Reporting data results #1

Data density

High data density

Guideline 1: **Aim for high data density.**

You should try to increase, as much as possible, the **data to ink ratio** in your graphs. This is the ratio of "ink" providing information to all ink used in the figure.

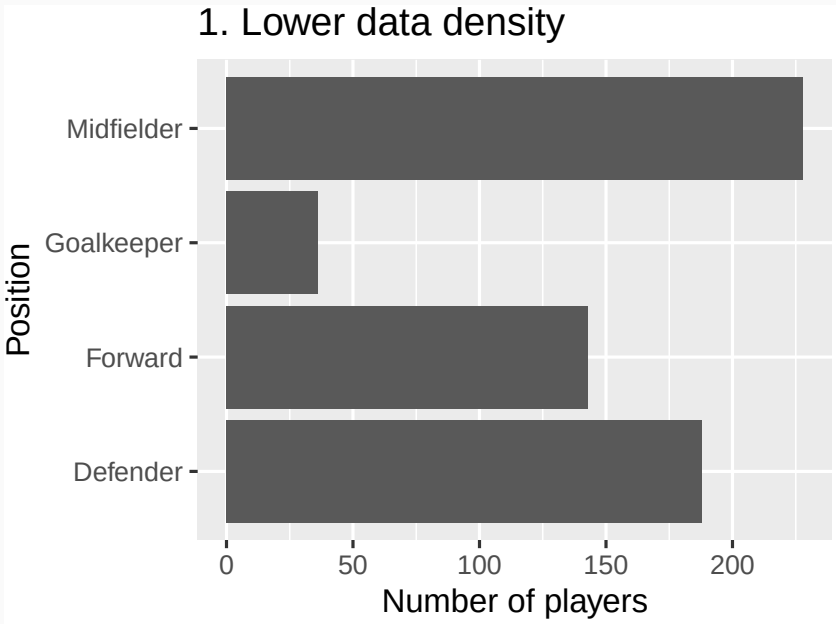
One way to think about this is that the only graphs you make that use up a lot of your printer's ink should be packed with information.

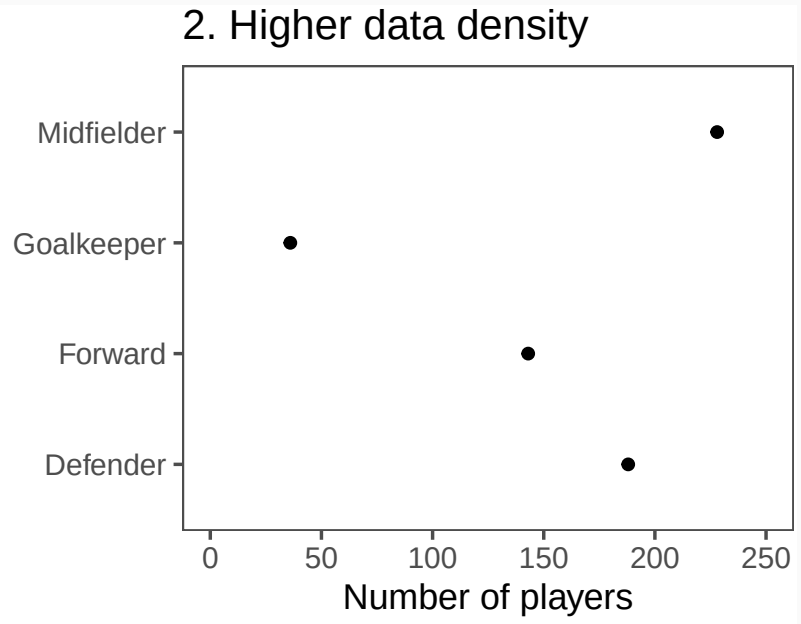
High data density

Guideline 1: **Aim for high data density.**

The two graphs below show the same information. Compare the amount of ink used in the left plot to the amount used in the right plot to see how graphs with the same information can have very different data densities.

```
## `summarise()` ungrouping output (override with `.groups` argu
```

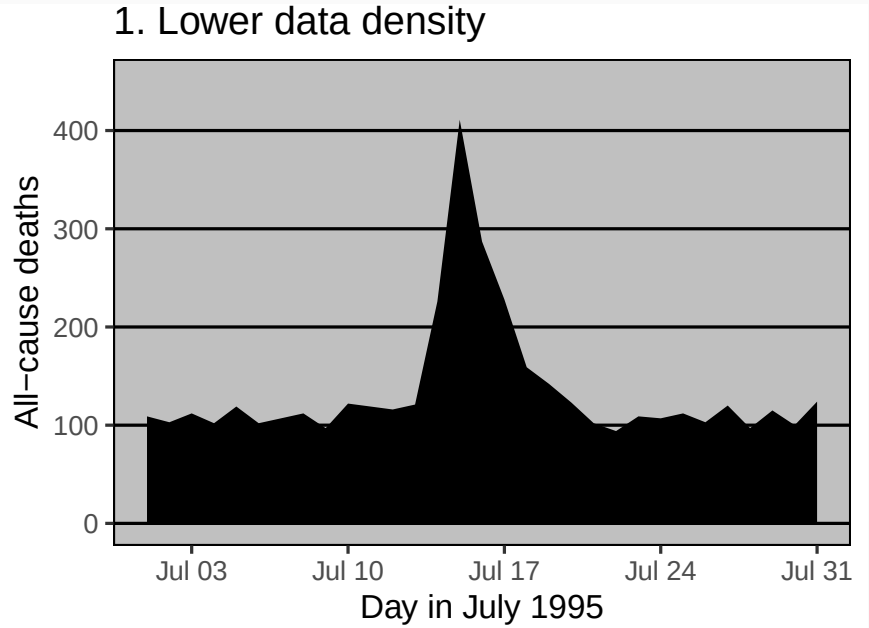


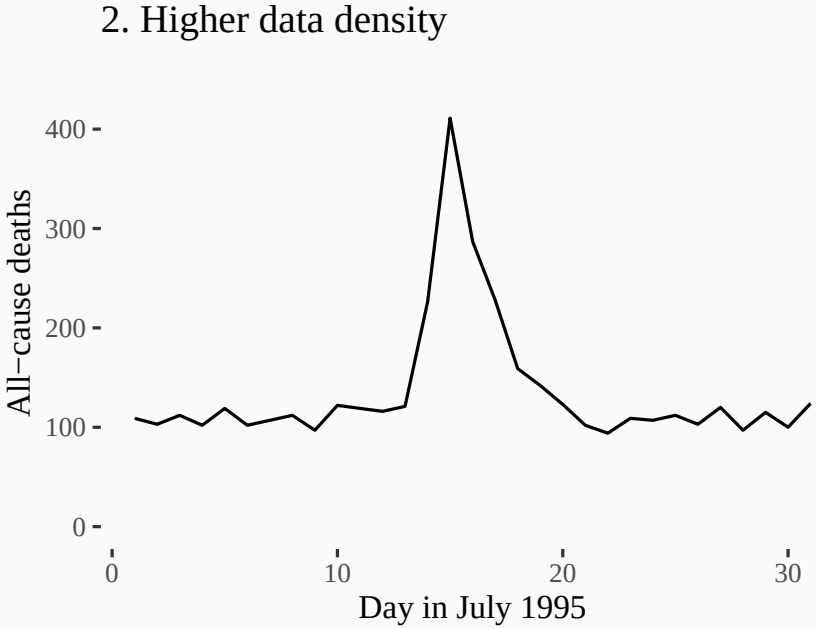


High data density

Guideline 1: **Aim for high data density.**

The two graphs below show another example of very different data densities in two plots showing the same information:





Data density

One quick way to increase data density in `ggplot2` is to change the *theme* for the plot. This essentially changes the "background" elements to a plot, including elements like the plot grid, background color, and the font used for labeling.

Some themes come with `ggplot2`, including:

- `theme_classic`
- `theme_bw`
- `theme_minimal`
- `theme_void`

The `ggthemes` packages has some excellent additional themes.

Data density

The following slides show some examples of the effects of using different themes. The following code creates a plot of daily deaths in Chicago in July 1995:

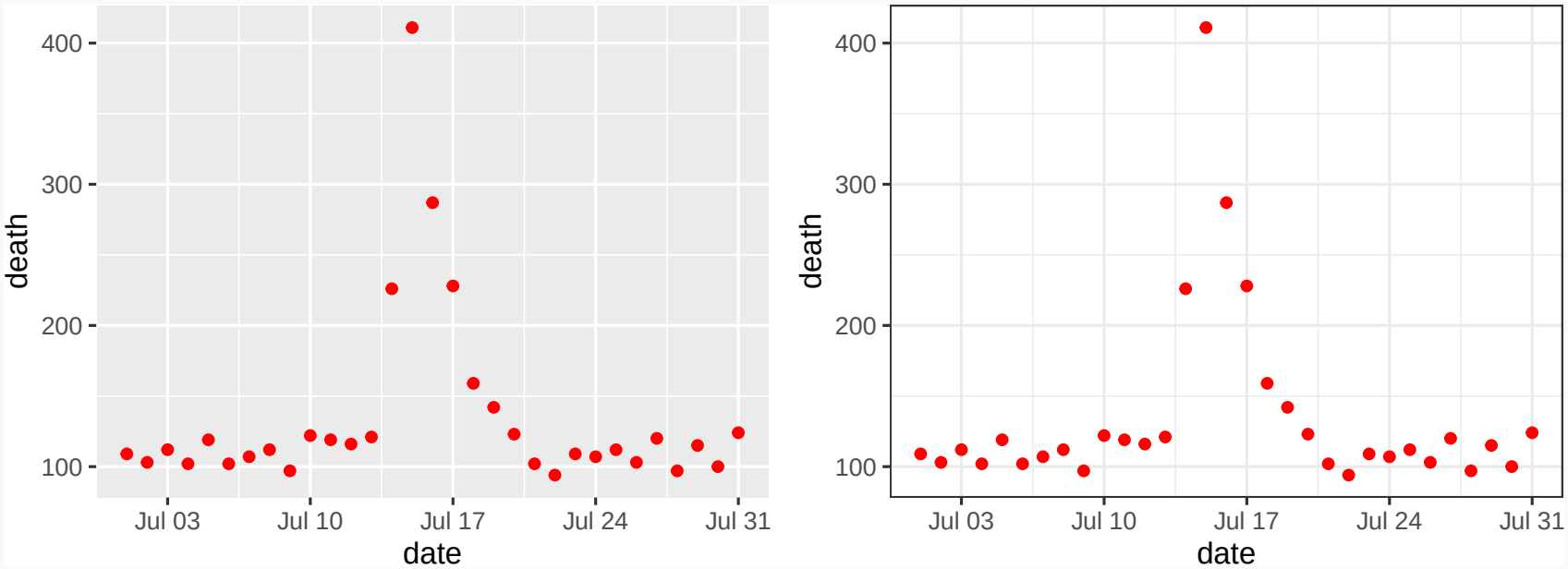
```
chic_plot <- ggplot(chic_july, aes(x = date, y = death))  +
        geom_point(color = "red")
```

Next, we can see how the graph looks with the default theme and with other themes.

Themes

The left graph shows the graph with the default theme, while the right shows the effect of adding the black-and-white theme that comes with ggplot2 as `theme_bw`:

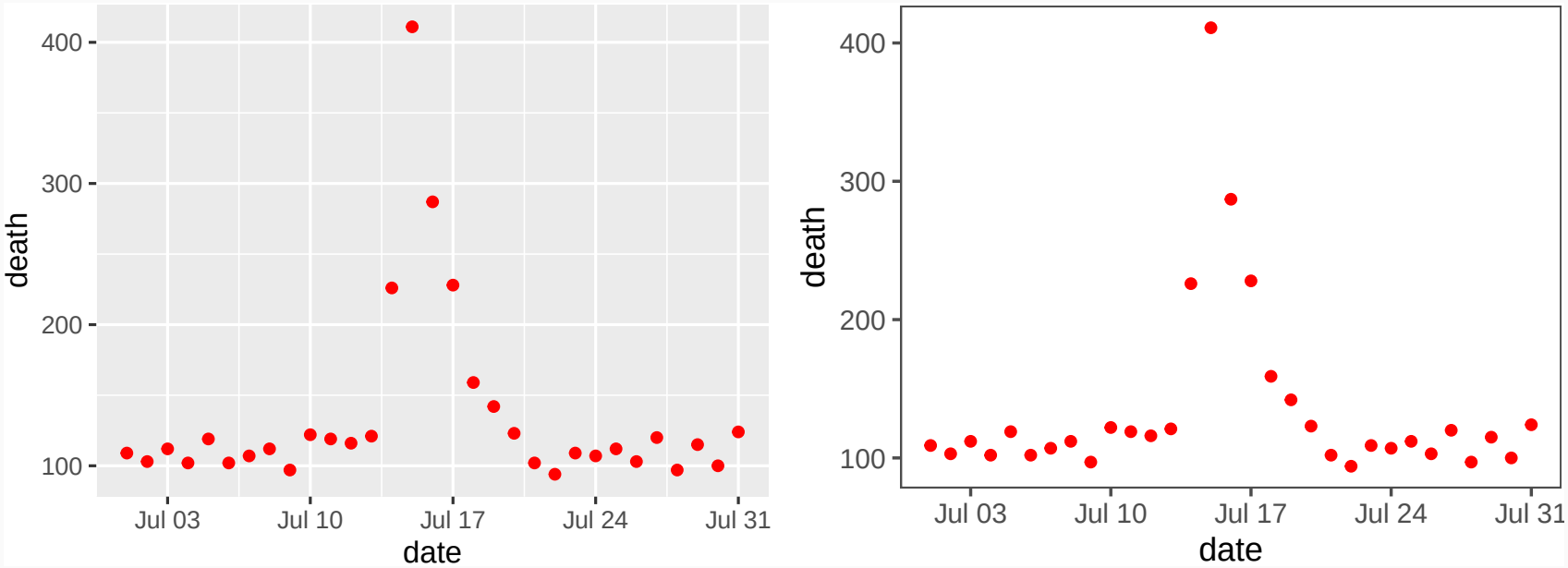
```
a <- chic_plot
b <- chic_plot + theme_bw()
grid.arrange(a, b, ncol = 2)
```



Themes

Stephen Few theme:

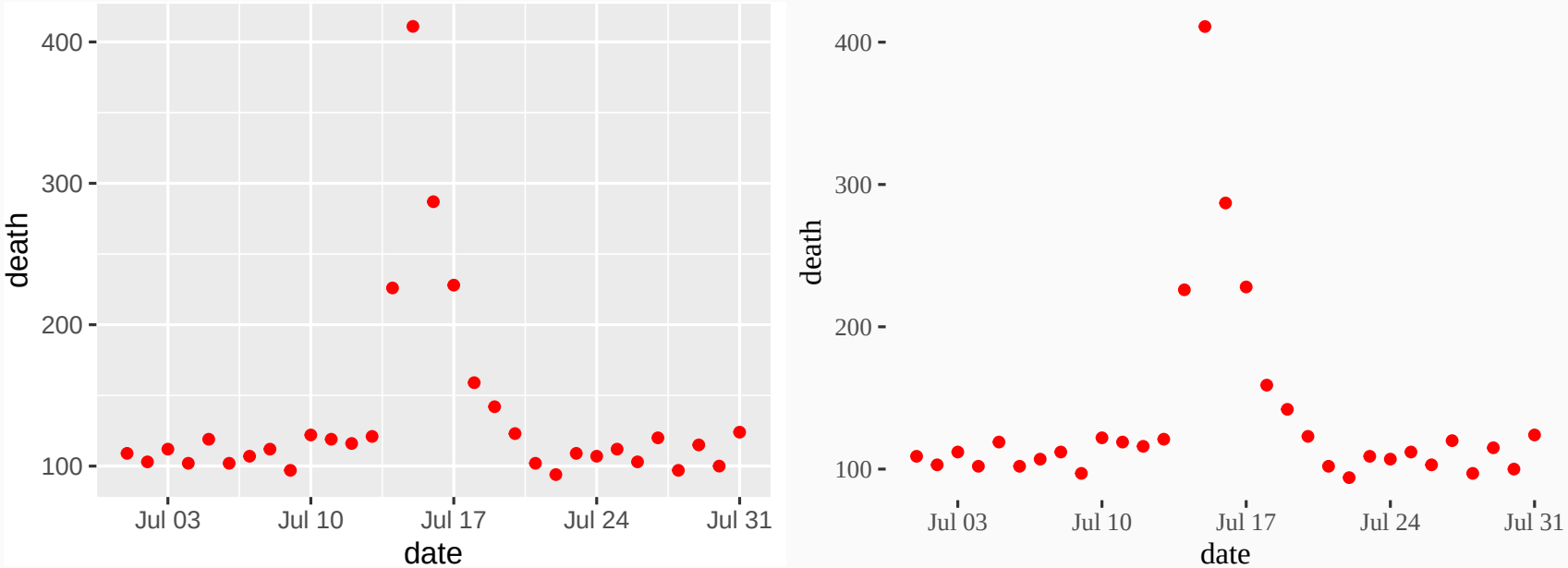
```
a <- chic_plot
b <- chic_plot + theme_few()
grid.arrange(a, b, ncol = 2)
```



Themes

Edward Tufte theme:

```
a <- chic_plot
b <- chic_plot + theme_tufte()
grid.arrange(a, b, ncol = 2)
```



Themes

You can even use themes to add some questionable choices for different elements. For example, `ggthemes` includes an Excel theme:

```
a <- chic_plot
b <- chic_plot + theme_excel()
grid.arrange(a, b, ncol = 2)
```