Fachhochschule Südwestfalen
University of Applied Sciences

Programmierung R

Applications - Graphics

May 24, SS 2022 || Hannah Behrens

Wir geben Impulse

Today's schedule

Creating graphics in R

- with `base`
- with `ggplot2` (Wickham 2016)
- outlook: interactive plots with `plotly` (Sievert 2020)

a mix of theory and exercises in between

Some simple plots in `base` R

Your turn

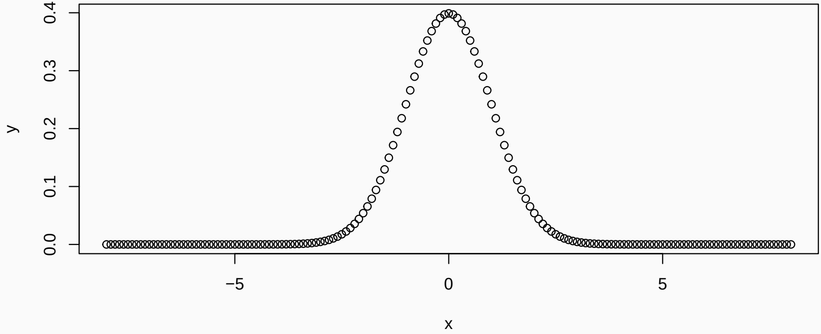
1. Make yourself familiar with the function `plot()`.
2. The variables `x` and `y` are defined as follows:

```r
x <- seq(from = -8, to = 8, by = 0.1)
y <- dnorm(x = x)
```

3. Plot `y` vs. `x`.
4. What did you plot? Hint: Type `?dnorm` into your console.

Some simple plots

```
plot(x = x, y = y)
```



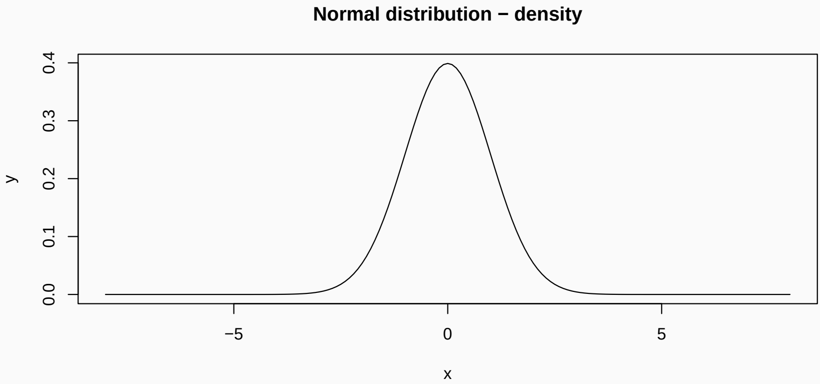
Some simple plots

Your turn

5. Add a suitable title to your plot.
6. How can you change the type of your plot in order to plot a line instead of points?

Some simple plots

```r
plot(x = x, y = y, main = "Normal distribution - density", type = "l")
```



The normal distribution

Your turn

Look at your plot and at the description of `dnorm()`.

What is the mean and standard deviation of your normal distribution?

- What will happen, if you change the mean?
- What will happen, if you change the standard deviation? Try it out!

The normal distribution - changing mean and standard deviation

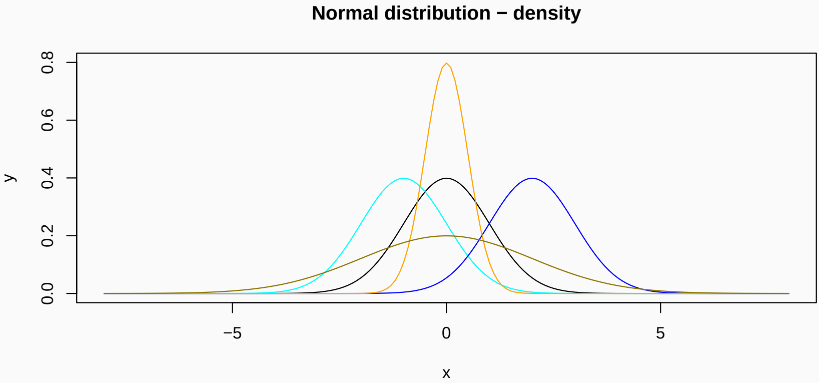
```r
y2 <- dnorm(x = x, mean = -1)
y3 <- dnorm(x = x, mean = 2)
y4<- dnorm(x = x, mean = 0, sd = 0.5)
y5 <- dnorm(x = x, mean = 0, sd = 2)
nd <- data.frame(x, y, y2, y3, y4, y5)
```

How can multiple lines be added to the plot?

The normal distribution

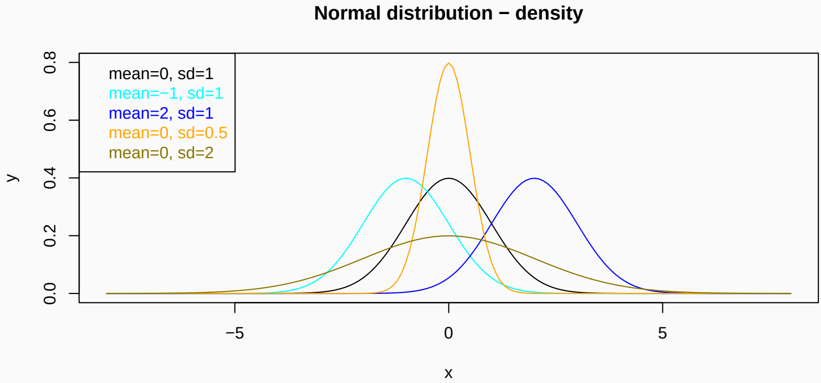
```r
plot(x = x, y = y, main = "Normal distribution - density", type = "l", ylim = c(0,0.8))
lines(x, y2, col = "cyan")
lines(x, y3, col = "blue")
lines(x, y4, col = "orange")
lines(x, y5, col = "gold4")
```

`ylim` limits the y-axis. The argument `xlim` works analogously for the x-axis.



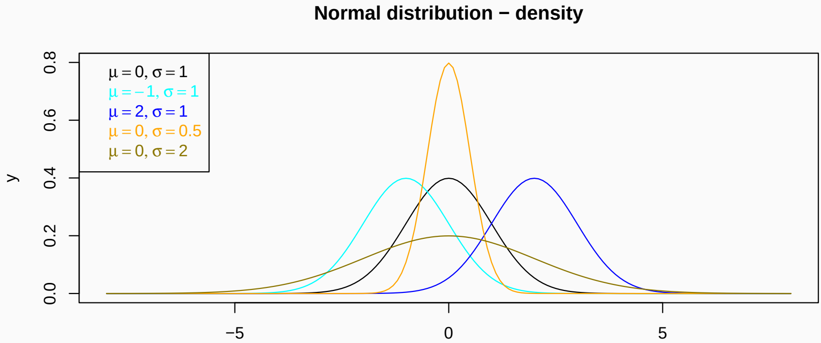
The normal distribution - Adding a legend

```r
plot(...)
... # see code from the slide before
legend("topleft", legend = c("mean=0, sd=1", "mean=-1, sd=1", "mean=2, sd=1",
"mean=0, sd=0.5", "mean=0, sd=2"),
text.col = c("black", "cyan", "blue", "orange", "gold4"))
```



Making the legend much prettier with `latex2exp` (Meschiari 2022)

```r
plot(...) ...
legend("topleft", legend=c(latex2exp::TeX("$\\mu = 0, \\sigma = 1$"),
latex2exp::TeX("$\\mu = -1, \\sigma = 1$"), latex2exp::TeX("$\\mu = 2, \\sigma = 1$"),
latex2exp::TeX("$\\mu = 0, \\sigma = 0.5$"), latex2exp::TeX("$\\mu = 0, \\sigma = 2$")),
text.col=c("black", "cyan", "blue", "orange", "gold4"))
```



R package `latex2exp` (Meschiari 2022)

Airquality data set

Your turn

Remember the data set `datasets::airquality` (R Core Team 2021).

```
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
dim(airquality)
```

```
## [1] 153   6
```

Airquality data - descriptive statistics

Your turn

You want to take a closer look at the values of the variables `Wind` and `Temp` of the `airquality` data set. You are interested in the range and distribution of the values of each variable.
Which graphical devices can you use?

Airquality data - descriptive statistics

Your turn

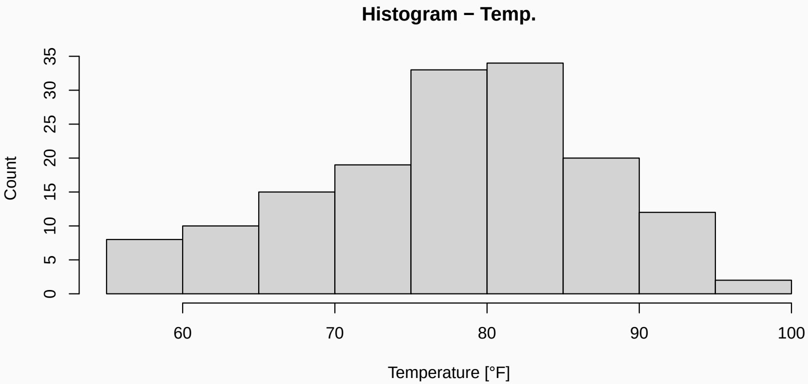
You want to take a closer look at the values of the variables `Wind` and `Temp` of the `airquality` data set. You are interested in the range and distribution of the values of each variable.

Which graphical devices can you use?

- Histogram
- Boxplot

Airquality data - Histogram of Temperature

```
h_temp <- hist(x = airquality$Temp, xlab = "Temperature [°F]", ylab = "Count",
               main = "Histogram - Temp.")
```



What is the benefit of assigning the histogram to a variable in general and here to `h_temp`?

Airquality data - Histogram of Temperature

```r
h_temp # getting information about the plotted data
```

```
## $breaks
##  [1]  55  60  65  70  75  80  85  90  95 100
##
## $counts
## [1]  8 10 15 19 33 34 20 12  2
##
## $density
## [1] 0.010458 0.013072 0.019608 0.024837 0.043137 0.044444 0.026144 0.015686
## [9] 0.002614
##
## $mids
## [1] 57.5 62.5 67.5 72.5 77.5 82.5 87.5 92.5 97.5
##
## $xname
## [1] "airquality$Temp"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

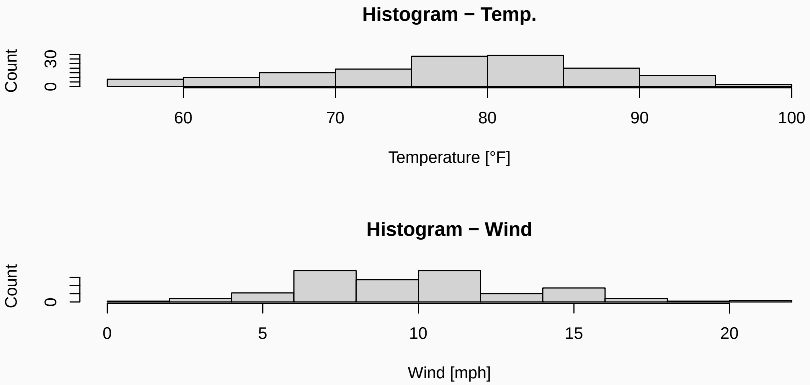
- `breaks`: the bin boundaries
- `counts`: the counts in (a,b]

- `density`: relative frequencies divided by binwidth, here: density = `h_temp$counts / sum(h_temp$counts))/5`
- `mids`: midpoints of the bins
- `equidist`: whether distances between breaks are the same

Airquality data - Histograms

```r
par(mfrow = c(2,1)) # c(r,c): c(number of rows, number of columns)
h_temp <- hist(x = airquality$Temp, xlab = "Temperature [°F]", ylab = "Count",
               main = "Histogram - Temp.")
h_wind <- hist(airquality$Wind, xlab = "Wind [mph]", ylab = "Count",
               main = "Histogram - Wind")
```

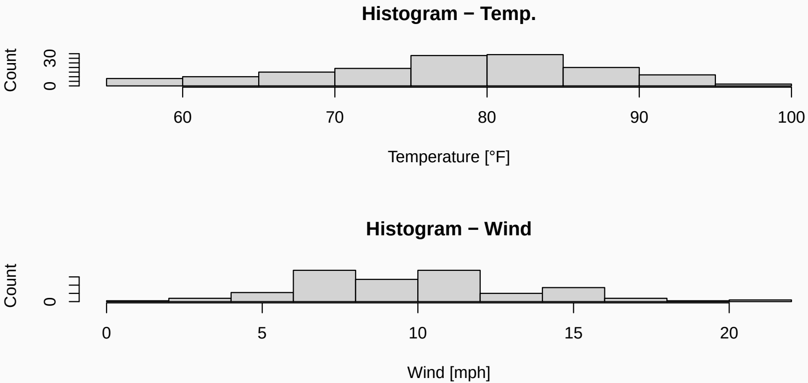
Set `main = ""` to leave out a title.



Airquality data - Histograms

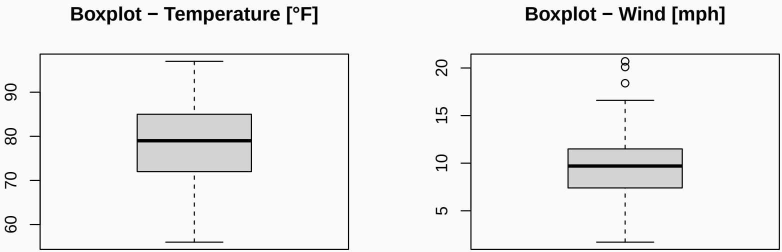
```r
par(mfcol = c(2,1)) # c(r,c): c(number of rows, number of columns)
h_temp <- hist(x = airquality$Temp, xlab = "Temperature [°F]", ylab = "Count",
               main = "Histogram - Temp.")
h_wind <- hist(airquality$Wind, xlab = "Wind [mph]", ylab = "Count",
               main = "Histogram - Wind")
```

`mfcol` is similar to `mfrow`.



Airquality data - Boxplot

```r
par(mfrow = c(1,2)) # c(r,c): c(number of rows, number of columns)
b1 <- boxplot(x = airquality$Temp, main = "Boxplot - Temperature [°F]")
b2 <- boxplot(x = airquality$Wind, main = "Boxplot - Wind [mph]")
```



What is the advantage of assigning the two boxplots to `b1` and `b2` respectively?

Airquality data - Boxplot - accessing values

```
b2
```

```
## $stats
##      [,1]
## [1,]  1.7
## [2,]  7.4
## [3,]  9.7
## [4,] 11.5
## [5,] 16.6
##
## $n
## [1] 153
##
## $conf
##        [,1]
## [1,]  9.176
## [2,] 10.224
##
## $out
## [1] 20.1 18.4 20.7
##
## $group
## [1] 1 1 1
##
## $names
## [1] ""
```

`stats`: lower whisker, $q_{0.25}$, $q_{0.5}$, $q_{0.75}$, upper whisker

`n`: number of non-NA observations

`conf`: lower and upper extremes of the notch

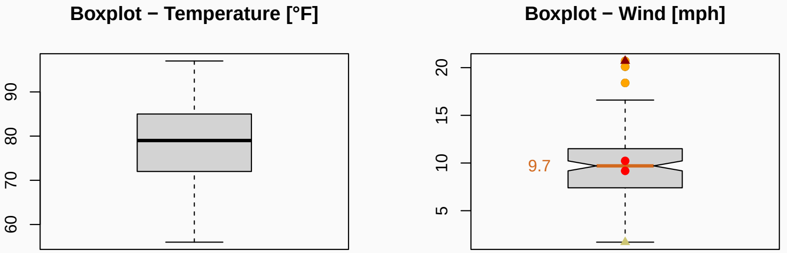
`out`: any data point *outside* the whiskers

`group`: indicating to which group the outliers belong

How far do the whiskers extend out from the box?

Airquality data - Boxplot

```r
par(mfrow = c(1,2)) # c(r,c): c(number of rows, number of columns)
b1 <- boxplot(x = airquality$Temp, main = "Boxplot - Temperature [°F]")
b2 <- boxplot(x = airquality$Wind, main = "Boxplot - Wind [mph]", medcol = "chocolate",
            notch = TRUE)
points(x = rep(1,length(b2$out)), y = b2$out, col = "orange", pch = 19)
points(x = c(1,1), y = b2$conf, col = "red", pch = 19)
points(x = 1, y = min(airquality$Wind), col = "khaki3", pch = 17)
points(x = 1, y = max(airquality$Wind), col = "darkred", pch = 17)
text(x = 0.7, y = b2$stats[3,], labels = b2$stats[3,], col = "chocolate")
```



Which values are colored in the boxplot on the right?

Colors in R

using colors in R independently of creating graphics with `base`, `ggplot2` or another R package

→ just define the corresponding argument which allows coloring text, points, lines etc. like `col` in `base` R by typing

1. the color's name e.g.

```
plot(..., col = "cyan")
```

   see a list of colors written as words which can be used in R: *colors in R as text* (Wei 2021)

2. the color's hexadecimal code, e.g. the hexadecimal code of cyan: #00FFFF

```
plot(..., col = "#00FFFF")
```

Adding information to a plot - points, lines and text

- Marking points in a plot with `points()`
- Drawing a (vertical, horizontal, ...) line in a plot with `lines()`
- Adding text in a plot with `text()`

Each of these functions needs the x and y coordinates,

- which point to mark,
- where to set a line (from x to y) or
- where to put the text.

You can *color* points, text and lines, change the *shape* of points and the *size* of points, text, lines etc. (just take a look at the corresponding functions).

Airquality data - Wind vs. Temperature

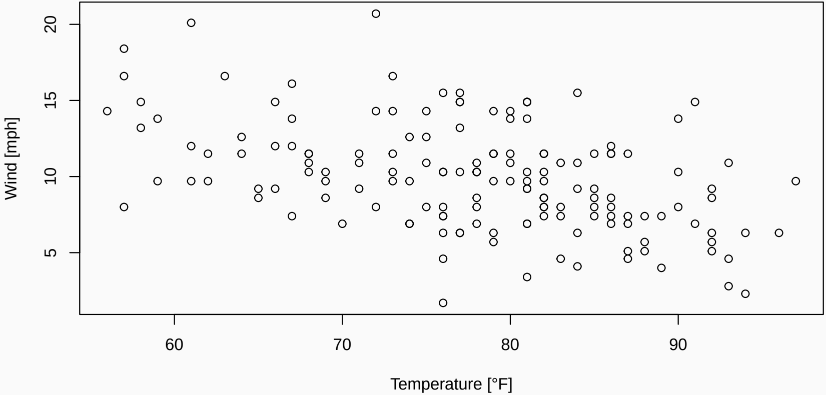
Your turn

- Plot the variable `Wind` vs. the variable `Temp`. Do not forget to label the x- and y-axis.
- What can you see in the plot?

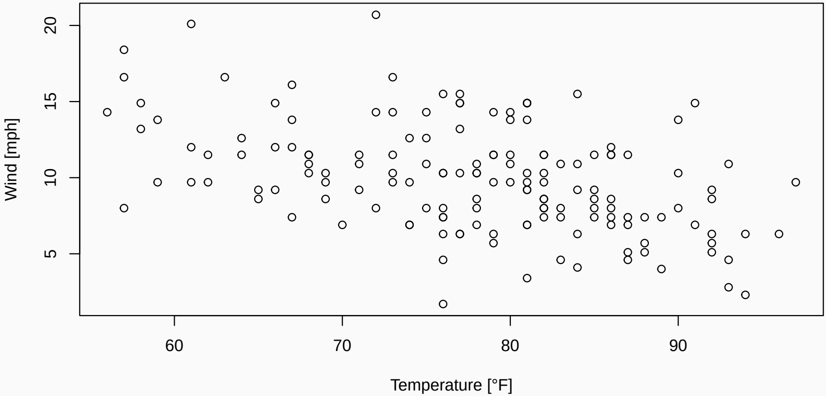
Airquality

```r
par(mar = c(4,4,2,2)) # margins
plot(x = airquality$Temp, y = airquality$Wind, xlab = "Temperature [°F]",
     ylab = "Wind [mph]")
```

For margins see https://r-graph-gallery.com/74-margin-and-oma-cheatsheet.html



Airquality

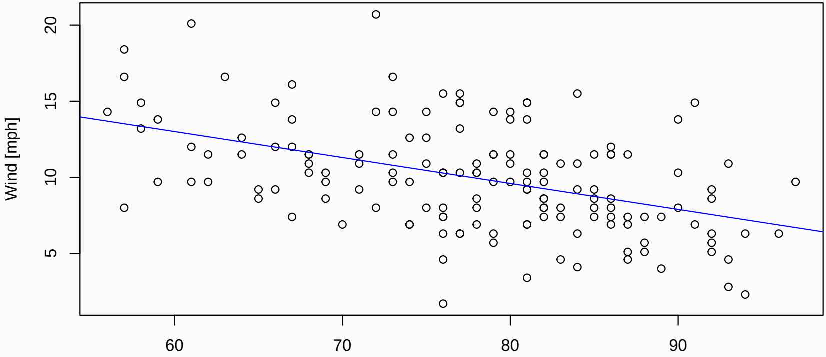


Your turn

Find a line that goes through the data points in the *best possible way* and add it to the plot.

Fitting a regression model to the data

```r
par(mar = c(4,4,2,2))
plot(x = airquality$Temp, y = airquality$Wind, xlab = "Temperature [°F]",
     ylab = "Wind [mph]")
air_model <- lm(Wind ~ 1 + Temp, data = airquality)
abline(air_model, col = "blue")
```



Why is it sufficient to apply `abline()` only to the model object?

Visualizing a regression line with `abline()`

```r
args(abline) # the first two arguments: a (the intercept), b (the slope)
```

```
## function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
##     coef = NULL, untf = FALSE, ...)
## NULL
```

```r
air_model
```

```
##
## Call:
## lm(formula = Wind ~ 1 + Temp, data = airquality)
##
## Coefficients:
## (Intercept)         Temp
##       23.23        -0.17
```

Since `air_model` is a regression object (see argument `reg` of function `abline()`), its coefficients (intercept and slope) will be extracted by calling `coef()` and a corresponding line will be drawn.

Creating plots with ggplot2

R package ggplot2 by Wickham (2016)

- is based on **the grammar of graphics (GoG)** (Wilkinson 2010)
- making (advanced) plots by defining different layers and connecting them by a “+”-sign:
- helpful by handling multiple variables

There is an own book: Wilkinson L.The Grammar of Graphics. 2nd ed.New York: Springer-Verlag; 2005.

The grammar of graphics (GoG) by Wilkinson (2010)

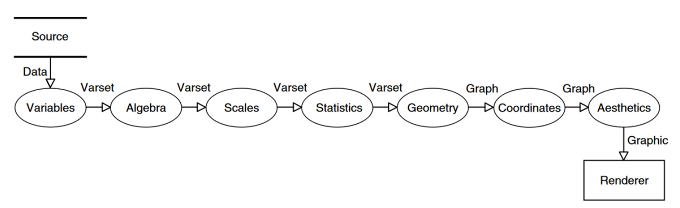


Figure 1: The grammar of graphics data flow from Wilkinson (2010).

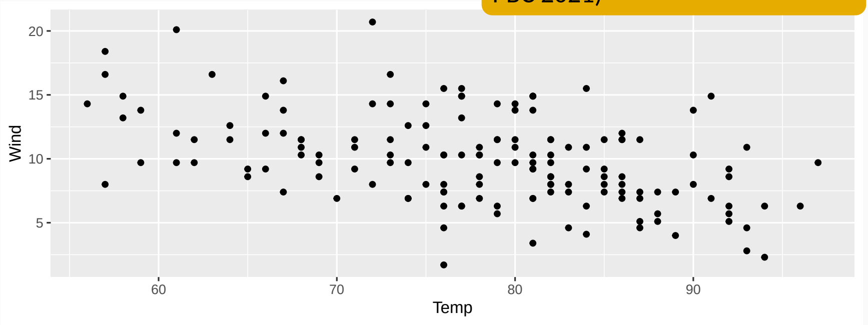
- variables: the variables to plot (*varset* = set of variables)
- algebra: combinations of variables, e.g. tuples (x_i, y_i), (x_i, z_i)
- scales: scaling the data e.g. log, ordering values, ...
- statistics: input a varset and output another varset after computing statistical summaries, e.g. summary statistics of boxplot
- geometry: geometric graphs like points, lines, area, ... the same statistic can be represented by multiple geometric objects
- coordinates: usually, Cartesian coordinates
- aesthetics: maps a graph to a graphic

The grammar of graphics in `ggplot2` (Wickham 2016)

Basics: a data set, mapping variables (aes), a coordinate system (grid) and geoms (representing data points)

```r
# option 1:
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+ # initial. a ggplot object
  geom_point()
# or option 2:
ggplot(data = airquality)+
  geom_point(mapping = aes(x = Temp, y = Wind))
```

Elements of a ggplot: see the cheat sheet of `ggplot2` (RStudio, PBC 2021)

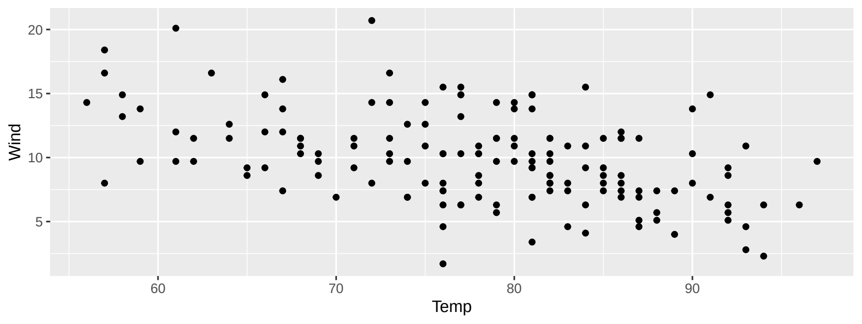


The grammar of graphics in ggplot2, see RStudio, PBC (2021)

Basics: a data set, mapping variables (aes), a coordinate system (grid) and geoms (representing data points)

```
# option 1:
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()
# or option 2:
ggplot(data = airquality)+
  geom_point(mapping = aes(x = Temp, y = Wind))
```

What is the difference between option 1 and 2?



The grammar of graphics in ggplot2, see RStudio, PBC (2021)

Basics: a data set, mapping variables (aes), a coordinate system (grid) and geoms (representing data points)

```
# option 1:
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()
# or option 2:
ggplot(data = airquality)+
  geom_point(mapping = aes(x = Temp, y = Wind))
# or option 3:
ggplot(data = airquality, mapping = aes(x = Temp))+
  geom_point(mapping = aes(y = Wind))
```

Difference: mapping the variables to the elements of the geom in diverse layers

The grammar of graphics in ggplot2, see RStudio, PBC (2021)

Implementing statistics and geometry by

- *Stats* and *Geoms* (see cheat sheet of `ggplot2`) depending on the number of variables and their scale of measurement (continuous, categorical) with their arguments `geom` and `stat` respectively, e.g.

```r
geom_bar(stat = "count") # is equal to
stat_count(geom = "bar")
```

```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point(stat = "identity") # stat = "identity" by default
# is equal to
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  stat_identity(geom = "point") # geom = "point" by default
```

The grammar of graphics in ggplot2, see RStudio, PBC (2021)

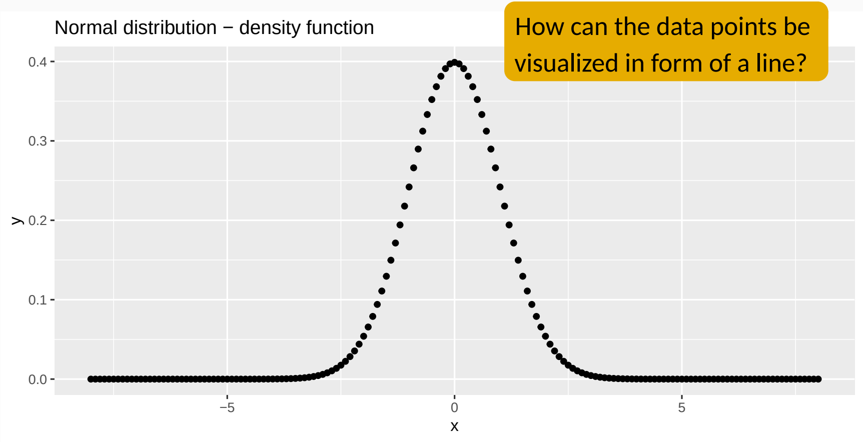
- Aes (aesthetics)
- Scales (scales)
- Coordinate Systems (coordinates)
- Faceting
- Position Adjustments
- Themes
- Labels and Legends
- (Zooming) → Take a look at the (ggplot2 cheat sheet by RStudio, PBC 2021)(`https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf`)

In the following, we will consider these components of a ggplot.

Normal distribution in ggplot2

```r
nd <- data.frame(x, y, y2, y3, y4, y5)
```

```r
ggplot(data = nd, aes(x = x, y = y))+ # initialization of a ggplot object
  geom_point()+ # adding points to a plot
  ggtitle("Normal distribution - density function")
```

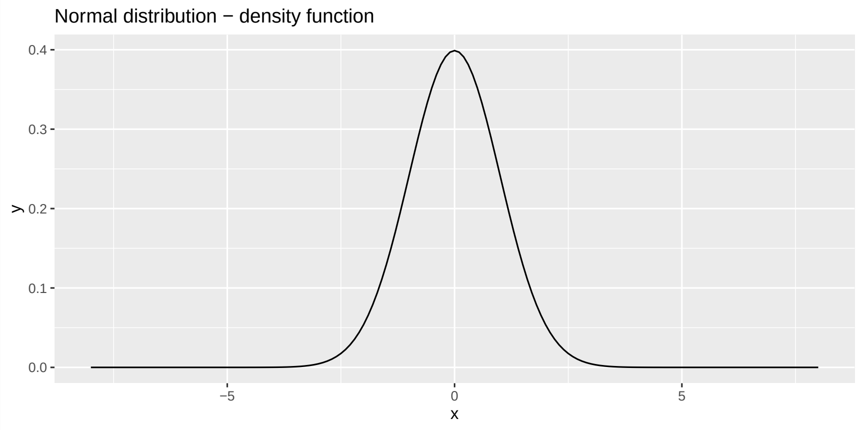


How can the data points be visualized in form of a line?

Normal distribution in ggplot2 - answer

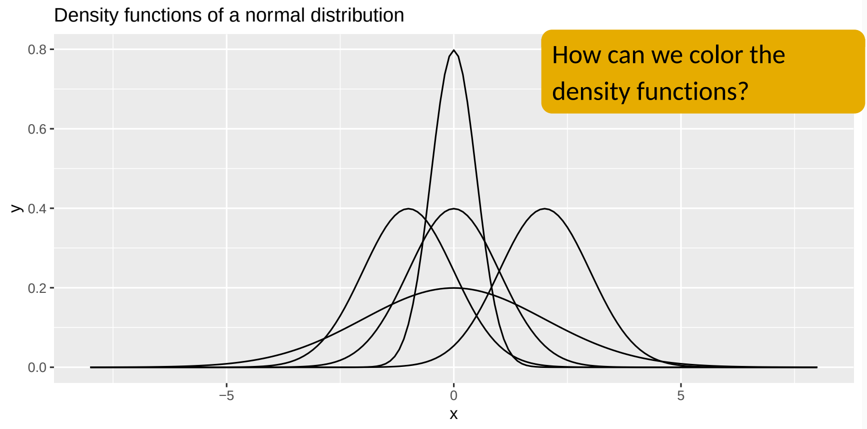
```r
ggplot(data = nd, aes(x = x, y = y))+ # initialization of a ggplot object
  geom_line()+ # add a line to the plot
  ggtitle("Normal distribution - density function")
```

Changing the geometry.



Normal distribution in `ggplot2` - multiple density functions

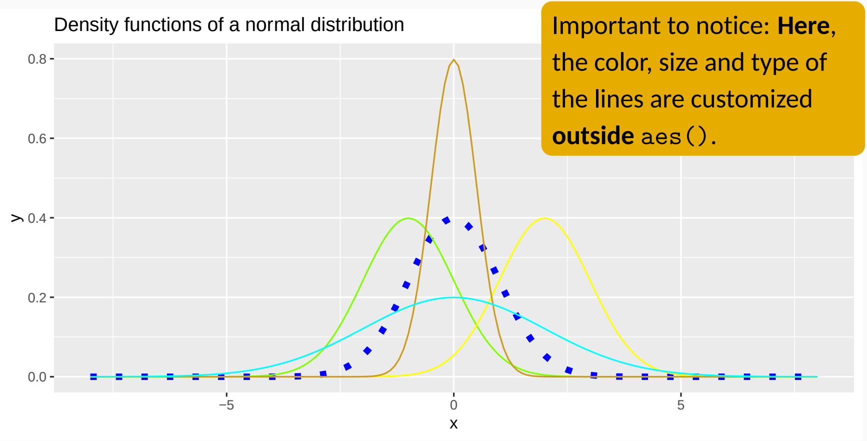
```r
ggplot(data = nd, mapping = aes(x = x))+ # initialization of a ggplot object
  geom_line(mapping = aes(y = y))+ # add a line to the plot
  geom_line(mapping = aes(y = y2))+
  geom_line(mapping = aes(y = y3))+
  geom_line(mapping = aes(y = y4))+
  geom_line(mapping = aes(y = y5))+
  ggtitle("Density functions of a normal distribution")
```



How can we color the density functions?

Coloring several density functions

```r
ggplot(data=nd, mapping = aes(x = x))+ # initialization of a ggplot object
  geom_line(mapping = aes(y = y), color = "blue", size = 2, linetype = "dotted")+
  geom_line(mapping = aes(y = y2), color = "#7FFF00")+ # add a line to the plot
  geom_line(mapping = aes(y = y3), color = "yellow")+
  geom_line(mapping = aes(y = y4), color = "goldenrod3")+
  geom_line(mapping = aes(y = y5), color = "cyan")+
  ggtitle("Density functions of a normal distribution")
```



Important to notice: **Here**, the color, size and type of the lines are customized **outside** `aes()`.

Task - normal distribution

Your turn

Plot the density functions of the normal distribution - saved in `nd` - as a ggplot and color them but this time without adding each single line by an extra layer. How can you do this?

Task - normal distribution - answer (1)

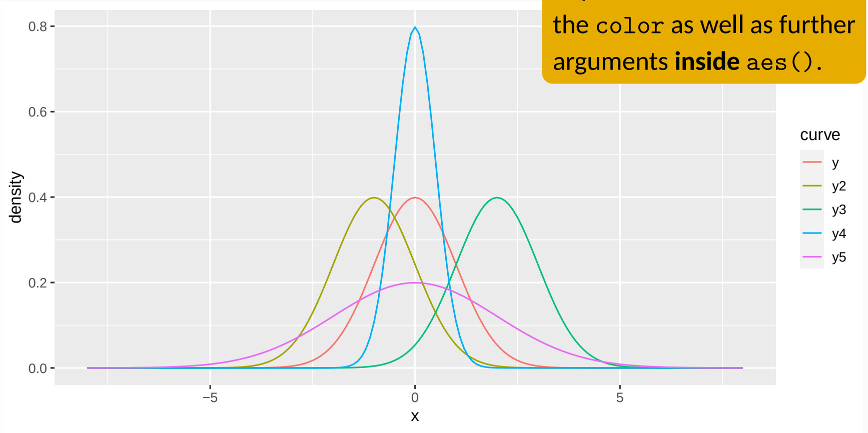
```r
nd2 <- data.frame(x = rep(x,5), density=c(y, y2, y3, y4, y5),
                  curve = c(rep("y", times = length(y)), rep("y2", times = length(y)),
                      rep("y3", times = length(y)), rep("y4", times = length(y)),
                      rep("y5", times = length(y))))
nd2
```

```
##         x   density curve
## 1    -8.0 5.052e-15     y
## 2    -7.9 1.119e-14     y
## 3    -7.8 2.453e-14     y
## 4    -7.7 5.324e-14     y
## 5    -7.6 1.144e-13     y
## 6    -7.5 2.434e-13     y
## 7    -7.4 5.128e-13     y
## 8    -7.3 1.069e-12     y
## 9    -7.2 2.208e-12     y
## 10   -7.1 4.514e-12     y
## 11   -7.0 9.135e-12     y
## 12   -6.9 1.830e-11     y
## 13   -6.8 3.631e-11     y
## 14   -6.7 7.131e-11     y
## 15   -6.6 1.387e-10     y
## 16   -6.5 2.670e-10     y
## 17   -6.4 5.088e-10     y
## 18   -6.3 9.601e-10     y
## 19   -6.2 1.794e-09     y
```

Solution: *lengthens* the data in order to define `y` as the density of the different curves and `color` as the name of the curves inside `aes()`

Task - normal distribution - answer (1)

```r
ggplot(nd2, mapping = aes(x = x, y = density, color = curve))+
  geom_line()
```



Important to notice: Define the `color` as well as further arguments **inside** `aes()`.

`aes()`, see RStudio, PBC (2021)

- **variables**: define the variable’s name **inside** the `geom`’ s `aes()` function, e.g.

```
ggplot(nd2, mapping = aes(x = x, y = density))+
  geom_line(mapping = aes(color = curve))
```

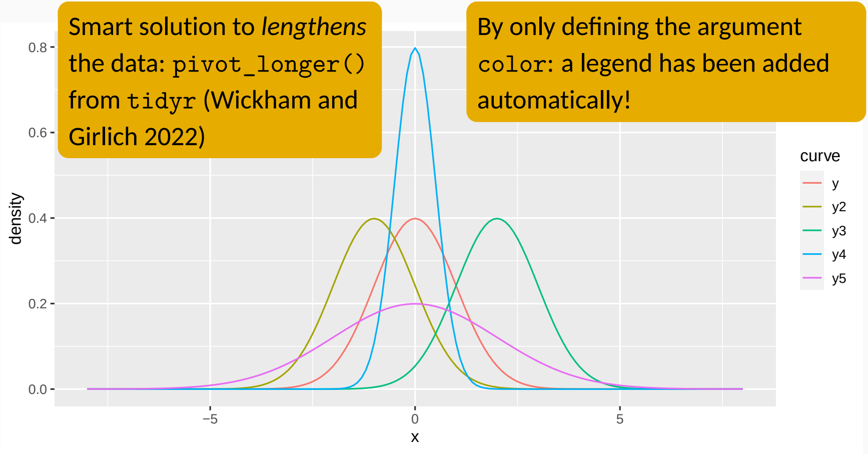
- **single** points, lines etc.: define the aesthetics arguments **outside** `aes()`, e.g.

```
ggplot(nd, mapping = aes(x = x, y = y))+
  geom_line(color = "blue")
```

Task - normal distribution - answer (2)

```r
nd2 <- nd %>% pivot_longer(cols = c(y, y2, y3, y4, y5)) # columns y, y2, y3, y4 and y5
# are concatenated to form one column, the other columns have been adjusted automatically
colnames(nd2)<-c("x", "curve", "density")

ggplot(data = nd2, mapping = aes(x = x, y = density))+
  geom_line(mapping = aes(color = curve))
```



Smart solution to *lengthens* the data: `pivot_longer()` from `tidyr` (Wickham and Girlich 2022)

By only defining the argument `color`: a legend has been added automatically!

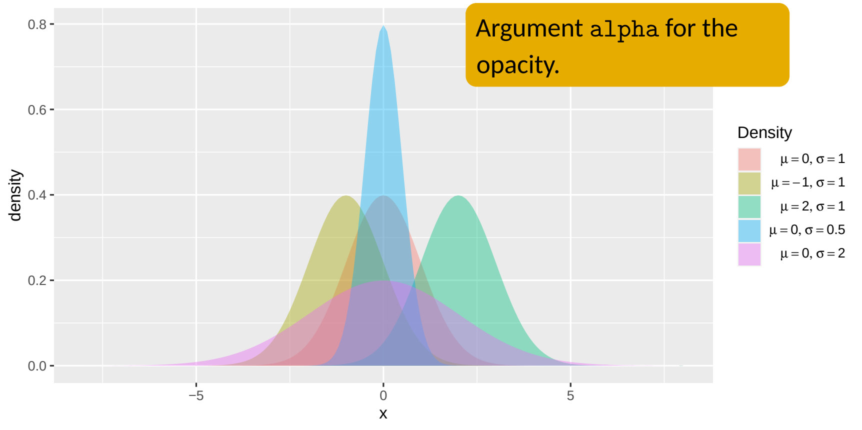
Task - normal distribution - Fill the curves

Your turn

1. Fill the curves of the density functions. Which `geom_*()`-function is appropriate? Do you have to specify arguments of your chosen `geom_*()`-function? If yes, which ones?
2. Customize the layout of your plot, i.e. add a nice legend. *Hint*: Take a look at the `ggplot2` cheat sheet .

Task - normal distribution - Fill the curves - answer

```r
ggplot(data = nd2, mapping = aes(x = x, y = density, fill = curve))+
  geom_polygon(alpha = 0.4)+
  scale_fill_discrete(name = "Density", labels= c(latex2exp::TeX("$\\mu=0, \\sigma=1$"),
  latex2exp::TeX("$\\mu= -1, \\sigma=1$"), latex2exp::TeX("$\\mu=2, \\sigma=1$"),
  latex2exp::TeX("$\\mu=0, \\sigma=0.5$"), latex2exp::TeX("$\\mu=0, \\sigma=2$")))
```



Argument `alpha` for the opacity.

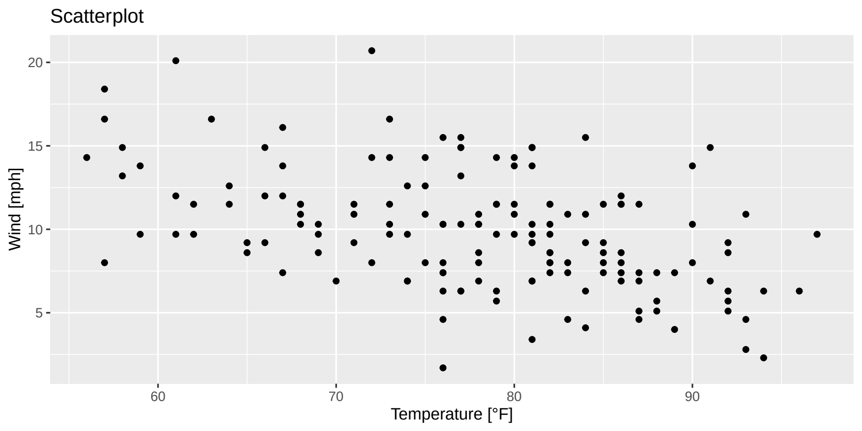
Airquality data set in `ggplot2`

Based on the data set `datasets::airquality` (R Core Team 2021), we want to

- create and customize
  - scatterplots,
  - histograms,
  - boxplots and
  - a linear regression model and
- generate subplots by ≥ 1 discrete variable(s) (called *faceting*).

Airquality data set in ggplot2 - Scatterplot

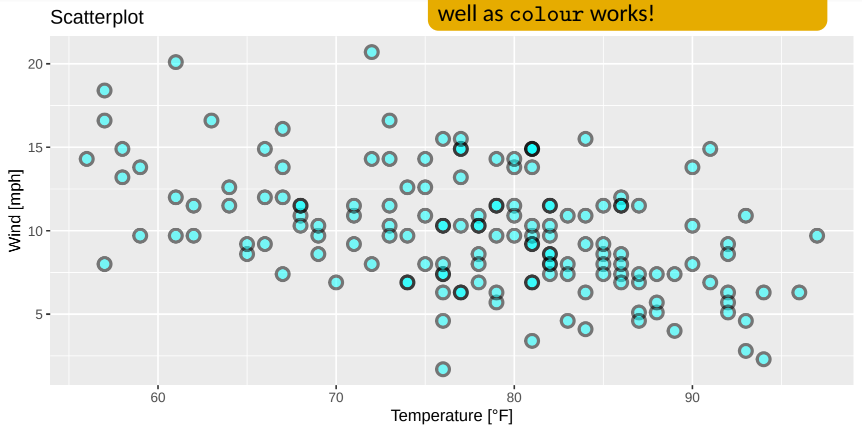
```r
ggplot(data = airquality, aes(x = Temp, y = Wind))+ # initialization of a ggplot object
  geom_point()+ # adding points to the plot
  ggtitle("Scatterplot")+ # adding a title to the plot
  xlab("Temperature [°F]")+ # labeling the x-axis
  ylab("Wind [mph]") # labeling the y-axis
```



Customizing points in ggplot2 (1) - Scatterplot

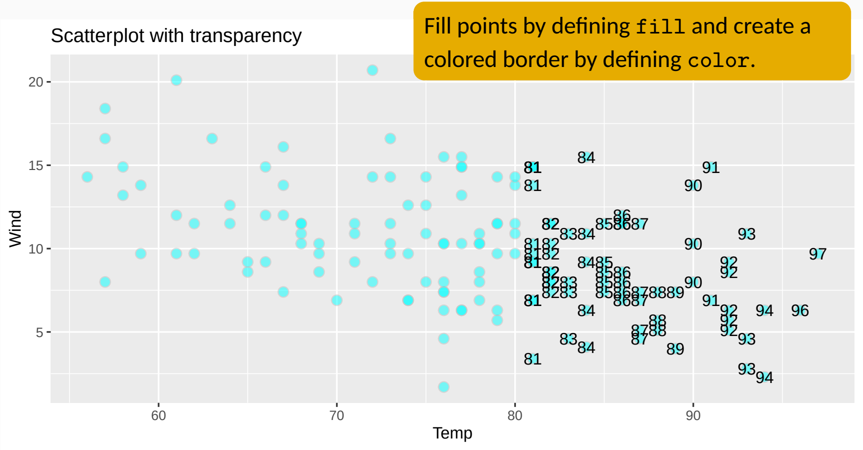
```
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+ # init. of a ggplot object
  # adding points to the plot and making them transparent
  geom_point(alpha = 0.5, shape = 21, color = "black", fill = "cyan", size = 3,
             stroke = 1.5)+
ggtitle("Scatterplot")+ # adding a title to the plot
  xlab("Temperature [°F]")+ # labeling the x-axis
  ylab("Wind [mph]") # labeling the y-axis
```

To define colors: argument color as well as colour works!



Customizing points in `ggplot2` (2) - Scatterplot

```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+ # init. of a ggplot object
  geom_point(alpha = 0.5, shape = 21, color = "grey", fill = "cyan", size = 3)+
  geom_text(mapping = aes(label = ifelse(Temp > 80, Temp, "")))+
ggtitle("Scatterplot with transparency") # adding a title to the plot
```

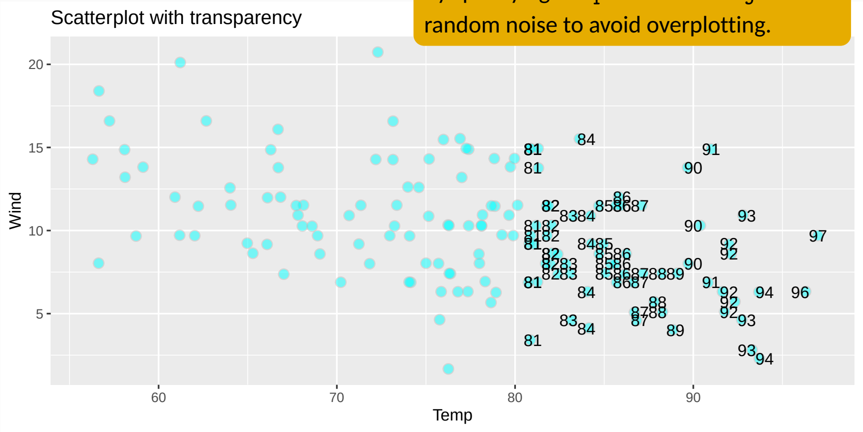


Fill points by defining `fill` and create a colored border by defining `color`.

Customizing points in `ggplot2` (2) - Scatterplot

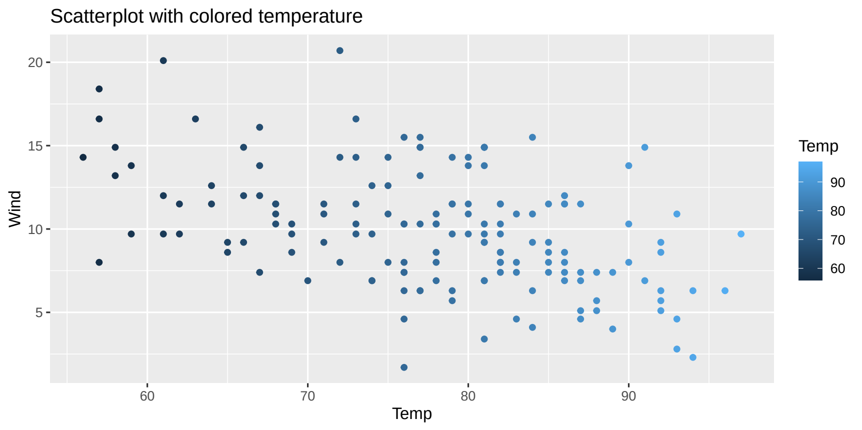
```
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+ # init. of a ggplot object
  geom_point(alpha = 0.5, shape = 21, color = "grey", fill = "cyan", size = 3,
             position = "jitter")+
  geom_text(mapping = aes(label = ifelse(Temp > 80, Temp, "")))+
ggtitle("Scatterplot with transparency") # adding a title to the plot
```

By specifying the `position = "jitter"`: random noise to avoid overplotting.



Customizing points in ggplot2 (3) - Scatterplot

```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point(mapping = aes(color = Temp))+
ggtitle("Scatterplot with colored temperature")
```



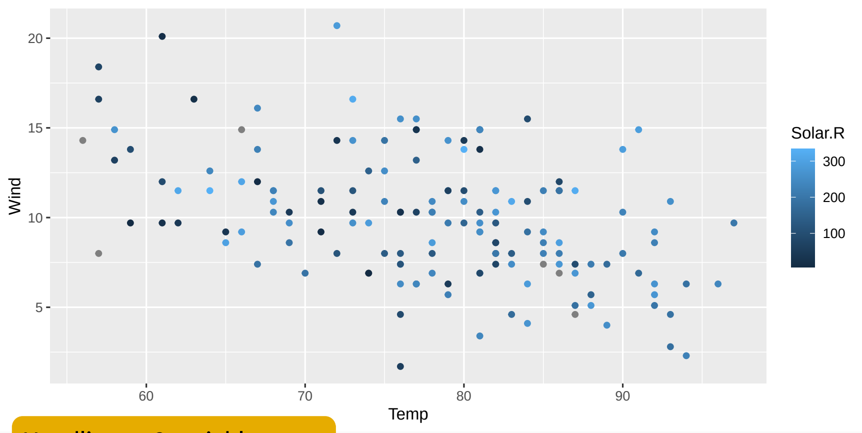
Task - Customizing points in `ggplot2` (4) - Scatterplot

Your turn

Plot `Wind` vs. `Temp` and color the plotted points by `Solar.R`.

Task - Customizing points in ggplot2 (4) - Scatterplot - answer

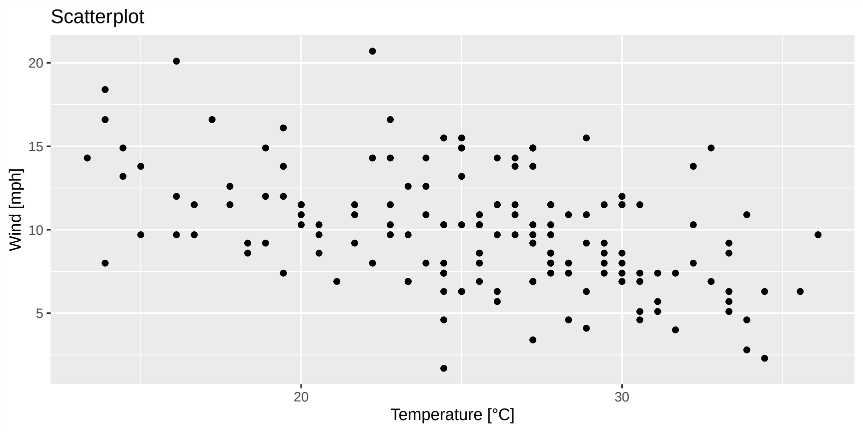
```
ggplot(airquality, mapping = aes(x = Temp, y = Wind, color = Solar.R))+
  geom_point()
```



Handling > 2 variables

Changing scales (1)

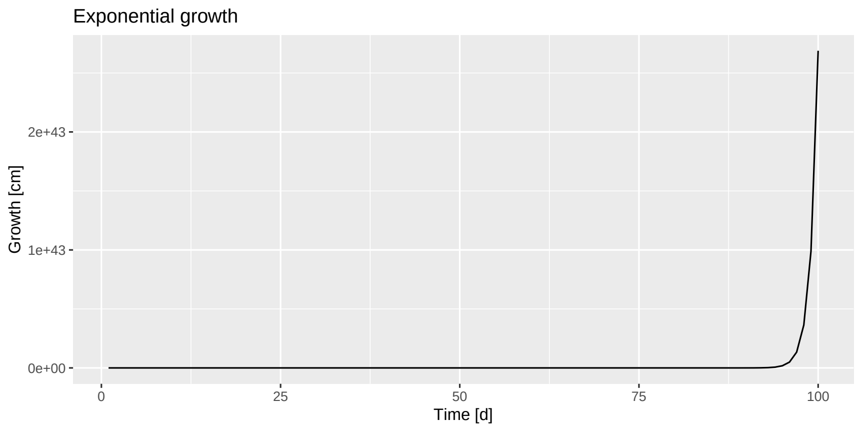
```r
ggplot(data=airquality, mapping = aes(x = (Temp-32)*5/9, y = Wind))+ # modify x,
# (National Institute of Standards and Technology (NIST) (2021))
  geom_point()+
  ggtitle("Scatterplot")+
  xlab("Temperature [°C]")+
  ylab("Wind [mph]")
```



Changing scales (2) - simulated exponential growth data

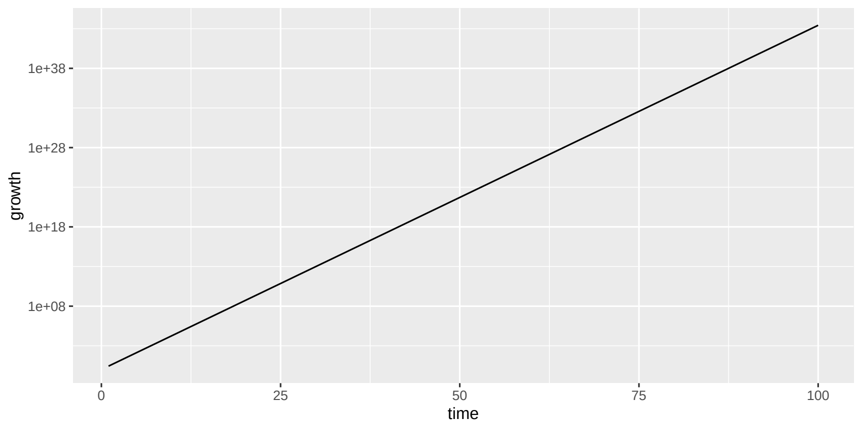
```r
time <- 1:100 # days
growth <- exp(time)
growth_df <- data.frame(time, growth)
```

```r
ggplot(data = growth_df, mapping = aes(x = time, y = growth))+
  geom_line() + xlab("Time [d]") + ylab("Growth [cm]") + ggtitle("Exponential growth")
```



Changing scales (2)

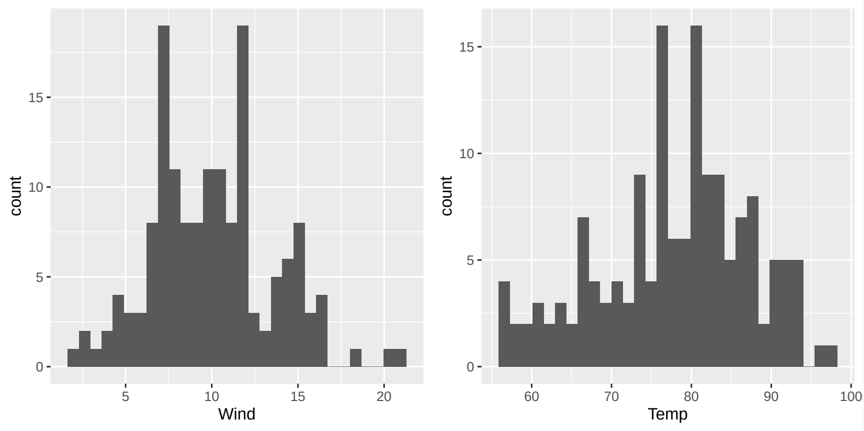
```
ggplot(data = growth_df, mapping = aes(x = time, y = growth))+
   scale_y_log10()+ # applying logarithm (base 10) to y values
  geom_line()+ xlab("Time [d]") + ylab("Growth [cm]")
# alternatively:
ggplot(data=growth_df, mapping = aes(x = time, y = growth))+
  geom_line()+ # first geom layer, then defining the scale
  scale_y_log10()+ xlab("Time [d]") + ylab("Growth [cm]")
```



Histograms

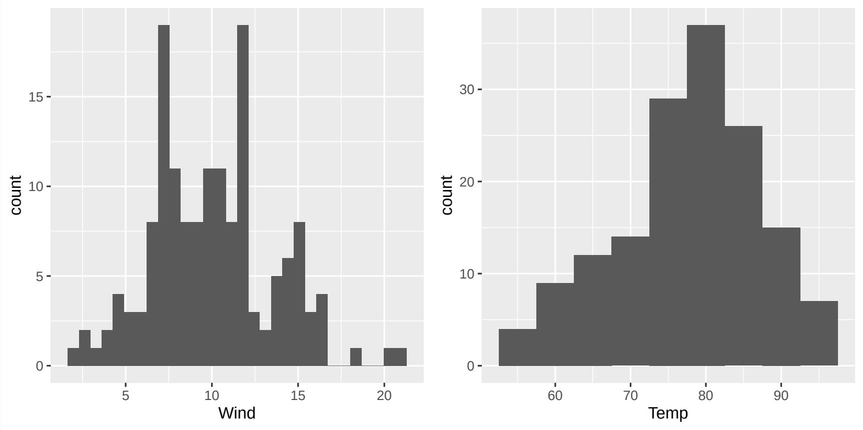
```r
g1 <- ggplot(data = airquality)+
  geom_histogram(mapping = aes(x = Wind))
g2 <- ggplot(data = airquality)+
  geom_histogram(mapping = aes(x = Temp))
grid.arrange(g1, g2, ncol = 2) # arrange both histograms in one row
```

R package `gridExtra` (Auguie 2017)



Histograms - binwidth

```r
g1 <- ggplot(data = airquality)+
  geom_histogram(mapping = aes(x = Wind))
g2 <- ggplot(data = airquality)+
  geom_histogram(mapping = aes(x = Temp), binwidth = 5) # binwidth has been changed
grid.arrange(g1, g2, ncol=2)
```



Extracting data from a ggplot - binwidth

access data by `ggplot_build()`
interval: (`xmin`, `xman`], `x`: center of interval, `count`: number of points in bin

```
typeof(ggplot_build(g2))
```

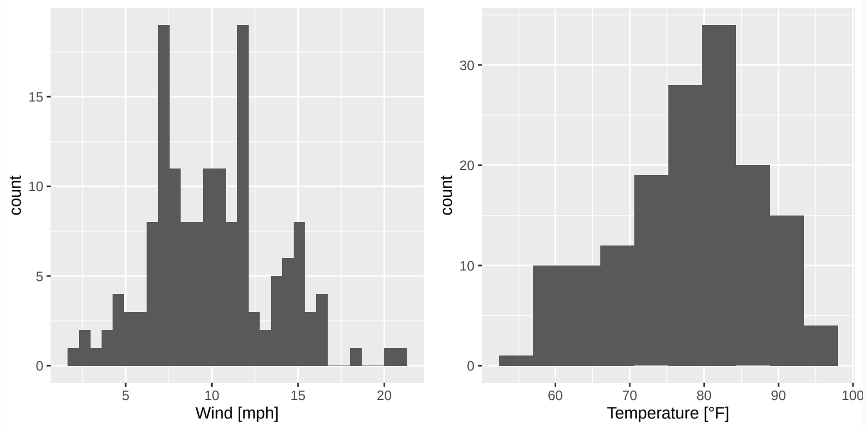
```
## [1] "list"
```

```
ggplot_build(g2)$data # number of rows: number of bins
```

```
## [[1]]
##    y count  x xmin xmax  density ncount ndensity flipped_aes PANEL group ymin
## 1  4     4 55 52.5 57.5 0.005229 0.1081   0.1081       FALSE     1    -1    0
## 2  9     9 60 57.5 62.5 0.011765 0.2432   0.2432       FALSE     1    -1    0
## 3 12    12 65 62.5 67.5 0.015686 0.3243   0.3243       FALSE     1    -1    0
## 4 14    14 70 67.5 72.5 0.018301 0.3784   0.3784       FALSE     1    -1    0
## 5 29    29 75 72.5 77.5 0.037908 0.7838   0.7838       FALSE     1    -1    0
## 6 37    37 80 77.5 82.5 0.048366 1.0000   1.0000       FALSE     1    -1    0
## 7 26    26 85 82.5 87.5 0.033987 0.7027   0.7027       FALSE     1    -1    0
## 8 15    15 90 87.5 92.5 0.019608 0.4054   0.4054       FALSE     1    -1    0
## 9  7     7 95 92.5 97.5 0.009150 0.1892   0.1892       FALSE     1    -1    0
##   ymax colour   fill size linetype alpha
## 1    4     NA grey35  0.5        1    NA
## 2    9     NA grey35  0.5        1    NA
## 3   12     NA grey35  0.5        1    NA
## 4   14     NA grey35  0.5        1    NA
## 5   29     NA grey35  0.5        1    NA
## 6   37     NA grey35  0.5        1    NA
## 7   26     NA grey35  0.5        1    NA
```

Histograms - number of bins

```r
g1 <- ggplot(data = airquality)+ ## initialization of a ggplot object
  geom_histogram(mapping = aes(x = Wind)) + labs(x = "Wind [mph]")
g2 <- ggplot(data = airquality)+ ## initialization of a ggplot object
  geom_histogram(mapping = aes(x = Temp), bins = 10) + labs(x = "Temperature [°F]")
grid.arrange(g1, g2, ncol = 2)
```



Extracting data from a ggplot - number of bins

```
ggplot_build(g2)$data
```

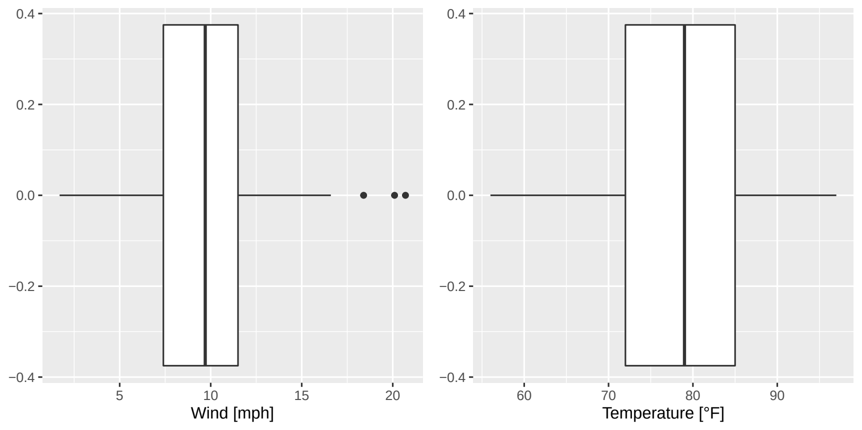
interval: (xmin, xman], x: center of interval

count: number of points in bin

```
## [[1]]
##      y count     x  xmin  xmax  density  ncount ndensity flipped_aes PANEL group
## 1   1     1 54.67 52.39 56.94 0.001435 0.02941  0.02941       FALSE     1    -1
## 2  10    10 59.22 56.94 61.50 0.014347 0.29412  0.29412       FALSE     1    -1
## 3  10    10 63.78 61.50 66.06 0.014347 0.29412  0.29412       FALSE     1    -1
## 4  12    12 68.33 66.06 70.61 0.017217 0.35294  0.35294       FALSE     1    -1
## 5  19    19 72.89 70.61 75.17 0.027260 0.55882  0.55882       FALSE     1    -1
## 6  28    28 77.44 75.17 79.72 0.040172 0.82353  0.82353       FALSE     1    -1
## 7  34    34 82.00 79.72 84.28 0.048780 1.00000  1.00000       FALSE     1    -1
## 8  20    20 86.56 84.28 88.83 0.028694 0.58824  0.58824       FALSE     1    -1
## 9  15    15 91.11 88.83 93.39 0.021521 0.44118  0.44118       FALSE     1    -1
## 10  4     4 95.67 93.39 97.94 0.005739 0.11765  0.11765       FALSE     1    -1
##    ymin ymax colour   fill size linetype alpha
## 1     0    1     NA grey35  0.5        1    NA
## 2     0   10     NA grey35  0.5        1    NA
## 3     0   10     NA grey35  0.5        1    NA
## 4     0   12     NA grey35  0.5        1    NA
## 5     0   19     NA grey35  0.5        1    NA
## 6     0   28     NA grey35  0.5        1    NA
## 7     0   34     NA grey35  0.5        1    NA
## 8     0   20     NA grey35  0.5        1    NA
## 9     0   15     NA grey35  0.5        1    NA
## 10    0    4     NA grey35  0.5        1    NA
```

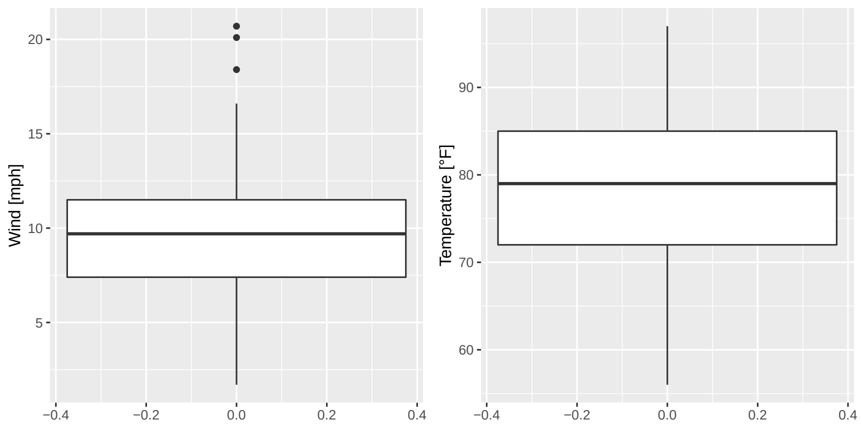
Boxplots

```r
g1 <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Wind)) + labs(x = "Wind [mph]")
g2 <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Temp)) + labs(x = "Temperature [°F]")
grid.arrange(g1, g2, ncol = 2)
```



Boxplots - flipping coordinates

```r
g1 <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Wind))+ labs(x = "Wind [mph]")+
  coord_flip() # flip coordinates
g2 <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Temp))+ labs(x = "Temperature [°F]")+
  coord_flip() # flip coordinates
grid.arrange(g1, g2, ncol = 2)
```



Extracting data from boxplots

```r
boxplot_g1 <- ggplot_build(g1)$data[[1]]

typeof(boxplot_g1)
```

```
## [1] "list"
```

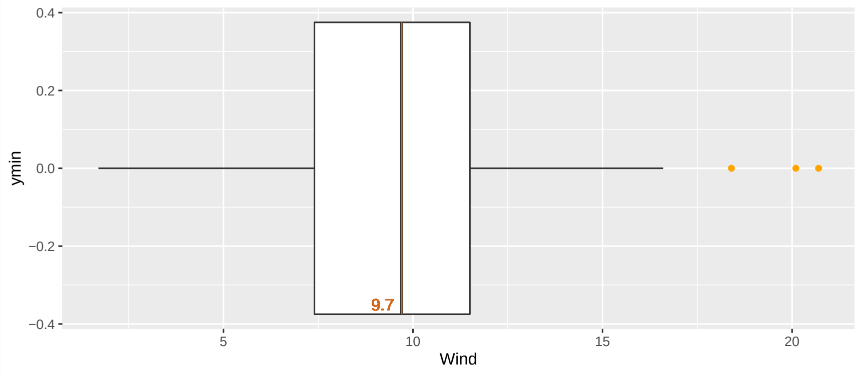
```r
boxplot_g1$outliers # accessing the outliers
```

```
## [[1]]
## [1] 20.1 18.4 20.7
```

Several further values of the boxplot are saved in `boxplot_g1`.

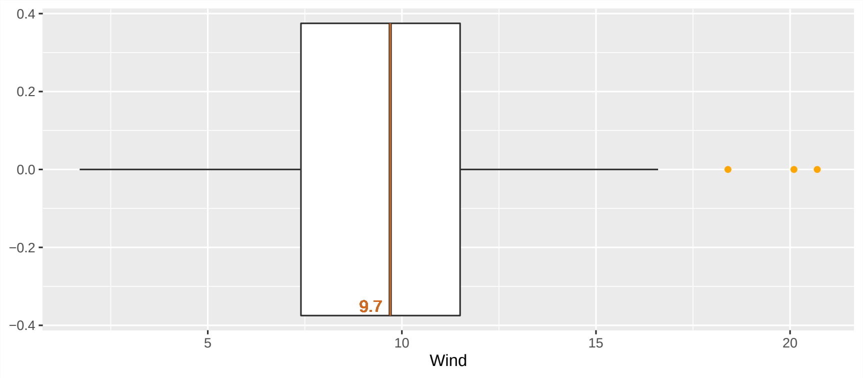
Boxplots - marking values

```r
wind_boxplot <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Wind), outlier.colour = "orange")+ # coloring outliers
  geom_segment(data = boxplot_g1, mapping = aes(x = xmiddle, xend = xmiddle,
               y = ymin, yend = ymax), colour = "chocolate")+ # ymin, ymax: coordinates
  # of box
  geom_text(x = 9.2, y = -0.35, label = paste(boxplot_g1$xmiddle), color = "chocolate")
wind_boxplot # call the defined boxplot
```



Boxplots - marking values - removing labeling of y-axis

```r
wind_boxplot <- ggplot(data = airquality)+
  geom_boxplot(mapping = aes(x = Wind), outlier.colour = "orange")+ # coloring outliers
  geom_segment(data = boxplot_g1, mapping = aes(x = xmiddle, xend = xmiddle,
               y = ymin, yend = ymax), colour = "chocolate")+ # ymin, ymax: coordinates
  # of box
  geom_text(x = 9.2, y = -0.35, label = paste(boxplot_g1$xmiddle), color = "chocolate")+
    theme(axis.title.y = element_blank()) # remove title of y-axis
wind_boxplot # call the defined boxplot
```



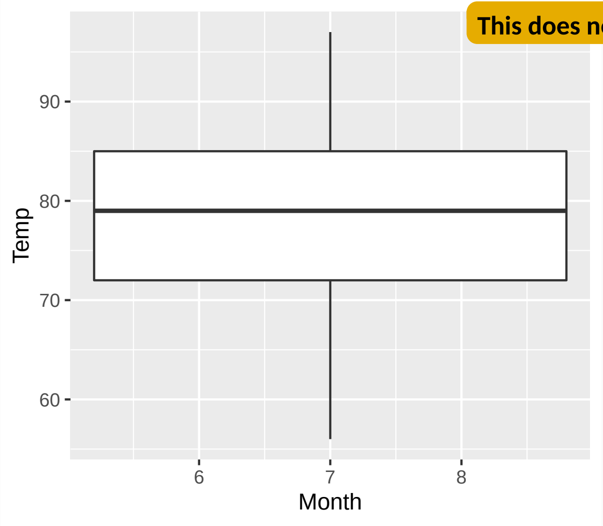
Task - airquality data set - boxplots for the single months

Your turn

Based on the `airquality` data set: How can you create several boxplots of `Temp` for the single months?

Task - airquality data set - boxplots of `Temp` for single months - answer

```r
ggplot(airquality, mapping = aes(x = Month, y = Temp))+
  geom_boxplot()
```

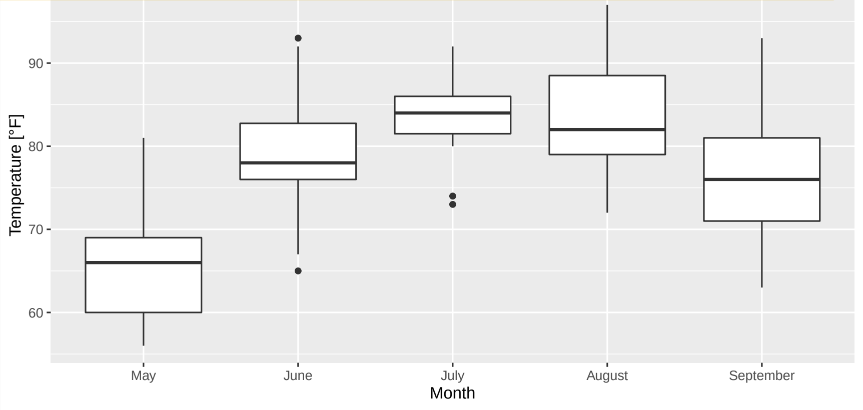


This does not work!

Task - airquality data set - boxplots of `Temp` for single months - answer

```r
airquality$Month_categ <- factor(airquality$Month, levels = c(5, 6, 7, 8, 9),
                            labels = c("May", "June", "July", "August", "September"))
ggplot(airquality, mapping = aes(x = Month_categ, y = Temp))+
  xlab("Month")+ ylab("Temperature [°F]")+
  geom_boxplot()
```

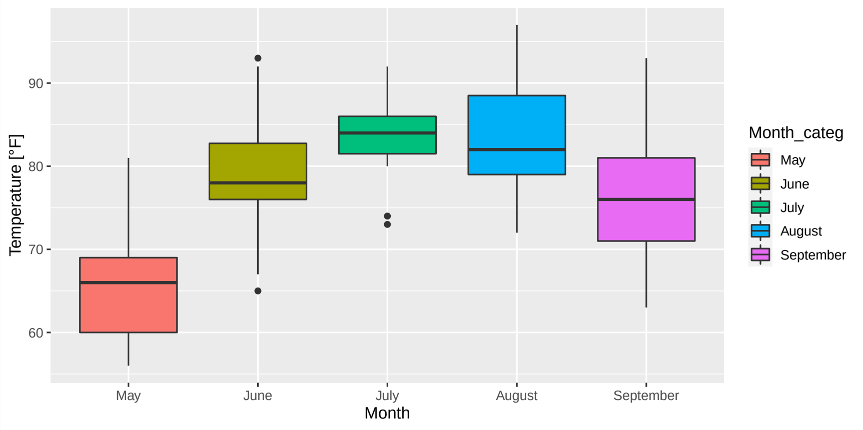
Transforming the numerical values of `Month` into categories!



Airquality - Boxplots of `Temp` for single months

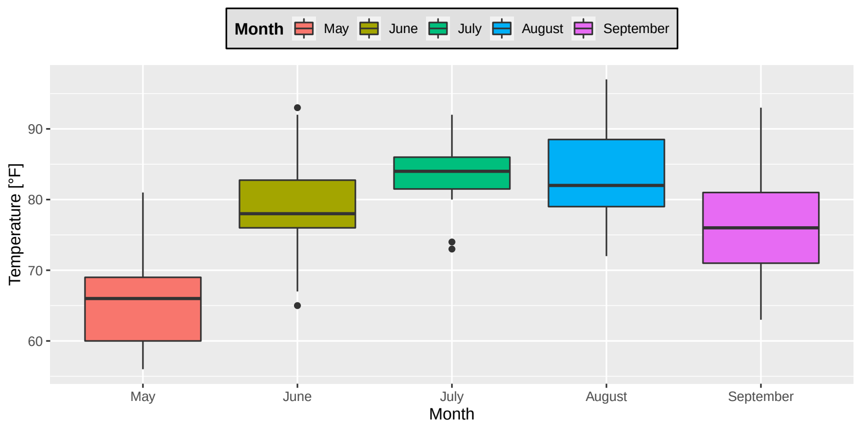
```r
boxplot_Temp_Month <- ggplot(airquality, mapping = aes(x = Month_categ, y = Temp,
  fill = Month_categ))+
  xlab("Month")+ ylab("Temperature [°F]")+
  geom_boxplot()
boxplot_Temp_Month # call the plot
```

By defining only the argument `fill`: a legend has automatically been added!



Airquality - Boxplots of `Temp` for single months

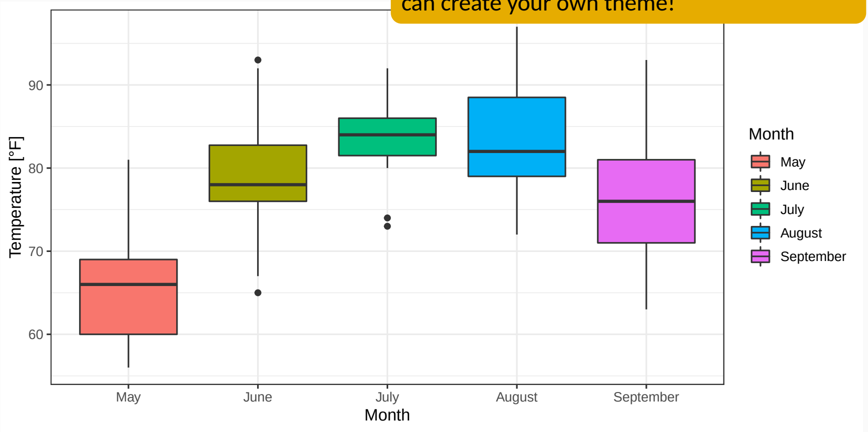
```r
boxplot_Temp_Month + guides(fill = guide_legend(title = "Month"))+ # title of legend
theme(legend.title = element_text(face = "bold"), # make title bold
      # fill legend and create a border
      legend.background = element_rect(fill = "gray88", colour = "black"),
      legend.position = "top") # change position of legend
```



Airquality - Boxplots of `Temp` for single months - predefined themes

```
boxplot_Temp_Month +
  guides(fill = guide_legend(title = "Month"))+
  theme_bw() # black and white
```

Some predefined themes already exist or you can create your own theme!



ggplot2 - theme() (Wickham 2016)

theme(): non-data components of plots (different arguments to access the different components)

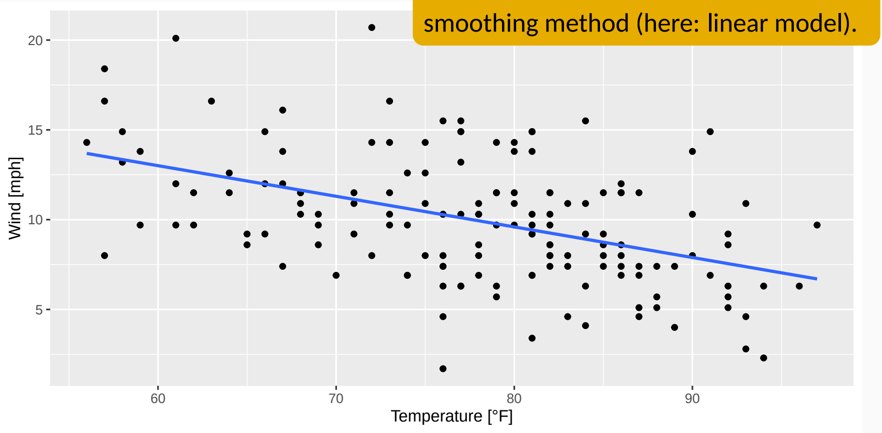
→ element_* functions to modify the attributes (e.g. color, size etc.)

- element_blank(): nothing is drawn e.g. to leave out a legend
- element_rect(): for borders and backgrounds (abbreviation for rectangle)
- element_line(): customize lines
- elment_text(): customize text
- and some more…

Linear regression in ggplot2 (Wickham 2016)

```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()+
  geom_smooth(method = "lm", se = FALSE)+
  labs(x = "Temperature [°F]", y = "Wind [mph]")
```

Argument `method` for specifying the smoothing method (here: linear model).

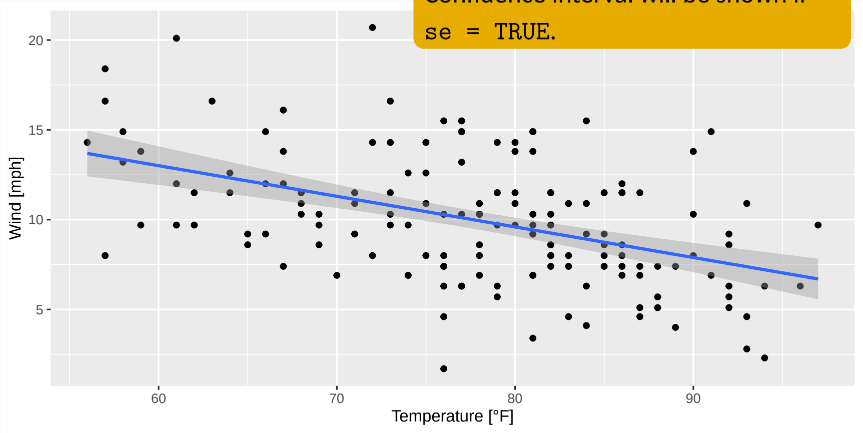


What happens when you set `se = TRUE`?

Linear regression in `ggplot2`

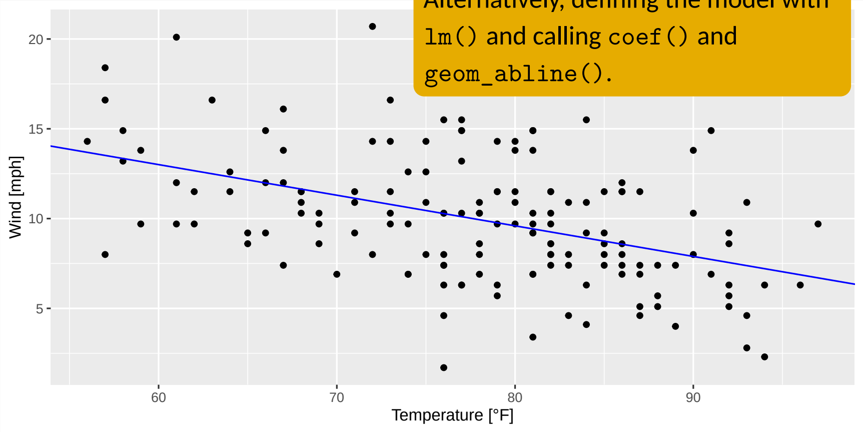
```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()+
  geom_smooth(method = "lm", se = TRUE)+ # se = TRUE by default
  labs(x = "Temperature [°F]", y = "Wind [mph]")
```

Confidence interval will be shown if `se = TRUE`.



Linear regression in `ggplot2`

```r
air_lmodel <- lm(Wind ~ 1 + Temp, data = airquality)
ggplot(airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()+
  geom_abline(slope = coef(air_lmodel)[2], intercept = coef(air_lmodel)[1],
              color = "blue")+ labs(x = "Temperature [°F]", y = "Wind [mph]")
```



Alternatively, defining the model with `lm()` and calling `coef()` and `geom_abline()`.

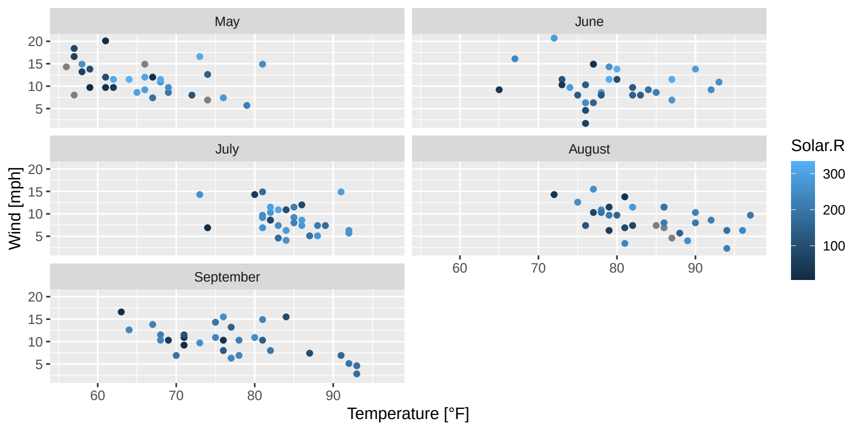
Task - airquality - multiple variables - faceting

Your turn

Plot `Wind` vs. `Temp` for the single months and color the plotted points by `Solar.R`.

Task - airquality - multiple variables - faceting - answer, see RStudio, PBC (2021)

```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind, color = Solar.R))+
  geom_point()+
  facet_wrap(~ Month_categ, nrow = 3, ncol = 2)+ # rectangular layout
  # (by default: nrow = 2, ncol = 3)
  labs(x = "Temperature [°F]", y = "Wind [mph]")
```



modified data set `datasets::airquality` (R Core Team 2021)

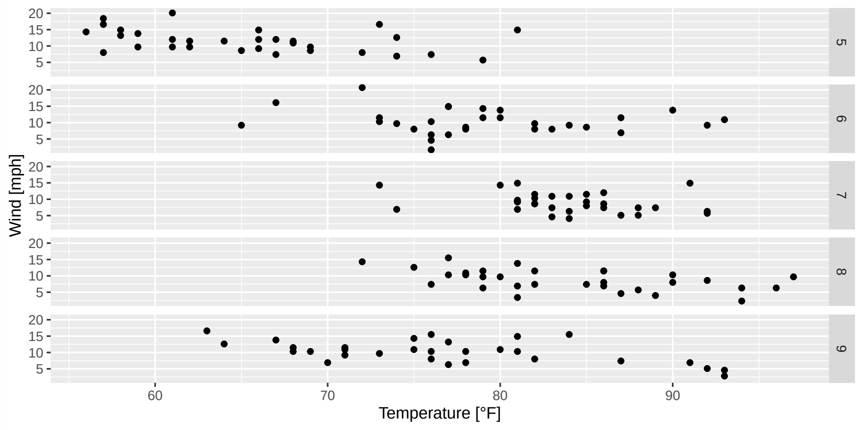
```
airquality
```

```
##     Ozone Solar.R Wind Temp Month Day Month_categ
## 1      41     190  7.4   67     5   1         May
## 2      36     118  8.0   72     5   2         May
## 3      12     149 12.6   74     5   3         May
## 4      18     313 11.5   62     5   4         May
## 5      NA      NA 14.3   56     5   5         May
## 6      28      NA 14.9   66     5   6         May
## 7      23     299  8.6   65     5   7         May
## 8      19      99 13.8   59     5   8         May
## 9       8      19 20.1   61     5   9         May
## 10     NA     194  8.6   69     5  10         May
## 11      7      NA  6.9   74     5  11         May
## 12     16     256  9.7   69     5  12         May
## 13     11     290  9.2   66     5  13         May
## 14     14     274 10.9   68     5  14         May
## 15     18      65 13.2   58     5  15         May
## 16     14     334 11.5   64     5  16         May
## 17     34     307 12.0   66     5  17         May
## 18      6      78 18.4   57     5  18         May
## 19     30     322 11.5   68     5  19         May
## 20     11      44  9.7   62     5  20         May
## 21      1       8  9.7   59     5  21         May
## 22     11     320 16.6   73     5  22         May
## 23      4      25  9.7   61     5  23         May
```

Airquality - multiple variables - faceting (Wickham 2016)

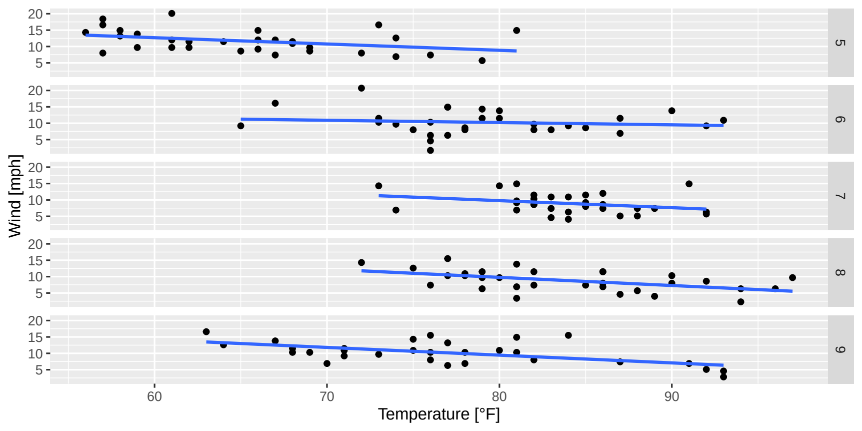
```r
air_ggplot <- ggplot(airquality, mapping = aes(x = Temp, y = Wind))+
  geom_point()+
  facet_grid(rows = vars(Month))+
  labs(x = "Temperature [°F]", y = "Wind [mph]")
air_ggplot
```

Forming subplots based on `Month` by calling `vars()`



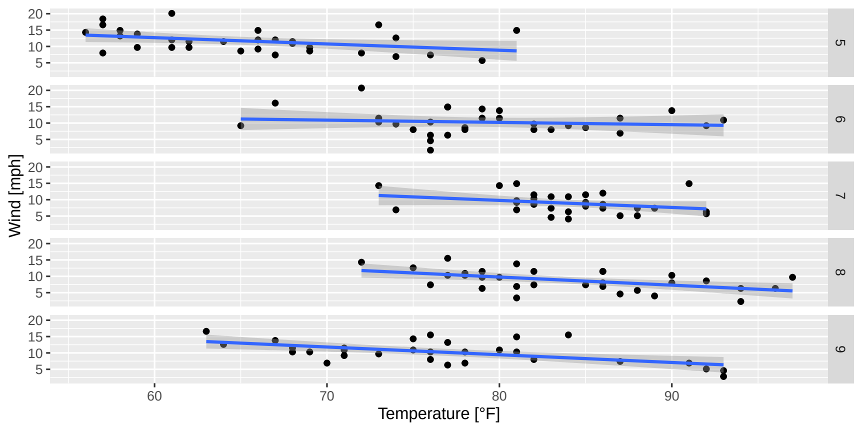
Faceting - linear regression in `ggplot2` (Wickham 2016)

```
air_ggplot + # call the already defined ggplot
  geom_smooth(method = "lm", se = FALSE) # add a linear model to each subplot
```



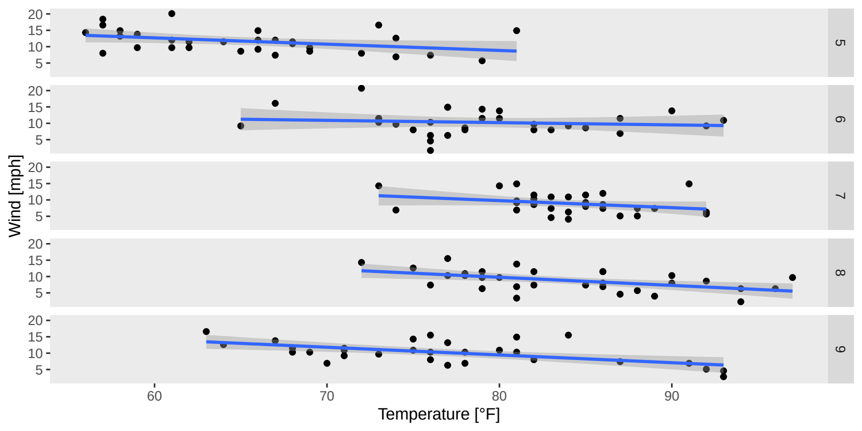
Faceting - linear regression in `ggplot2` (Wickham 2016)

```r
air_ggplot +
  geom_smooth(method = "lm", se = TRUE)
```



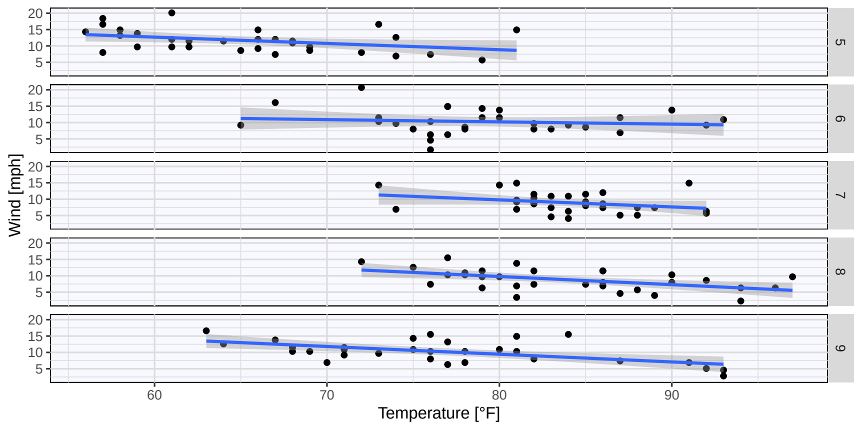
Additional features in `ggplot2` - Grid (1)

```r
air_ggplot +
  geom_smooth(method = "lm", se = TRUE)+
  theme(panel.grid = element_blank()) # turning the grid off
```



Additional features in ggplot2 - Grid (2)

```r
air_ggplot +
  geom_smooth(method = "lm", se = TRUE)+
  # coloring the background and border:
theme(panel.background = element_rect(fill = "ghostwhite", colour = "black"),
        panel.grid = element_line(colour = "gainsboro")) # coloring the grid lines
```



Task - bar plots in `ggplot2` (Wickham 2016)

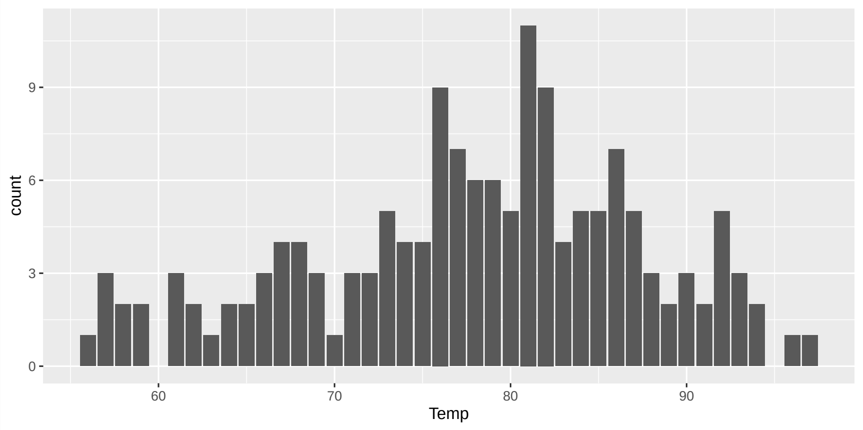
Your turn

So far, we have seen different geoms (for scatterplots, histograms, boxplots and smoothing methods).
Now, it is your turn: Explore the functions `geom_bar()` and `geom_col()` by applying it to the `datasets::airquality` (R Core Team 2021) data set. What will happen if you change the position argument?

bar plots in ggplot2 - airquality - answer

```r
gbar_temp <- ggplot(airquality, mapping = aes(x = Temp))+
  geom_bar()
gbar_temp
```

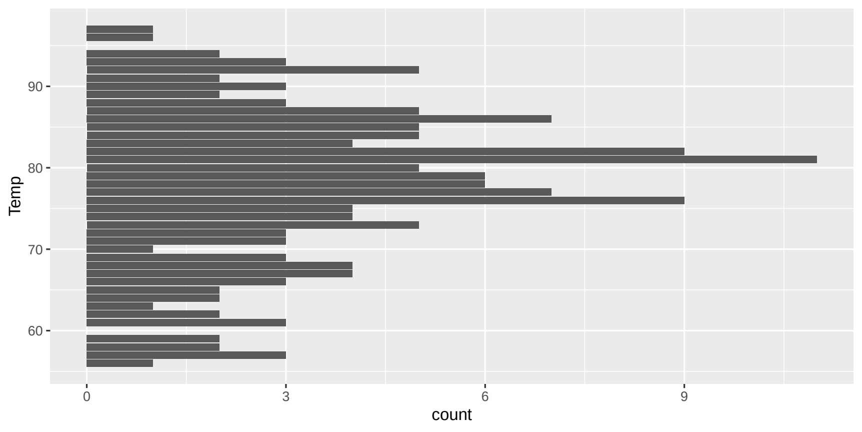
Counts the different temperatures; vertical bars



Task - bar plots in `ggplot2` - airquality - answer

```
ggplot(data = airquality, mapping = aes(y = Temp))+
  geom_bar()
```

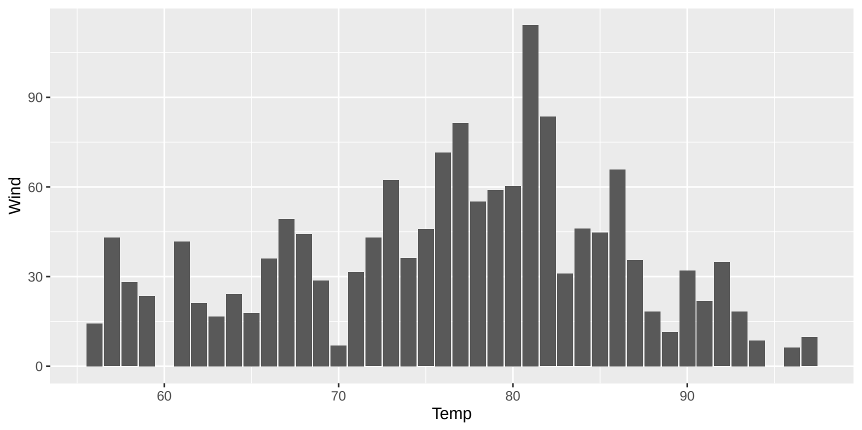
Counts the different temperatures; horizontal bars



Task - bar plots in ggplot2 - airquality - answer

```r
gbar_temp_wind <- ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
  geom_col()
gbar_temp_wind
```

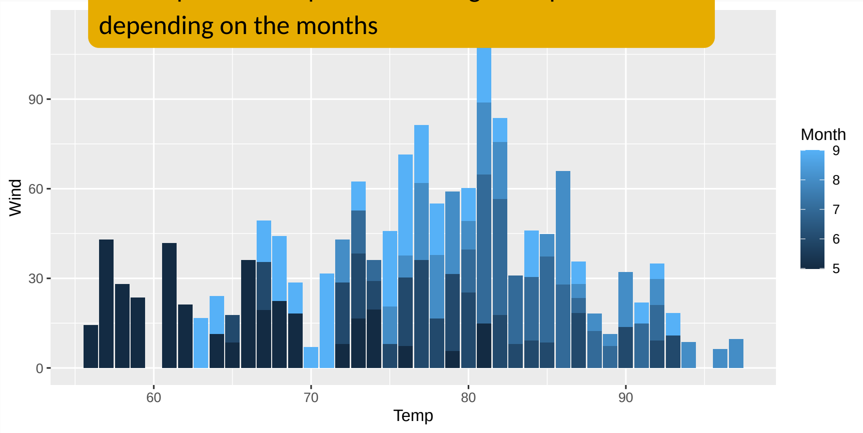
Sums up the wind speed at the single temperatures



Task - bar plots in `ggplot2` - airquality - answer

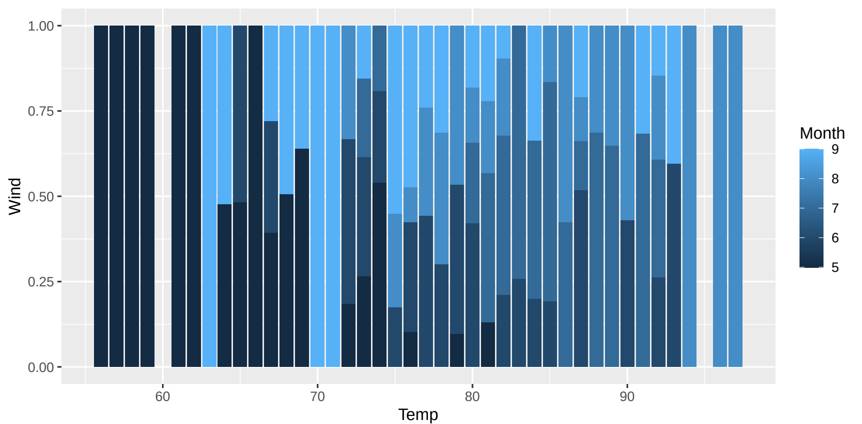
```r
ggplot(data = airquality, mapping = aes(x = Temp, y = Wind, fill = Month))+
  geom_col(position = "stack") # by default, position = "stack"
```

Sums up the wind speed at the single temperatures depending on the months



Task - bar plots in `ggplot2` - airquality - answer

```r
ggplot(airquality, mapping = aes(x = Temp, y = Wind, fill = Month))+
  geom_col(position = "fill")
```

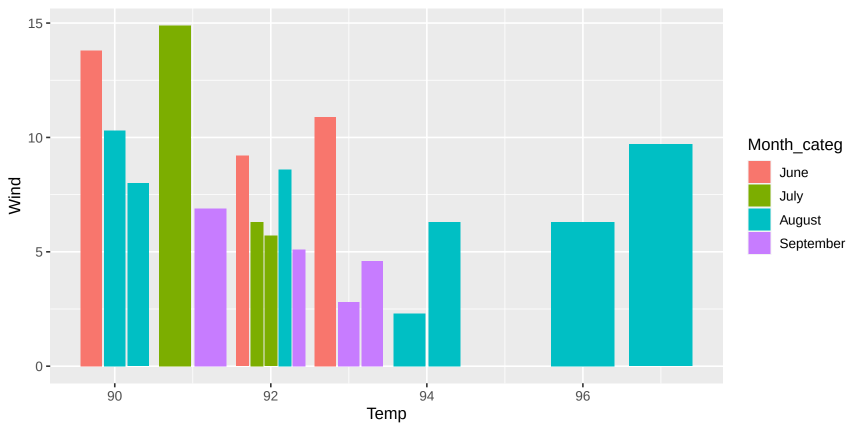


Shows the relative proportions of the summed wind speed at the single temperatures depending on the months

Task - bar plots in `ggplot2` - airquality - answer

```r
airquality %>% filter(Temp >= 90) %>%
ggplot(mapping = aes(x = Temp, y = Wind, fill = Month_categ))+
  geom_col(position = "dodge2")
```

Month as category, for each value: one bar



Advanced plots with plotly (Sievert 2020)

- Interactive plots
- (Interactive) 3D plots

R package `plotly` (Sievert 2020)

Interactive plots

```
air_lm_ggplot <- ggplot(data = airquality, mapping = aes(x = Temp, y = Wind))+
geom_point()+
geom_smooth(method = "lm", se = FALSE)
ggplotly(air_lm_ggplot) # converts ggplot2 to a plotly object
```

R package `plotly` (Sievert 2020)

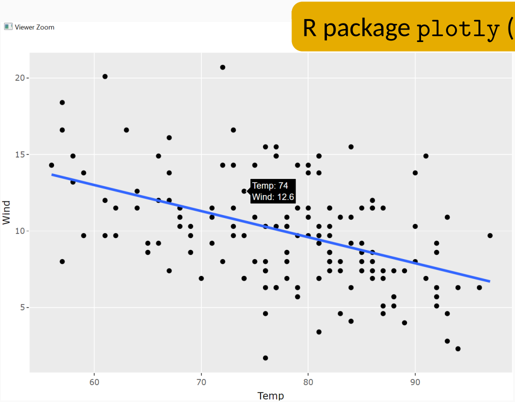


Figure 2: Two-dimensional plot (`plotly` object) based on `air_lm_ggplot`.

Interactive three-dimensional plots (Sievert 2020)

```
plot_ly(airquality, x = ~Temp, y = ~Wind, z = ~Solar.R) %>%
  add_markers(color = ~Month)
```

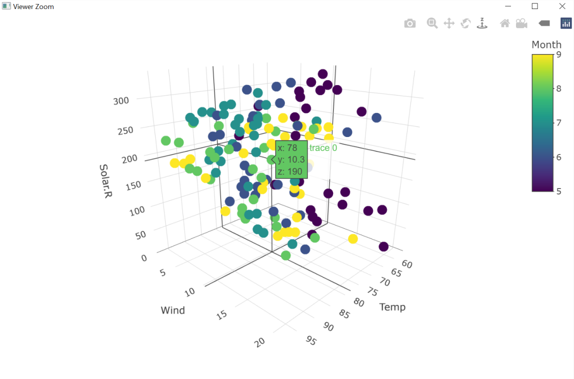


Figure 3: Three-dimensional plot (`plotly` object) based on `datasets::airquality` (R Core Team 2021).

Task - Repetition

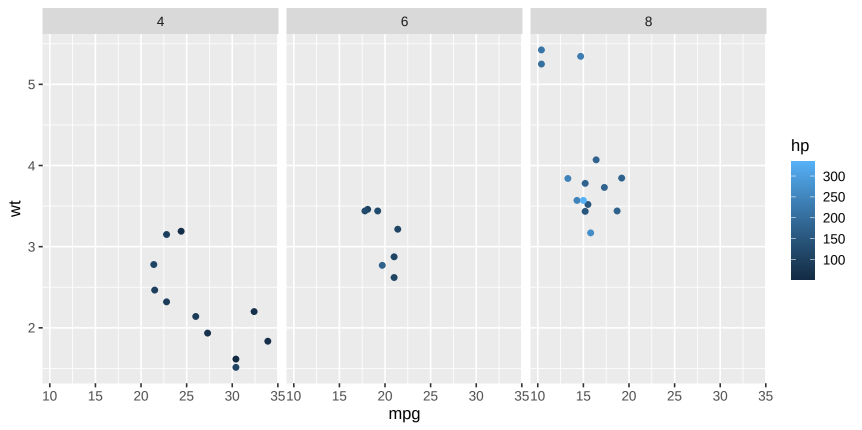
Your turn

In your R environment the `datasets::mtcars` data set (R Core Team 2021) is available.

1) Plot the cars' weights vs. the miles per gallon, colored by horsepower dependent on the number of cylinders.
2) Add to your plot from 1) a horizontal line that goes through the point (0, 2.5).
3) Change the plot you have created so far (1)-2)) by coloring all data points that lie below the horizontal line in red, whereas all other data points are colored in green-blue.
4) Change the plot you have created so far (1)-3)) by differentiating between those observations whose number of cylinders is equal to four and the rest.
5) Label the resulting panels by setting `labeller = (cyl = label_both)` and see what happens.
6) Plot miles per gallon vs. weight and fit a linear regression model. Create a plot on the one hand with `base` R and on the other hand with `ggplot2`.

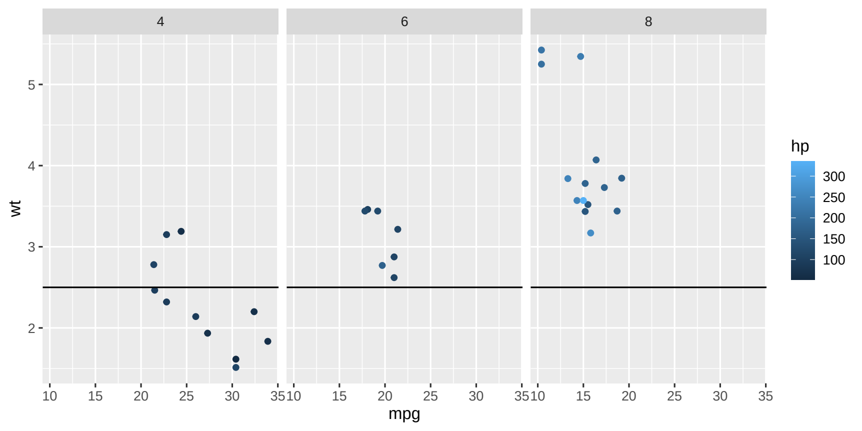
Task - Repetition - 1) - answers

```
mtcars %>% ggplot(mapping = aes(x = mpg, y = wt, color = hp))+
  geom_point()+
  facet_wrap(~ cyl)
```



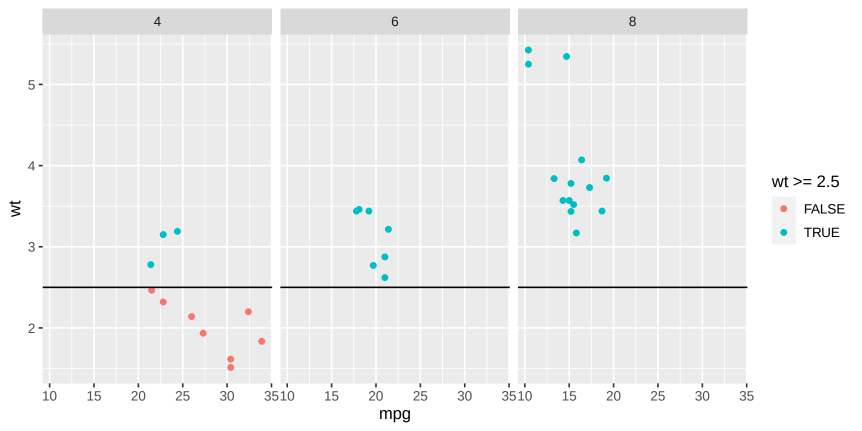
Task - Repetition - 2) - answers

```r
mtcars %>% ggplot(mapping = aes(x = mpg, y = wt, color = hp))+
  geom_point()+
  facet_wrap(~ cyl)+
  geom_hline(yintercept = 2.5)
```



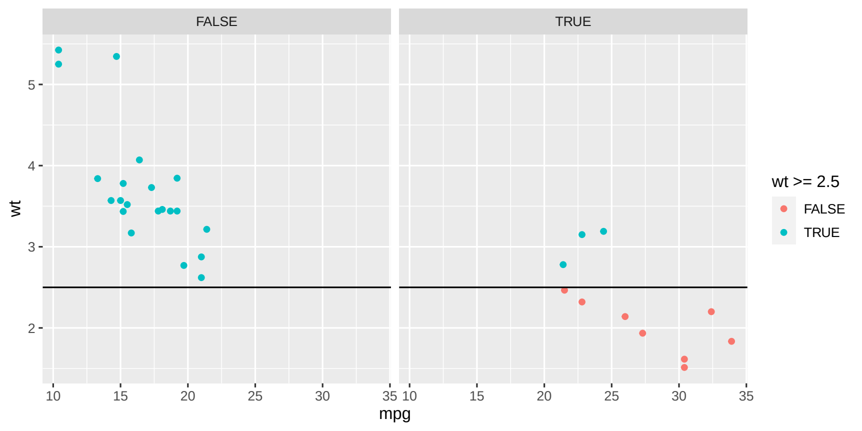
Task - Repetition - 3) - answers

```r
mtcars %>% ggplot(mapping = aes(x = mpg, y = wt, color = wt >= 2.5))+
  geom_point()+
  facet_wrap(~ cyl)+
  geom_hline(yintercept = 2.5)
```



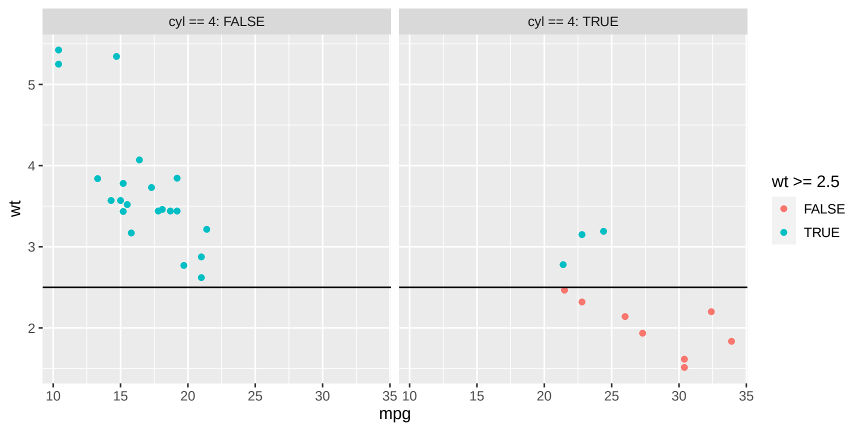
Task - Repetition - 4) - answers

```r
mtcars %>% ggplot(mapping = aes(x = mpg, y = wt, color = wt >= 2.5))+
  geom_point()+
  facet_wrap(~ cyl == 4)+
  geom_hline(yintercept = 2.5)
```



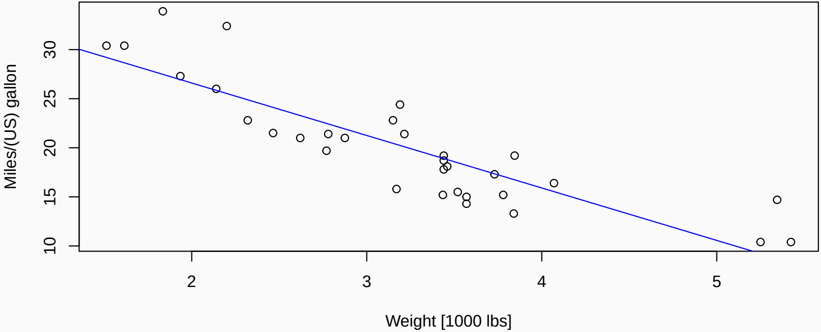
Task - Repetition - 5) - answers

```
mtcars %>% ggplot(mapping = aes(x = mpg, y = wt, color = wt >= 2.5))+
  geom_point()+
  facet_wrap(~ cyl == 4, labeller = (cyl = label_both))+
  geom_hline(yintercept = 2.5)
```



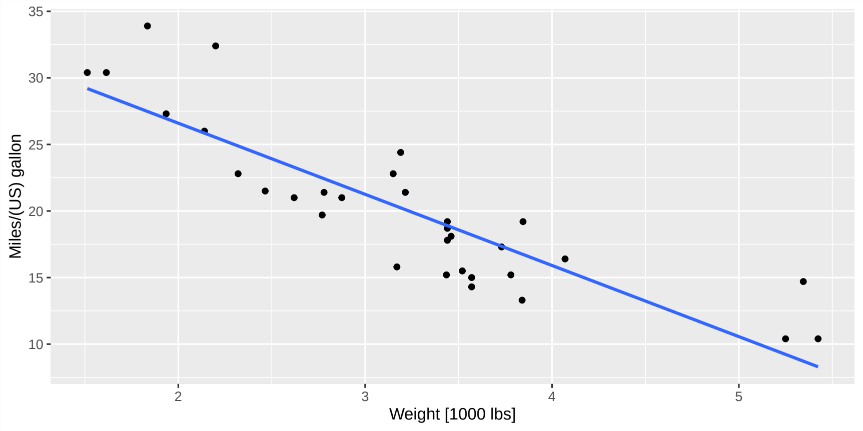
Task - Repetition - 6) in base R - answer

```r
plot(mtcars$wt, mtcars$mpg, xlab="Weight [1000 lbs]", ylab="Miles/(US) gallon")
wt_mpg <- lm(mpg ~ 1 + wt, data = mtcars)
abline(wt_mpg, col = "blue")
```



Task - Repetition - 6) - ggplot2 - answer

```
ggplot(data = mtcars, mapping = aes(x = wt, y = mpg))+
  geom_point()+
  geom_smooth(method = "lm", se = FALSE)+
  xlab("Weight [1000 lbs]")+ylab("Miles/(US) gallon")
```



More information

- Call the help page of the single functions → lots of arguments to specify
- DataCamp courses about ggplot2
  - *Introduction to data visualization with ggplot2*
  - *Intermediate data visualization with ggplot2*
- Just search the internet: diverse forums like stackoverflow help, e.g.: `https://stackoverflow.com/questions/30002257/change-color-median-line-ggplot-geom-boxplot`
- Click here: *Legends (ggplot2)* [accessed: May 11, 2022]
- Click here: *Interactive web-based data visualization with R, plotly, and shiny* by Sievert (2020)

For home

Customize the different plots shown here in this presentation (label the axes, add a title etc. if they are missing).

References i

Auguie, Baptiste. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. `https://CRAN.R-project.org/package=gridExtra`.

Meschiari, Stefano. 2022. *Latex2exp: Use LaTeX Expressions in Plots*. `https://CRAN.R-project.org/package=latex2exp`.

National Institute of Standards and Technology (NIST). 2021. *SI Units – Temperature*. USA, Gaithersburg. `https://www.nist.gov/pml/weights-and-measures/si-units-temperature`.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. `https://www.R-project.org/`.

RStudio, PBC. 2021. *Data Visualization with Ggplot2::CHEAT SHEET*. `https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf`.

Sievert, Carson. 2020. *Interactive Web-Based Data Visualization with r, Plotly, and Shiny*. Chapman; Hall/CRC. `https://plotly-r.com`.

References ii

Wei, Y. 2021. *Colors in r*. Department of Biostatistics, Columbia University. `http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf`.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. `https://ggplot2.tidyverse.org`.

Wickham, Hadley, and Maximilian Girlich. 2022. *Tidyr: Tidy Messy Data*. `https://CRAN.R-project.org/package=tidyr`.

Wilkinson, Leland. 2010. “The Grammar of Graphics.” *Wiley Interdisciplinary Reviews. Computational Statistics* 2 (6): 673–77.