StatComp_11931695

for Exercises

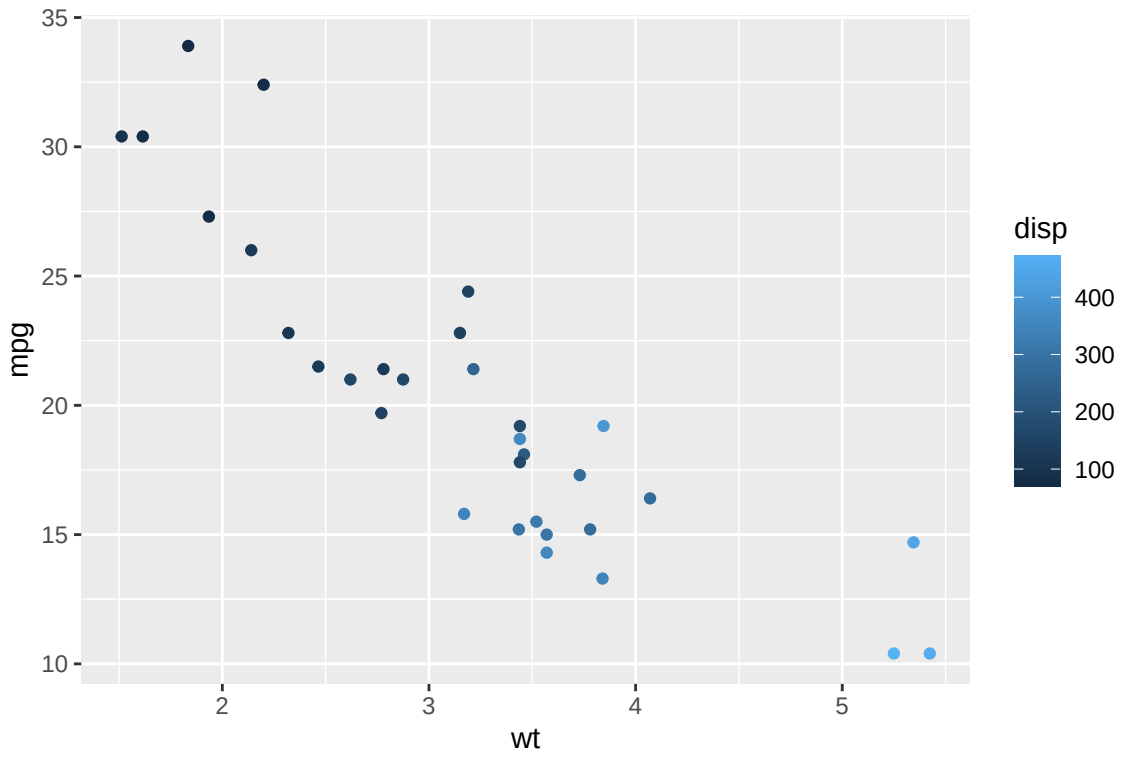
Eszter Katalin Bognar

19-05-2020

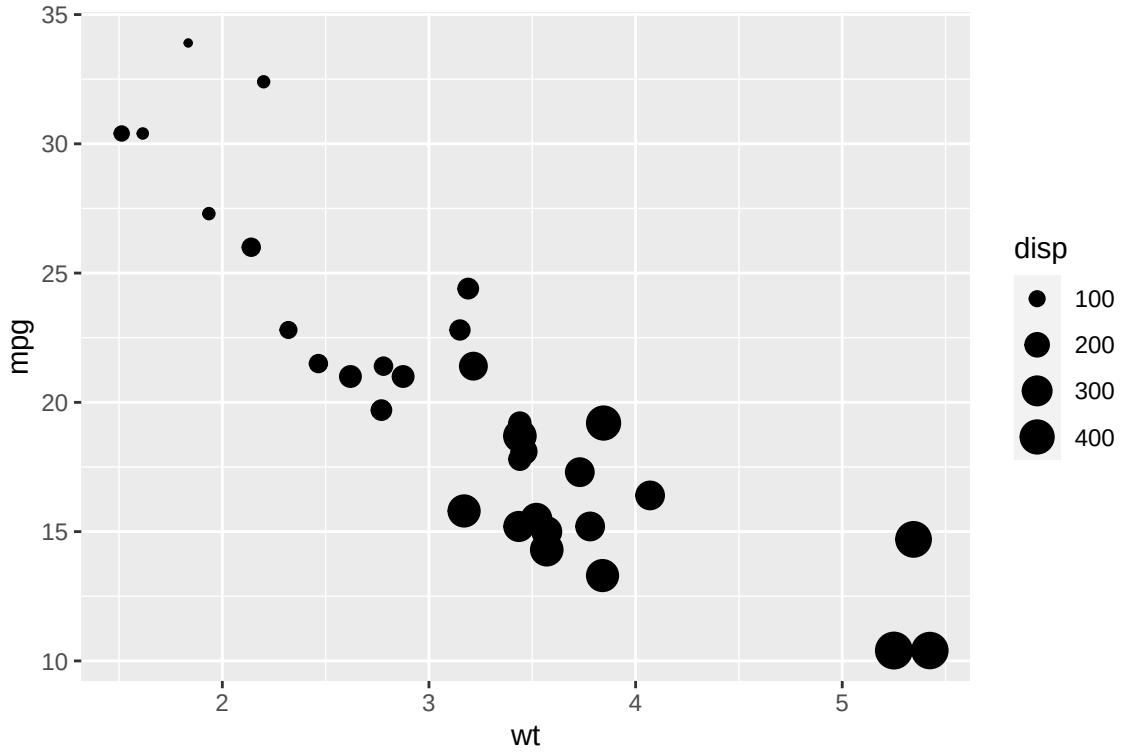
Introduction to Data Visualization with ggplot2

mapping onto a color gradient or onto a continuous size scale or shape (only for categorial variables)

```
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg, color = disp)) + geom_point()
```




```
ggplot(mtcars, aes(x = wt, y = mpg, size = disp)) + geom_point()
```

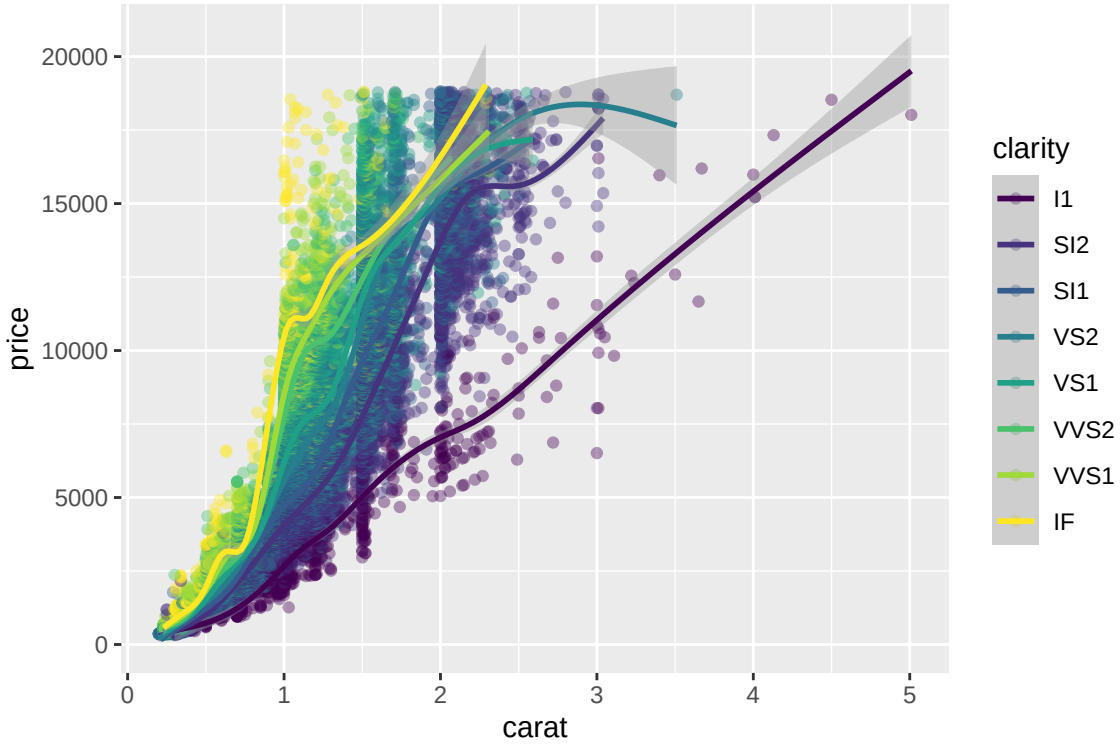


##

Adding geometries Common geom layer functions: -geom_point() adds points (as in a scatter plot) -geom_smooth() adds a smooth trend curve We can also make change to individual geoms, such as making the points 40% opaque

```
ggplot(diamonds, aes(x = carat, y = price, color = clarity)) +
  geom_point(alpha=0.4) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

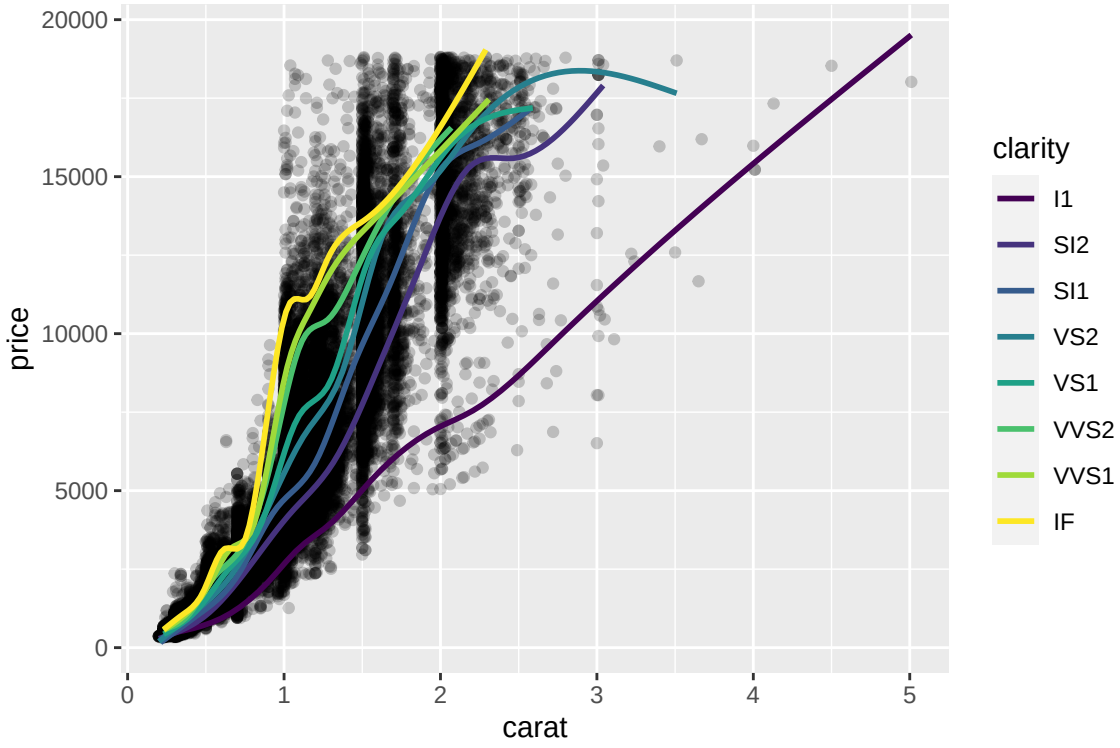


##

Make changes to individual geoms and saving plots as variables

```
plt_price_vs_carat <- ggplot(diamonds, aes(carat, price))

# Edit this to map color to clarity,
# Assign the updated plot to a new object
plt_price_vs_carat_by_clarity <- plt_price_vs_carat + geom_point(alpha = 0.2) + geom_smooth(aes(col = cl

# See the plot
plt_price_vs_carat_by_clarity
```

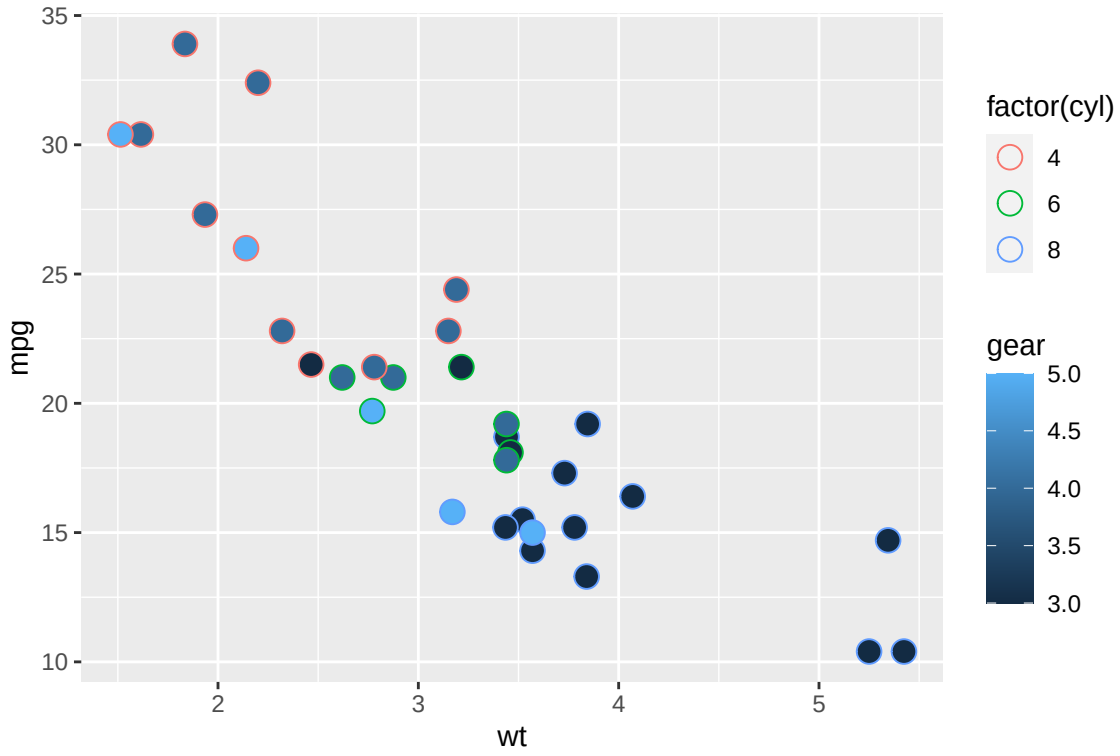
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



##

Visible aesthetics - color, fill, shape and size

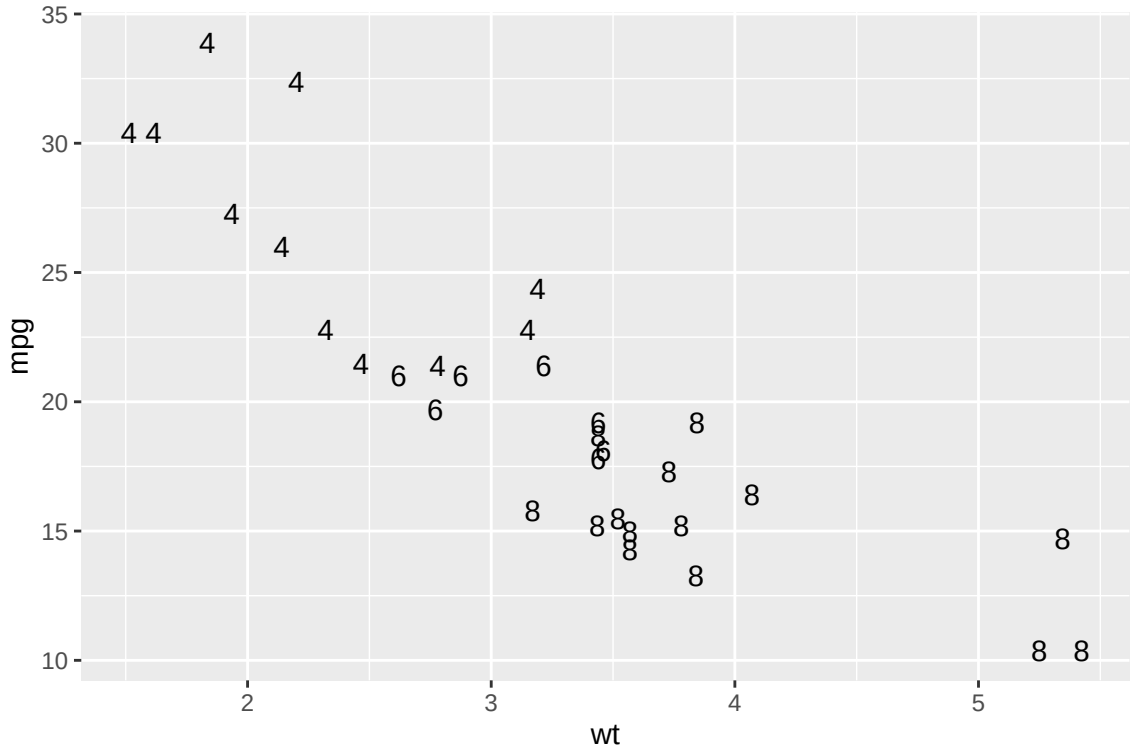
```
# Map cyl to fill and
ggplot(mtcars, aes(x = wt, y = mpg, fill=gear, color = factor(cyl))) +
  geom_point(shape = 21, size = 4)
```



Us-

ing geom_text

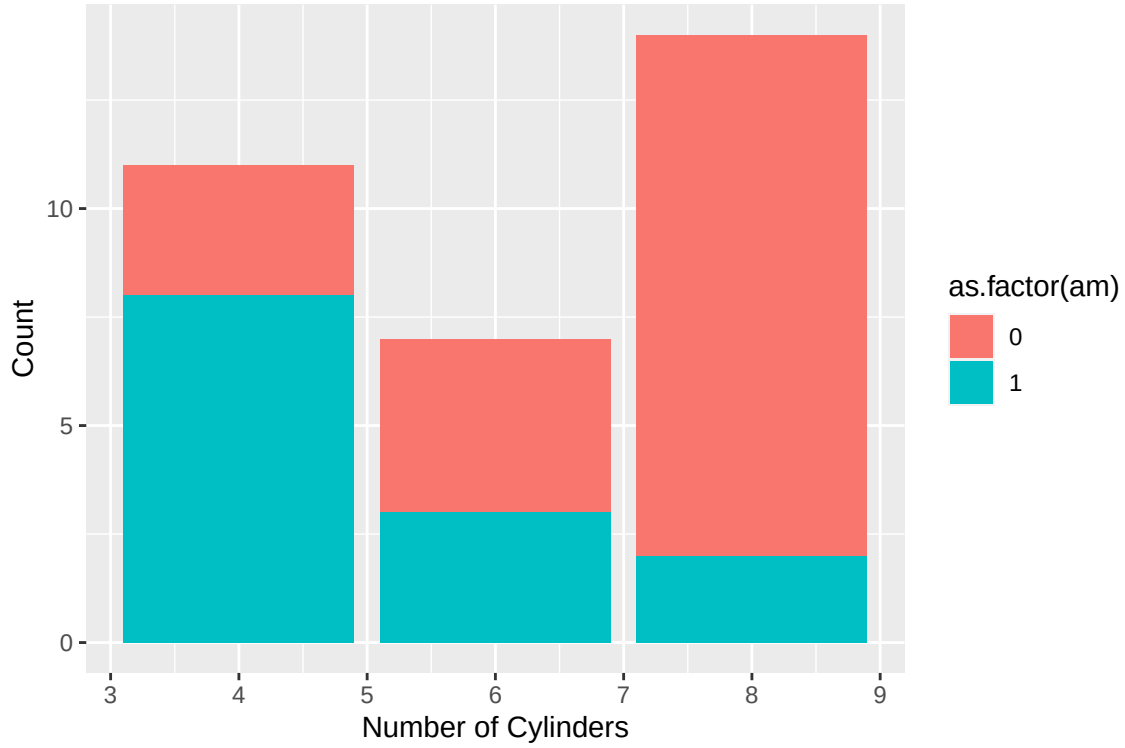
```
plt_mpg_vs_wt <- ggplot(mtcars, aes(wt, mpg))

# Use text layer and map fcyl to label
plt_mpg_vs_wt +
  geom_text(aes(label = cyl))
```



##

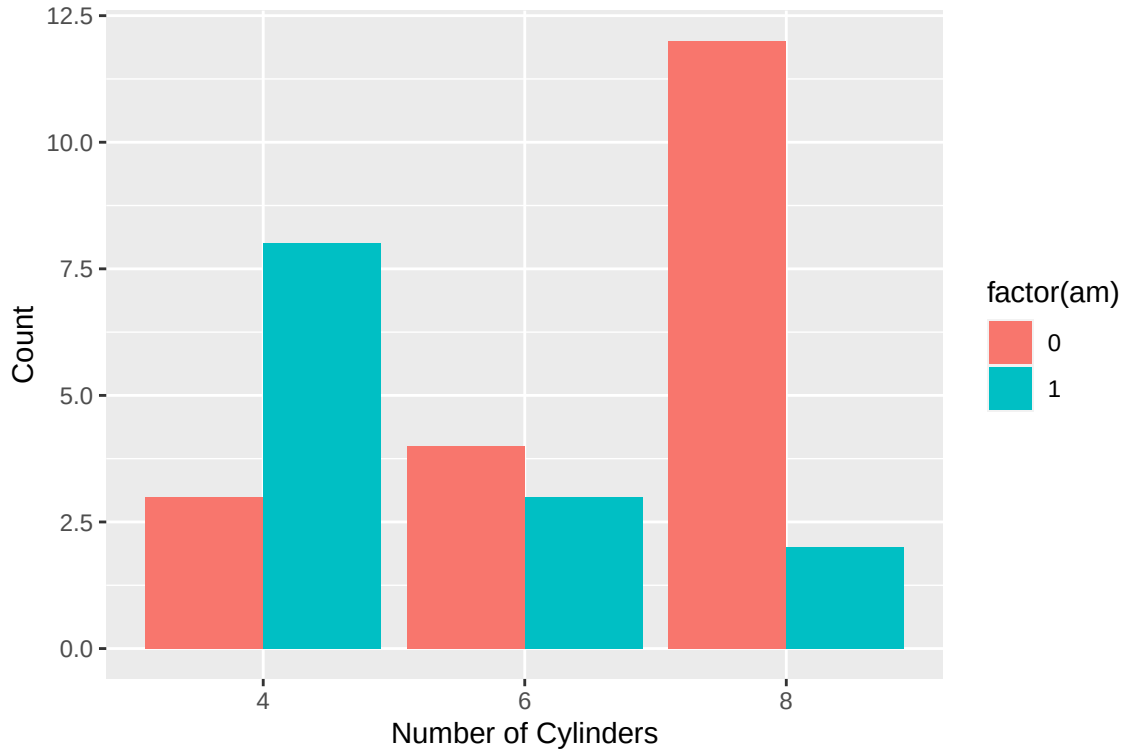
Updating aesthetic labels

```
ggplot(mtcars, aes(cyl, fill = as.factor(am))) +
  geom_bar() +
  # Set the axis labels
  labs(x = "Number of Cylinders", y = "Count")
```



Using custom color scale

```
palette <- c(automatic = "#377EB8", manual = "#E41A1C")

# Set the position
ggplot(mtcars, aes(factor(cyl), fill = factor(am))) +
  geom_bar(position = "dodge") +
  labs(x = "Number of Cylinders", y = "Count")
```



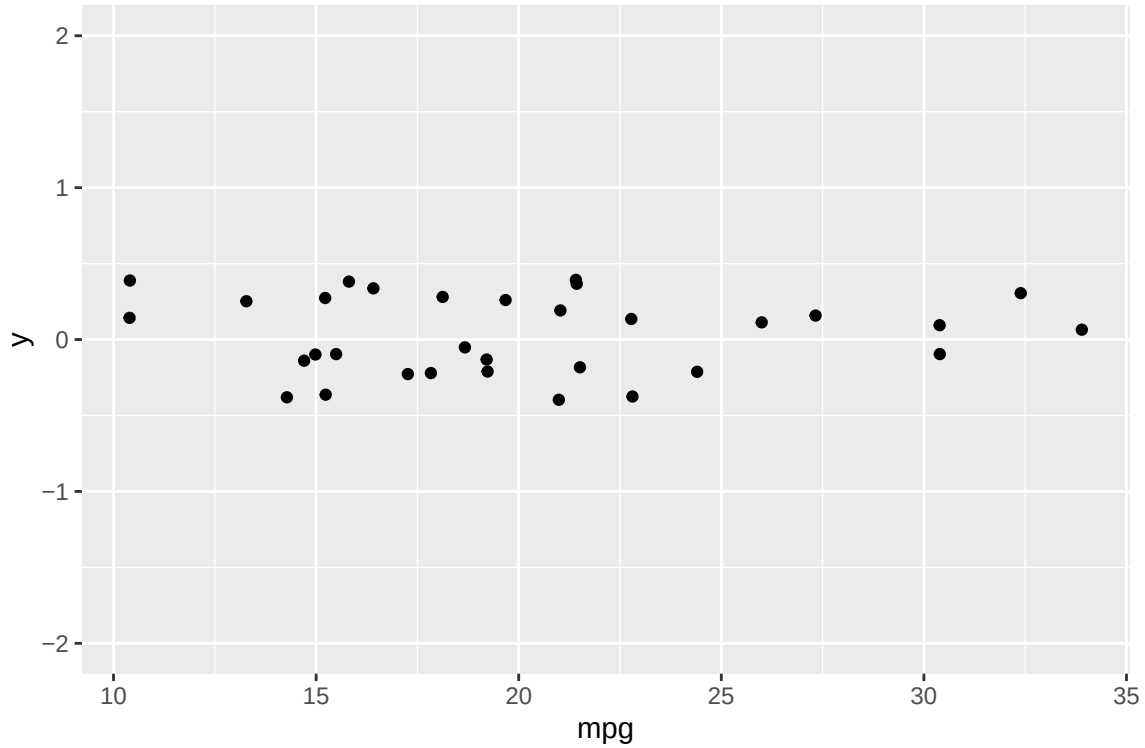

```
  scale_fill_manual("Transmission", values = palette)
```

```
## <ggproto object: Class ScaleDiscrete, Scale, gg>
##     aesthetics: fill
##     axis_order: function
##     break_info: function
##     break_positions: function
##     breaks: waiver
##     call: call
##     clone: function
##     dimension: function
##     drop: TRUE
##     expand: waiver
##     get_breaks: function
##     get_breaks_minor: function
##     get_labels: function
##     get_limits: function
##     guide: legend
##     is_discrete: function
##     is_empty: function
##     labels: waiver
##     limits: NULL
##     make_sec_title: function
##     make_title: function
##     map: function
##     map_df: function
##     n.breaks.cache: NULL
##     na.translate: TRUE
```

```
##     na.value: NA
##     name: Transmission
##     palette: function
##     palette.cache: NULL
##     position: left
##     range: <ggproto object: Class RangeDiscrete, Range, gg>
##         range: NULL
##         reset: function
##         train: function
##         super:  <ggproto object: Class RangeDiscrete, Range, gg>
##     rescale: function
##     reset: function
##     scale_name: manual
##     train: function
##     train_df: function
##     transform: function
##     transform_df: function
##     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
```

Setting a dummy aesthetic

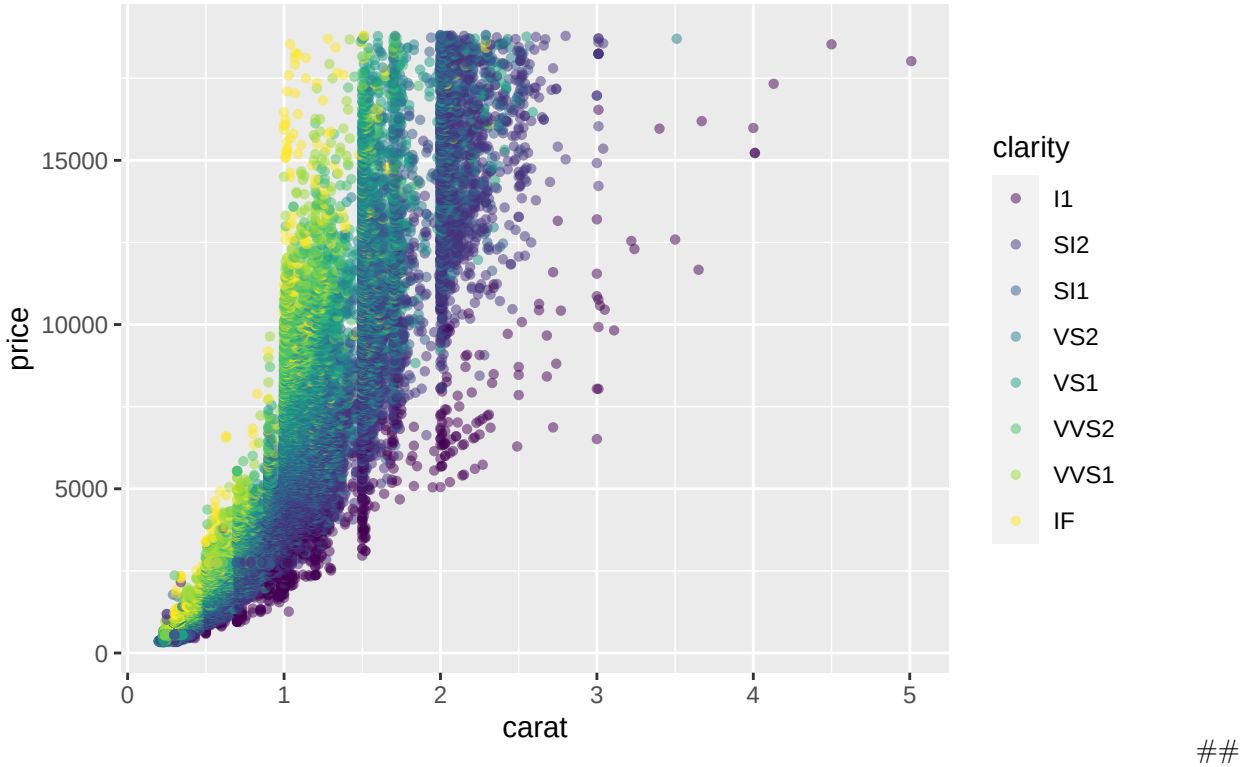
```
ggplot(mtcars, aes(mpg, 0)) +
  geom_jitter() +
  # Set the y-axis limits
  ylim(limits = c(-2,2))
```



##

Scatter plots

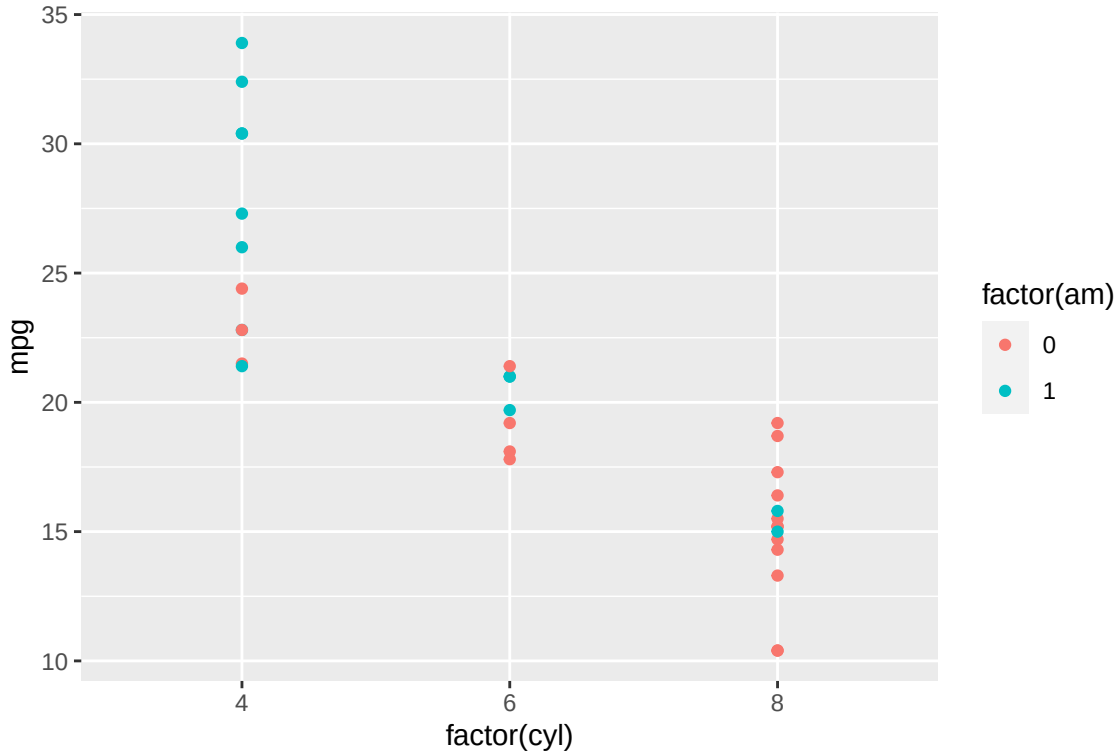
```
# Plot price vs. carat, colored by clarity
plt_price_vs_carat_by_clarity <- ggplot(diamonds, aes(carat, price, color = clarity))

# Set transparency to 0.5
plt_price_vs_carat_by_clarity + geom_point(alpha = 0.5, shape = 16)
```



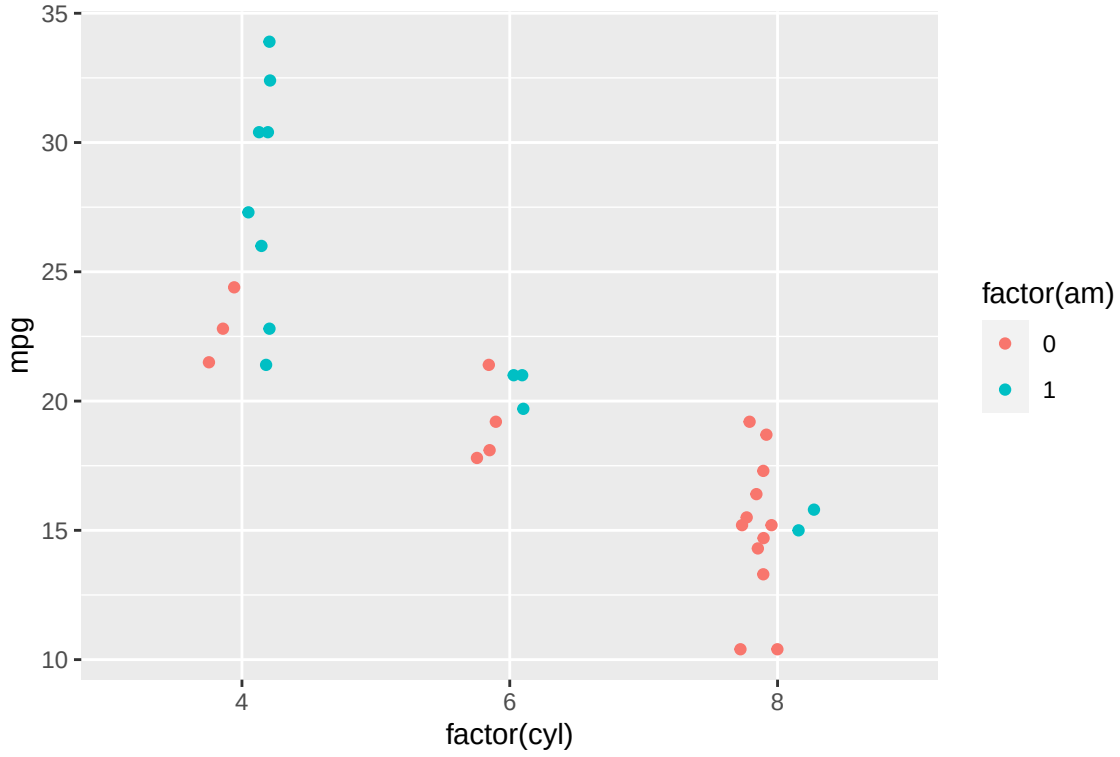
##

Aligned values

```
# Plot base
plt_mpg_vs_fcyl_by_fam <- ggplot(mtcars, aes(factor(cyl), mpg, color = factor(am)))

# Default points are shown for comparison
plt_mpg_vs_fcyl_by_fam + geom_point()
```




```
# Now jitter and dodge the point positions
plt_mpg_vs_fcyl_by_fam + geom_point(position = position_jitterdodge(jitter.width = 0.3, dodge.width = 0
```



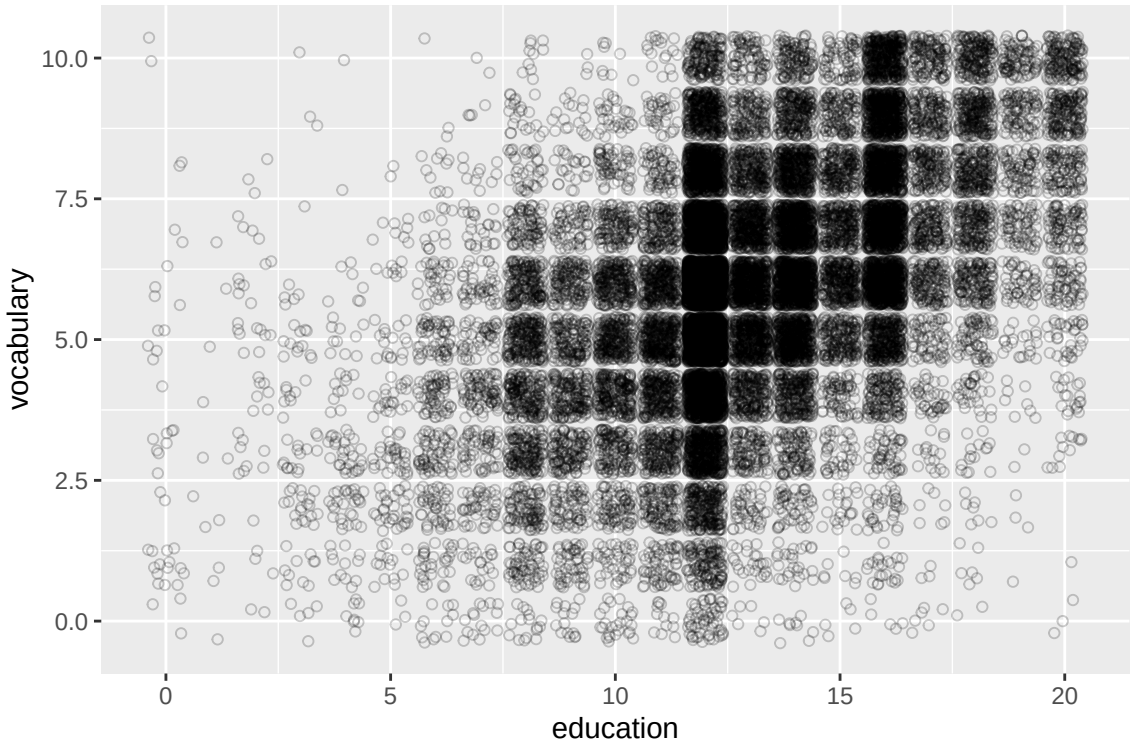
##

Applying jittering and alpha blending as solution to the overplotting

```
library(car)
```

```
## Loading required package: carData
```

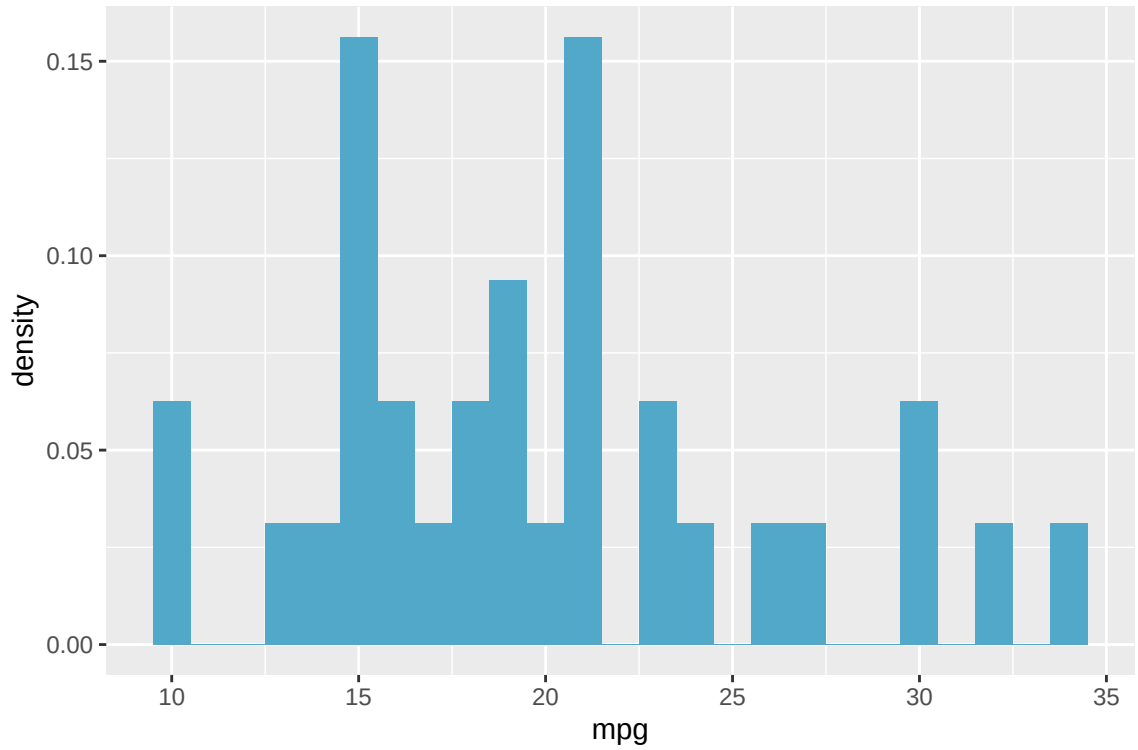
```
ggplot(Vocab, aes(education, vocabulary)) +
  # Set the shape to 1
  geom_jitter(alpha = 0.2, shape=1)
```



##

Histograms

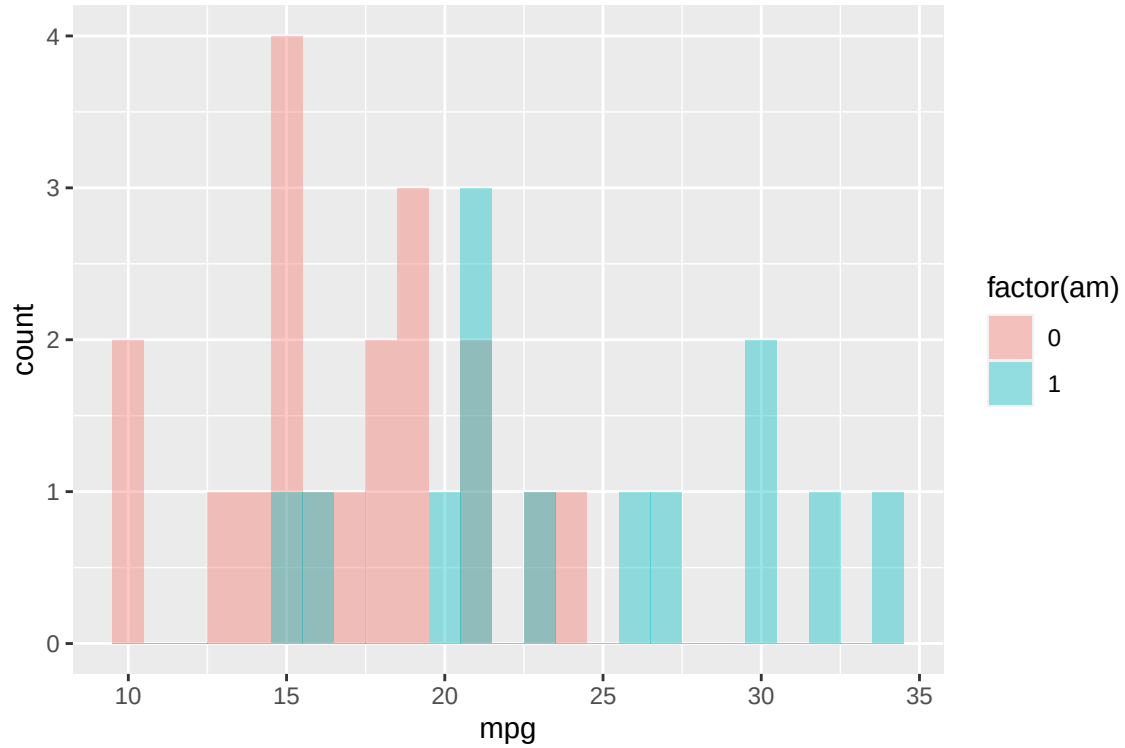
```
datacamp_light_blue <- "#51A8C9"

ggplot(mtcars, aes(mpg, ..density..)) +
  # Set the fill color to datacamp_light_blue
  geom_histogram(binwidth = 1, fill=datacamp_light_blue)
```



##

Putting bars on top of eachother

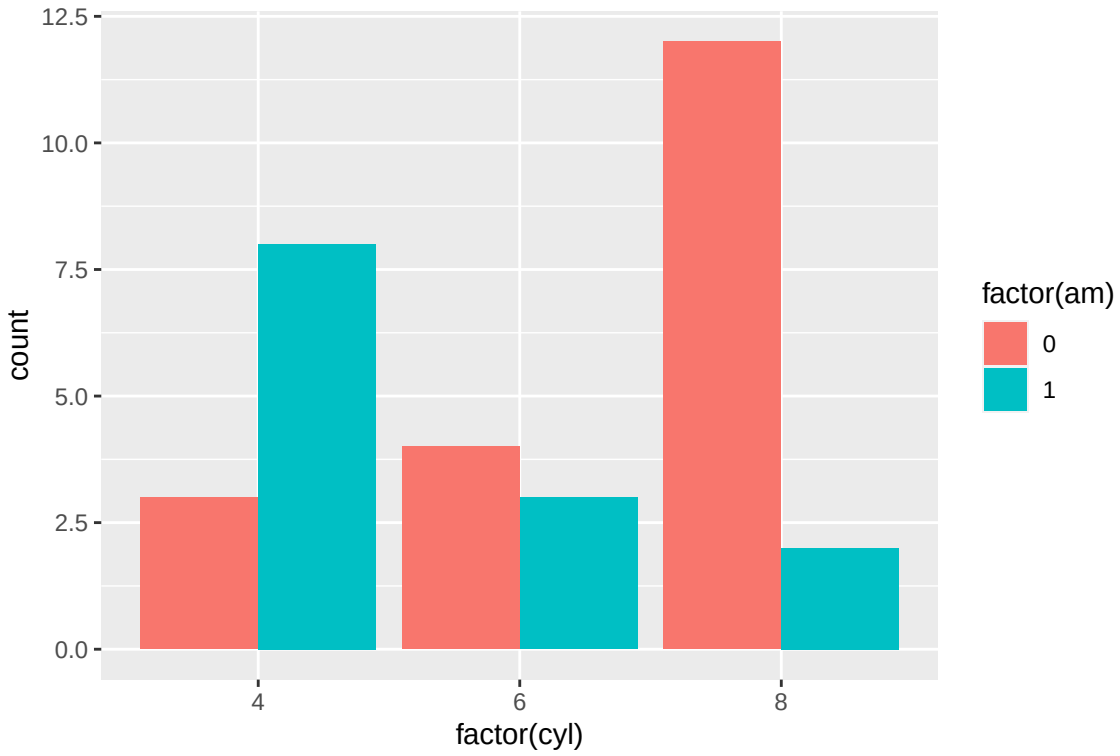
```
ggplot(mtcars, aes(mpg, fill = factor(am))) +
  # Change the position to identity, with transparency 0.4
  geom_histogram(binwidth = 1, position = "identity", alpha=0.4)
```



##

Bar plots

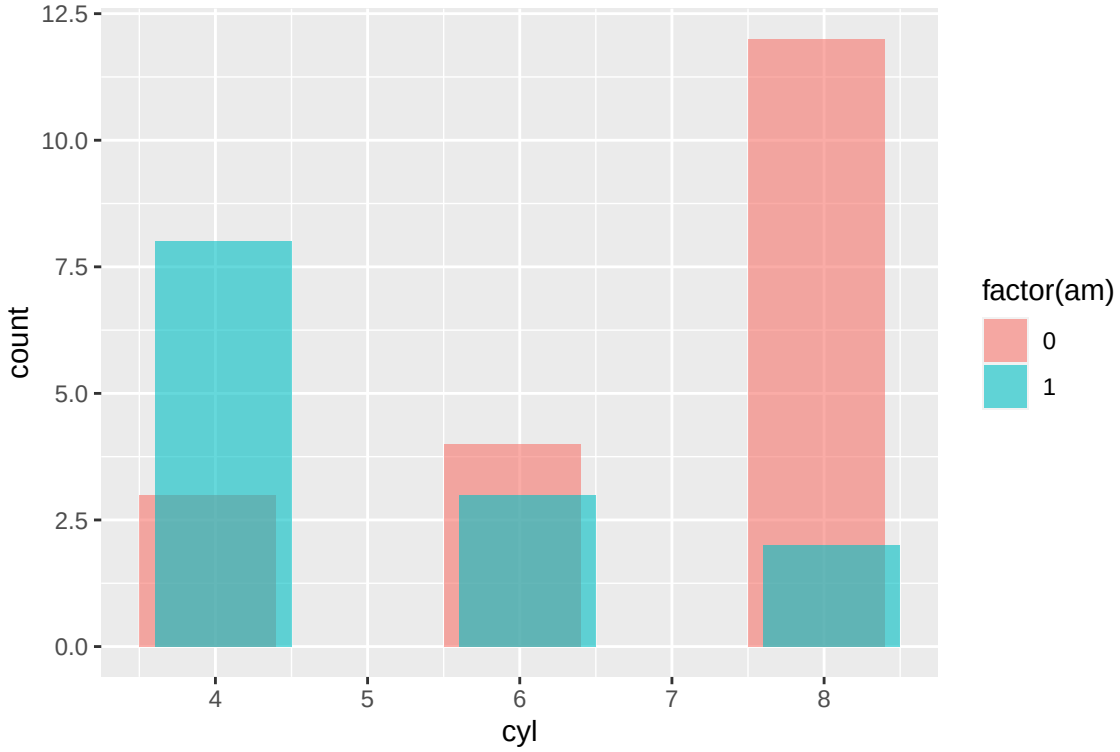
```
ggplot(mtcars, aes(factor(cyl), fill = factor(am))) +
  # Change the position to "dodge"
  geom_bar(position = "dodge")
```



##

Overlapping bar plots

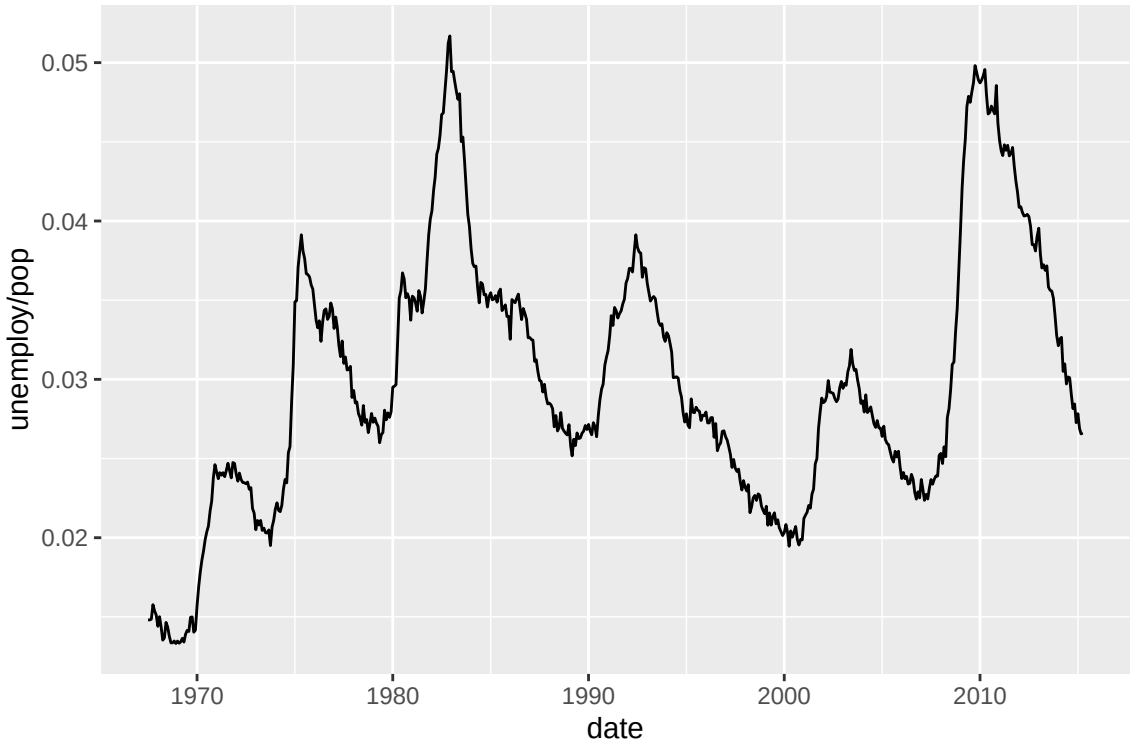
```
ggplot(mtcars, aes(cyl, fill = factor(am))) +
  # Set the transparency to 0.6
  geom_bar(position = position_dodge(width = 0.2),alpha=0.6)
```



##

Line plots

```
# Change the y-axis to the proportion of the population that is unemployed
ggplot(economics, aes(date,unemploy/pop)) +
  geom_line()
```



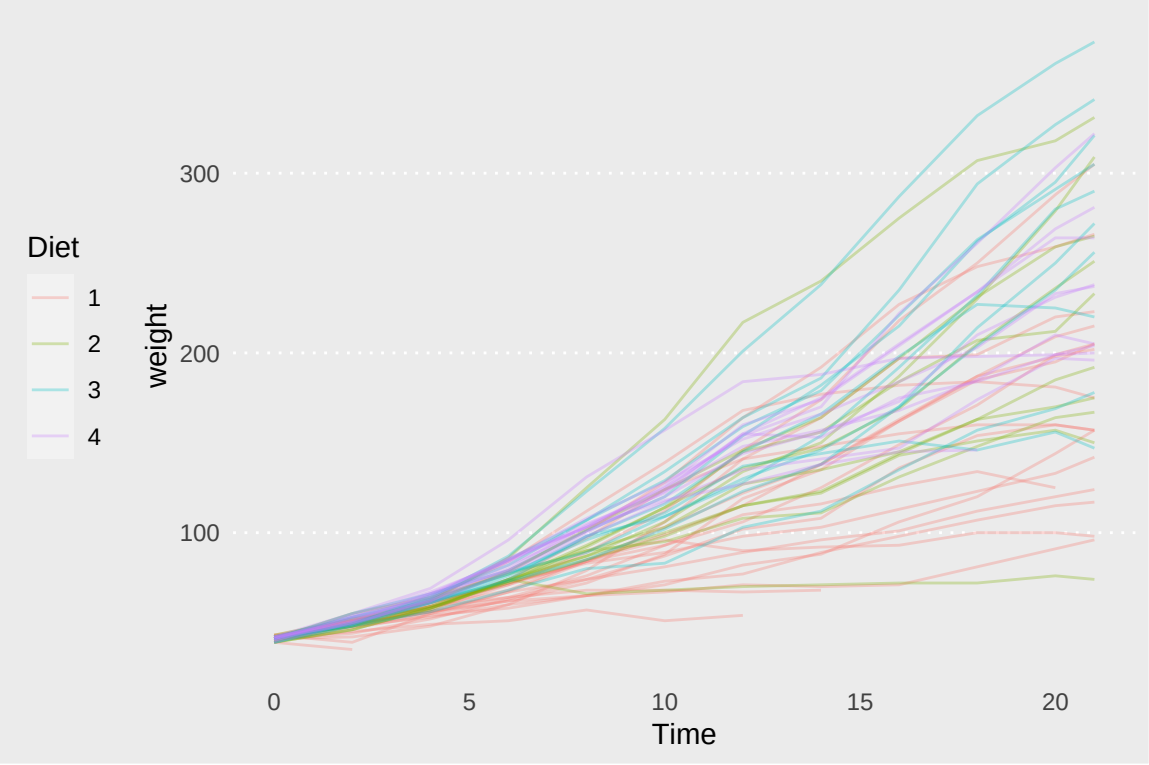
##

Themes layer

```
# Position the legend inside the plot at (0.6, 0.1)
p=ggplot(ChickWeight,
    aes(x = Time, y = weight, col = Diet)) +
  geom_line(
    aes(group = Chick), alpha = 0.3) +
  theme(legend.position = "left")
```

Modifying theme elements

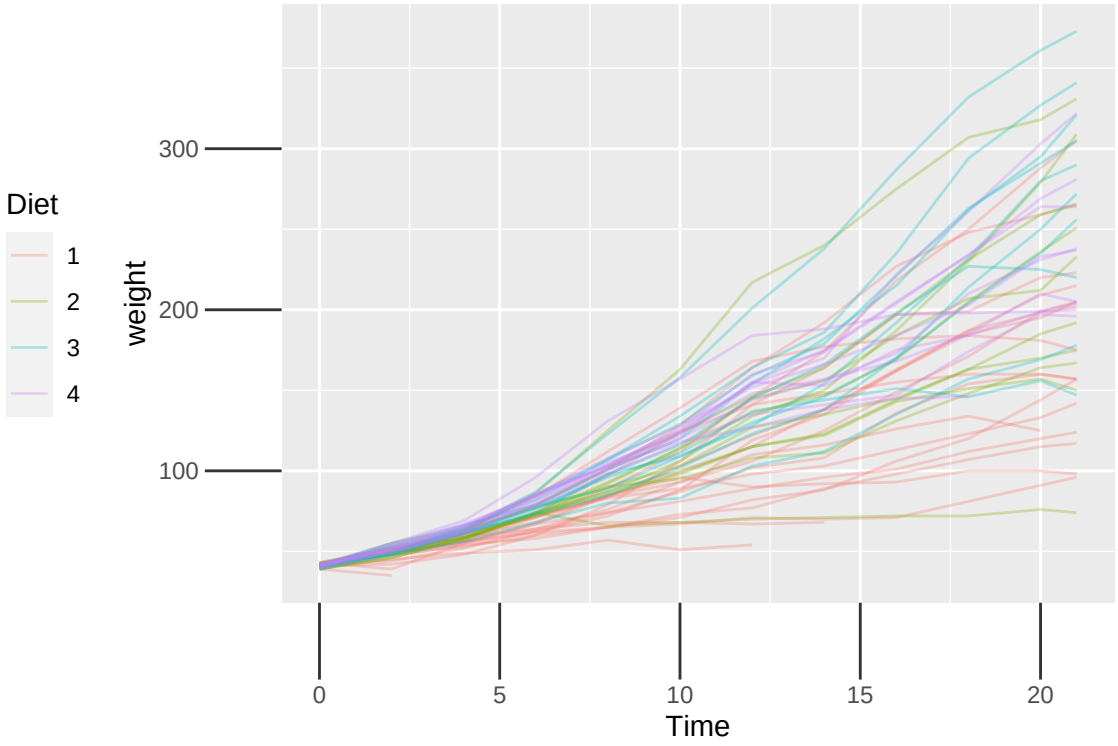
```
p +
  theme(
    rect = element_rect(fill = "grey92"),
    legend.key = element_rect(color = NA),
    axis.ticks = element_blank(),
    panel.grid = element_blank(),
    panel.grid.major.y = element_line(
      color = "white",
      size = 0.5,
      linetype = "dotted"
    ),
    # Set the axis text color to grey25
    axis.text = element_text(color = "grey25"),
    # Set the plot title font face to italic and font size to 16
   plot.title=element_text(size=16, face="italic")
  )
```



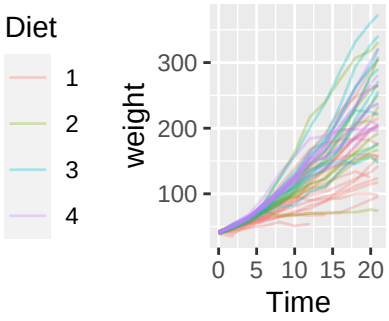
##

Modifying whitespace

```
p +
  theme(
    # Set the axis tick length to 2 lines
    axis.ticks.length = unit(2,"lines")
  )
```



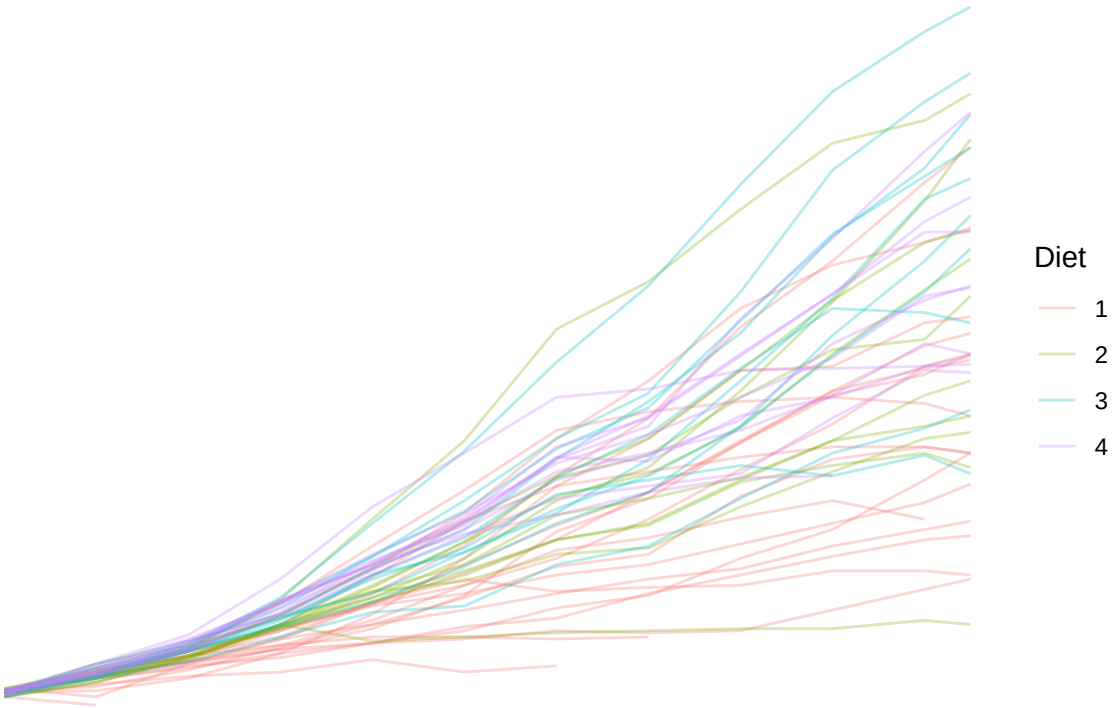

```
p +
  theme(
    # Set the plot margin to (10, 30, 50, 70) millimeters
    plot.margin = margin(10, 30, 50, 70, "mm")
  )
```



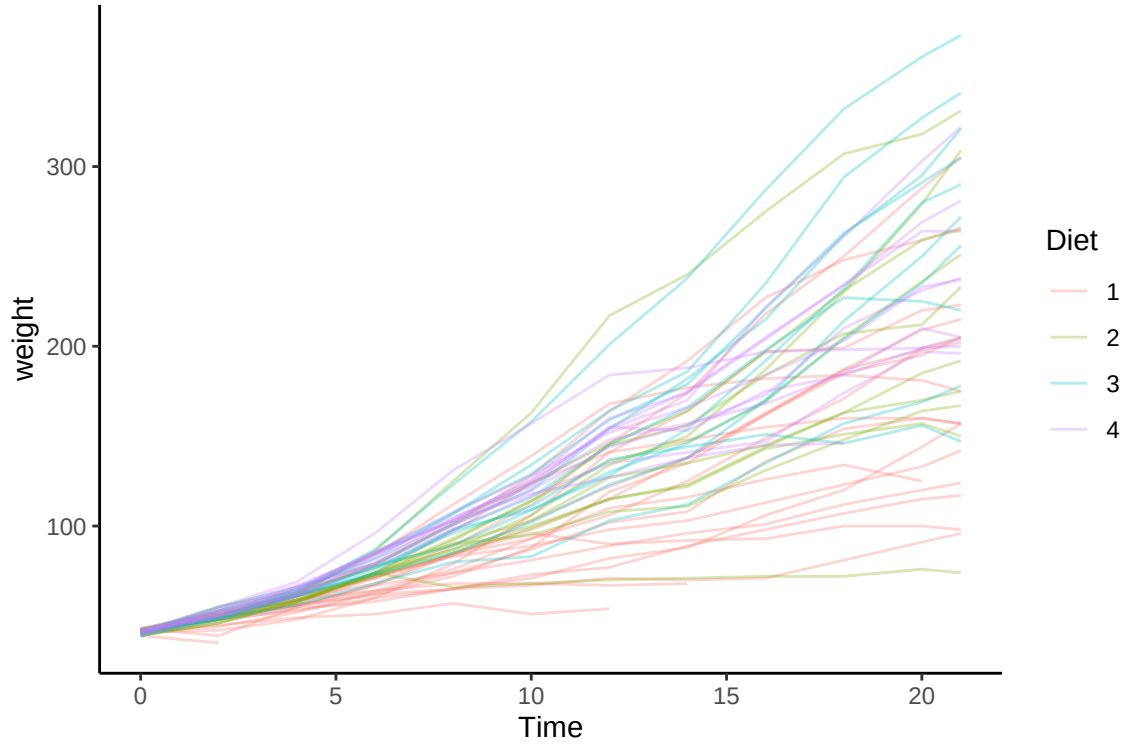
##

Built-in themes

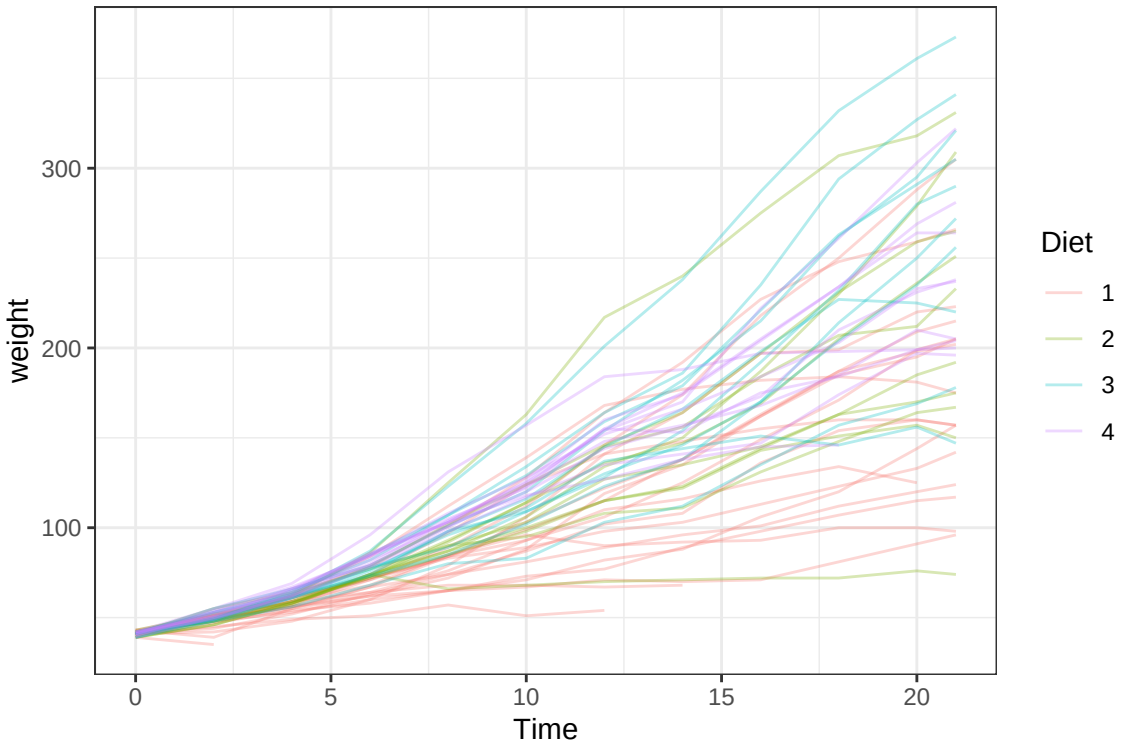
```
# Add a void theme
p +
  theme_void()
```



```
# Add a classic theme
p +
  theme_classic()
```




```
# Add a bw theme
p +
  theme_bw()
```

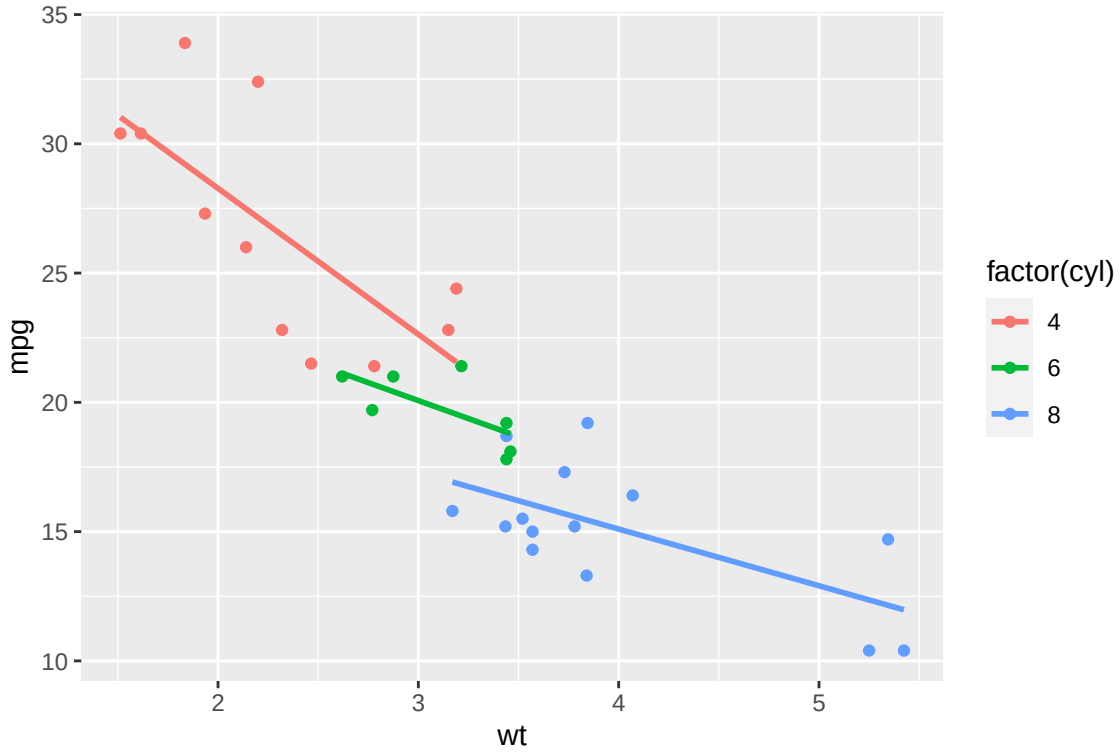


#

Intermediate Data Visualitatiton with ggplot2 ## Statistics

```
ggplot(mtcars,
    aes(x = wt, y = mpg, color = factor(cyl))) +
  # Add a point layer
  geom_point() +
  # Add a smooth lin reg stat, no ribbon
  stat_smooth(method="lm",se=FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



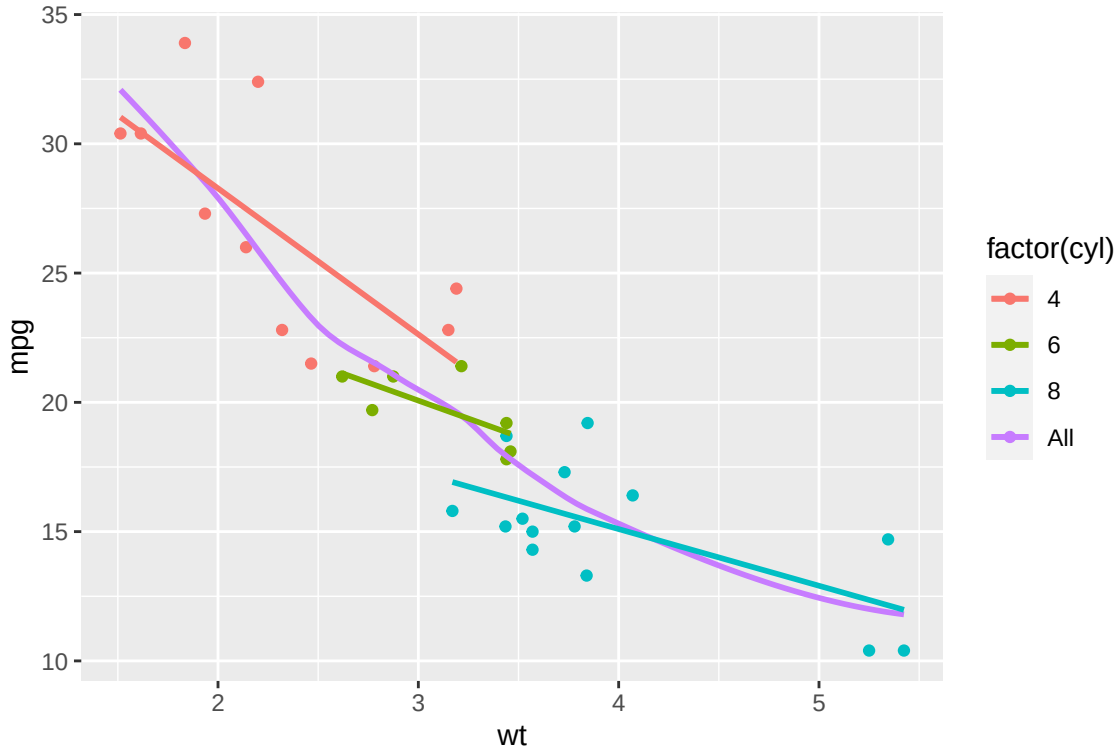
##

Amend the plot

```
# Amend the plot
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point() +
  # Map color to dummy variable "All"
  stat_smooth(se = FALSE,aes(color="All")) +
  stat_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## `geom_smooth()` using formula 'y ~ x'
```

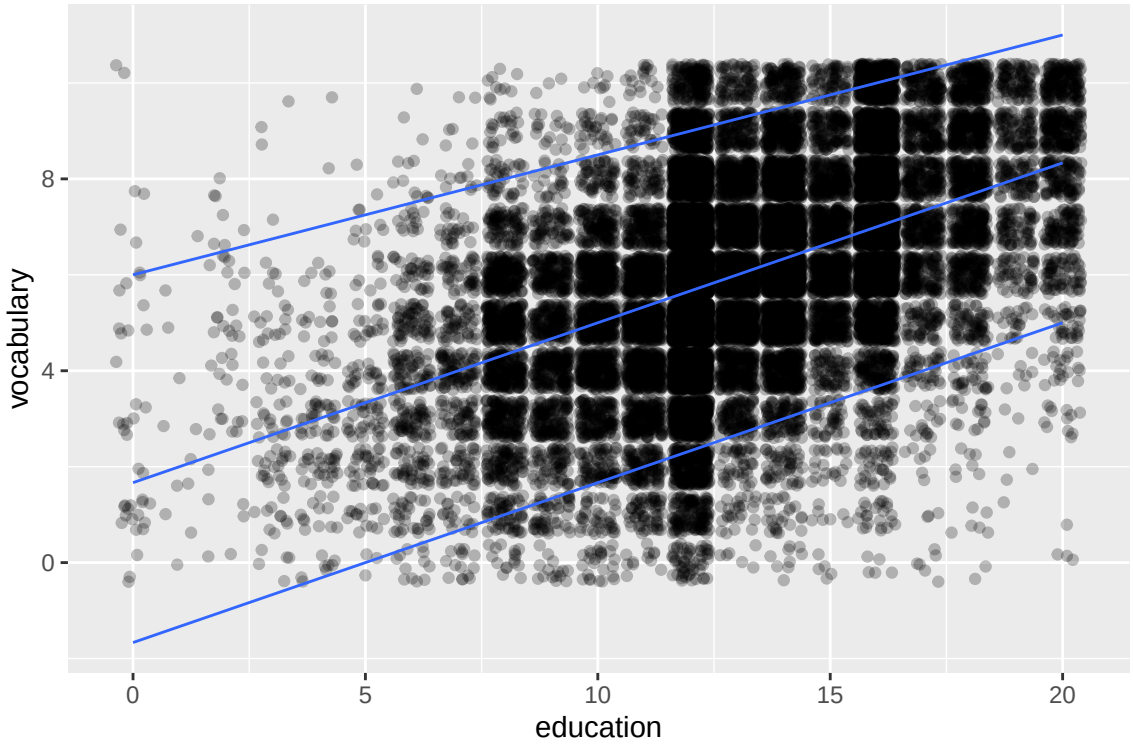


##

Quantile

```
ggplot(Vocab, aes(x = education, y = vocabulary)) +
  geom_jitter(alpha = 0.25) +
  stat_quantile(quantiles = c(0.05, 0.5, 0.95))
```

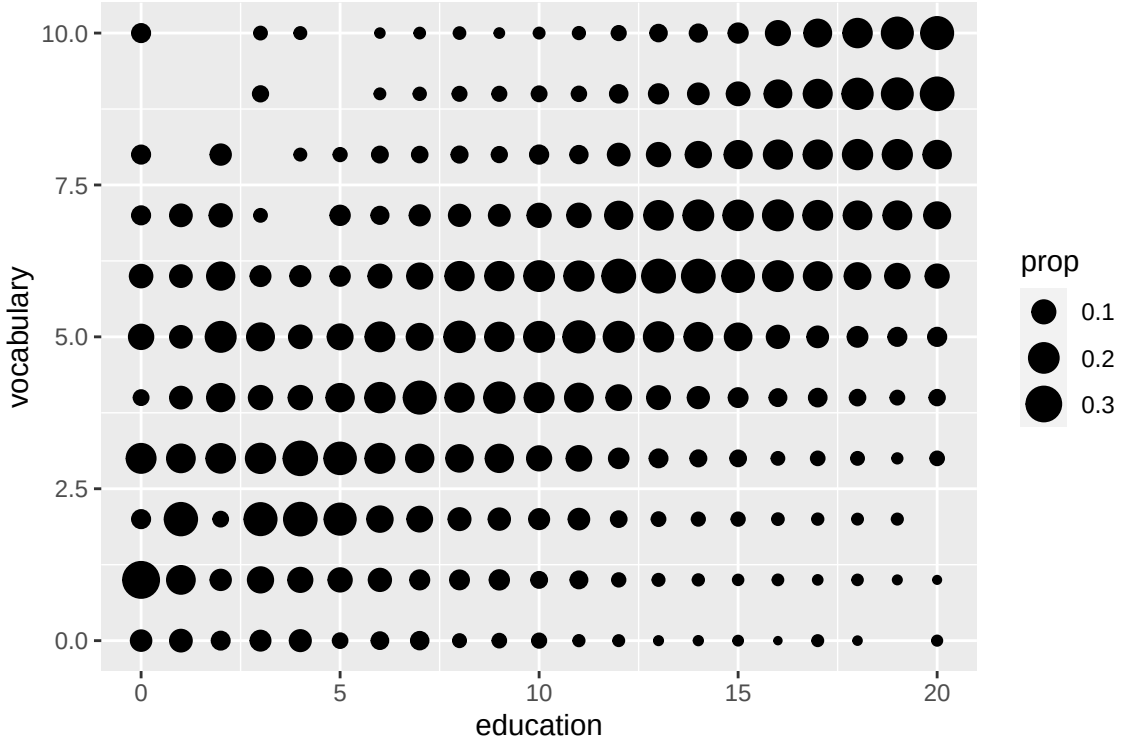
```
## Smoothing formula not specified. Using: y ~ x
```



##

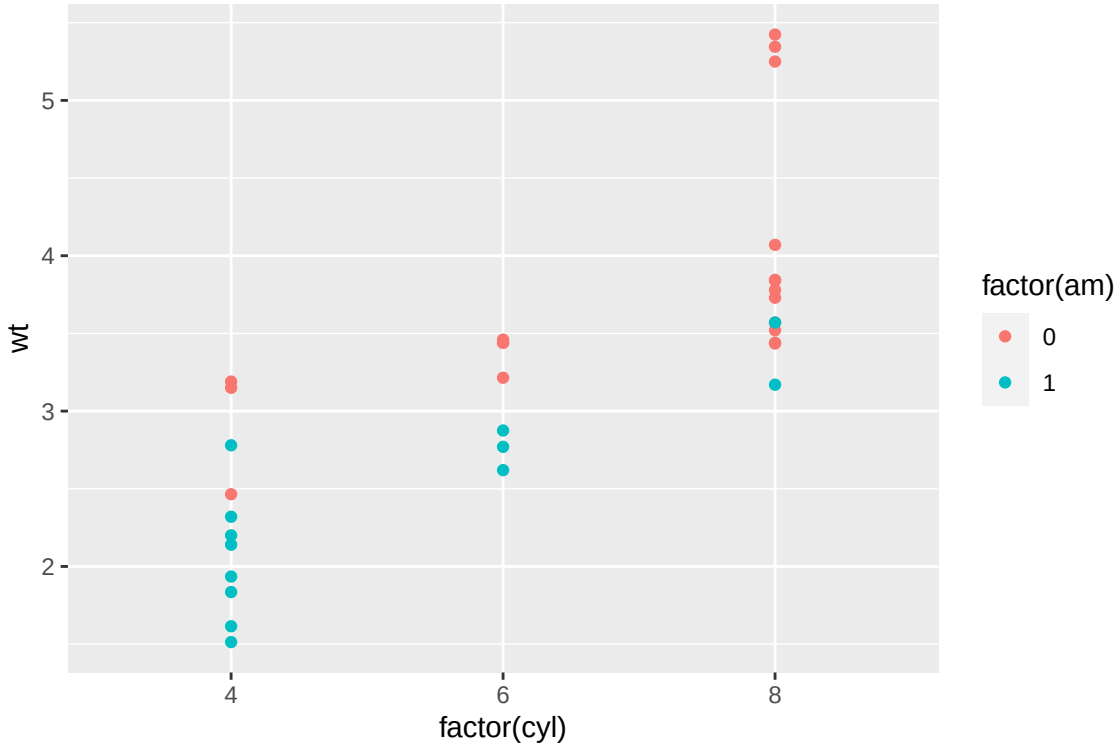
Sum

```
ggplot(Vocab, aes(x = education, y = vocabulary, group = education)) +
  stat_sum(aes(size = ..prop..))
```



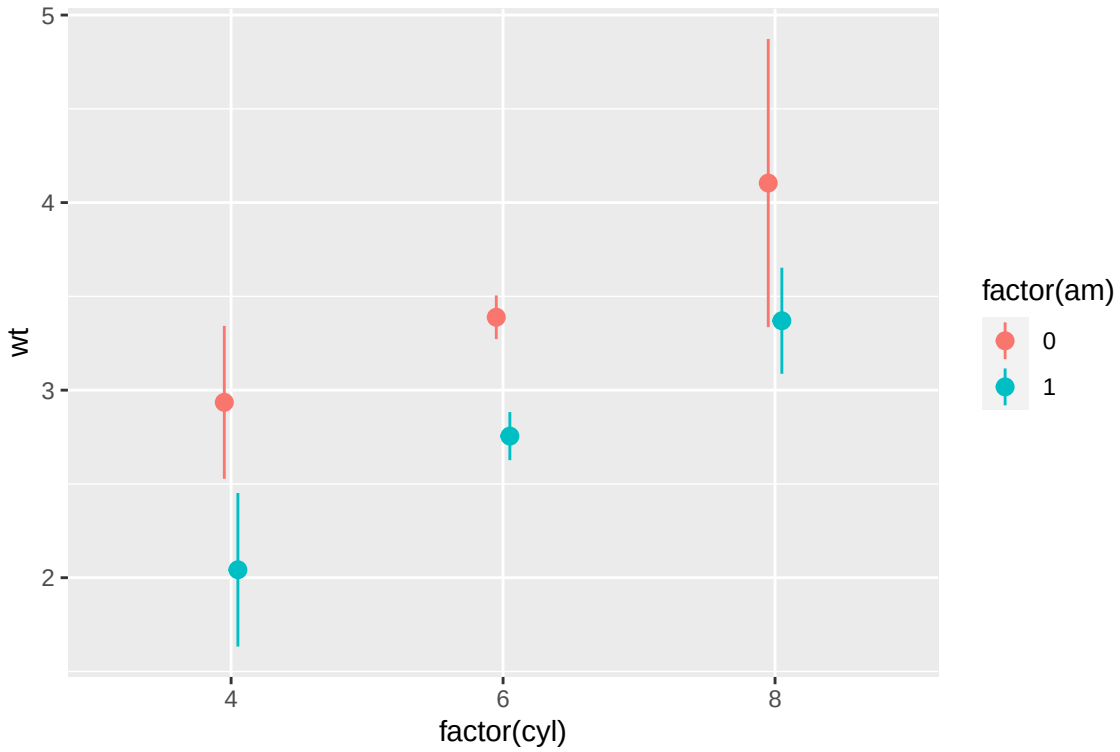
Using stats outside geoms

```
posn_j <- position_jitter(width = 0.2)
posn_d <- position_dodge(width = 0.1)
posn_jd <- position_jitterdodge(jitter.width = 0.2, dodge.width = 0.1)

# Create the plot base: wt vs. fcyl, colored by fam
p_wt_vs_fcyl_by_fam_jit <- ggplot(mtcars,aes(factor(cyl),wt,color=factor(am)))

# Add a point layer
p_wt_vs_fcyl_by_fam_jit +
  geom_point()
```



Using the stat_summary function

```
p_wt_vs_fcyl_by_fam_jit +
  # Add a summary stat of std deviation limits
  stat_summary(fun.data = mean_sdl,
  fun.args = list(mult = 1),
  position = posn_d)
```

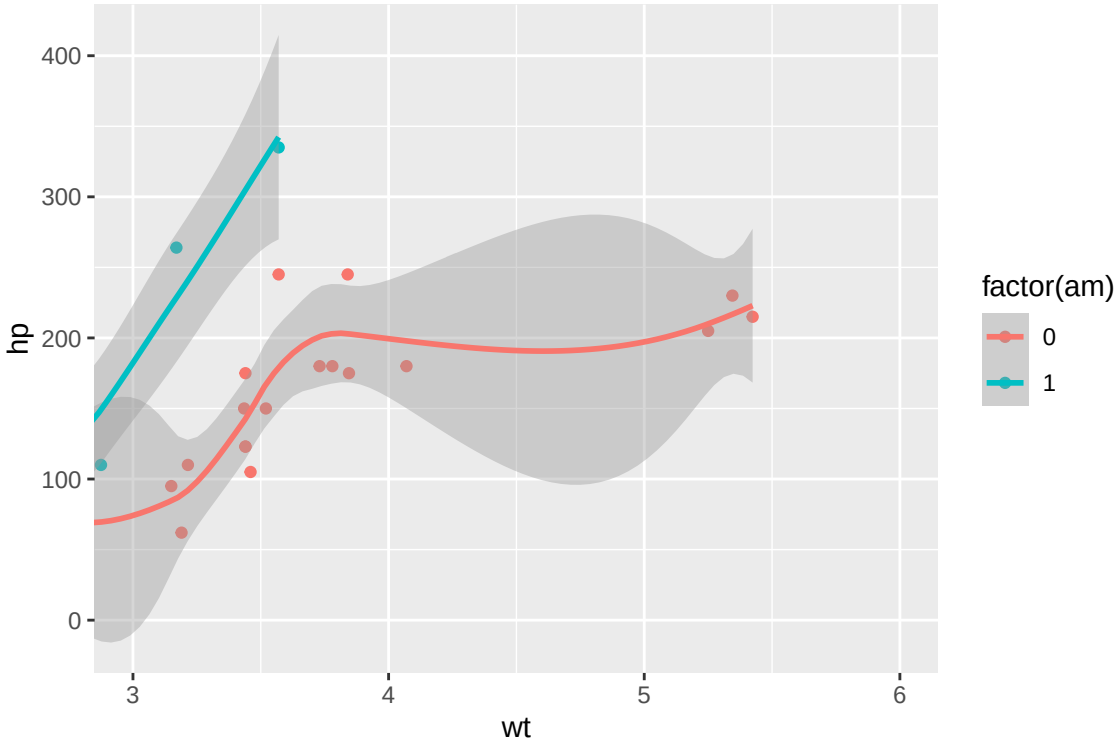


##

Coordinates layer

```
ggplot(mtcars, aes(x = wt, y = hp, color = factor(am))) +
  geom_point() +
  geom_smooth() +
  # Add Cartesian coordinates with x limits from 3 to 6
  coord_cartesian(xlim = c(3,6))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

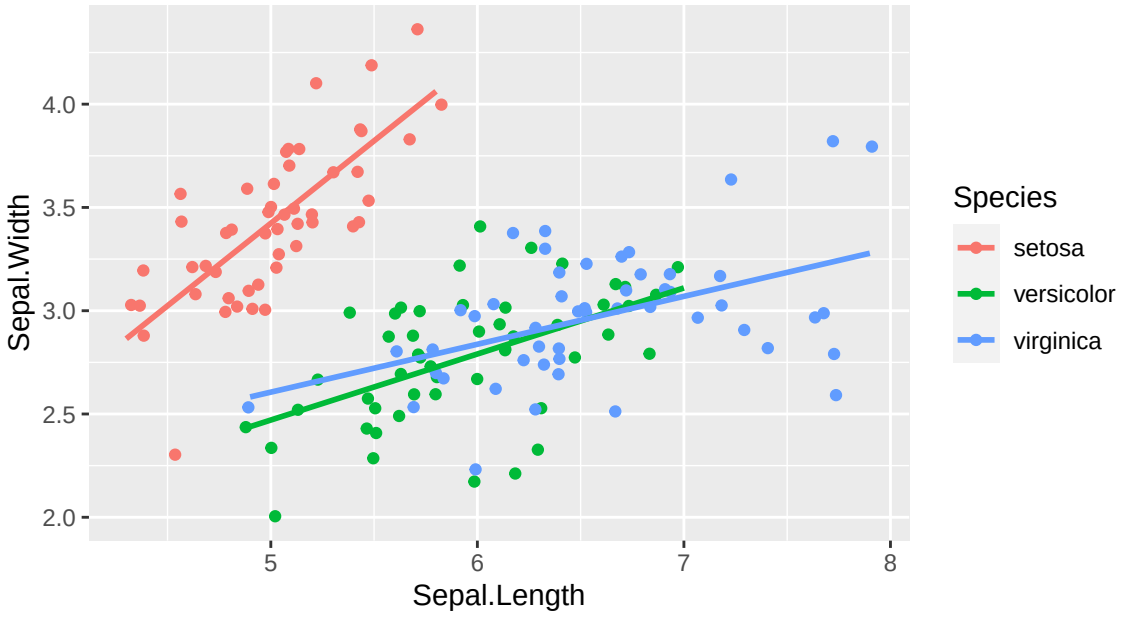


##

Setting aspect ratio

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_jitter() +
  geom_smooth(method = "lm", se = FALSE) +
  # Fix the coordinate ratio
  coord_equal()
```

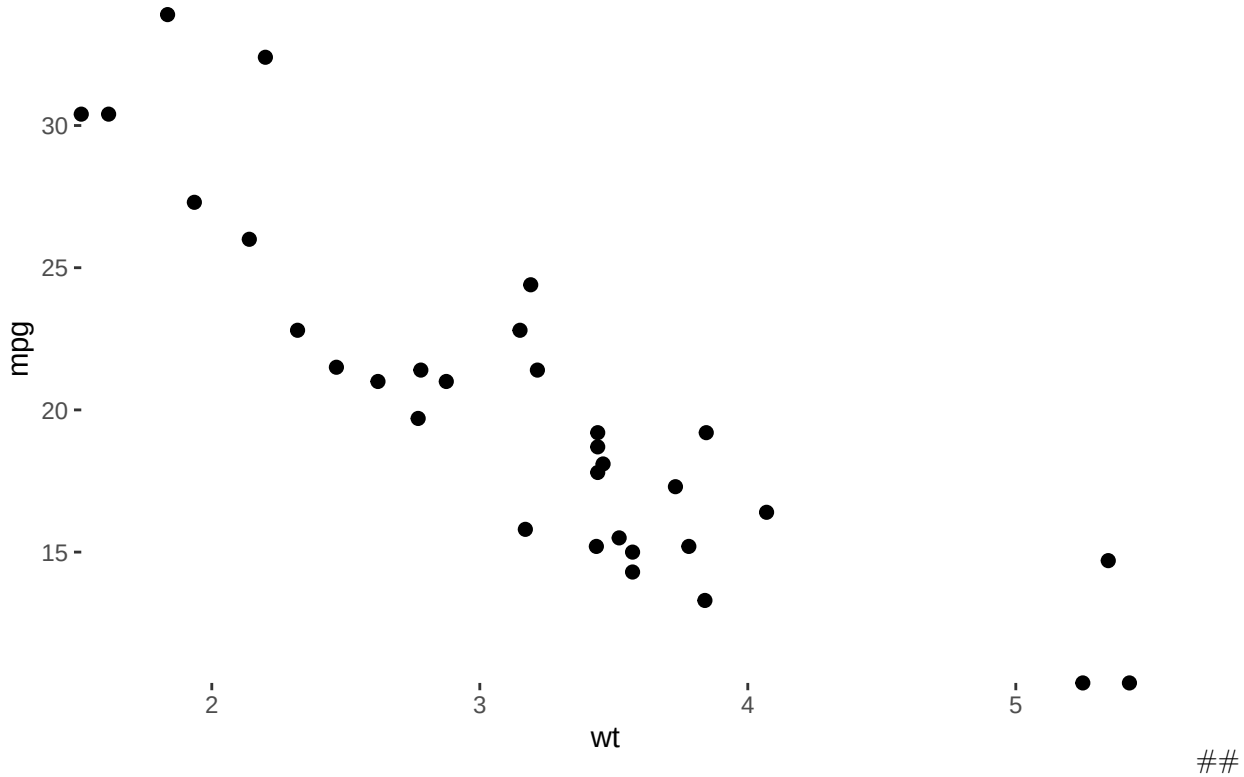
```
## `geom_smooth()` using formula 'y ~ x'
```



##

Expand and clip

```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(size = 2) +
  # Turn clipping off
  coord_cartesian(expand = 0, clip="off") +
  theme_classic() +
  # Remove axis lines
  theme(axis.line=element_blank())
```

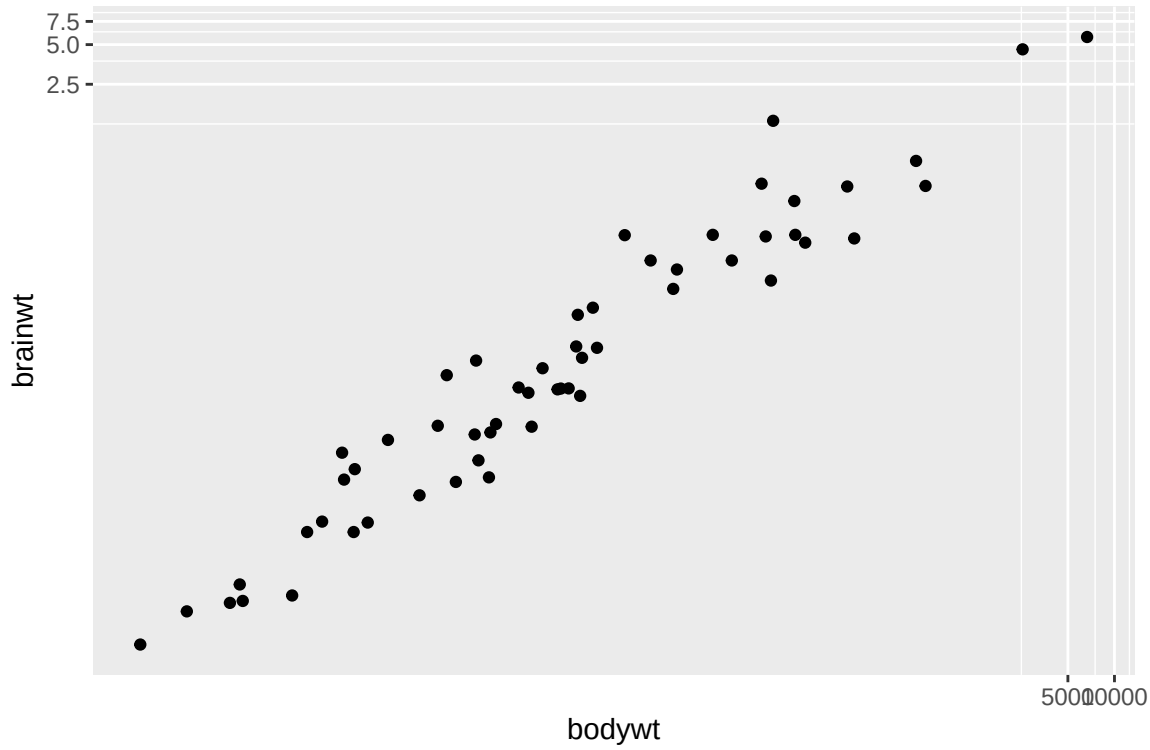


##

Log-transforming scales

```
# Perform a log10 coordinate system transformation
ggplot(msleep, aes(bodywt, brainwt)) +
  geom_point() +
  coord_trans(x="log10",y="log10")
```

```
## Warning: Removed 27 rows containing missing values (geom_point).
```



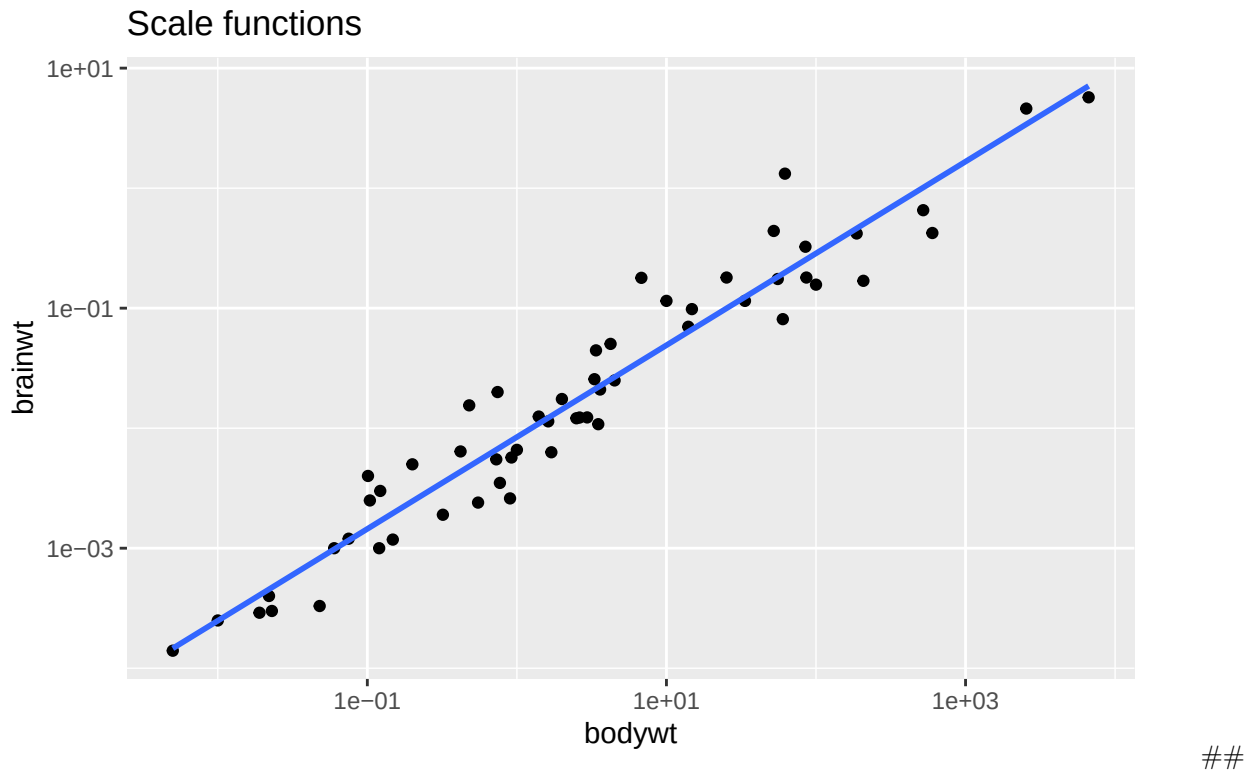
Log-transforming scales

```
# Plot with a scale_*_*() function:
ggplot(msleep, aes(bodywt, brainwt)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  # Add a log10 x scale
  scale_x_log10() +
  scale_y_log10() +
  ggtitle("Scale functions")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 27 rows containing non-finite values (stat_smooth).
```

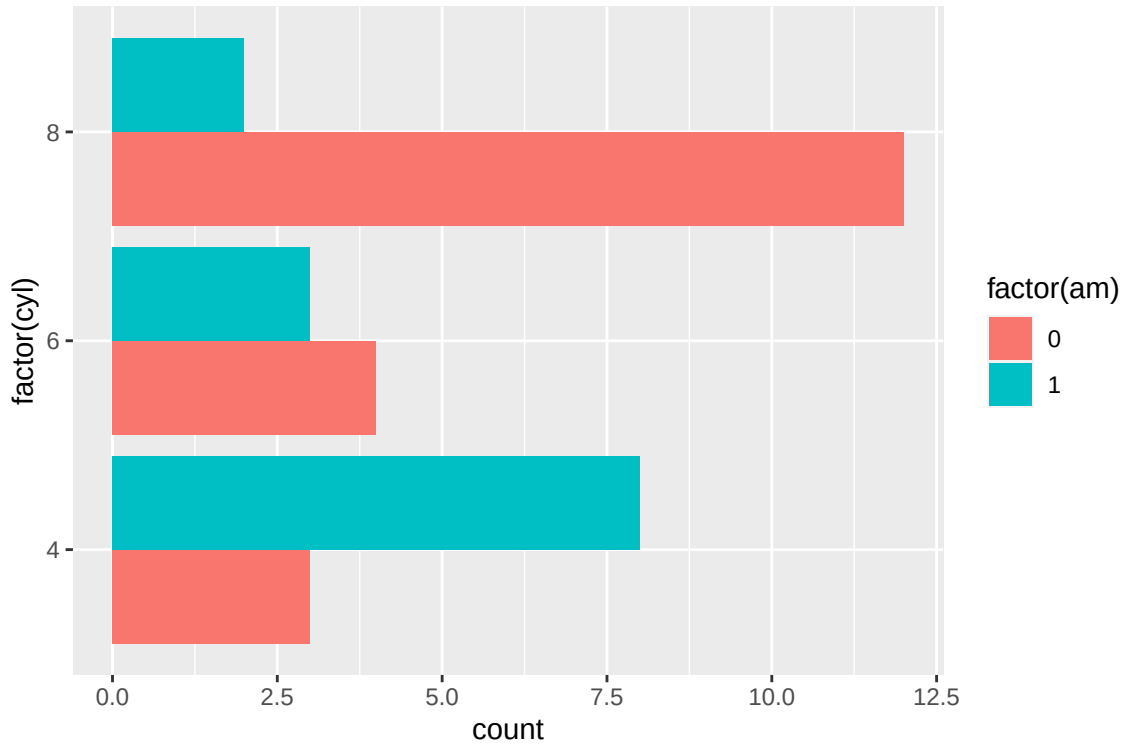
```
## Warning: Removed 27 rows containing missing values (geom_point).
```



##

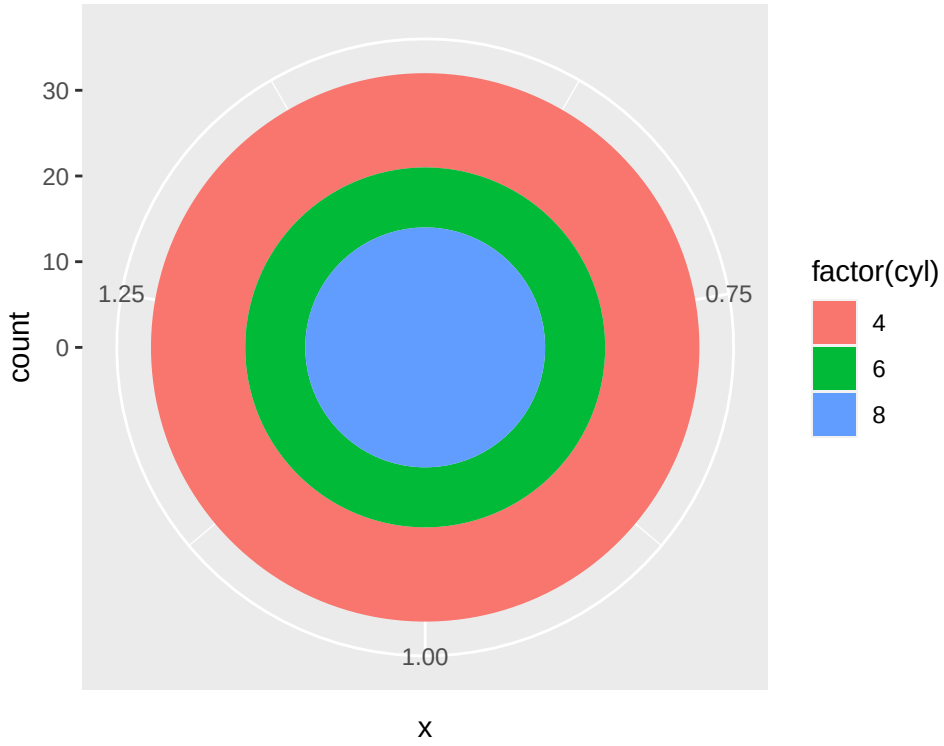
Flipped axes

```
ggplot(mtcars, aes(factor(cyl), fill = factor(am))) +
  # Set a dodge width of 0.5 for partially overlapping bars
  geom_bar(position = "dodge") +
  coord_flip()
```



Pie chart - Polar coordinates

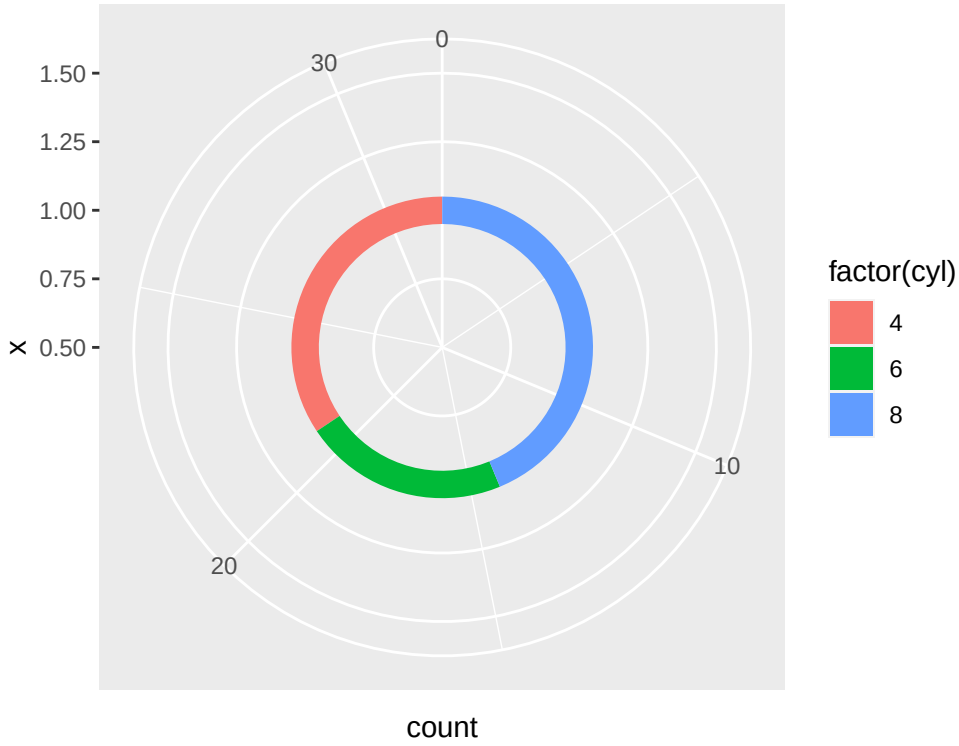
```
ggplot(mtcars, aes(x = 1, fill = factor(cyl))) +
  geom_bar() +
  # Add a polar coordinate system
  coord_polar()
```



#

Make some adjustment

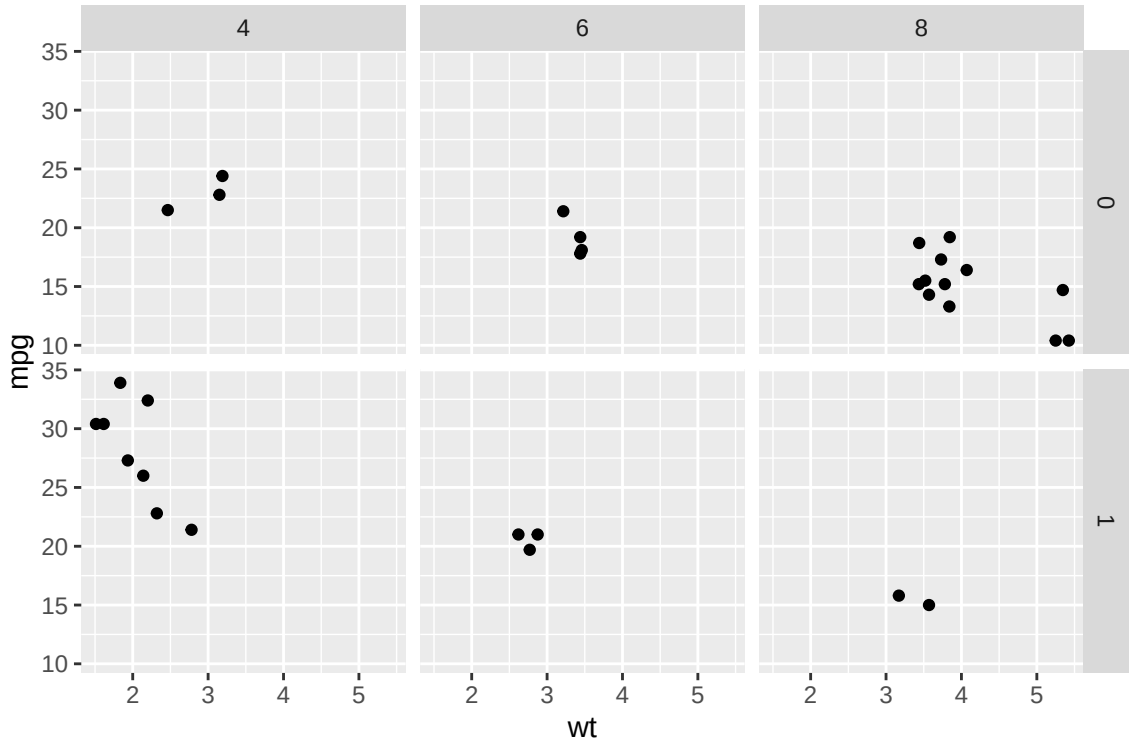
```
ggplot(mtcars, aes(x = 1, fill = factor(cyl))) +
  # Reduce the bar width to 0.1
  geom_bar(width=0.1) +
  coord_polar(theta = "y") +
  # Add a continuous x scale from 0.5 to 1.5
  scale_x_continuous(limits=c(0.5,1.5))
```



##

The facets layer

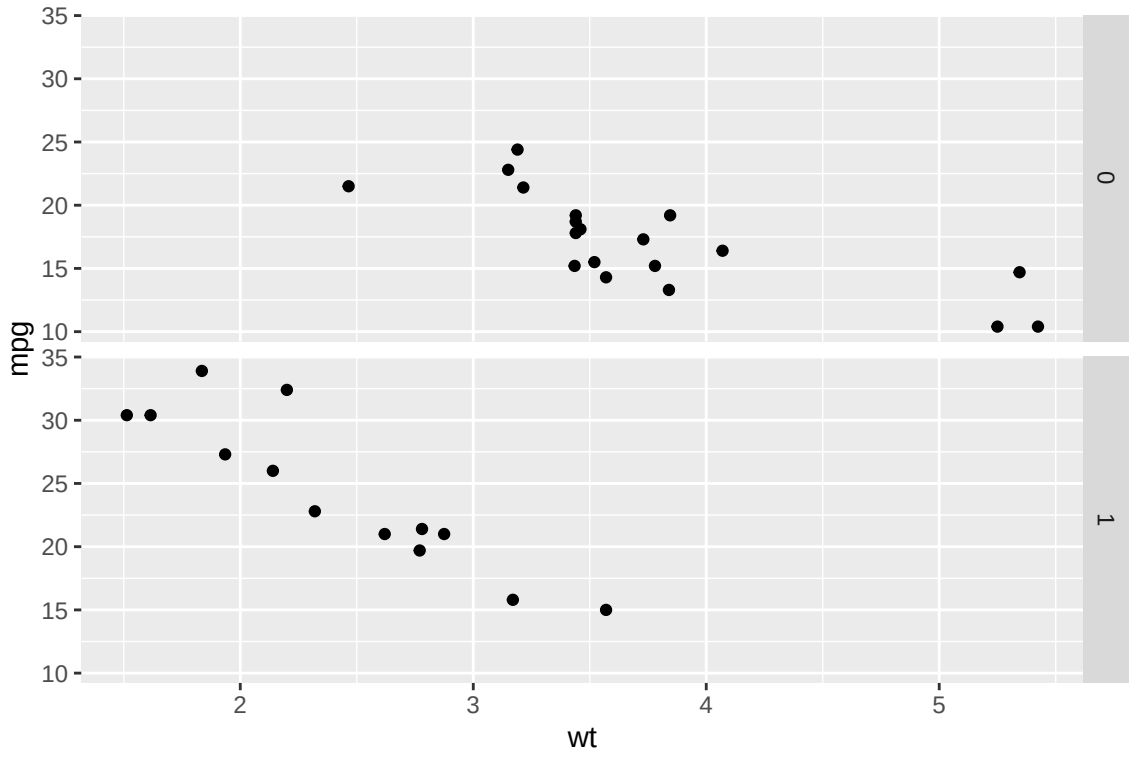
```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  # Facet rows by am and columns by cyl
   facet_grid(rows=vars(am),cols=vars(cyl))
```



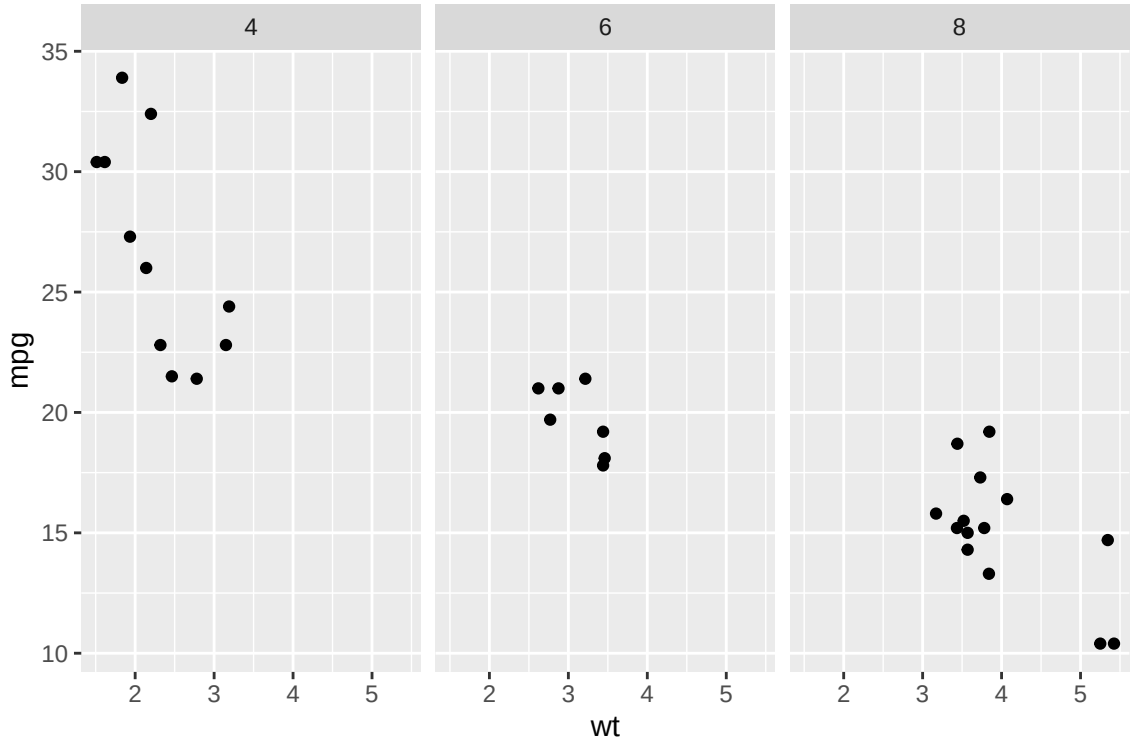
##

Facet formula formulation

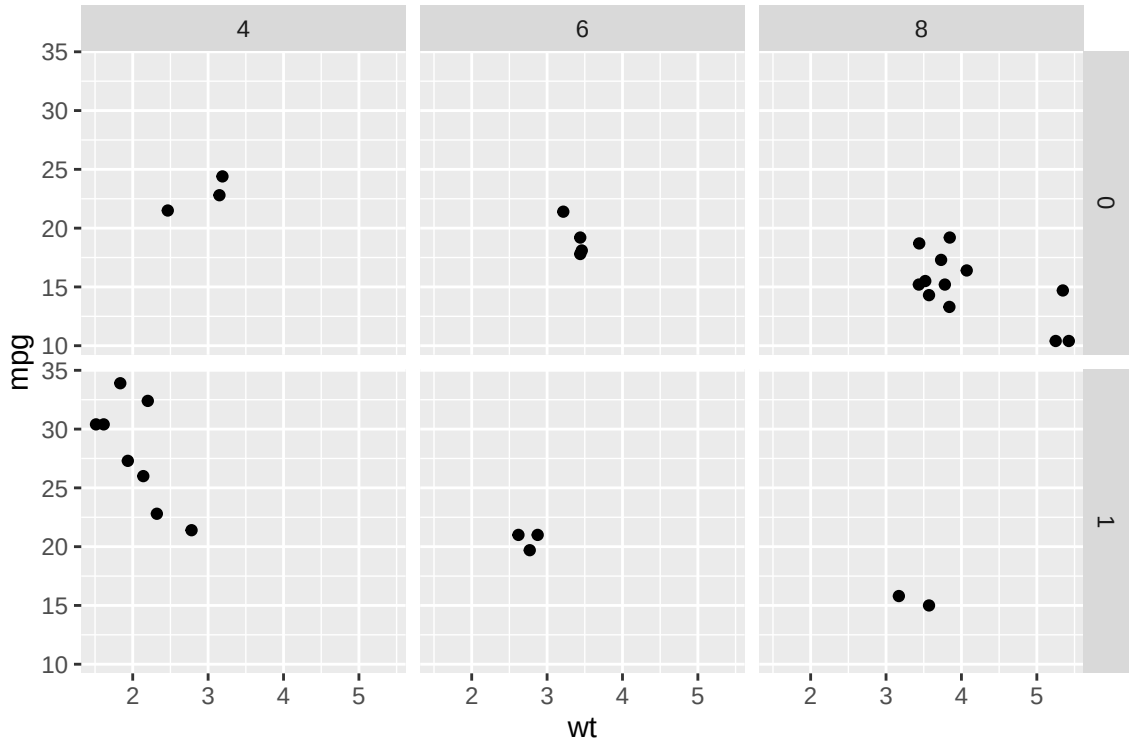
```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  # Facet rows by am using formula notation
  facet_grid(am ~ .)
```




```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  # Facet columns by cyl using formula notation
  facet_grid(. ~ cyl)
```

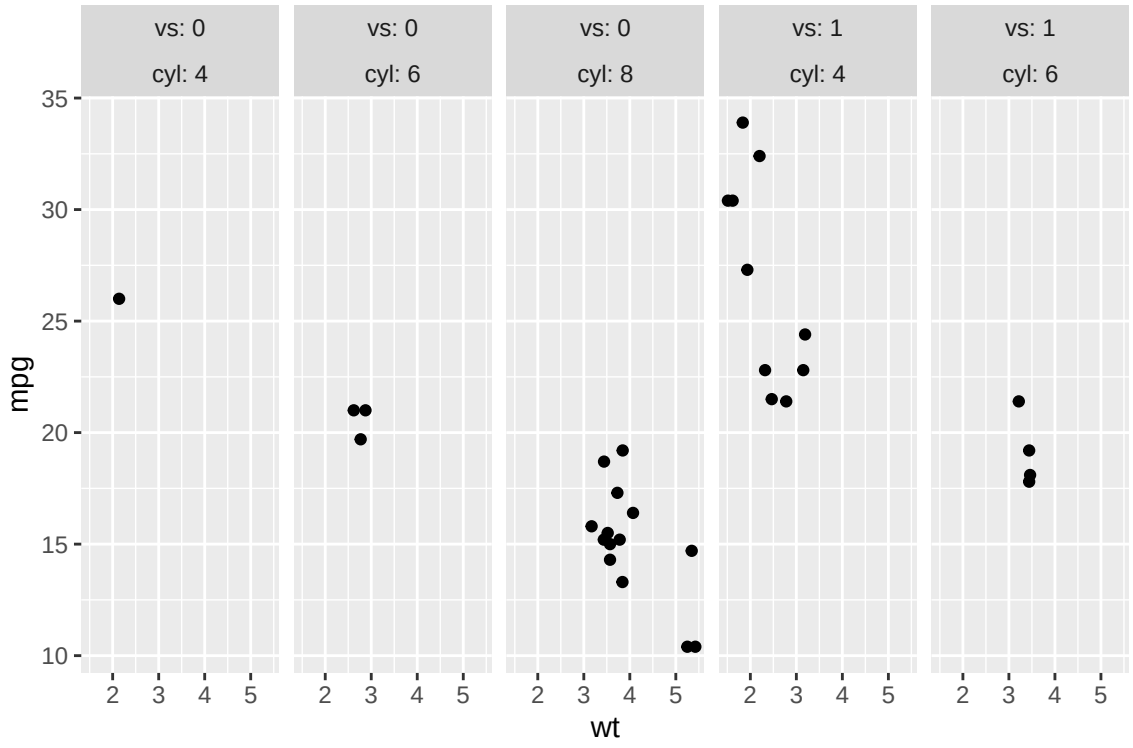


```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  # Facet rows by am and columns by cyl using formula notation
  facet_grid(am ~ cyl)
```



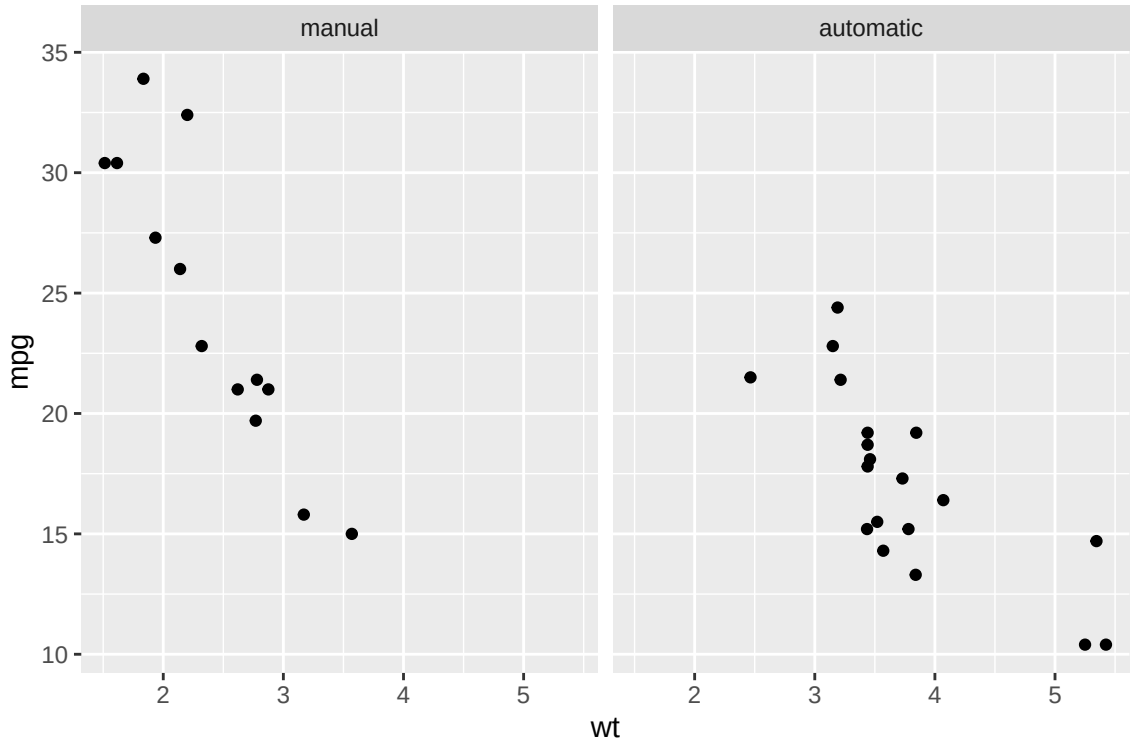
Labeling facets

```
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  # Two variables
  facet_grid(cols = vars(vs, cyl), labeller = label_context)
```



Labeling facets

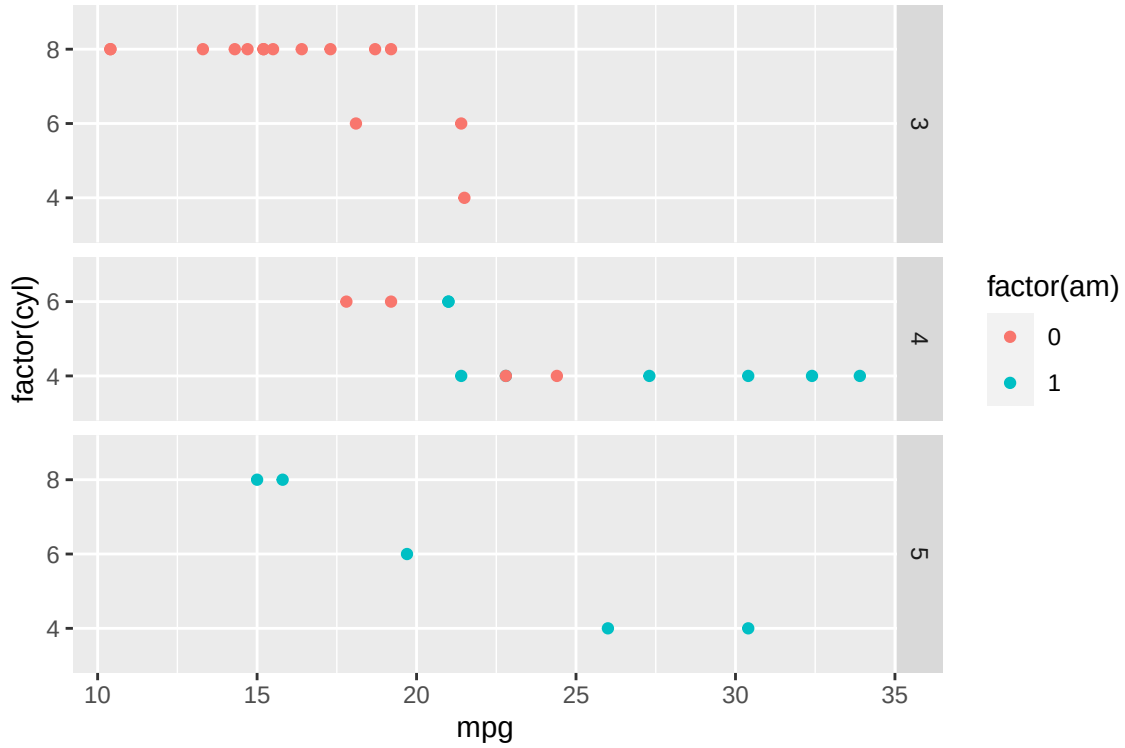
```
# Make factor, set proper labels explictly, and
# manually set the label order
mtcars$fam <- factor(mtcars$am,
                     levels = c(1, 0),
                     labels = c("manual", "automatic"))

# View again
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  facet_grid(cols = vars(fam))
```



##

Scaling and spacing

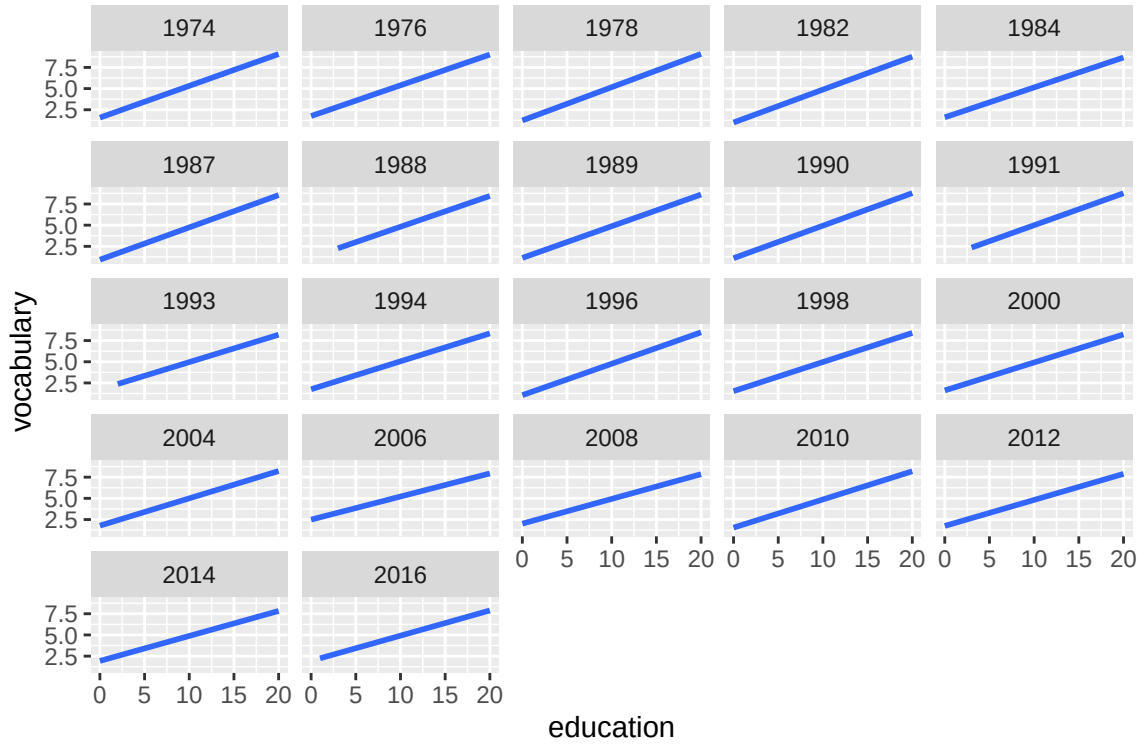
```
ggplot(mtcars, aes(x = mpg, y = factor(cyl), color = factor(am))) +
  geom_point() +
  # Free the y scales and space
  facet_grid(rows = vars(gear),space="free_y",scales="free_y")
```



##Facet wrapping for many levels

```
ggplot(Vocab, aes(x = education, y = vocabulary)) +
  stat_smooth(method = "lm", se = FALSE) +
  # Create facets, wrapping by year, using vars()
  facet_wrap(vars(year))
```

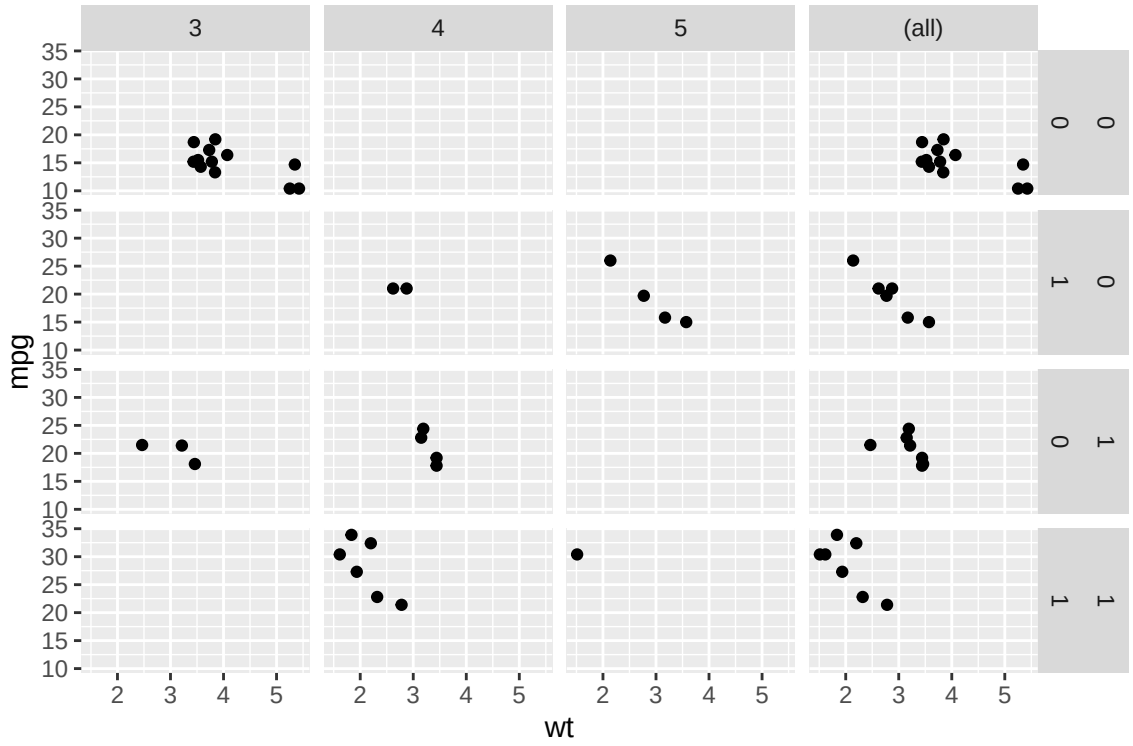
```
## `geom_smooth()` using formula 'y ~ x'
```



##

Margin plots

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  # Update the facets to only show margins on gear and fvs
  facet_grid(rows = vars(factor(vs), factor(am)), cols = vars(gear), margins = c("gear","fvs"))
```

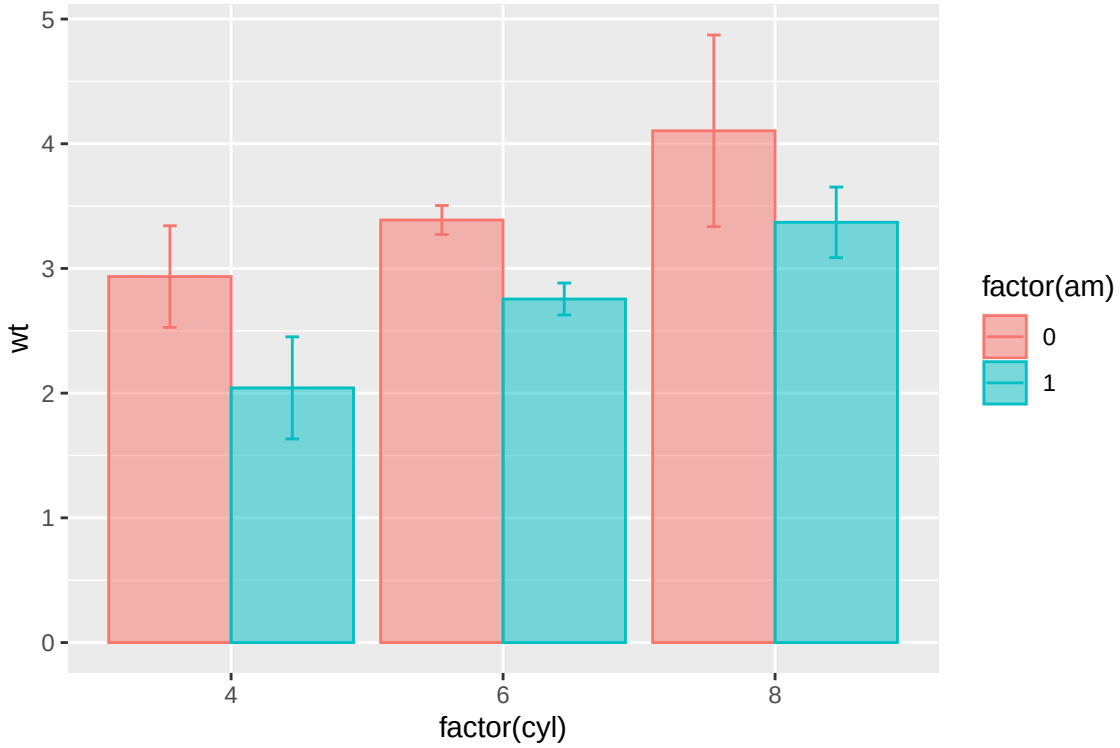


##

Best practices - Bar plots: dynamite plots

```
# Define a dodge position object with width 0.9
posn_d <- position_dodge(width = 0.9)

# For each summary stat, update the position to posn_d
ggplot(mtcars, aes(x = factor(cyl), y = wt, color = factor(am), fill = factor(am))) +
  stat_summary(fun.y = mean, geom = "bar", position = posn_d, alpha = 0.5) +
  stat_summary(fun.data = mean_sdl, fun.args = list(mult = 1), width = 0.1, position = posn_d, geom = "
```

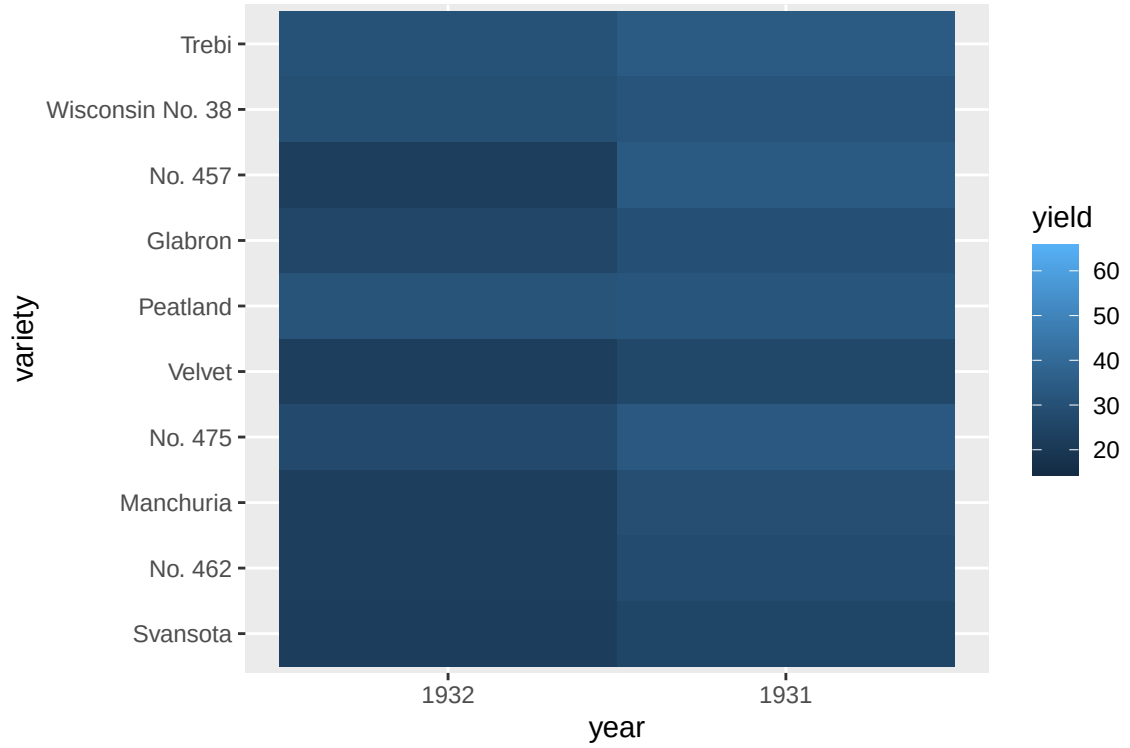
```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



##

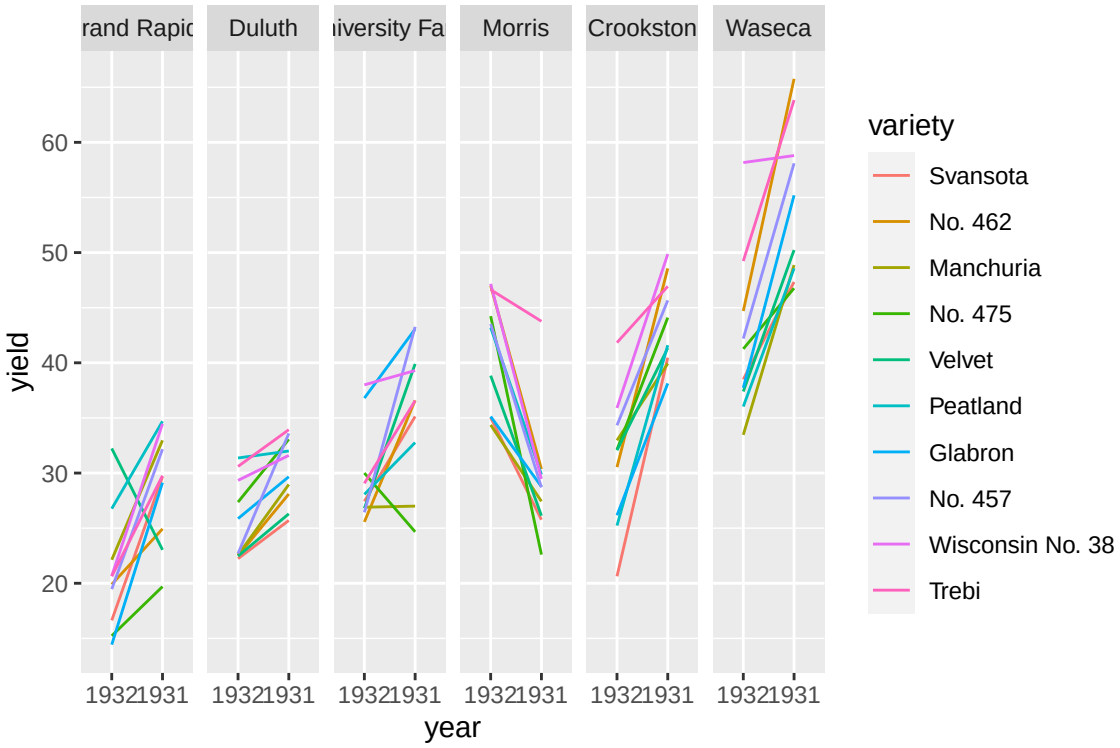
Heatmaps

```
library(lattice)
ggplot(barley,aes(year,variety,fill=yield)) +
  # Add a tile geom
  geom_tile()
```



##Heatmap alternatives

```
# Using barley, plot yield vs. year, colored and grouped by variety
ggplot(barley,aes(year,yield,group=variety,color=variety)) +
  # Add a line layer
  geom_line() +
  # Facet, wrapping by site, with 1 row
  facet_wrap( ~ site, nrow = 1)
```


rand Rapic
Duluth
iversity Fa
Morris
Crookston
Waseca
yield
60
50
40
30
20
1932
1931
year
variety
Svansota
No. 462
Manchuria
No. 475
Velvet
Peatland
Glabron
No. 457
Wisconsin No. 38
Trebi