp = gpar(fill = "pink"))

> grid.edit("myCircle",
            x = unit(0.6, "npc"),
            r = unit(0.6, "npc"))
```

Viewports

Windows onto which we draw graphic outputs

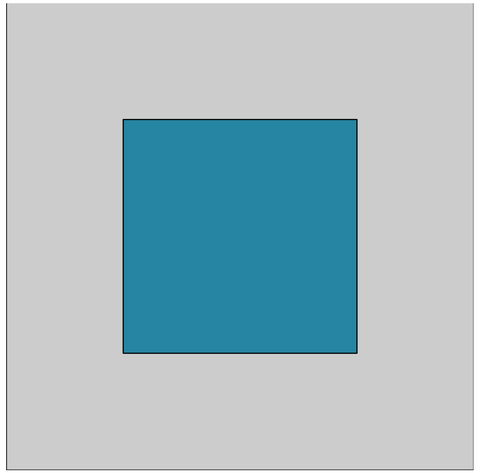
```
> grid.rect(gp = gpar(fill = "grey80"))
> vp <- viewport(x = 0.5, y = 0.5, w = 0.5, h = 0.5,
                 just = "center")
> pushViewport(vp)
```



Viewports

Windows onto which we draw graphic outputs

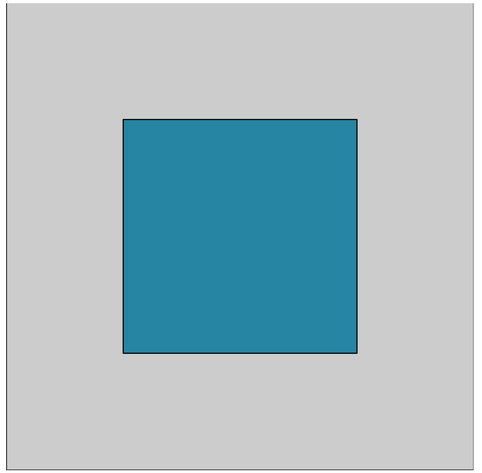
```
> grid.rect(gp = gpar(fill = "grey80"))
> vp <- viewport(x = 0.5, y = 0.5, w = 0.5, h = 0.5,
                 just = "center")
> pushViewport(vp)
> grid.rect(gp = gpar(fill = "#2685A2"))
```



Viewports

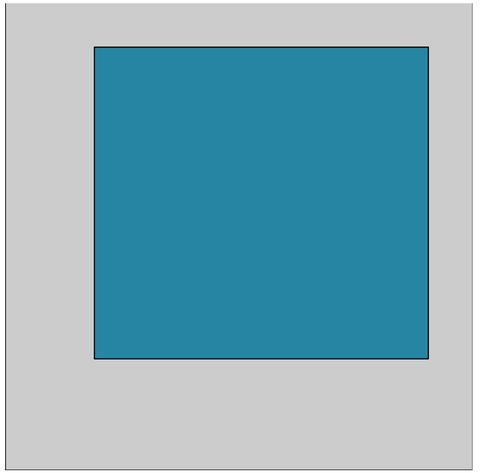
Windows onto which we draw graphic outputs

```
> grid.rect(gp = gpar(fill = "grey80"))
> vp <- viewport(x = 0.5, y = 0.5, w = 0.5, h = 0.5,
                 just = "center", name = "vp1")
> pushViewport(vp)
> grid.rect(gp = gpar(fill = "#2685A2"))
```



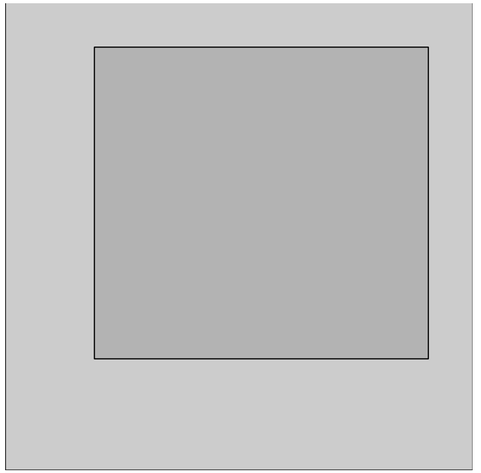
plotViewport

```
> grid.rect(gp = gpar(fill = "grey80"))
> mar <- c(5, 4, 2, 2)
> vp_plot <- plotViewport(margins = mar, name = "vp2")
> pushViewport(vp_plot)
> grid.rect(gp = gpar(fill = "#2685A2"))
```



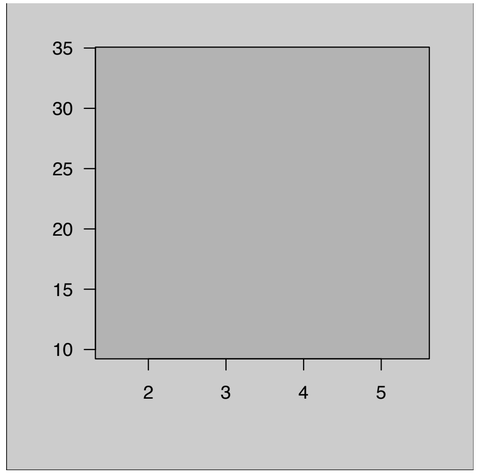
dataViewport

```
> grid.rect(gp = gpar(fill = "grey80"))
> mar <- c(5, 4, 2, 2)
> vp_plot <- plotViewport(margins = mar, name = "vp2")
> pushViewport(vp_plot)

> vp_data <- dataViewport(mtcars$wt, mtcars$mpg)
> pushViewport(vp_data)
> grid.rect(gp = gpar(fill = "grey70"))
```



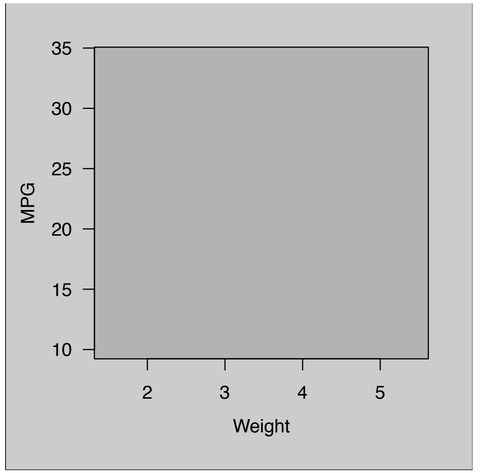
dataViewport

```
> grid.rect(gp = gpar(fill = "grey80"))
> mar <- c(5, 4, 2, 2)
> vp_plot <- plotViewport(margins = mar, name = "vp2")
> pushViewport(vp_plot)

> vp_data <- dataViewport(mtcars$wt, mtcars$mpg)
> pushViewport(vp_data)
> grid.rect(gp = gpar(fill = "grey70"))

> grid.xaxis()
> grid.yaxis()
```



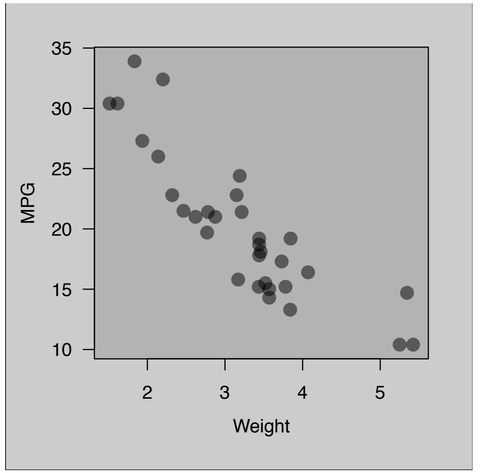
dataViewport

```
> grid.rect(gp = gpar(fill = "grey80"))
> mar <- c(5, 4, 2, 2)
> vp_plot <- plotViewport(margins = mar, name = "vp2")
> pushViewport(vp_plot)

> vp_data <- dataViewport(mtcars$wt, mtcars$mpg)
> pushViewport(vp_data)
> grid.rect(gp = gpar(fill = "grey70"))

> grid.xaxis()
> grid.yaxis()

> grid.text("Weight", y = unit(-3, "lines"))
> grid.text("MPG", x = unit(-3, "lines"), rot = 90)
```



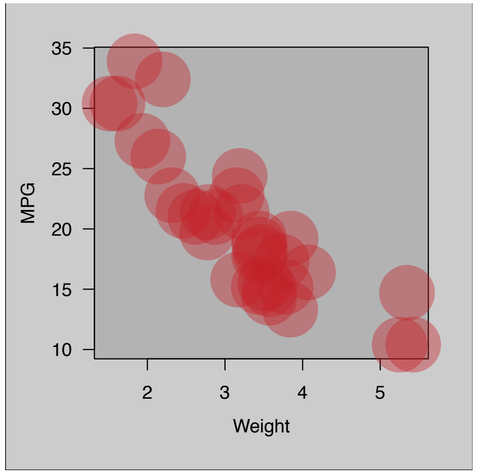
dataViewport

```
> grid.rect(gp = gpar(fill = "grey80"))
> mar <- c(5, 4, 2, 2)
> vp_plot <- plotViewport(margins = mar, name = "vp2")
> pushViewport(vp_plot)

> vp_data <- dataViewport(mtcars$wt, mtcars$mpg)
> pushViewport(vp_data)
> grid.rect(gp = gpar(fill = "grey70"))

> grid.xaxis()
> grid.yaxis()

> grid.text("Weight", y = unit(-3, "lines"))
> grid.text("MPG", x = unit(-3, "lines"), rot = 90)

> grid.points(mtcars$wt, mtcars$mpg, pch = 16,
              gp = gpar(col = "#00000080"),
              name = "data")
```



grid.edit

```
> grid.rect(gp = gpar(fill = "grey80"))

...

> grid.points(mtcars$wt,mtcars$mpg, pch = 16,
              gp = gpar(col = "#00000080"),
              name = "datapoints")

> grid.edit("datapoints",
            gp = gpar(col = "#C3212766", cex = 4))
```



DATA VISUALIZATION WITH GGPLOT2

Let’s practice!

DATA VISUALIZATION WITH GGPLOT2

Grid graphics in ggplot2

Grobs

- Graphical objects = grobs
- ggplot2 object = collection of grobs

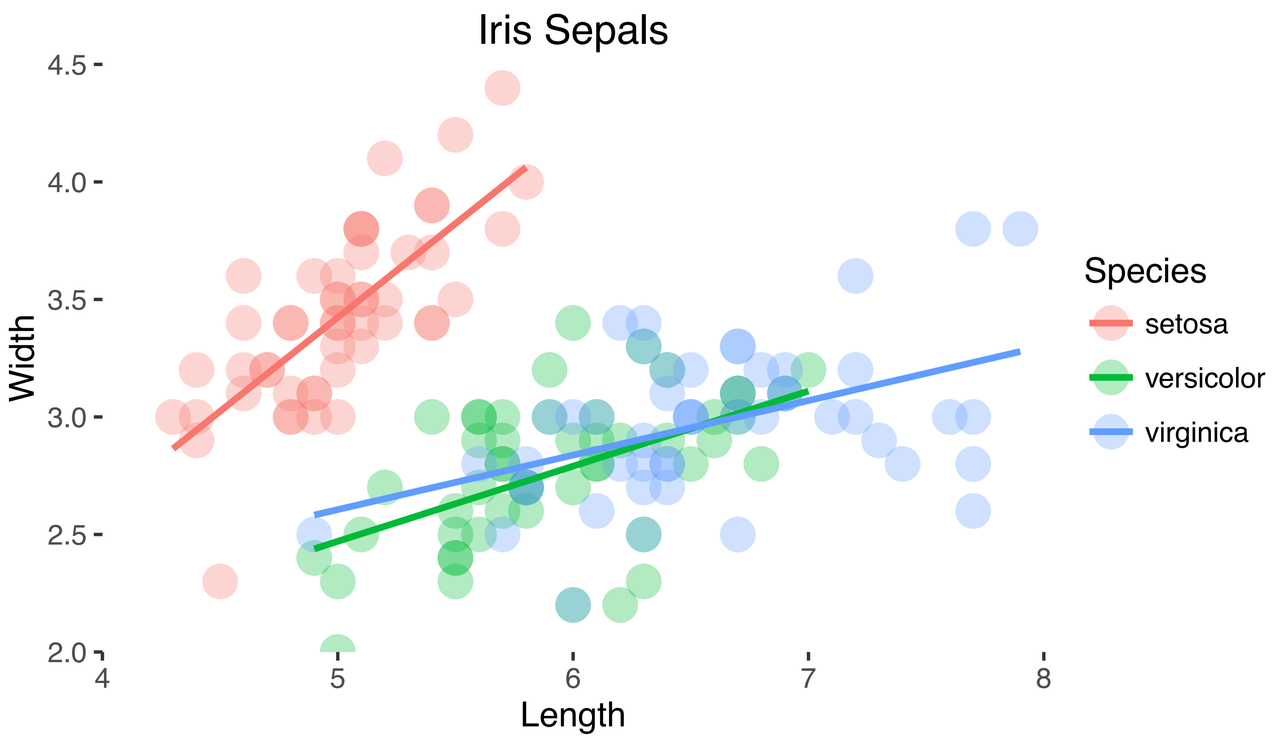
Graphic Output	Graphics Object
grid.rect()	rectGrob()
grid.lines()	linesGrob()
grid.circle()	circleGrob()
grid.polygon()	polygonGrob()
grid.text()	textGrob()

ggplot2 example

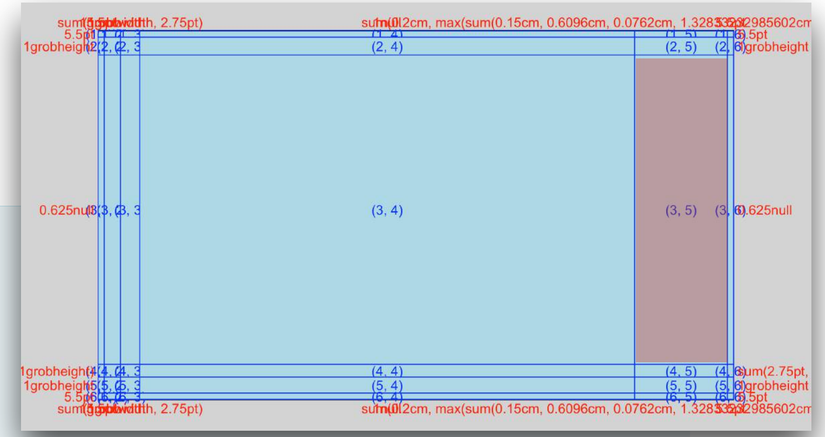
```
> p <- ggplot(iris, aes(x = Sepal.Length,
                        y = Sepal.Width,
                        col = Species)) +
    geom_point(alpha = 0.3, size = 5, shape = 16) +
    geom_smooth(method = "lm", se = FALSE) +
    scale_y_continuous("Width", limits = c(2, 4.5), expand = c(0,0)) +
    scale_x_continuous("Length", limits = c(4, 8), expand = c(0,0)) +
    coord_equal() +
    ggtitle("Iris Sepals") +
    theme(rect = element_blank())
```

ggplot2 example

```
> p
```



Accessing grobs



```
> library(grid)
> g <- ggplotGrob(p)
> g
TableGrob (6 x 6) "layout": 9 grobs
  z     cells       name                                                grob
1 0 (1-6,1-6) background zeroGrob[plot.background..zeroGrob.23938]
2 3 (3-3,3-3)     axis-l     absoluteGrob[GRID.absoluteGrob.23907]
3 1 (4-4,3-3)     spacer                                zeroGrob[NULL]
4 2 (3-3,4-4)      panel                       gTree[GRID.gTree.23893]
5 4 (4-4,4-4)     axis-b     absoluteGrob[GRID.absoluteGrob.23900]
6 5 (5-5,4-4)       xlab  titleGrob[axis.title.x..titleGrob.23910]
7 6 (3-3,2-2)       ylab  titleGrob[axis.title.y..titleGrob.23913]
8 7 (3-3,5-5)  guide-box                          gtable[guide-box]
9 8 (2-2,4-4)      title    titleGrob[plot.title..titleGrob.23937]
```

List of grobs

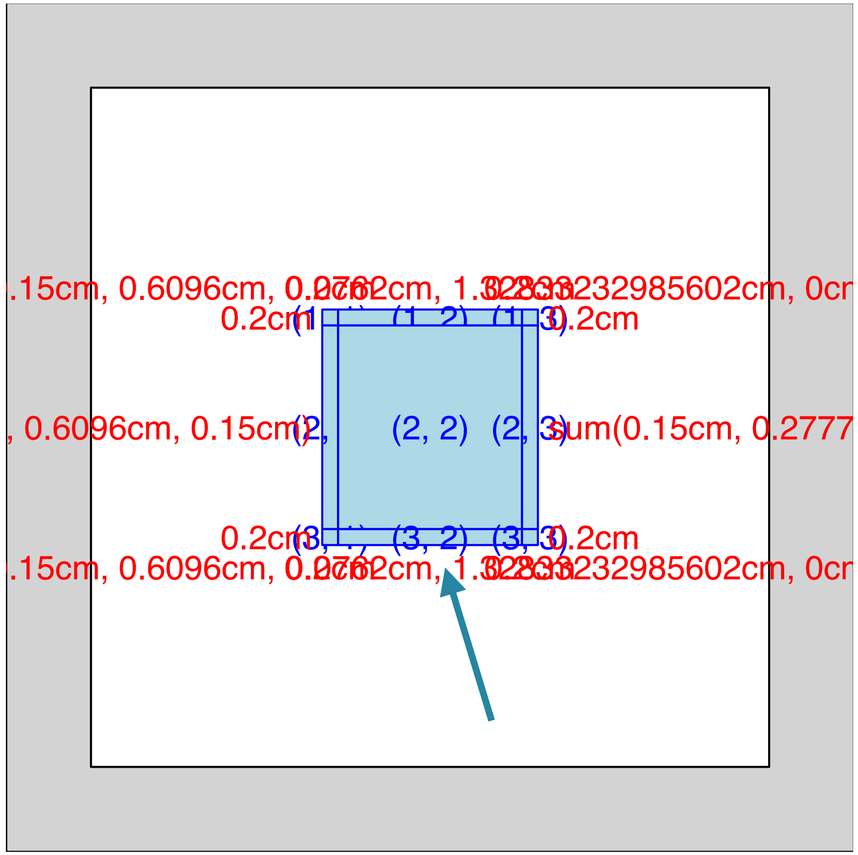
```
> g$grob

[[1]]
zeroGrob[plot.background..zeroGrob.24133]

...

[[6]]
titleGrob[axis.title.x..titleGrob.24105]

[[7]]
titleGrob[axis.title.y..titleGrob.24108]

[[8]]
TableGrob (3 x 3) "guide-box": 1 grobs
                                    z     cells   name           grob
99_cf2b20daa6ef538a0def731fa7c3e7db 1 (2-2,2-2) guides gtable[layout]
...
```

Legend grob

```
> g$grob[[8]]

TableGrob (3 x 3) "guide-box": 1 grobs
                                     z     cells   name           grob
99_cf2b20daa6ef538a0def731fa7c3e7db 1 (2-2,2-2) guides gtable[layout]

> grid.draw(g$grob[[8]])
```

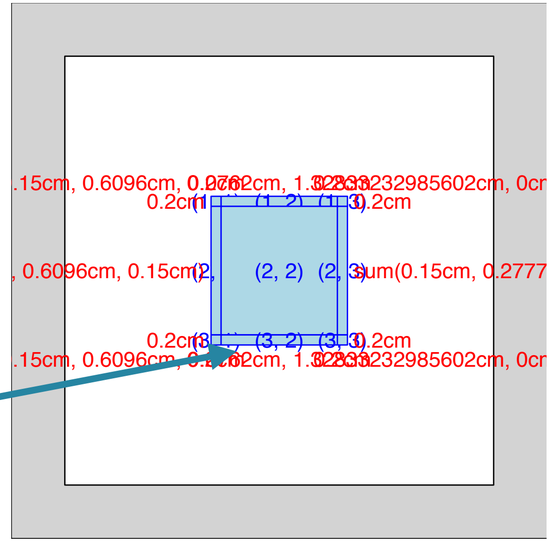
Structure of legend

```
> library(gtable)
> gtable_show_layout(g$grob[[8]])
```



Update legend

```
> my_text <- textGrob(label = "Anderson, 1936",
                      gp = gpar(fontsize = 7,
                                col = "gray25"))

> leg2 <- gtable_add_grob(g$grob[[8]], my_text, 3, 2)
> leg2
TableGrob (3 x 3) "guide-box": 2 grobs
            z     cells      name                    grob
99_cf2.... 1 (2-2,2-2)    guides          gtable[layout]
           2 (3-3,2-2) guide-box text[GRID.text.24881]
```

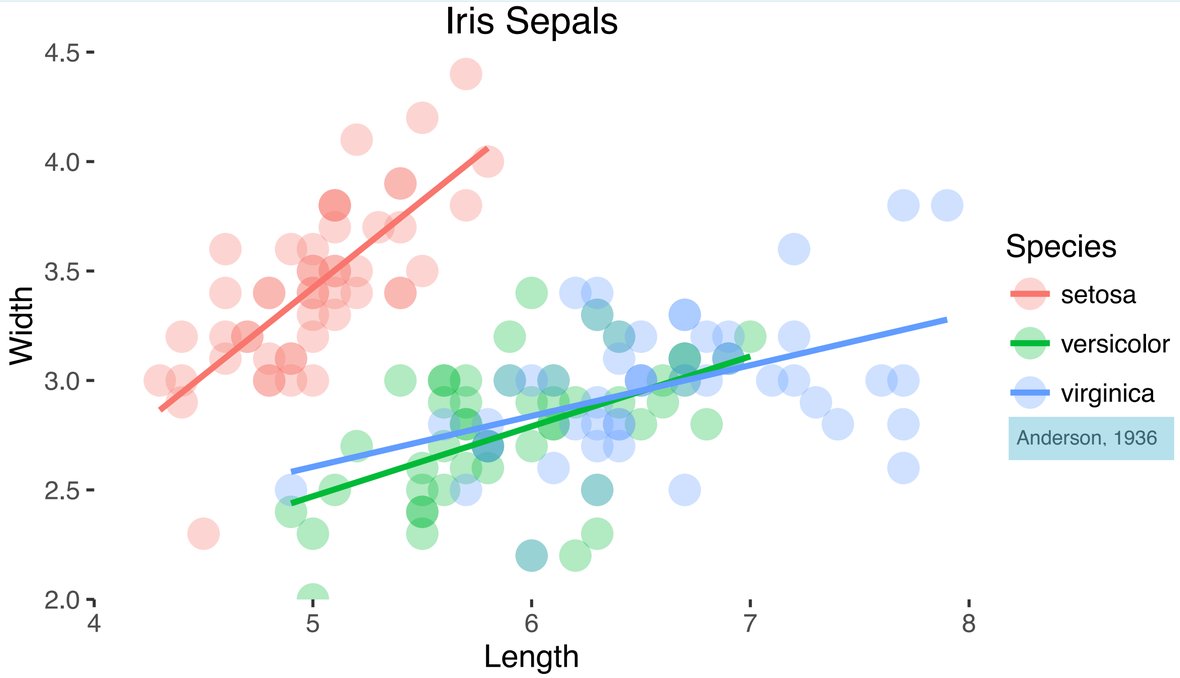


Update legend

```
> my_text <- textGrob(label = "Anderson, 1936",
                      gp = gpar(fontsize = 7,
                                col = "gray25"))

> leg2 <- gtable_add_grob(g$grob[[8]], my_text, 3, 2)
> leg2
TableGrob (3 x 3) "guide-box": 2 grobs
            z     cells      name                    grob
99_cf2.... 1 (2-2,2-2)    guides          gtable[layout]
           2 (3-3,2-2) guide-box text[GRID.text.24881]

> grid.draw(leg2)
```

Update legend (2)

```
> my_text <- textGrob(label = "Anderson, 1936",
                      gp = gpar(fontsize = 7,
                                col = "gray25"))

> g$grobs[[8]] <- gtable_add_grob(g$grob[[8]], my_text, 3, 2)
> grid.draw(g)
```



DATA VISUALIZATION WITH GGPLOT2

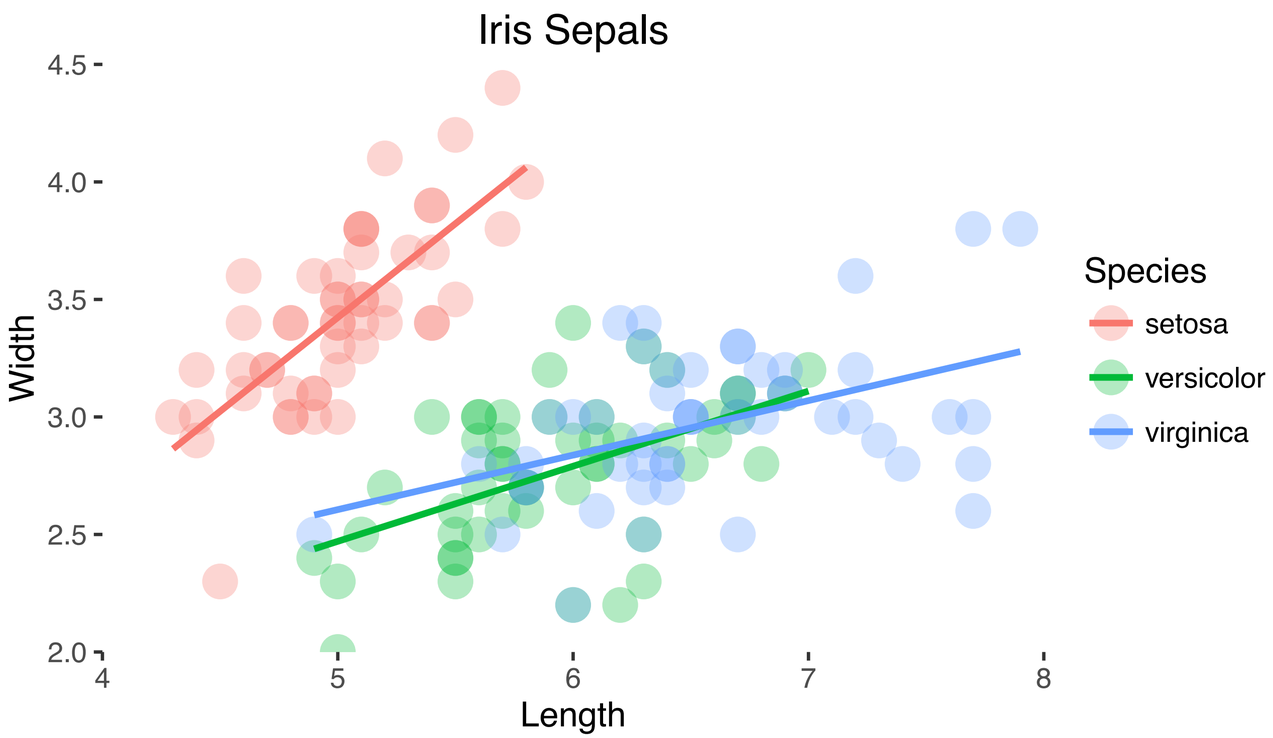
Let’s practice!

DATA VISUALIZATION WITH GGPLOT2

ggplot2 Objects

ggplot2 example

```
> p
```



Accessing grobs

```
> library(grid)
> g <- ggplotGrob(p)
> g
TableGrob (6 x 6) "layout": 9 grobs
  z     cells       name                                                grob
1 0 (1-6,1-6) background zeroGrob[plot.background..zeroGrob.23938]
2 3 (3-3,3-3)     axis-l     absoluteGrob[GRID.absoluteGrob.23907]
3 1 (4-4,3-3)     spacer                                zeroGrob[NULL]
4 2 (3-3,4-4)      panel                      gTree[GRID.gTree.23893]
5 4 (4-4,4-4)     axis-b     absoluteGrob[GRID.absoluteGrob.23900]
6 5 (5-5,4-4)       xlab  titleGrob[axis.title.x..titleGrob.23910]
7 6 (3-3,2-2)       ylab  titleGrob[axis.title.y..titleGrob.23913]
8 7 (3-3,5-5)  guide-box                          gtable[guide-box]
9 8 (2-2,4-4)      title    titleGrob[plot.title..titleGrob.23937]
```

ggplot object

```
> e <- ggplot()

> class(e)
[1] "gg"     "ggplot"

> names(e)
[1] "data"        "layers"      "scales"      "mapping"     "theme"
[6] "coordinates" "facet"       "plot_env"    "labels"
```

p

```
> p <- ggplot(iris, aes(x = Sepal.Length,
                        y = Sepal.Width,
                        col = Species)) +
    geom_point(alpha = 0.3, size = 5, shape = 16) +
    geom_smooth(method = "lm", se = FALSE) +
    scale_y_continuous("Width", limits = c(2, 4.5), expand = c(0,0)) +
    scale_x_continuous("Length", limits = c(4, 8), expand = c(0,0)) +
    coord_equal() +
    ggtitle("Iris Sepals") +
    theme(rect = element_blank())

> names(p)
[1] "data"        "layers"      "scales"      "mapping"     "theme"
[6] "coordinates" "facet"       "plot_env"    "labels"
```

```
> p$data

    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          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
...

> p$layers
> p$scales
> p$mapping
> p$theme
> p$coordinates
> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers

[[1]]
geom_point: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity

[[2]]
geom_smooth: na.rm = FALSE
stat_smooth: na.rm = FALSE, method = lm, formula = y ~ x, se =
FALSE
position_identity

> p$scales
> p$mapping
> p$theme
> p$coordinates
> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales

<ggproto object: Class ScalesList>
    add: function
    clone: function
    find: function
    get_scales: function
    has_scale: function
    input: function
    n: function
    non_position_scales: function
    scales: list
    super:  <ggproto object: Class ScalesList>

> p$mapping
> p$theme
> p$coordinates
> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping

* x      -> Sepal.Length
* y      -> Sepal.Width
* colour -> Species

> p$theme
> p$coordinates
> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping
> p$theme

List of 1
 $ rect: list()
  ..- attr(*, "class")= chr [1:2] "element_blank" "element"
 - attr(*, "class")= chr [1:2] "theme" "gg"
 - attr(*, "complete")= logi FALSE
 - attr(*, "validate")= logi FALSE

> p$coordinates
> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping
> p$theme
> p$coordinates

<ggproto object: Class CoordFixed, CoordCartesian, Coord>
    aspect: function
    distance: function
    expand: TRUE
    is_linear: function
    labels: function
    limits: list
    range: function
    ratio: 1
    render_axis_h: function
    ...

> p$facet
> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping
> p$theme
> p$coordinates
> p$facet

facet_null()

> p$plot_env
> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping
> p$theme
> p$coordinates
> p$facet
> p$plot_env

<environment: R_GlobalEnv>

> p$labels
```

```
> p$data
> p$layers
> p$scales
> p$mapping
> p$theme
> p$coordinates
> p$facet
> p$plot_env
> p$labels

$title
[1] "Iris Sepals"

$x
[1] "Sepal.Length"

$y
[1] "Sepal.Width"

$colour
[1] "Species"
```

ggplot_build

```
> p_build <- ggplot_build(p)

> names(p_build)
[1] "data"  "panel" "plot"
```

data

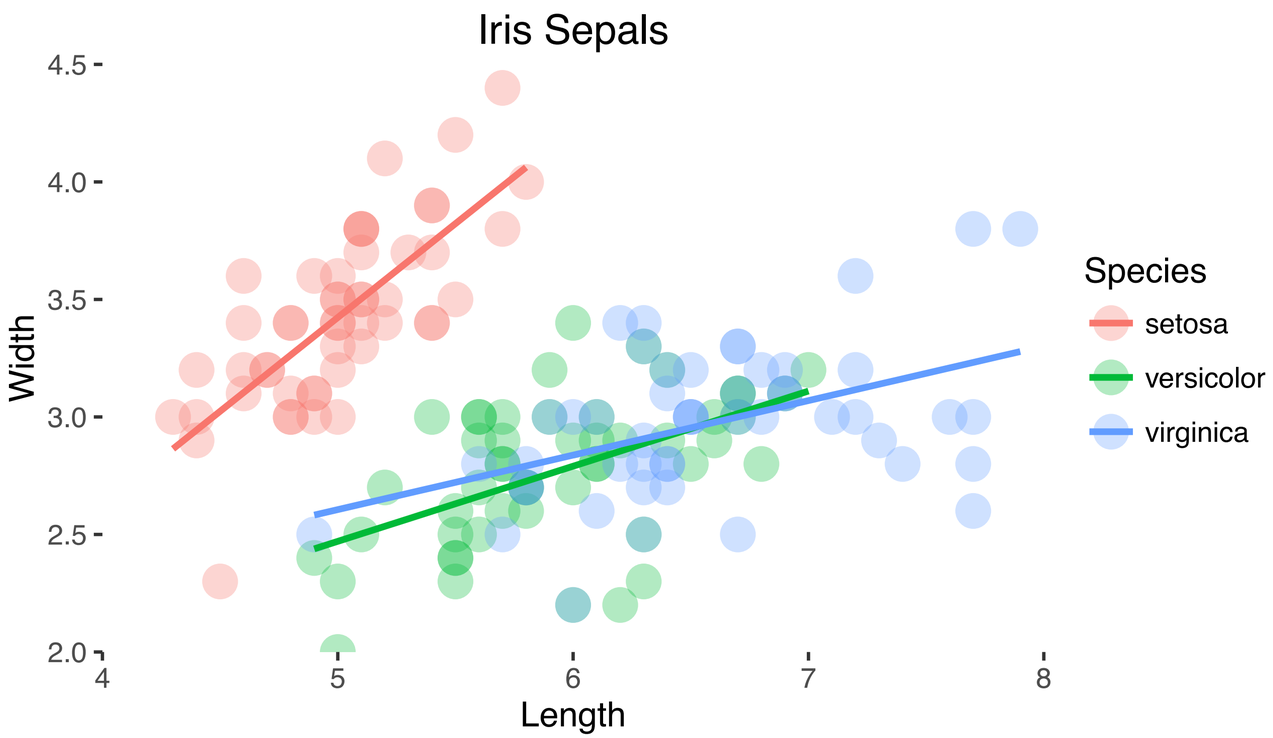
```
> p_build$data
[[1]]
     colour   y   x PANEL group shape size fill alpha stroke
1   #F8766D 3.5 5.1     1     1    16    5   NA   0.3    0.5
2   #F8766D 3.0 4.9     1     1    16    5   NA   0.3    0.5
3   #F8766D 3.2 4.7     1     1    16    5   NA   0.3    0.5
4   #F8766D 3.1 4.6     1     1    16    5   NA   0.3    0.5
...

[[2]]
     colour        x        y PANEL group   fill size linetype weight alpha
1   #F8766D 4.300000 2.864239     1     1 grey60    1        1      1   0.4
2   #F8766D 4.318987 2.879401     1     1 grey60    1        1      1   0.4
3   #F8766D 4.337975 2.894563     1     1 grey60    1        1      1   0.4
4   #F8766D 4.356962 2.909725     1     1 grey60    1        1      1   0.4
...
```

panel

```
> p_build$panel
$layout
  PANEL ROW COL SCALE_X SCALE_Y
1     1   1   1       1       1

$shrink
[1] TRUE

$x_scales
$x_scales[[1]]
<ScaleContinuousPosition>
 Range:   4.3 --  7.9
 Limits:  4.3 --    8

$y_scales
$y_scales[[1]]
<ScaleContinuousPosition>
 ...
```

plot

```
> p_build$plot
```

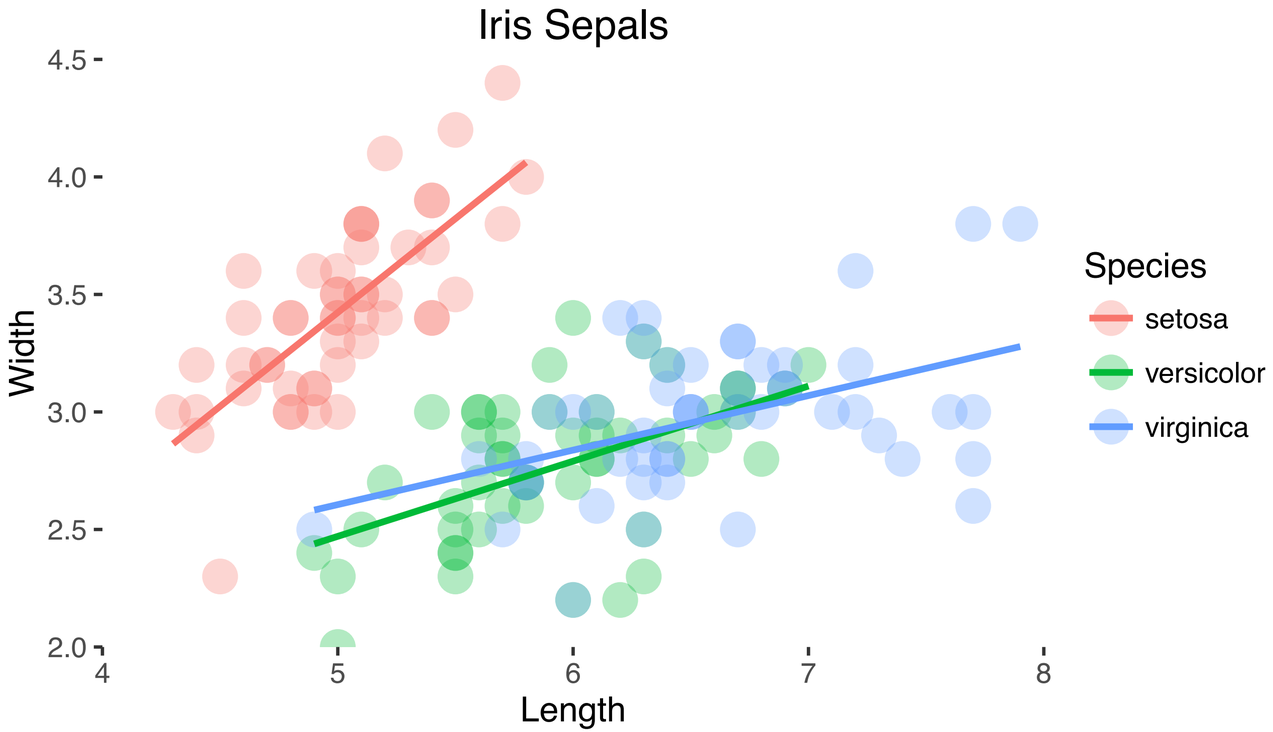


gtable

```
> p_build <- ggplot_build(p)
> gtab <- ggplot_gtable(p_build)

> gtab
TableGrob (6 x 6) "layout": 9 grobs
  z     cells       name                                           grob
1 0 (1-6,1-6) background zeroGrob[plot.background..zeroGrob.25361]
2 3 (3-3,3-3)     axis-l     absoluteGrob[GRID.absoluteGrob.25330]
3 1 (4-4,3-3)     spacer                         zeroGrob[NULL]
4 2 (3-3,4-4)      panel                  gTree[GRID.gTree.25316]
5 4 (4-4,4-4)     axis-b     absoluteGrob[GRID.absoluteGrob.25323]
6 5 (5-5,4-4)       xlab  titleGrob[axis.title.x..titleGrob.25333]
7 6 (3-3,2-2)       ylab  titleGrob[axis.title.y..titleGrob.25336]
8 7 (3-3,5-5)  guide-box                       gtable[guide-box]
9 8 (2-2,4-4)      title    titleGrob[plot.title..titleGrob.25360]

> gtab2 <- ggplotGrob(p) # same thing
```

grid.draw

```
> library(grid)
> grid.draw(gtab)
```

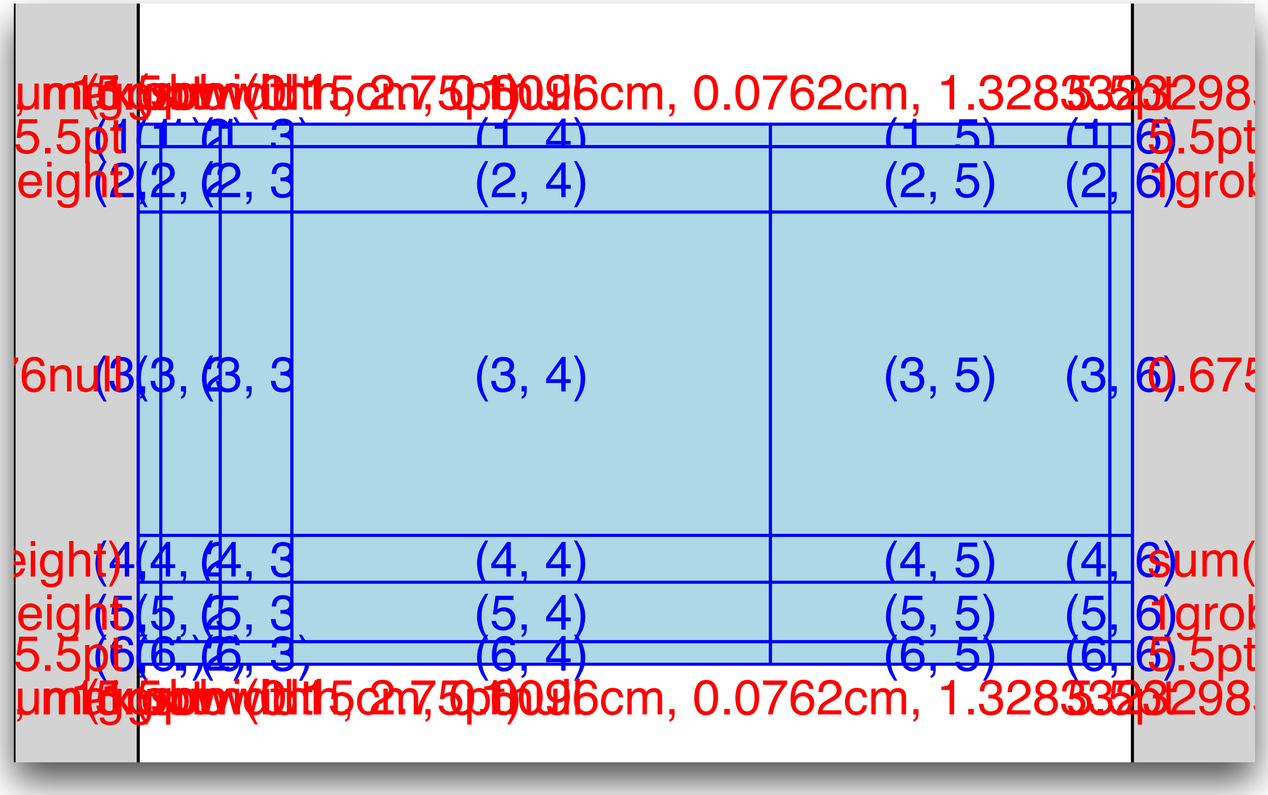


gtab

```
> names(gtab)
 [1] "grobs"          "layout"         "widths"         "heights"
 [5] "respect"        "rownames"       "colnames"       "name"
 [9] "gp"             "vp"             "children"       "childrenOrder"
```

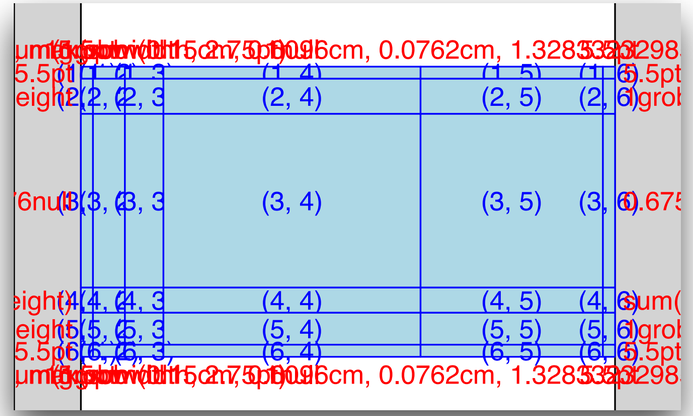
layout

```
> gtable_show_layout(gtab)
```



layout (2)

```
> gtab$layout
  t l b r z clip       name
9 1 1 6 6 0   on background
1 3 3 3 3 3  off     axis-l
2 4 3 4 3 1  off     spacer
3 3 4 3 4 2   on      panel
4 4 4 4 4 4  off     axis-b
5 5 4 5 4 5  off       xlab
6 3 2 3 2 6  off       ylab
7 3 5 3 5 7  off  guide-box
8 2 4 2 4 8  off      title
```

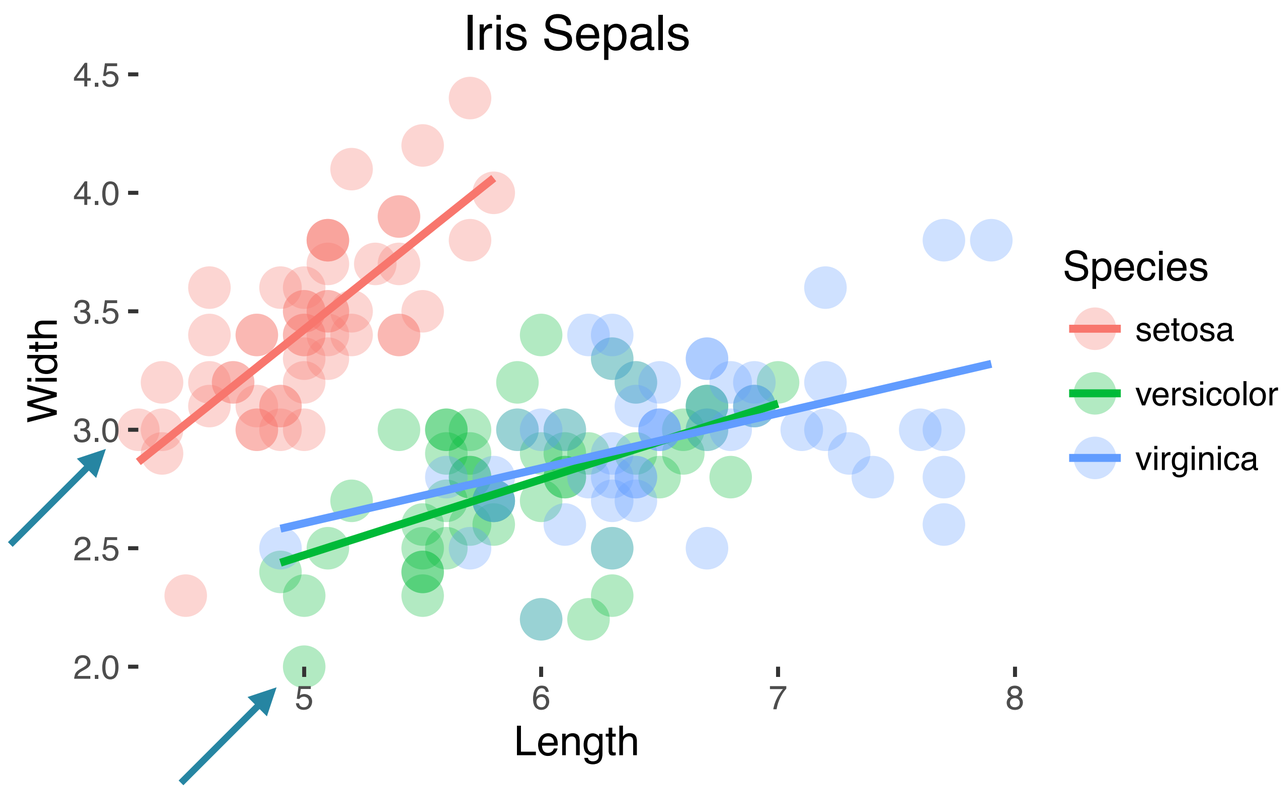


Update clipping

```
> gtab$layout$clip
[1] "on"  "off" "off" "on"  "off" "off" "off" "off" "off"

> gtab$layout$clip[gtab$layout$name == "panel"] <- "off"
```

Redraw

```
> library(grid)
> grid.draw(gtab)
```



DATA VISUALIZATION WITH GGPLOT2

Let’s practice!

DATA VISUALIZATION WITH GGPLOT2

gridExtra

gridExtra

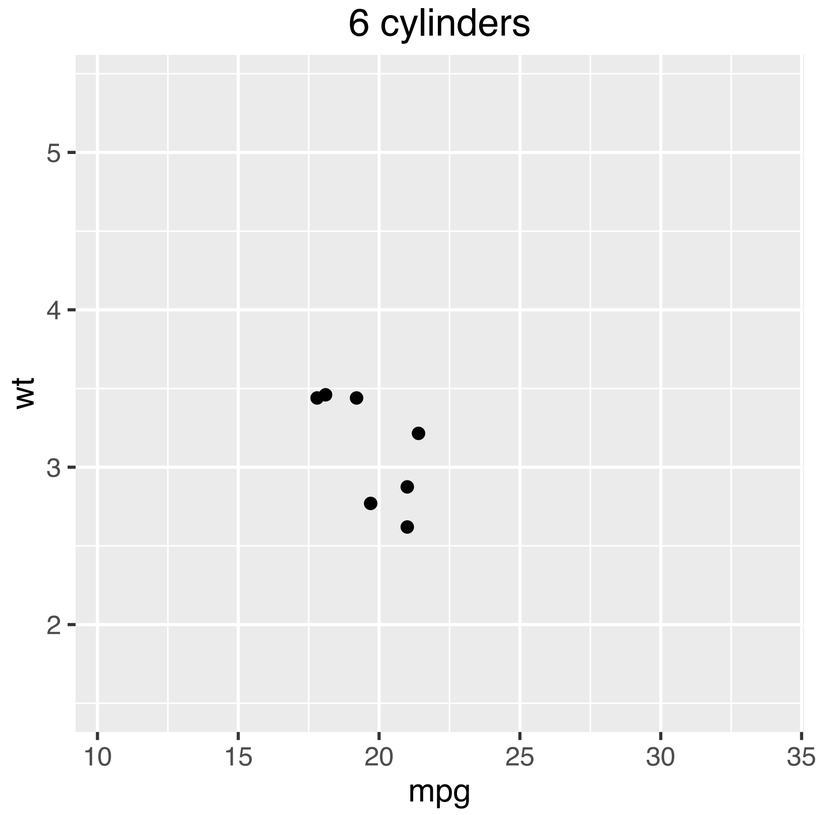
- Manage multiple plotting objects
- Reasons
  - Avoid giant facetted plot
  - Defer plotting
  - Arrange multiple plots in layout
  - Make a multiple page pdf of plots

Build multiple plots

```
> library(plyr)

> my_plots <- dlply(mtcars, .(cyl), function(df) {
    ggplot(df, aes(mpg, wt)) +
      geom_point() +
      xlim(range(mtcars$mpg)) +
      ylim(range(mtcars$wt)) +
      ggtitle(paste(df$cyl[1], "cylinders"))})

> length(my_plots)
[1] 3

> names(my_plots)
[1] "4" "6" "8"
```

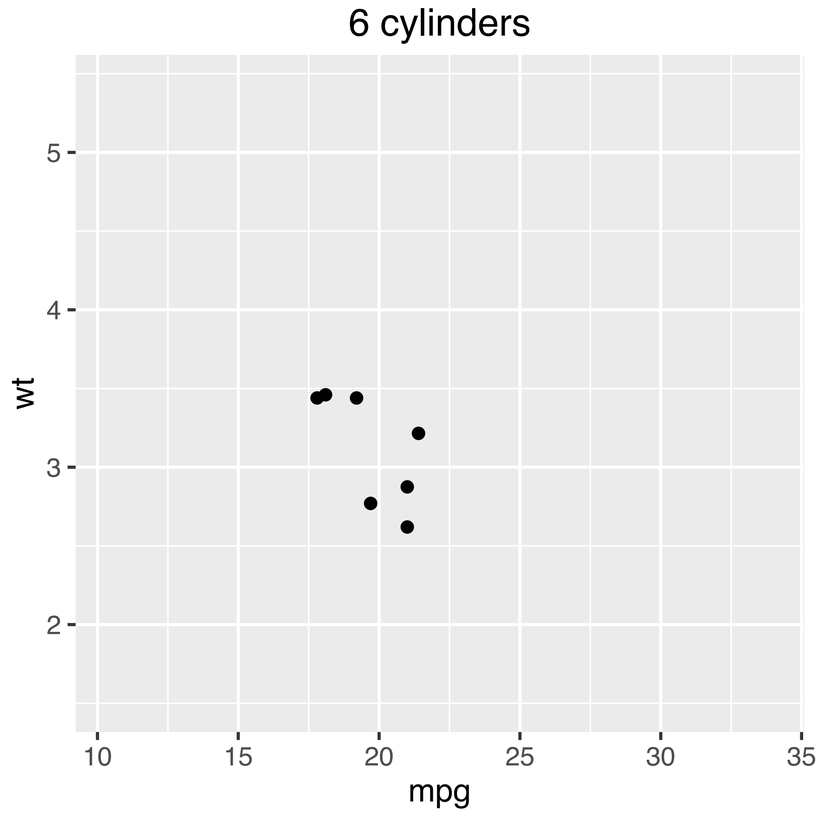
Plot by position

```
> my_plots[[2]]
```



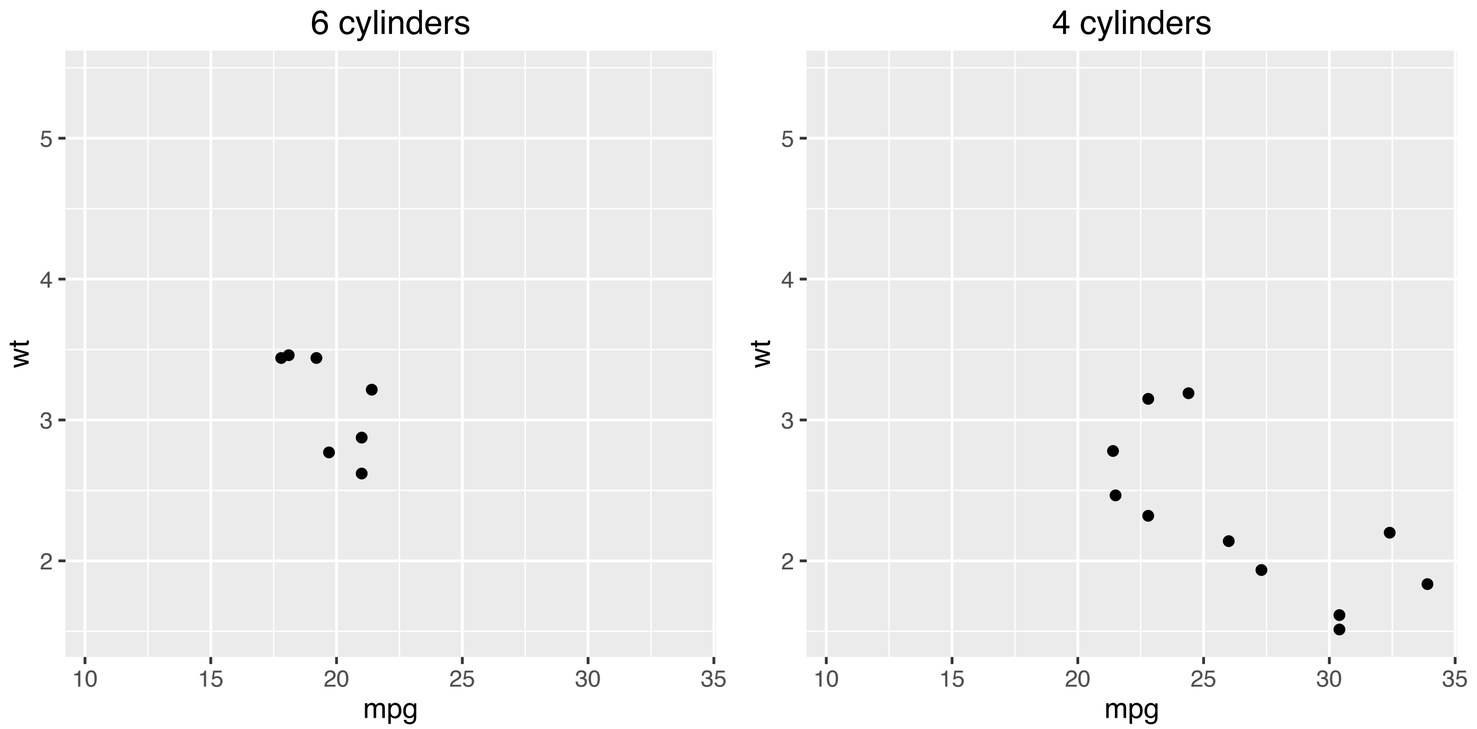
Plot by name

```
> my_plots[["6"]]
```



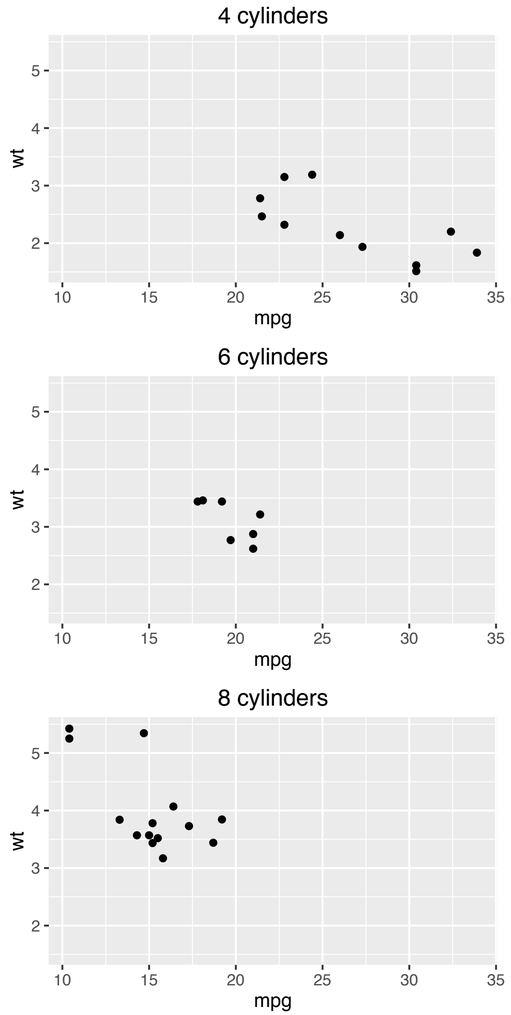
Combine plots (1)

```
> library(gridExtra)
> grid.arrange(my_plots[[2]], my_plots[[1]], ncol = 2)
```

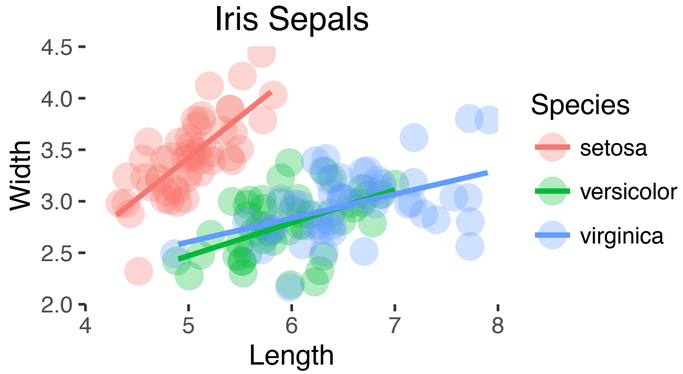


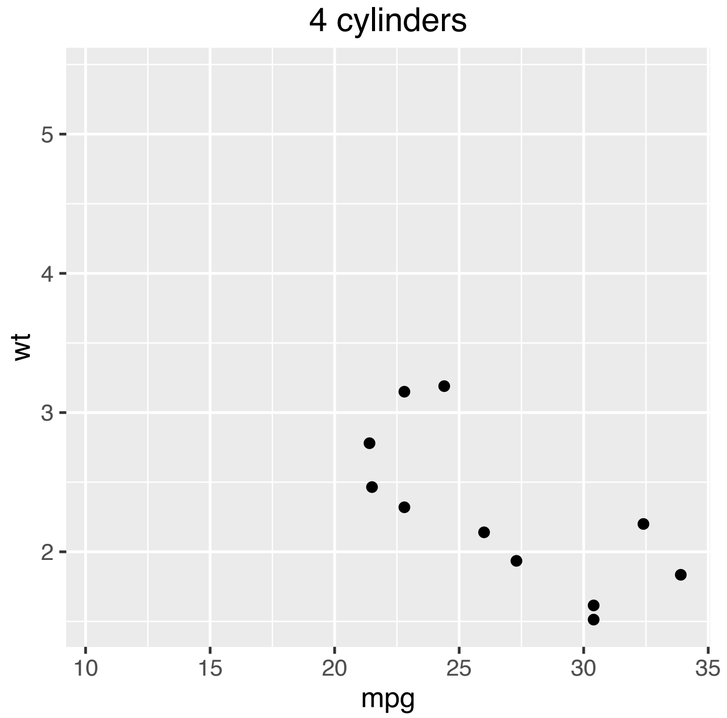
Combine plots (2)

```
> do.call(grid.arrange, my_plots)
```



Combine plots (3)






```
> grid.arrange(p, my_plots[[1]])
```

Why grid.arrange()?

- You are not able to make manual adjustments
- Creating many of the same composite plots
  - Slight variations (different dataset or variables)

DATA VISUALIZATION WITH GGPLOT2

Let’s practice!