Lecture - Behind the scenes of ggplot2

Data Visualisation

dr. Sanne Willems

Statistical layers vs. geometrical layers

Statistics in geometrical layers

Behind the scenes, some geometrical layers are calculating some statistics before they plot output:

Example: geom_bar is calculating counts

```
geom_bar
```

```
## function (mapping = NULL, data = NULL, stat = "count", position = "stack",
##     ..., just = 0.5, width = NULL, na.rm = FALSE, orientation = NA,
##     show.legend = NA, inherit.aes = TRUE)
## {
##     layer(data = data, mapping = mapping, stat = stat, geom = GeomBar,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
##         params = list2(just = just, width = width, na.rm = na.rm,
##             orientation = orientation, ...))
## }
## <bytecode: 0x7ff2a7d89110>
## <environment: namespace:ggplot2>
```

Statistics in geometrical layers

Behind the scenes, some geometrical layers are calculating some statistics before they plot output:

Example: geom_histogram is binning the data

```
geom_histogram
```

```
## function (mapping = NULL, data = NULL, stat = "bin", position = "stack",
##     ..., binwidth = NULL, bins = NULL, na.rm = FALSE, orientation = NA,
##     show.legend = NA, inherit.aes = TRUE)
## {
##     layer(data = data, mapping = mapping, stat = stat, geom = GeomBar,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
##         params = list2(binwidth = binwidth, bins = bins, na.rm = na.rm,
##             orientation = orientation, pad = FALSE, ...))
## }
## <bytecode: 0x7fdee9f01d28>
## <environment: namespace:ggplot2>
```

Statistics in geometrical layers

Behind the scenes, some geometrical layers are calculating some statistics before they plot output:

Example: geom_density is estimating the density

```
geom_density
```

```
## function (mapping = NULL, data = NULL, stat = "density", position = "identity",
##     ..., na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE,
##     outline.type = "upper")
## {
##     outline.type <- arg_match0(outline.type, c("both", "upper",
##         "lower", "full"))
##     layer(data = data, mapping = mapping, stat = stat, geom = GeomDensity,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
##         params = list2(na.rm = na.rm, orientation = orientation,
##             outline.type = outline.type, ...))
## }
## <bytecode: 0x7f7c0c1e6910>
## <environment: namespace:ggplot2>
```

Statistics in geometrical layers

But some geometrical layers do not require any calculations, so these use the data “as is” (“identity”):

Example: geom_line

```
geom_line
```

```
## function (mapping = NULL, data = NULL, stat = "identity", position = "identity",
##     na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE,
##     ...)
## {
##     layer(data = data, mapping = mapping, stat = stat, geom = GeomLine,
##         position = position, show.legend = show.legend, inherit.aes = inherit.aes,
##         params = list2(na.rm = na.rm, orientation = orientation,
##             ...))
## }
## <bytecode: 0x7f7bdd2a8138>
## <environment: namespace:ggplot2>
```

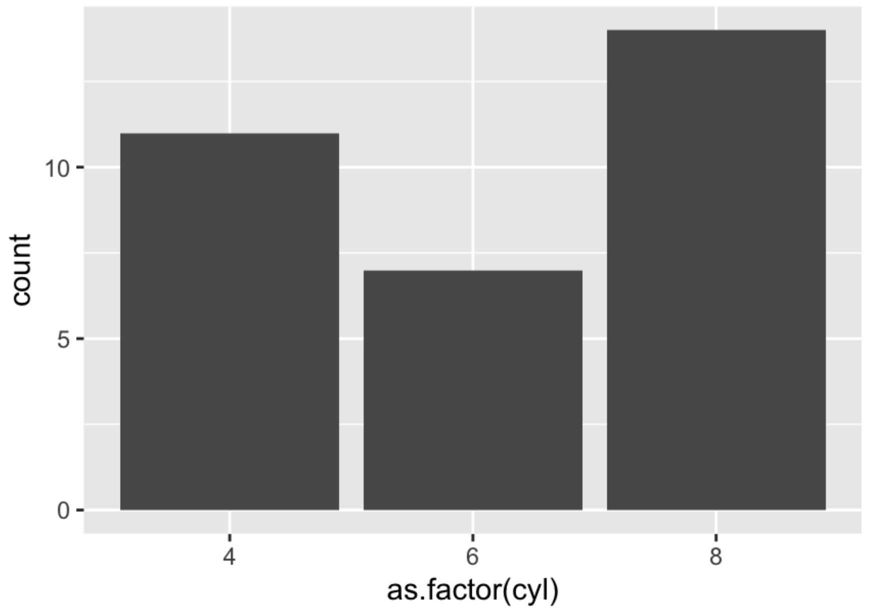
Example:

Using the fact that ggplot can do calculations in the background, we can ask it to plot the number of cars grouped by number of cylinders in a bar chart:

geom_bar will count the number of cars in each group:

```
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl))) +
  geom_bar()

# in the background:
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl))) +
  geom_bar(stat = 'count')
```

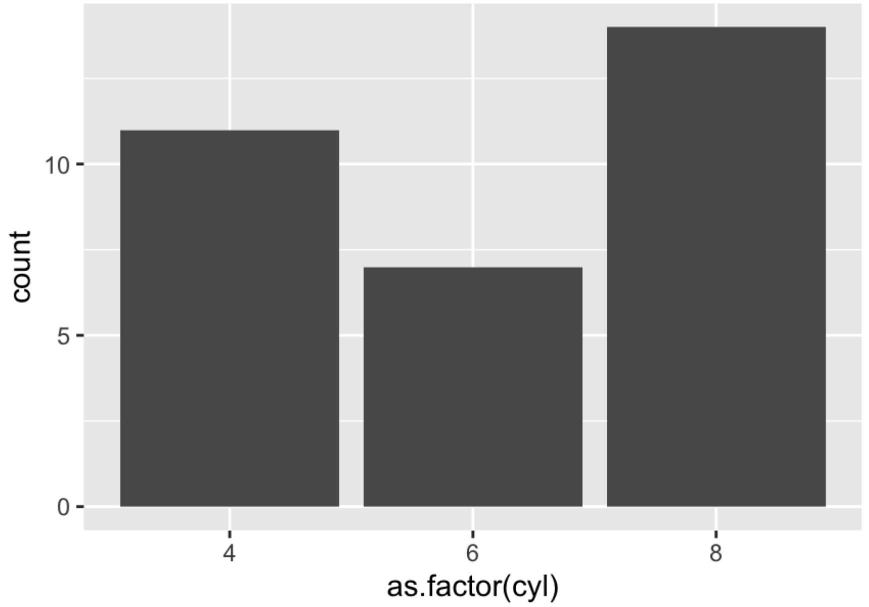


Reverse: geometric object to plot statistics

You can also do this the other way around with the stat_*() functions.

- First, specify which statistic ggplot should calculate
- Then, specify the type of plot you want to make with it

```
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl))) +
  stat_count(geom = "bar")
```

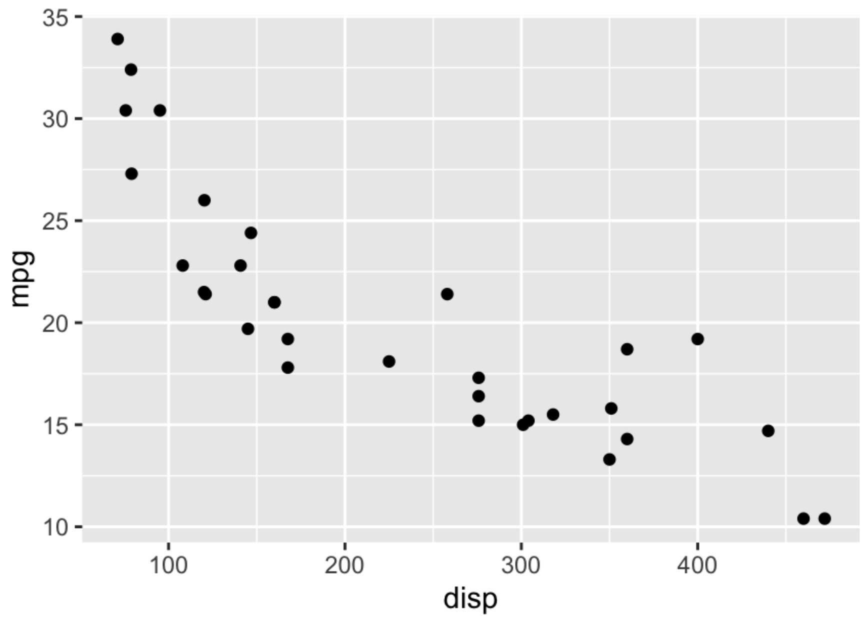


Reverse: geometric object to plot statistics

You can also do this the other way around with the stat_*() functions.

- First, specify which statistic ggplot should calculate
- Then, specify the type of plot you want to make with it

```
ggplot(data = mtcars,
       mapping = aes(x = disp,
                     y = mpg)) +
  stat_identity(geom = "point")
```

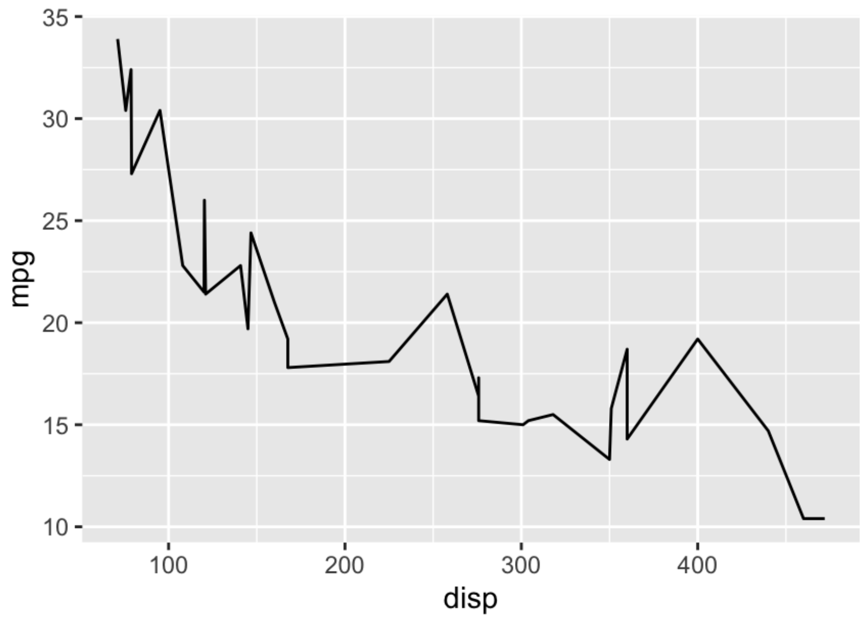


Reverse: geometric object to plot statistics

You can also do this the other way around with the stat_*() functions.

- First, specify which statistic ggplot should calculate
- Then, specify the type of plot you want to make with it

```r
ggplot(data = mtcars,
       mapping = aes(x = disp,
                     y = mpg)) +
  stat_identity(geom = "line")
```



Statistical layers vs. Geometrical layers

- In this course we focus on geometrical layers that calculate statistics
- This puts more focus on the visual aspects of the plot

The stages of ggplot

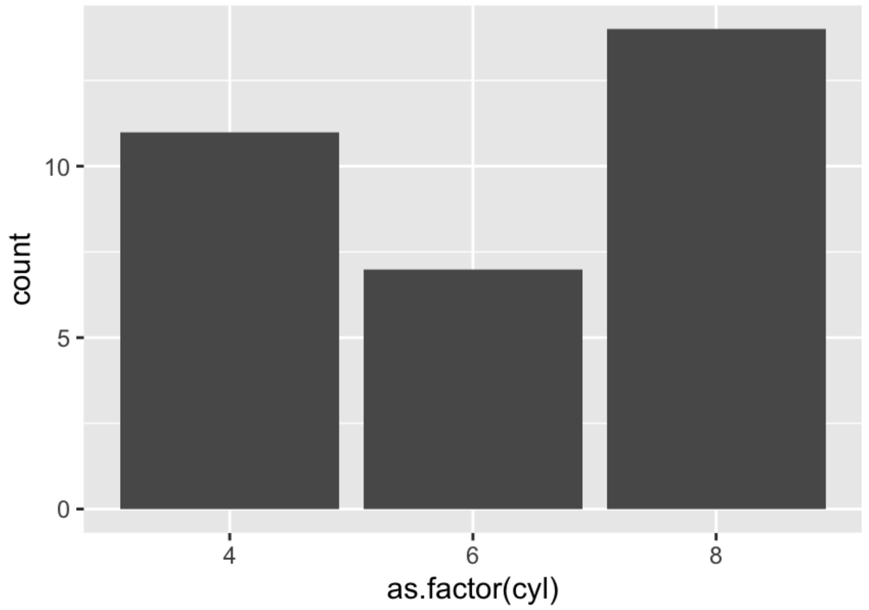
The stages of ggplot

- Stage 1: direct input - plot data provided by the user (including calculating statistics)
- Stage 2: after stat transformation - statistics have been calculated and can be used for plotting
- Stage 3: after scale transformation - values have been maped to scales

Using the stages of ggplot

Stage 1: direct input

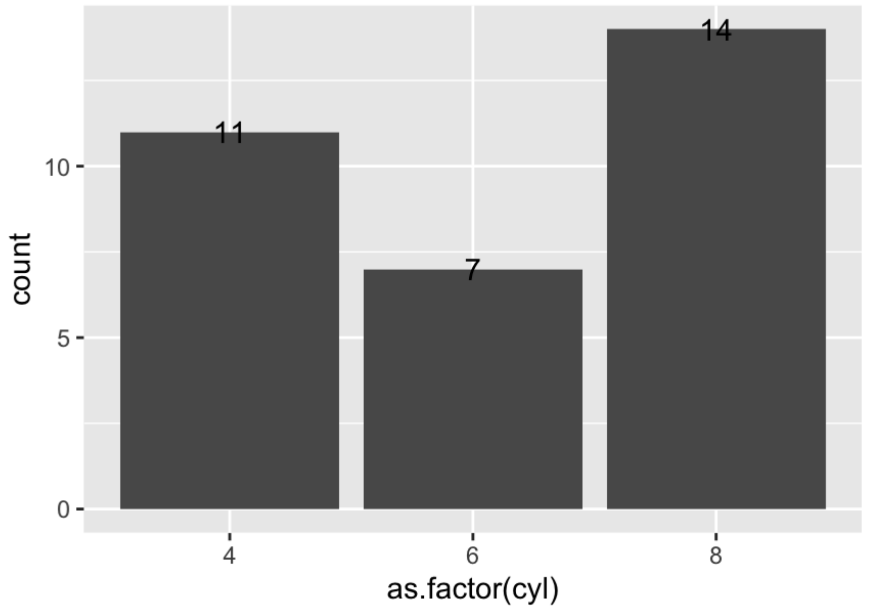
```
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl))) +
  geom_bar(stat = 'count')
```



Using the stages of ggplot

Stage 2: after stat transformation

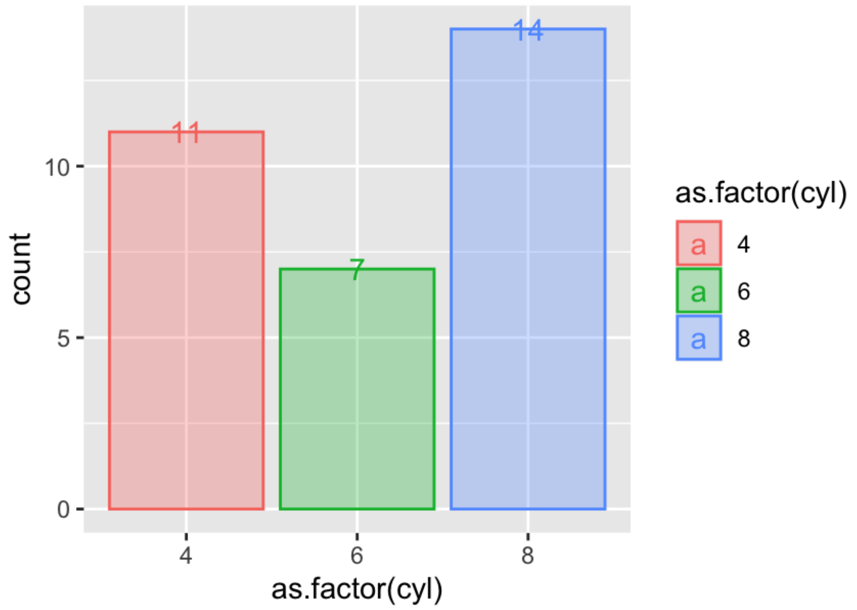
```
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl))) +
  geom_bar(stat = 'count') +
  geom_text(mapping = aes(
              label = after_stat(count)),
            stat = 'count')
```



Using the stages of ggplot

Stage 3: after scale transformation

```
ggplot(data = mtcars,
       mapping = aes(x = as.factor(cyl),
                     color = as.factor(cyl),
                     fill = after_scale(
                       alpha(color, 0.3)))) +
  geom_bar(stat = 'count') +
  geom_text(mapping = aes(
                label = after_stat(count)),
            stat = 'count')
```



The stages of ggplot

- Stage 1: direct input - plot data provided by the user (including calculating statistics)
- Stage 2: after stat transformation - statistics have been calculated and can be used for plotting
- Stage 3: after scale transformation - values have been maped to scales

More info: https://ggplot2.tidyverse.org/reference/aes_eval.html