R markdown main workfile

Basic stuff and data operations

Useful links and resources

!!! The most useful book for R markdown - "R Markdown: The Definitive Guide" Ebook "R for Graduate Students"
Ebook "R for Data Science"
Ebook "Statistical Inference via Data Science: A ModernDive into R and Tidyverse"
Gradient generator
ggthemes
R studio cheatsheets
Introduction to ggplot2
Themes for Bootstrap

Versions and plugins

To update R version: go to R GUI (old), run 1) install.packages("installr"); library(installr); choose installr option on the tab next to the option "Help", then update... voila!

Packages installation and loading:

```
getRversion()

install.packages(c("afex", "tidyverse", "emmeans", "writexl",
    "readxl", "ggthemes", "ggplot2", "extrafont", "reshape",
    "reshape2", "gcookbook", "readr", "swirl", "ggpubr", "lubridate",
    "rmarkdown", "formatR", "knitr", "magrittr", "glue", "viridis",
    "ggsci", "gcookbook"))

install.packages("plotly")
```

```
library(afex)
library(emmeans)
library(tidyverse)
library(writexl)
library(readxl)
library(ggthemes)
library(extrafont)
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.1.2
```

```
library(reshape2)
library(gcookbook)
library(readr)
library(swirl)
library(ggpubr)
library(ggthemes)
library(lubridate)
library(rmarkdown)
library(formatR)
library(knitr)
library(magrittr)
library(glue)
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.1.2
```

```
library(ggsci)
```

```
## Warning: package 'ggsci' was built under R version 4.1.2
```

```
library(gcookbook)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.1.2
```

```
font_import()  # for extrafont package
loadfonts(device = "win")  # loading fonts from extrafont
```

Functionality

- Ctrl + shift + R : creates headers for your R script
- Alt + shift + K : displays all programmed keyboard shortcuts for R
- Ctrl + shift + M : write "%>%" (tidyverse pipe)

Basic commands

- Save as .csv: `write_csv(name, "filename.csv")`
- Import data from .txt file: `read.csv("filename.txt", sep="\t", header=FALSE)`
- read.csv("filename.csv") # import data from csv file
- ironmansuits <- read.csv("C:/Users/evgen/Desktop/R_work_folder/Iron_Man_Suits.csv", stringsAsFactors=TRUE)
- mutate() # creates new columns or modifies current variables in the dataset
- diamonds %>% mutate(depth_times_price = depth * price)
- diamonds %>% mutate(depth_times_price = depth * price, is_cut_ideal = cut == "Ideal")
- analog of f-string in R: library(glue): `glue("Variable {varname}")`
- Export graph as .tiff: `ggsave("test.tiff", units="in", width=8, height=6, dpi=300, compression = 'lzw')`

Operators AND (&), OR (|)

- diamonds %>% filter(cut == "Ideal", carat == "0.23", price > 400)
- select() # gives us specific columns
- diamonds %>% select (cut, price)
- arrange() # arranges the values in data frame by values in a variable
- diamonds %>% arrange(price)
- diamonds %>% arrange(desc(price))
- diamonds %>% arrange(price, carat)
- group_by() / summarise()
- diamonds %>% group_by(carat) %>% summarise(mean(price), count = n())
- diamonds %>% group_by(carat) %>% mutate(mean_price = mean(price))
- runif() # generates random deviates of uniform distribution

Describing data

- str # describe the structure of dataset by type of variable
- mean, median, max, min, sd
- dim(my_data) ## show dimensions of a table
- dim(my_data)
- summary(my_data)
- summary(my_data$chrom)
- ?ggplot
- seq(1,10)
- data() # show a list of built-in datasets

Datasets

Pre-loaded datasets:

```
data()  # check a list of pre-loaded datasets
```

```
# Create brand-new dataset
data <- data.frame(Chr = c("Chr1", "Chr2", "Chr2", "Chr3", "Chr4",
    "Chr11"), Type = c("1_Promoter", "1_Promoter", "2_Enhancer",
    "3_Activator", "1_Promoter", "3_Activator"), Value = c(1e+09,
    5e+10, 6e+09, 9e+10, 8e+10, 5e+10))
data$Chr = gsub("Chr", "", data$Chr)  # remove 'Chr' from column 1
data$Chr <- factor(data$Chr, levels = c(seq(1, 11), "X", "Y"))  # reorder the chromosomes numerically


# Create dataset from existing
diamonds2 <- tibble(diamonds$cut, diamonds$clarity)
names(diamonds2)[1:4] <- c("Cut", "Clarity")  # renames the first 4 columns
```

```
## Warning: The `value` argument of `names<-` must have the same length as `x` as of tibble 3.0.0.
## `names` must have length 2, not 4.
```

```
# Add new columns
diamonds2 <- diamonds %>%
    mutate(price200 = price - 200, pricepercarat = price/carat)
```

```r
# Update the in-table columns with new values
diamonds %>%
    mutate(cut = recode(cut, Ideal = "IDEAL", Good = "GOOD"))
```

Filtering

- `diamonds %>% filter(cut=="Fair" | cut=="Good", price <= 600)`
- `recode () # modifies the values within a variable`
- `filter () # gives us specific rows`
- `population2 <- population %>% filter(country == 'Brazil' | country == 'Argentina', year >= 2005)`
- `df$col1 == "name 1"`
- `df$col1 %in% c("name1", "name2")`

Data processing

- Rename df columns by name: `df.rename(columns={"A": "a", "B": "c"})`
- Rename df columns by index: `df.rename(index={0: "x", 1: "y", 2: "z"})`

Information below about tidy data is taken from this link.

Dataset - a collection of quantitative and qualitative values. Every value belongs to a variable and an observation.

Untidy (messy) data - rows and columns are mixed up with observations and variables.

Tidy data - a standard way of mapping the meaning of a dataset to its structure. Variables are in columns and observations - in rows. Tidy data makes it easy to extract variables.

Examples:

Untidy (messy) data - 6 observations but only 3 rows - 'wide' dataset:

```r
# define data
df <- data.frame(. = c("John", "Jane", "Mary"), Treatment_A = c(16,
    3, 2), Treatment_B = c(0, 4, 5), Treatment_C = c(0, 8, 10))
df
```

```
##      . Treatment_A Treatment_B Treatment_C
## 1 John          16           0           0
## 2 Jane           3           4           8
## 3 Mary           2           5          10
```

Tidy data structure - three variables - Person/name, Treatment, and Result - 'long/tall' dataset - made by melting the untidy one:

```r
df <- data.frame(`Person/name` = c("John", "Jane", "Mary", "John",
    "Jane", "Mary", "John", "Jane", "Mary"), Treatment = c("a",
    "a", "a", "b", "b", "b", "c", "c", "c"), Result = c(16, 3,
    2, 0, 4, 5, 0, 8, 10))
df
```

```
##   Person.name Treatment Result
## 1        John         a     16
## 2        Jane         a      3
## 3        Mary         a      2
## 4        John         b      0
## 5        Jane         b      4
## 6        Mary         b      5
## 7        John         c      0
## 8        Jane         c      8
## 9        Mary         c     10
```

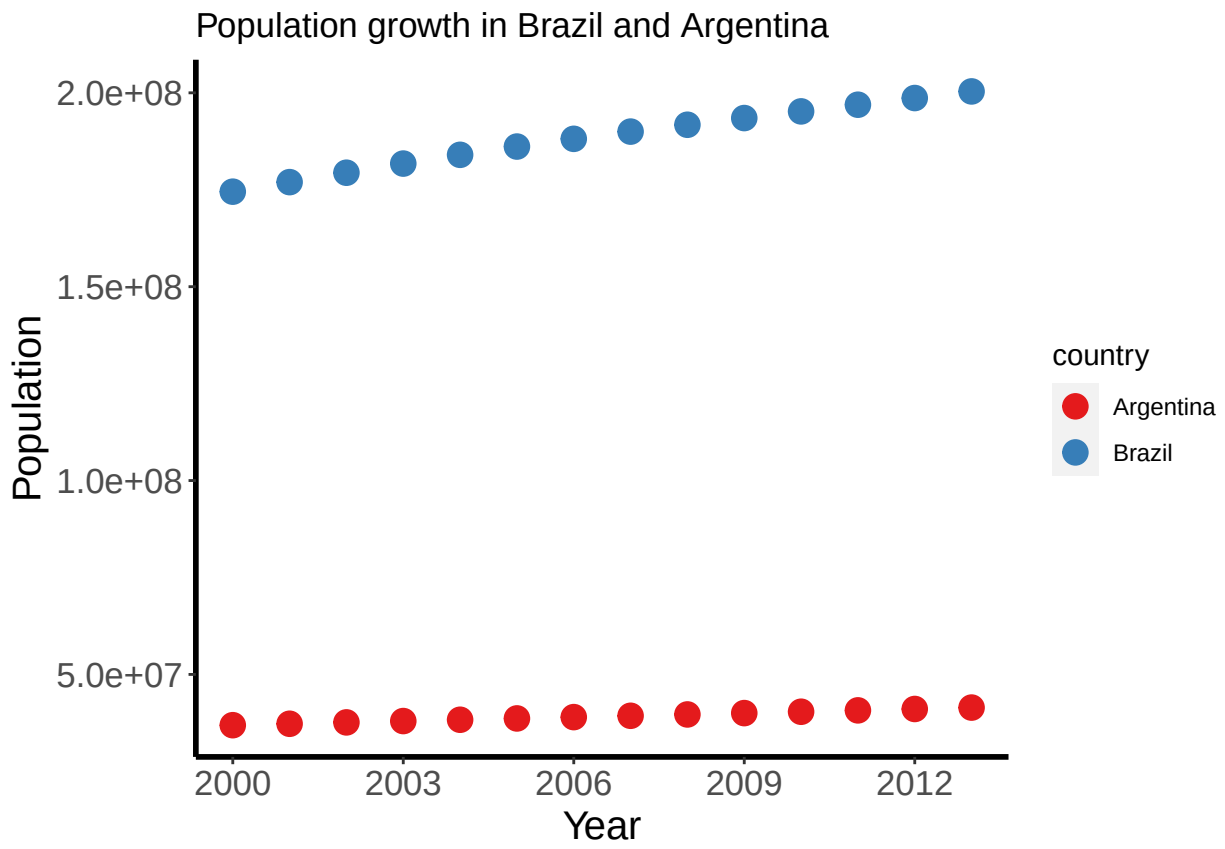
Graphs and Figures

By convention, independent variable is on X axis, and dependent variable is on Y axis.

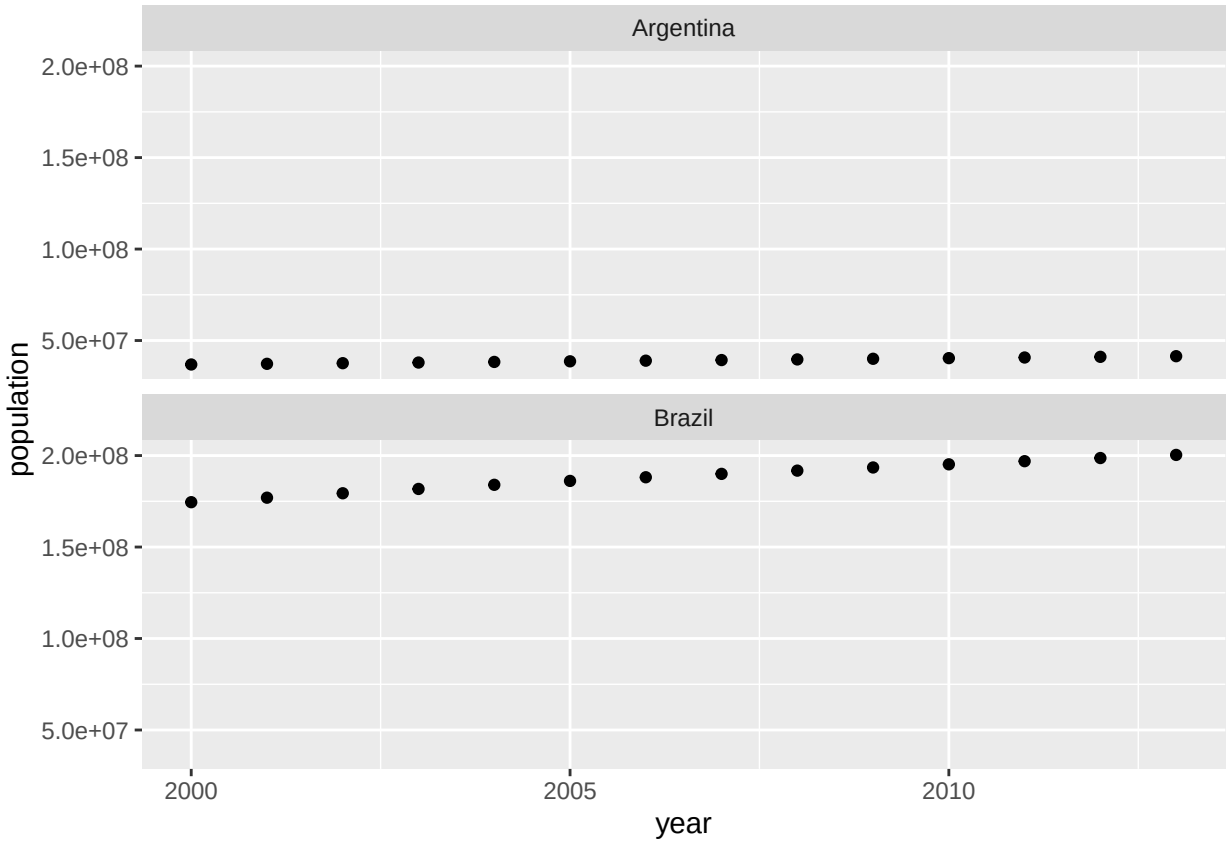
Dotplots

Dataset 'population'

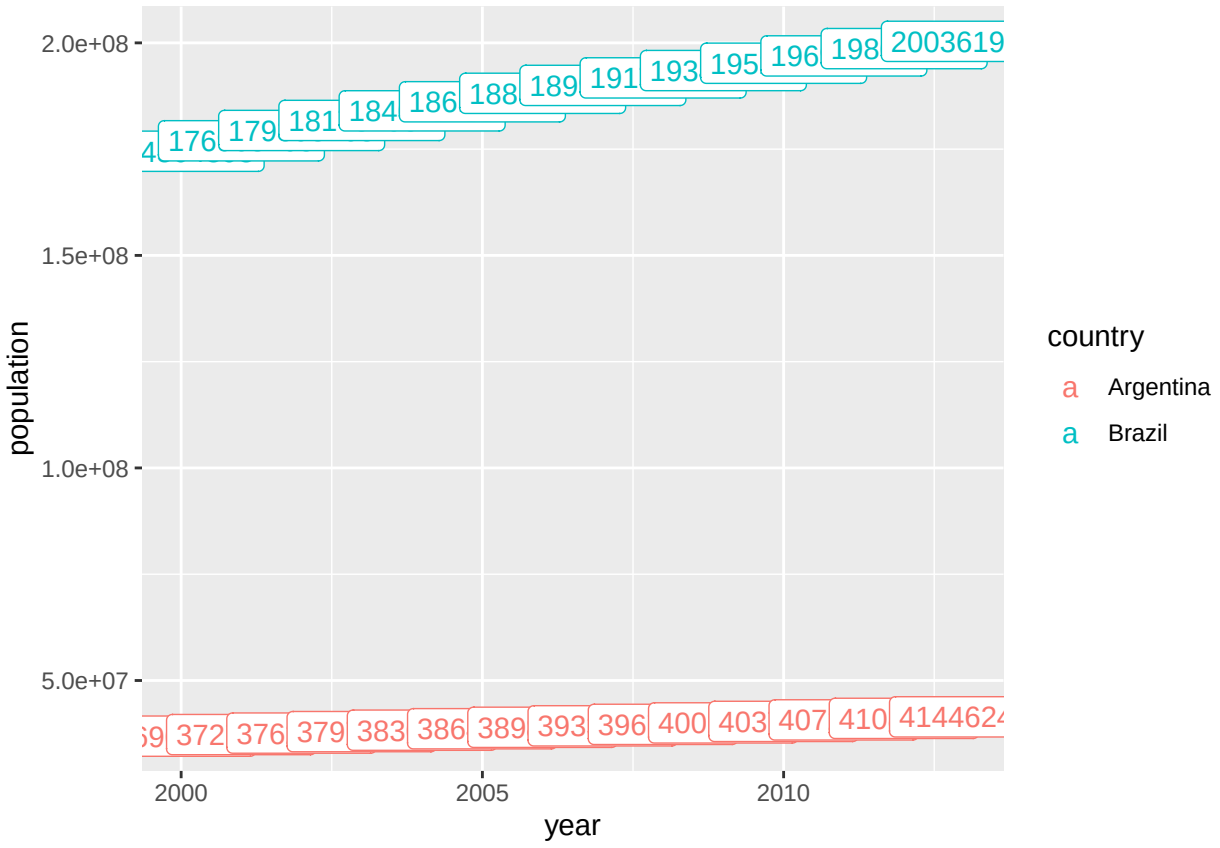
```
# Filtering a sample dataset 'population'
population2 <- population %>%
    filter(country == "Brazil" | country == "Argentina", year >=
        2000)
# population2

# Simple dotplot
ggplot(population2, aes(x = year, y = population, color = country)) +
    geom_point(size = 4) + scale_x_continuous(breaks = (seq(min(population2$year),
    max(population2$year), by = 3))) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), axis.text = element_text(size = 13),
    axis.title = element_text(size = 15), panel.background = element_blank(),
    axis.line = element_line(size = 1)) + scale_color_brewer(palette = "Set1") +
    labs(title = "Population growth in Brazil and Argentina",
        x = "Year", y = "Population")
```




```
# Faceted dotplot
ggplot(population2, aes(x = year, y = population)) + geom_point() +
    facet_wrap(~country, nrow = 2)
```




```
# Dotplot with callouts
ggplot(population2, aes(x = year, y = population, color = country,
    label = population)) + geom_text(check_overlap = TRUE) +
    geom_label()
```

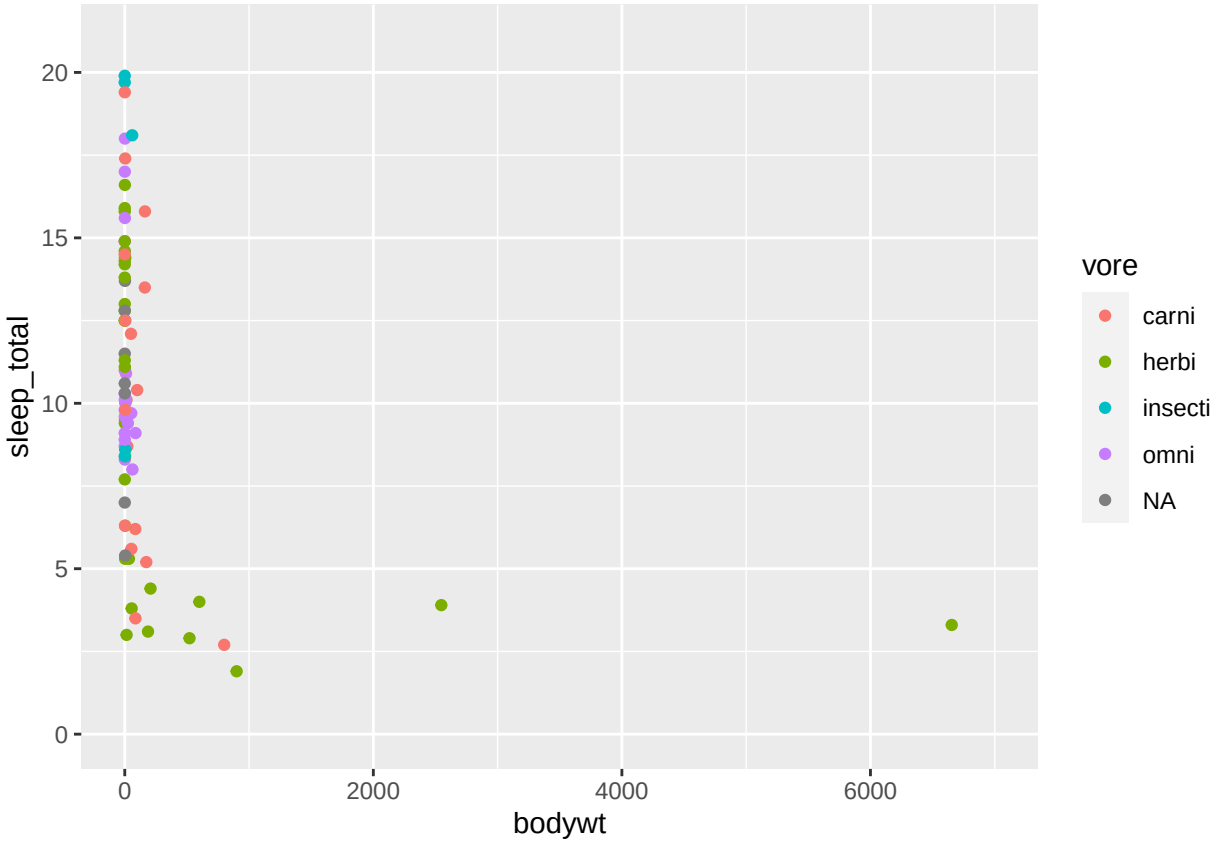


Scatterplots

```
msleep
```

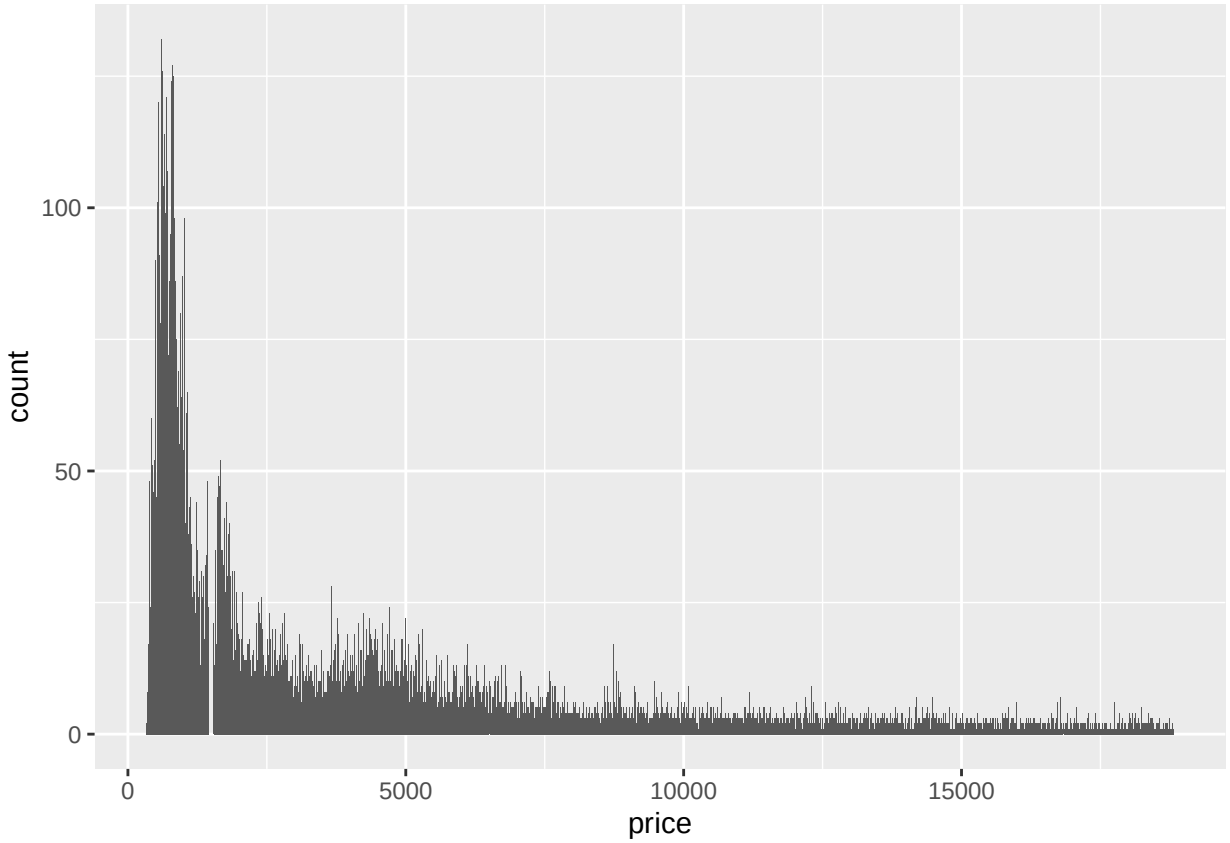
```
## # A tibble: 83 x 11
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Cheet~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
##  2 Owl m~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
##  3 Mount~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA       9.6
##  4 Great~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133   9.1
##  5 Cow    Bos   herbi Arti~ domesticated         4         0.7       0.667  20
##  6 Three~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767   9.6
##  7 North~ Call~ carni Carn~ vu                   8.7       1.4       0.383  15.3
##  8 Vespe~ Calo~ <NA>  Rode~ <NA>                 7        NA        NA      17
##  9 Dog    Canis carni Carn~ domesticated        10.1       2.9       0.333  13.9
## 10 Roe d~ Capr~ herbi Arti~ lc                   3        NA        NA      21
## # ... with 73 more rows, and 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# Scatterplot with df msleep
ggplot(msleep, aes(x = bodywt, y = sleep_total, colour = vore)) +
    geom_point() + xlim(0, 7000) + ylim(0, 21)
```

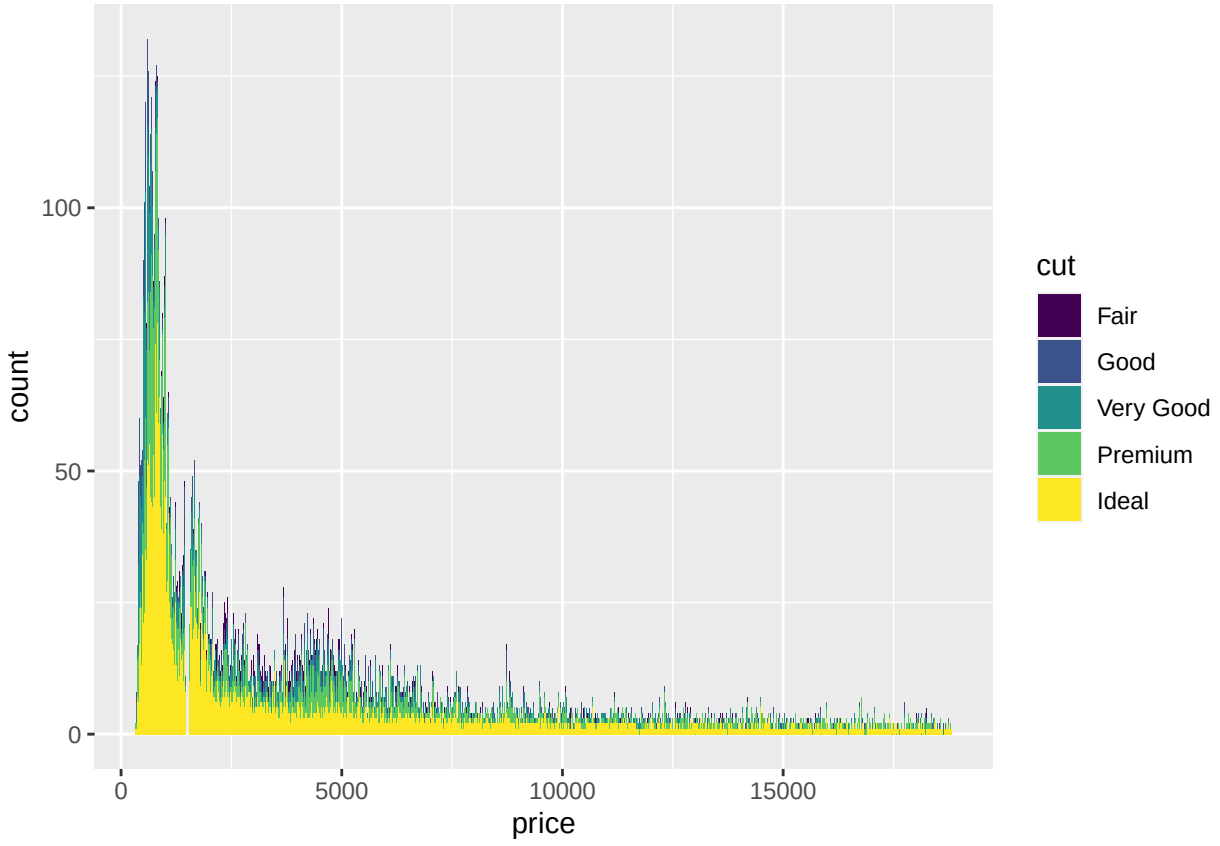


Histograms

```
ggplot(diamonds, aes(x = price)) + geom_bar()
```



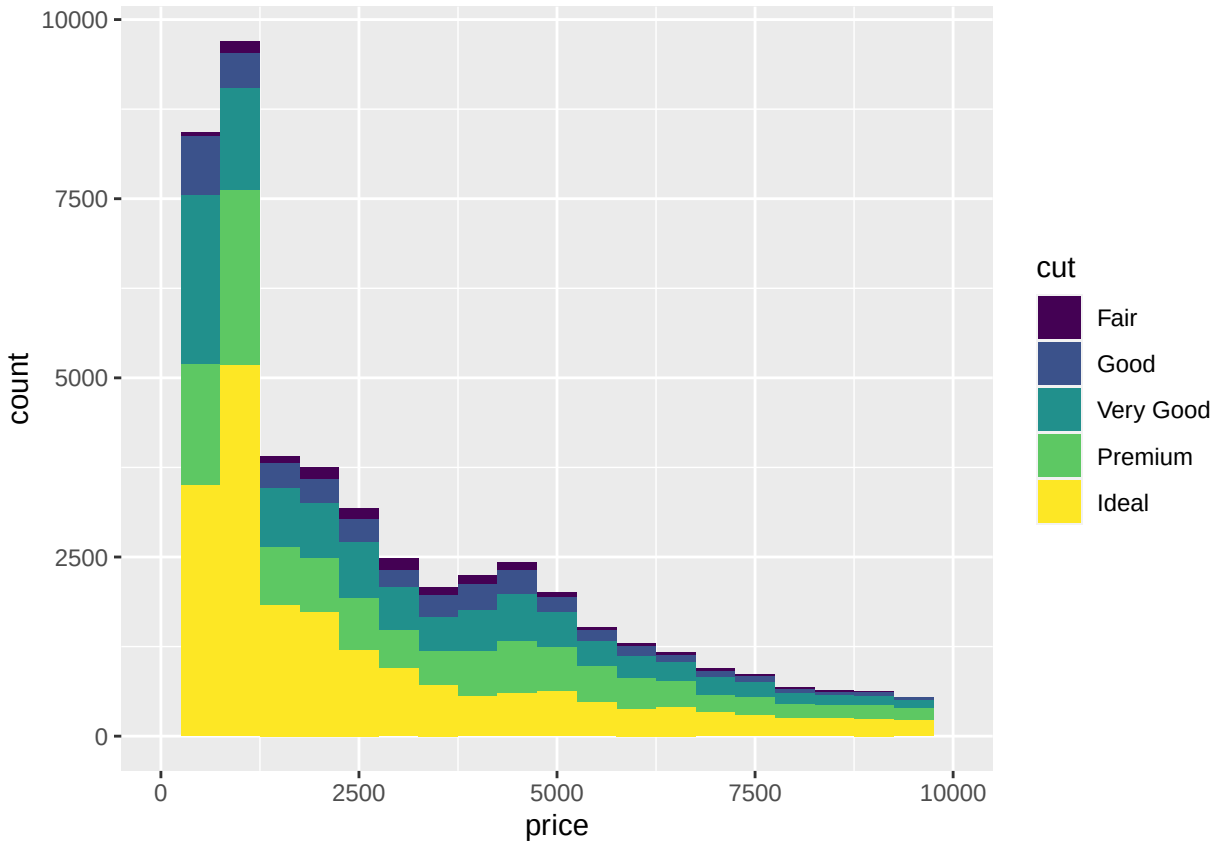

```
ggplot(diamonds, aes(x = price, fill = cut)) + geom_bar()
```




```
ggplot(diamonds, aes(x = price, fill = cut)) + geom_histogram(binwidth = 500) +
    xlim(0, 10000)
```

```
## Warning: Removed 5222 rows containing non-finite values (stat_bin).
```

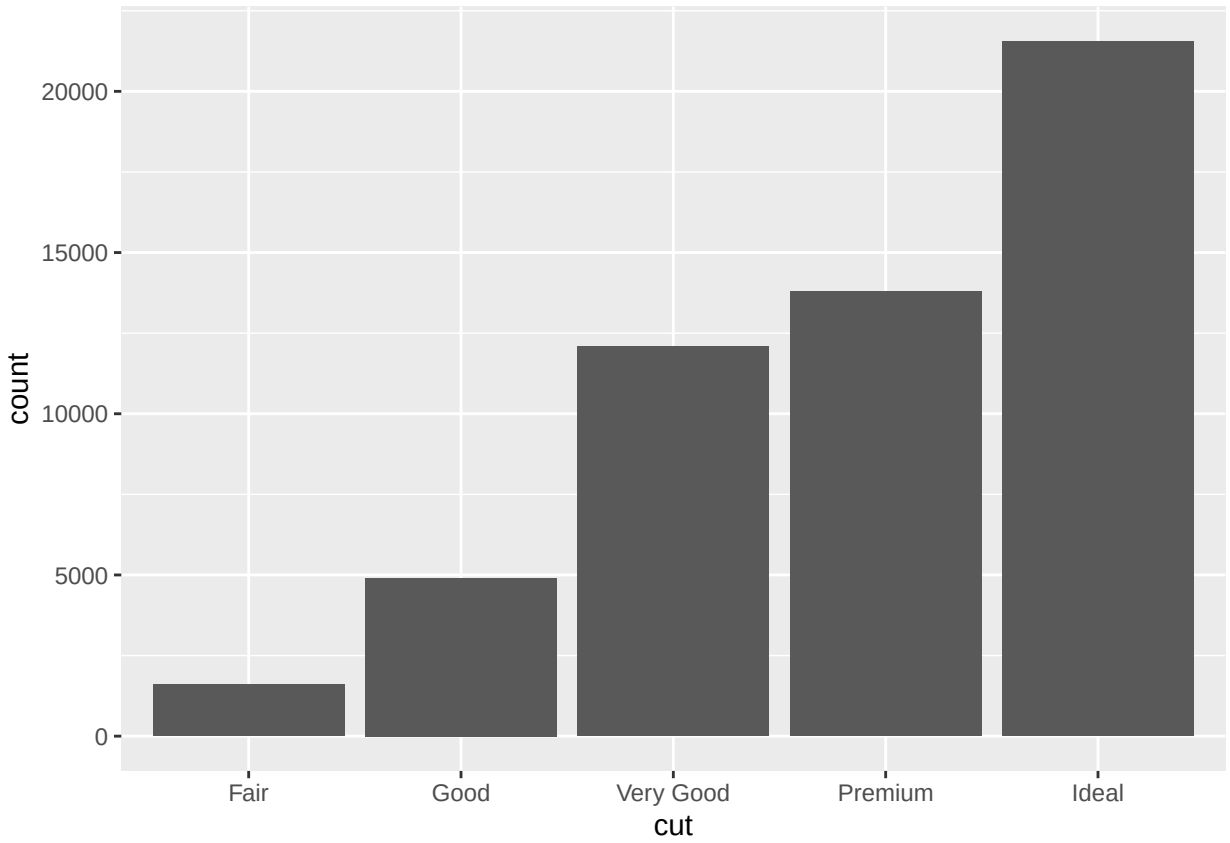
```
## Warning: Removed 10 rows containing missing values (geom_bar).
```



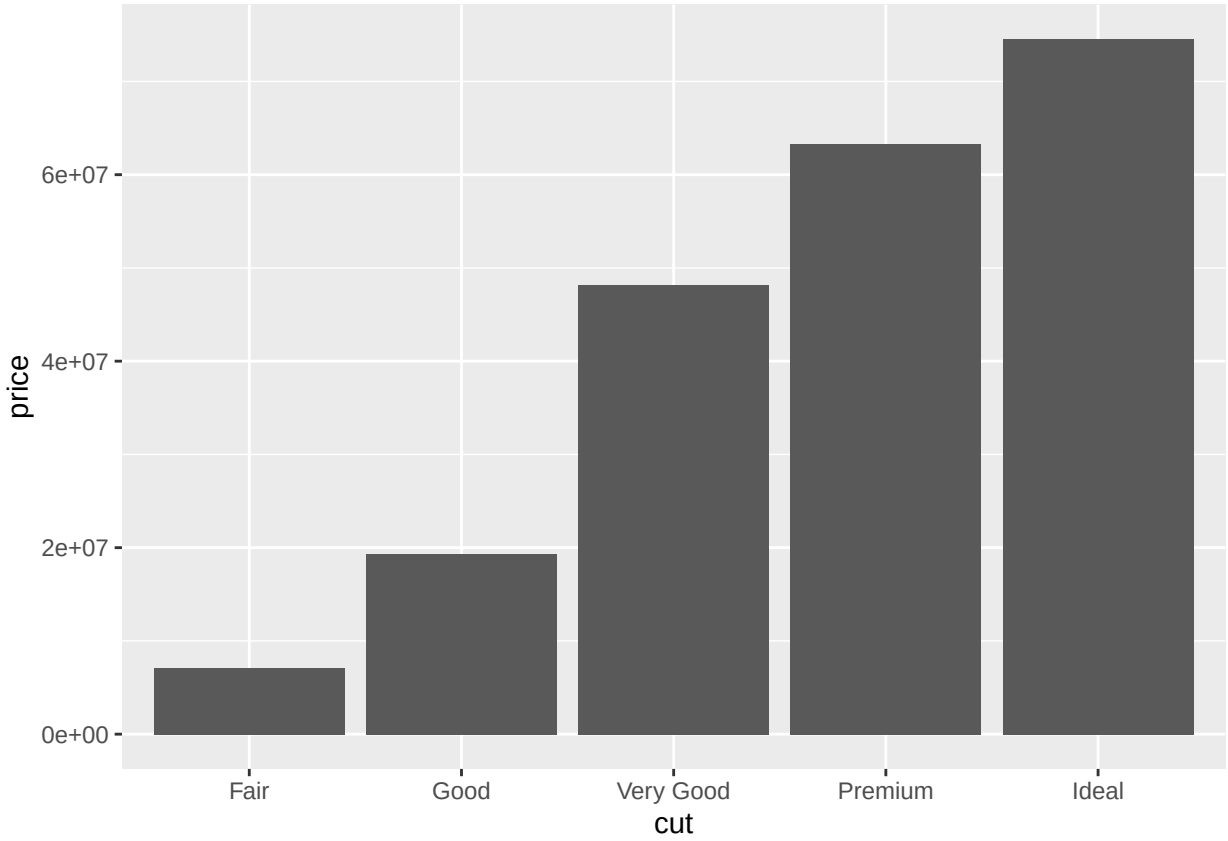
Bar graphs

For geom_bar, aggregate numbers of rows for each x-value can be counted by default with argument stat=count. Alternatively, the values can be provided manually for the bar graph by stat="identity".

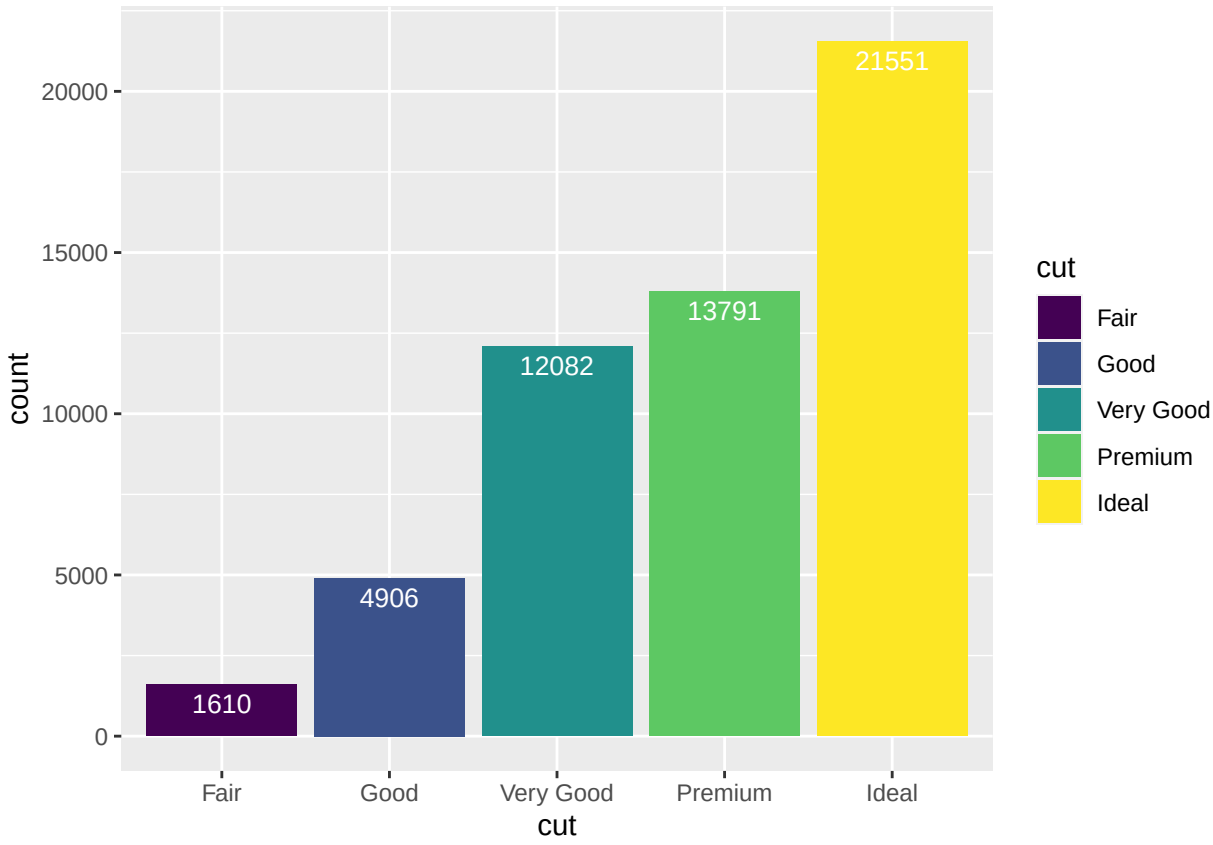
```
ggplot(diamonds, aes(x = cut)) + geom_bar()
```



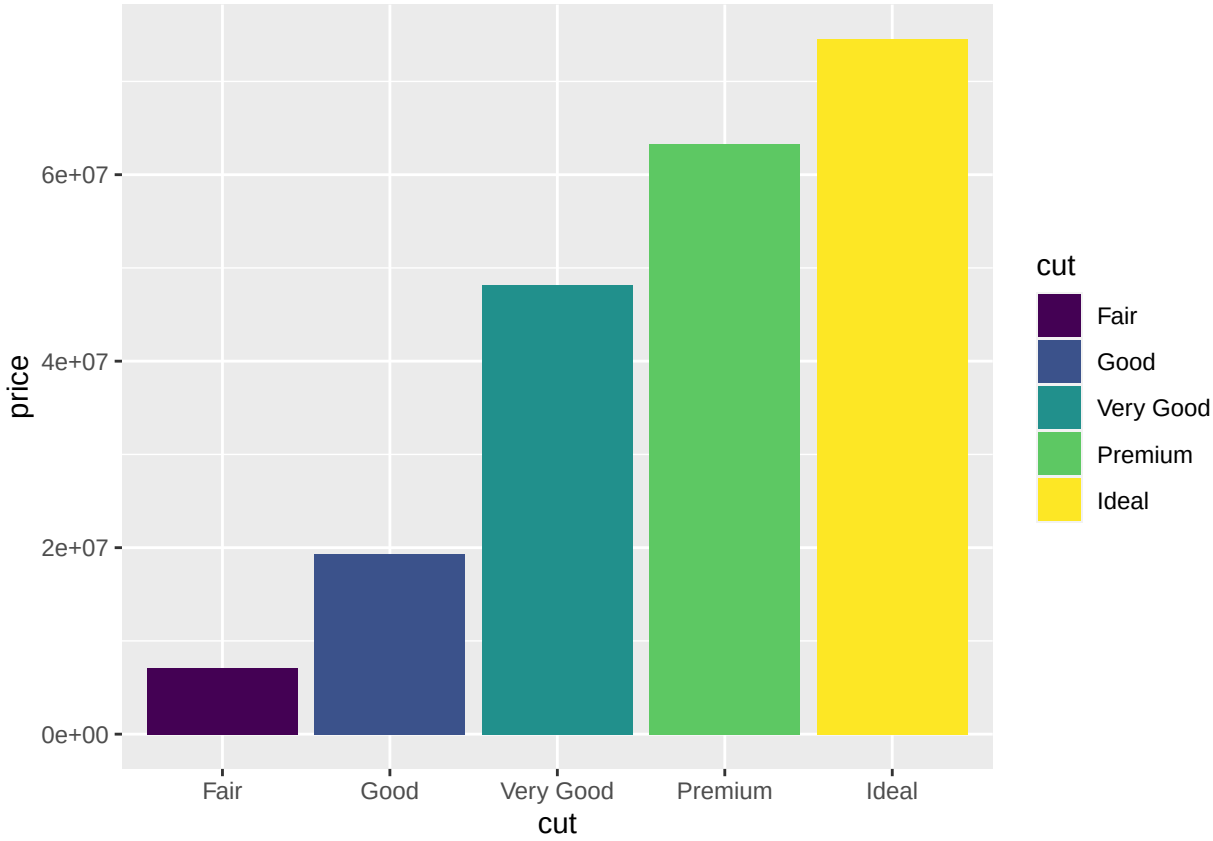

```
ggplot(diamonds, aes(x = cut, y = price)) + geom_bar(stat = "identity")
```




```
ggplot(diamonds, aes(x = cut, fill = cut)) + geom_bar() + geom_text(stat = "count",
    aes(label = ..count..), vjust = 1.5, size = 3.5, colour = "white")
```

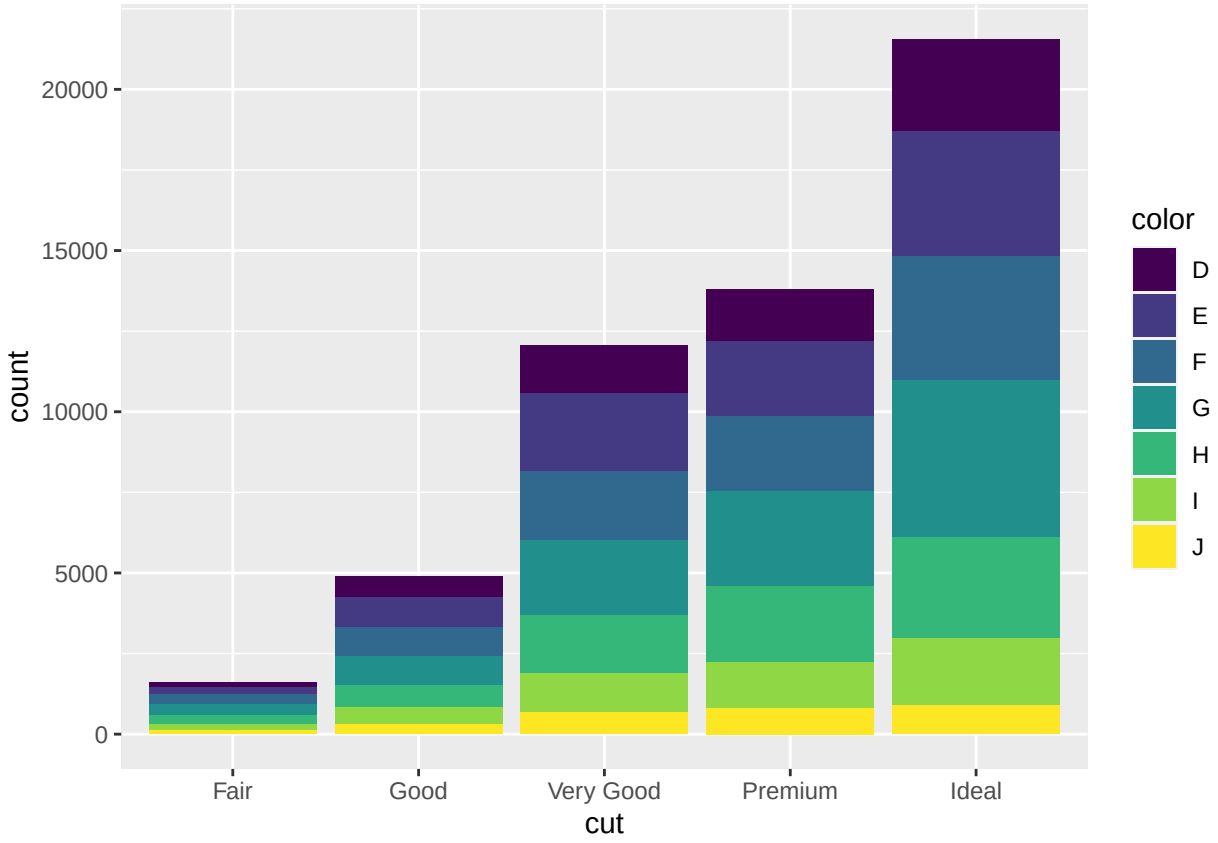



```
ggplot(diamonds, aes(x = cut, y = price, fill = cut)) + geom_col()
```

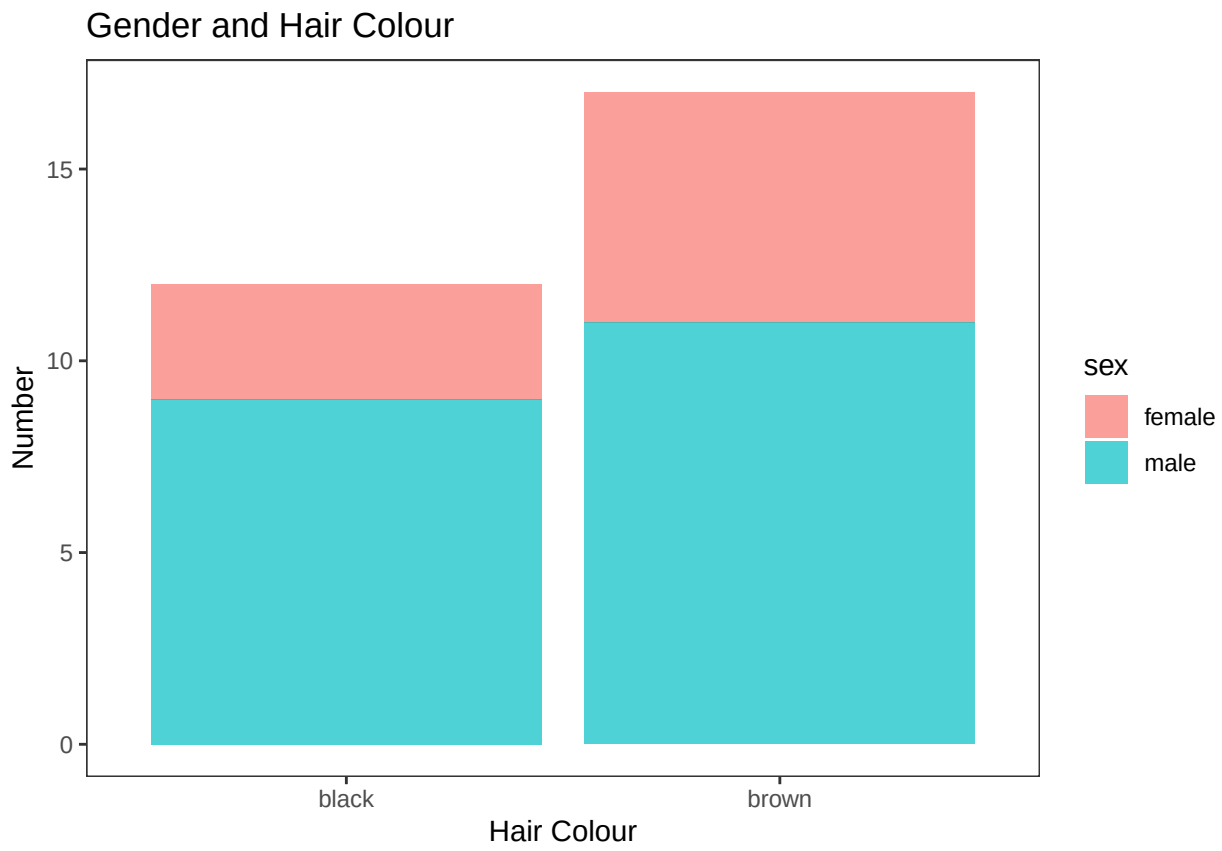


Stacked component bar graphs

```
ggplot(diamonds, aes(x=cut, fill=color)) +
  geom_bar()
```




```
starwars %>%
  filter(hair_color == "black" |       # | = OR
           hair_color == "brown") %>%
  drop_na(sex) %>%
  ggplot(aes(x = hair_color, fill = sex)) +
  geom_bar(alpha = 0.7) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(title = "Gender and Hair Colour",
       x = "Hair Colour",
       y = "Number")
```

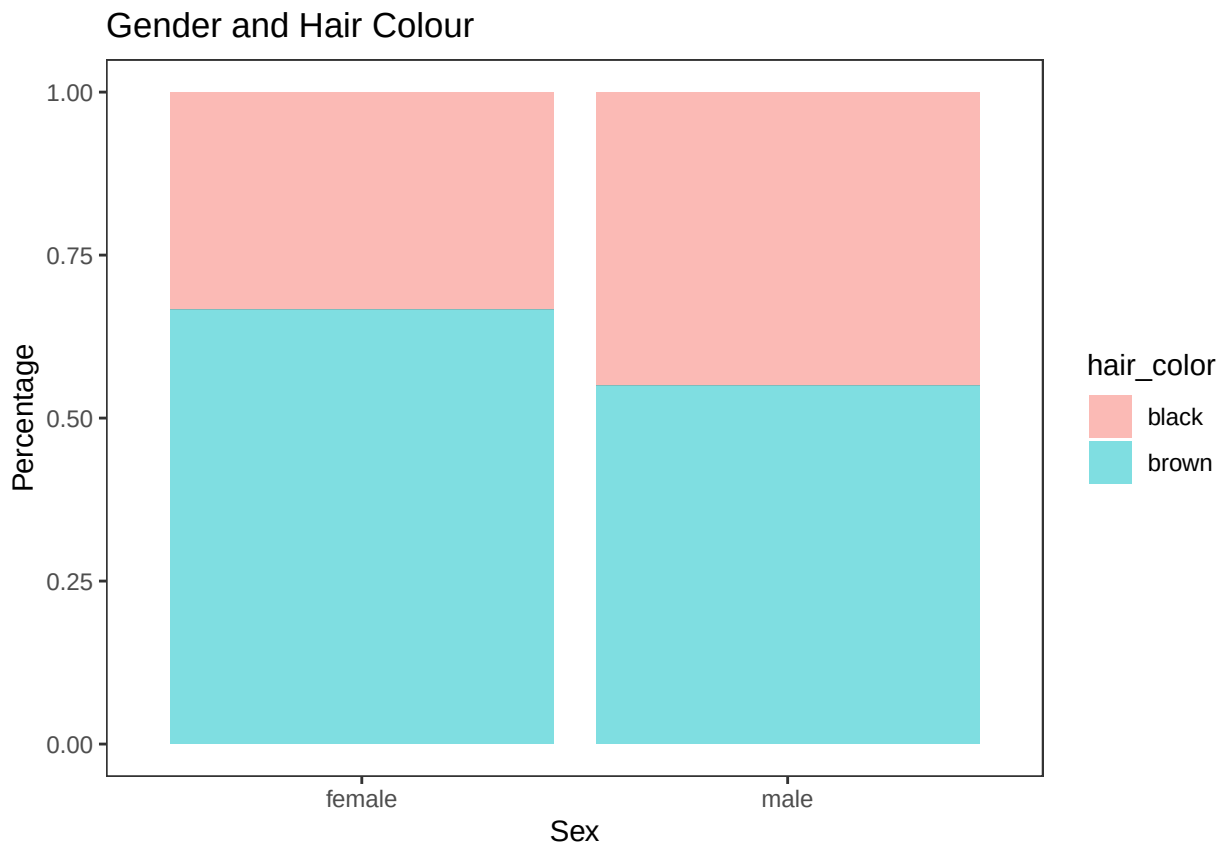


Stacked percentage bar graphs

```
library(reshape)

# Example 1
starwars
```

```
## # A tibble: 87 x 14
##    name    height  mass hair_color  skin_color eye_color birth_year sex   gender
##    <chr>    <int> <dbl> <chr>       <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke S~    172    77 blond       fair       blue            19   male  mascu~
##  2 C-3PO      167    75 <NA>        gold       yellow         112   none  mascu~
##  3 R2-D2       96    32 <NA>        white, bl~ red             33   none  mascu~
##  4 Darth ~    202   136 none        white      yellow          41.9 male  mascu~
##  5 Leia O~    150    49 brown       light      brown           19   fema~ femin~
##  6 Owen L~    178   120 brown, grey light      blue            52   male  mascu~
##  7 Beru W~    165    75 brown       light      blue            47   fema~ femin~
##  8 R5-D4       97    32 <NA>        white, red red             NA   none  mascu~
##  9 Biggs ~    183    84 black       light      brown           24   male  mascu~
## 10 Obi-Wa~    182    77 auburn, wh~ fair       blue-gray       57   male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

```r
starwars %>%
  filter(hair_color == "black" |       # | = OR
           hair_color == "brown") %>%
  drop_na(sex) %>%
  ggplot(aes(x = sex, fill = hair_color)) +
  geom_bar(position = "fill",
           alpha = 0.5) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  labs(title = "Gender and Hair Colour",
       x = "Sex",
       y = "Percentage")
```

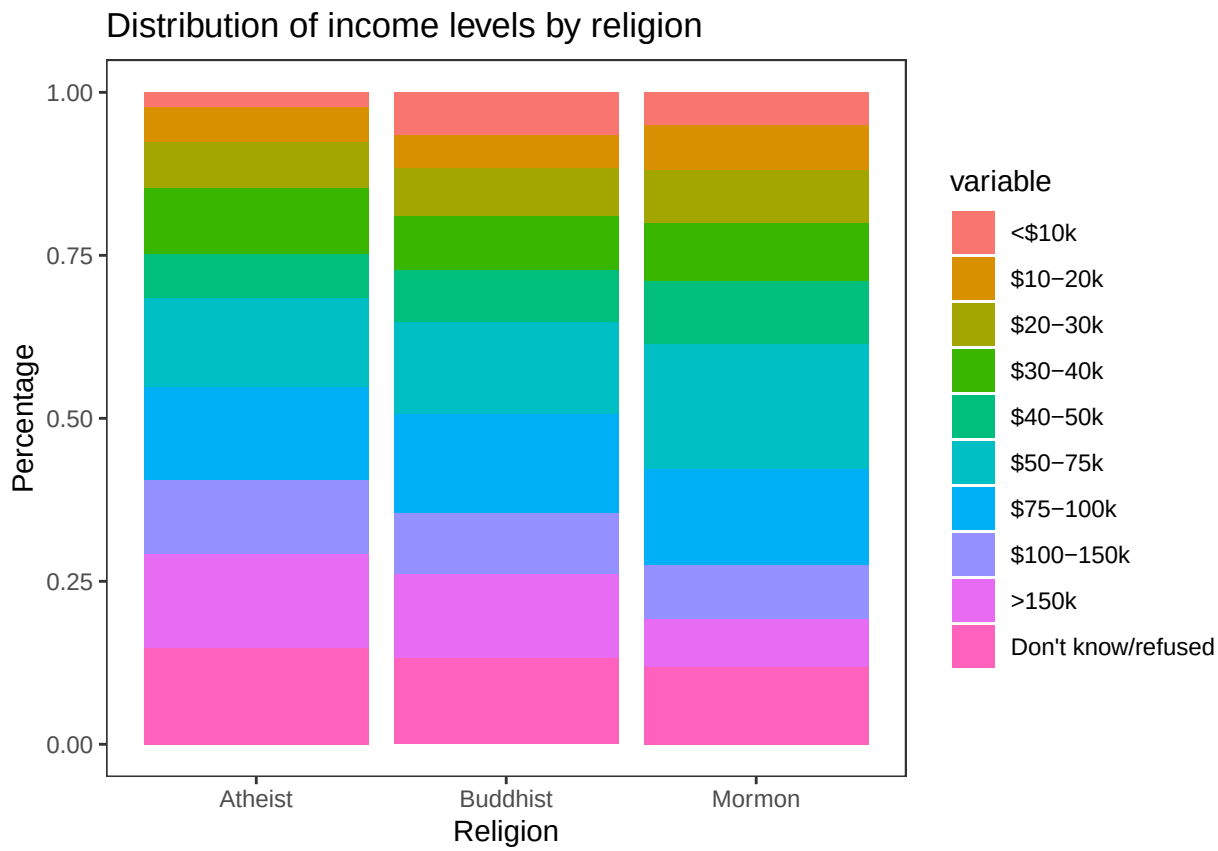



```r
# Example 2
relig_income
```

```
## # A tibble: 18 x 11
##    religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k` `$50-75k` `$75-100k`
##    <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
##  1 Agnostic      27        34        60        81        76       137        122
##  2 Atheist       12        27        37        52        35        70         73
##  3 Buddhist      27        21        30        34        33        58         62
##  4 Catholic     418       617       732       670       638      1116        949
##  5 Don't k~      15        14        15        11        10        35         21
```

```
##  6 Evangel~     575       869      1064       982       881      1486       949
##  7 Hindu          1         9         7         9        11        34        47
##  8 Histori~     228       244       236       238       197       223       131
##  9 Jehovah~      20        27        24        24        21        30        15
## 10 Jewish        19        19        25        25        30        95        69
## 11 Mainlin~     289       495       619       655       651      1107       939
## 12 Mormon        29        40        48        51        56       112        85
## 13 Muslim         6         7         9        10         9        23        16
## 14 Orthodox      13        17        23        32        32        47        38
## 15 Other C~       9         7        11        13        13        14        18
## 16 Other F~      20        33        40        46        49        63        46
## 17 Other W~       5         2         3         4         2         7         3
## 18 Unaffil~     217       299       374       365       341       528       407
## # ... with 3 more variables: $100-150k <dbl>, >150k <dbl>,
## #   Don't know/refused <dbl>
```

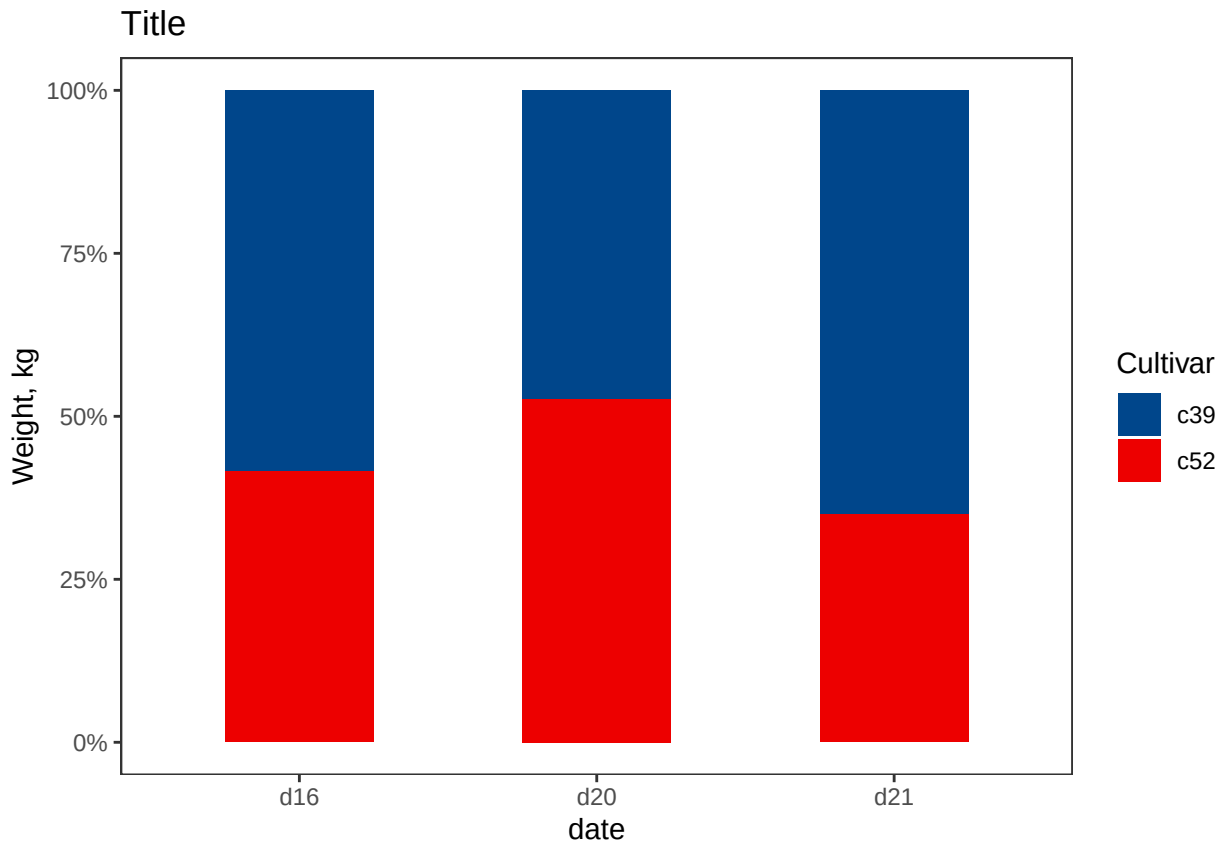
```
relig_income %>%
  filter(religion == "Atheist" |
         religion == "Mormon" |
         religion == "Buddhist") %>%
  melt(id.vars = "religion") %>%
  group_by(religion, variable) %>%
  ggplot(aes(x = religion, y = value, fill = variable)) +
  geom_bar(stat="identity", position = "fill") +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_text()) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Distribution of income levels by religion",
       x = "Religion",
       y = "Percentage")
```




```
# library gcookbook
cabbage_exp
```

```
##   Cultivar Date Weight        sd  n         se
## 1      c39  d16   3.18 0.9566144 10 0.30250803
## 2      c39  d20   2.80 0.2788867 10 0.08819171
## 3      c39  d21   2.74 0.9834181 10 0.31098410
## 4      c52  d16   2.26 0.4452215 10 0.14079141
## 5      c52  d20   3.11 0.7908505 10 0.25008887
## 6      c52  d21   1.47 0.2110819 10 0.06674995
```

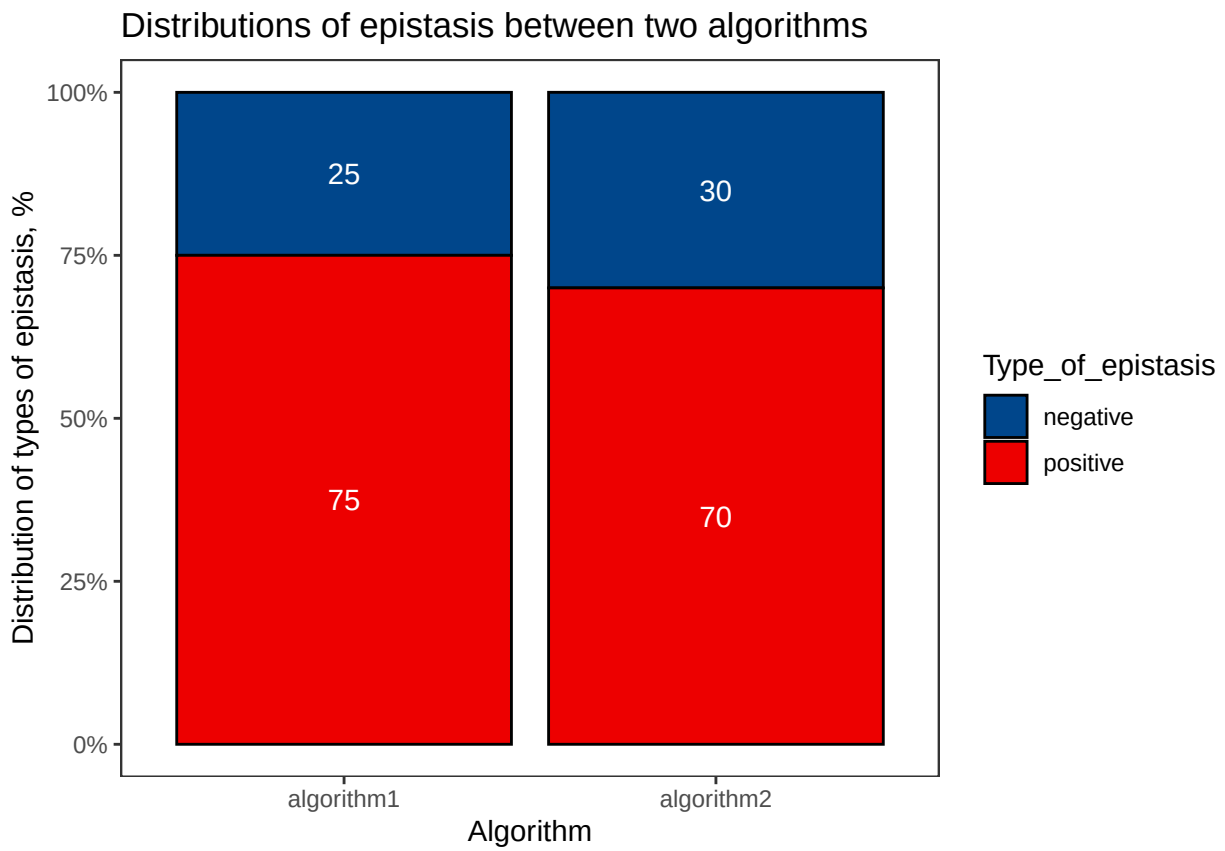
```
ggplot(cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar)) +
  geom_col(position = "fill", width=0.5) +
  scale_y_continuous(labels = scales::percent) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
    scale_fill_lancet() + # module ggsci - palettes from sci journals
  labs(title = "Title",
       x = "date",
       y = "Weight, kg")
```




```
# Custom dataset
df <- data.frame (Type_of_epistasis = c('negative', 'positive', 'negative', 'positive'),
                  n = c(250, 750, 300, 700),
                  algorithm = c('algorithm1', 'algorithm1', 'algorithm2', 'algorithm2'),
                  percentage = c(0.25, 0.75, 0.3, 0.7),
                  percentage_100 = c(25, 75, 30, 70))
df
```

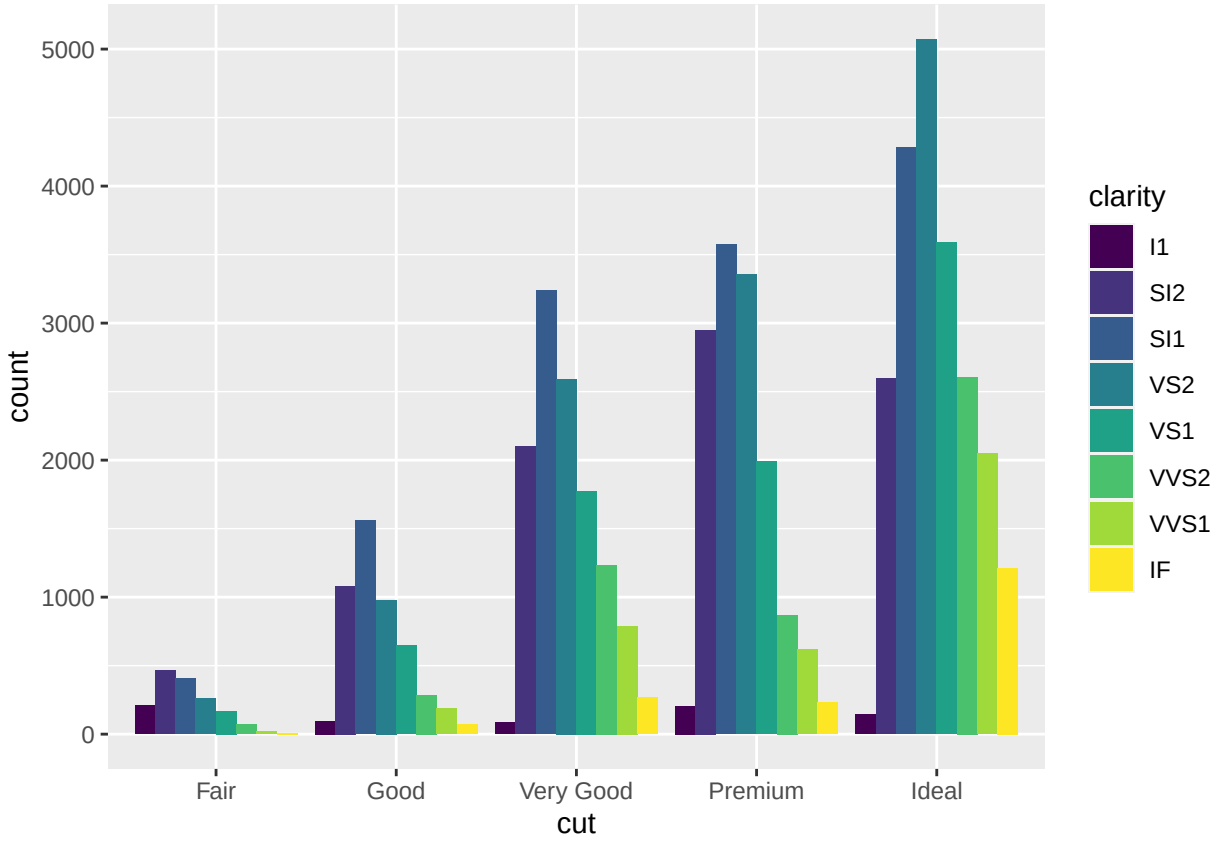
```
##   Type_of_epistasis   n  algorithm percentage percentage_100
## 1          negative 250 algorithm1       0.25             25
## 2          positive 750 algorithm1       0.75             75
## 3          negative 300 algorithm2       0.30             30
## 4          positive 700 algorithm2       0.70             70
```

```
ggplot(df, aes(x=algorithm, y=percentage, fill=Type_of_epistasis)) +
  geom_col(position='fill', colour='black') +
  scale_y_continuous(labels = scales::percent) +
  geom_text(aes(label = percentage_100), size = 4, position=position_stack(vjust=0.5), color='white') +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  scale_fill_lancet() + # module ggsci - palettes from sci journals
  labs(title = "Distributions of epistasis between two algorithms",
       x = "Algorithm",
       y = "Distribution of types of epistasis, %")
```

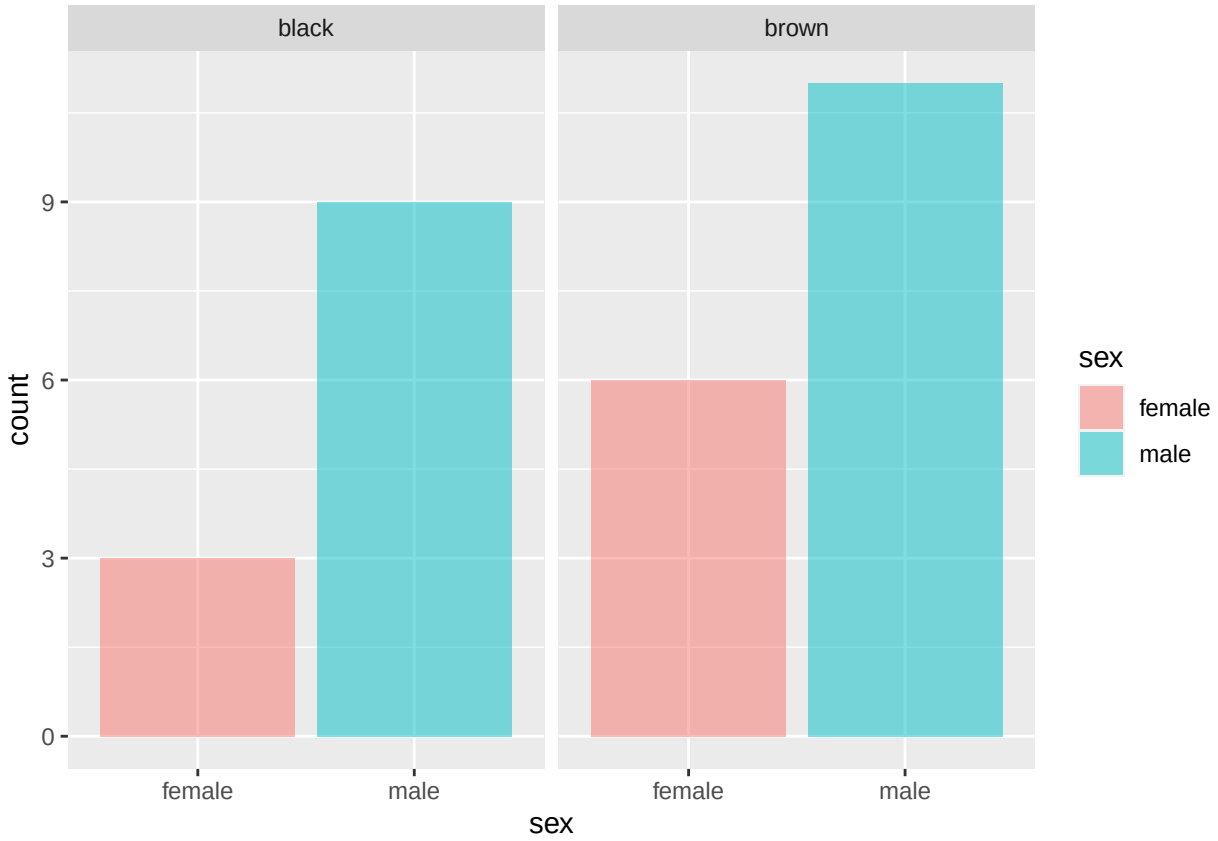


Grouped bar graphs

```
ggplot(diamonds, aes(x = cut, fill = clarity)) + geom_bar(position = "dodge")
```




```
# grouped facet
starwars %>%
    filter(hair_color %in% c("black", "brown")) %>%
    drop_na(sex) %>%
    ggplot(aes(sex)) + geom_bar(aes(fill = sex), alpha = 0.5) +
    facet_wrap(~hair_color)
```




```
# grouped bar graphs with error bars
head(ToothGrowth)
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

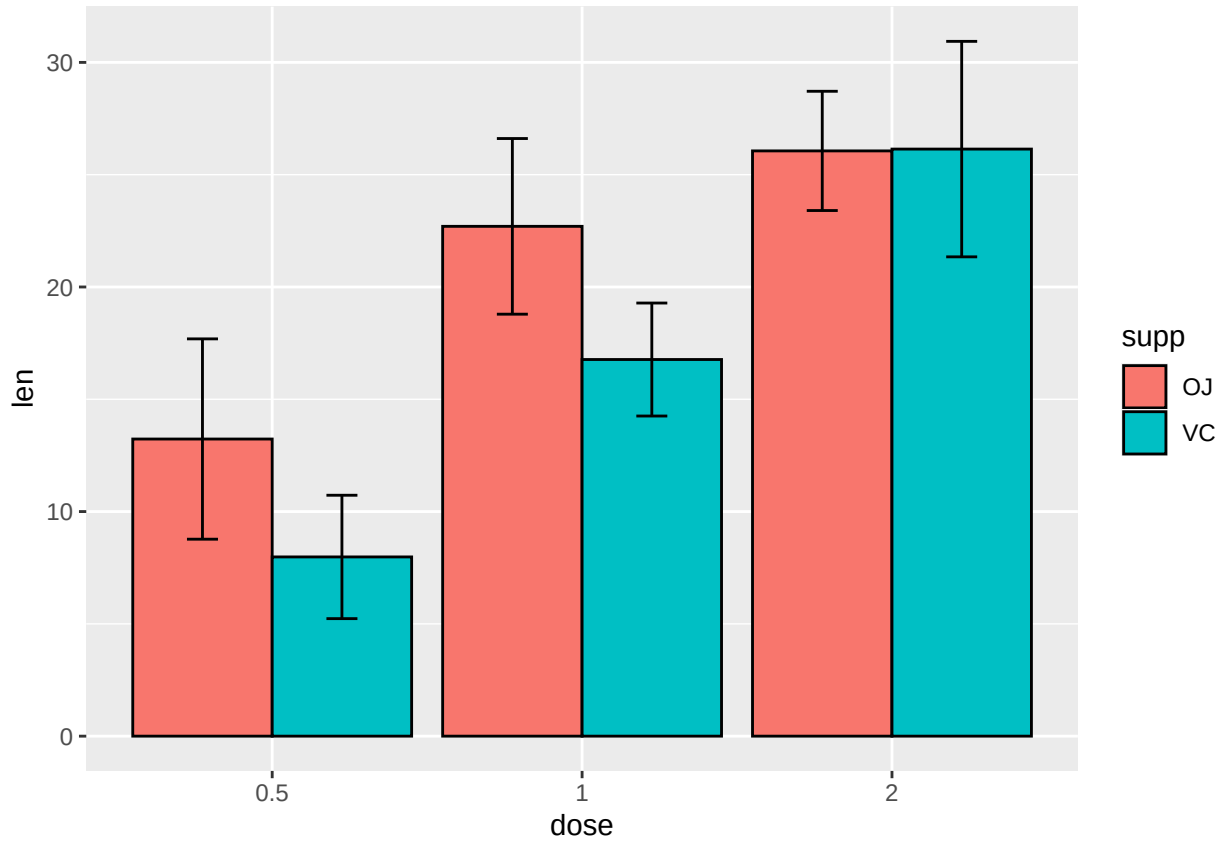
```
df <- ToothGrowth
df$dose <- as.factor(df$dose)
data_summary <- function(data, varname, groupnames) {
    require(plyr)
    summary_func <- function(x, col) {
        c(mean = mean(x[[col]], na.rm = TRUE), sd = sd(x[[col]],
            na.rm = TRUE))
    }
    data_sum <- ddply(data, groupnames, .fun = summary_func,
        varname)
    data_sum <- rename(data_sum, c(mean = varname))
    return(data_sum)
}
df2 <- data_summary(ToothGrowth, varname = "len", groupnames = c("supp",
    "dose"))
```

```
## Loading required package: plyr

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise

## The following object is masked from 'package:ggpubr':
##
##     mutate

## The following objects are masked from 'package:reshape':
##
##     rename, round_any

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

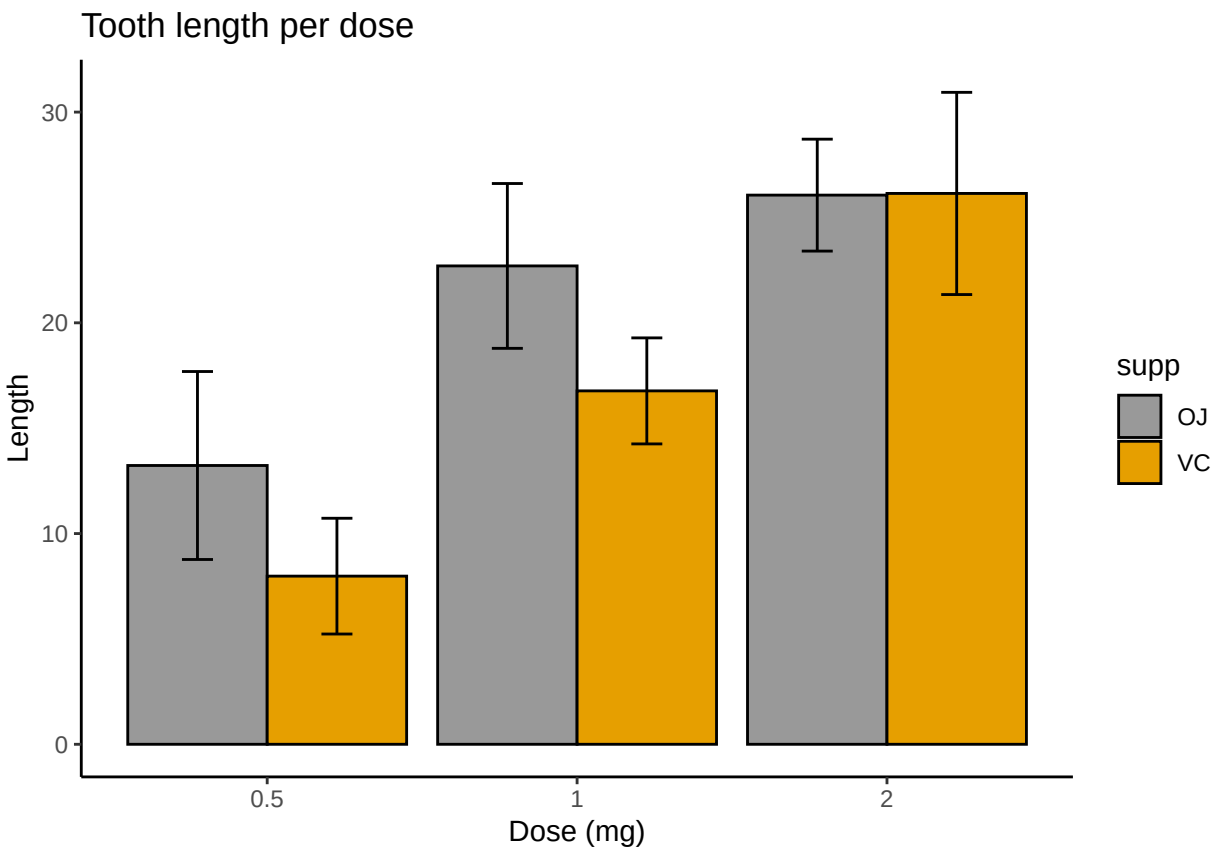
```
## Convert dose to a factor variable
df2$dose = as.factor(df2$dose)
head(df2)
```

```
##   supp dose   len       sd
## 1   OJ  0.5 13.23 4.459709
## 2   OJ    1 22.70 3.910953
## 3   OJ    2 26.06 2.655058
## 4   VC  0.5  7.98 2.746634
## 5   VC    1 16.77 2.515309
## 6   VC    2 26.14 4.797731
```

```
p <- ggplot(df2, aes(x = dose, y = len, fill = supp)) + geom_bar(stat = "identity",
    color = "black", position = position_dodge()) + geom_errorbar(aes(ymin = len -
    sd, ymax = len + sd), width = 0.2, position = position_dodge(0.9))
print(p)
```

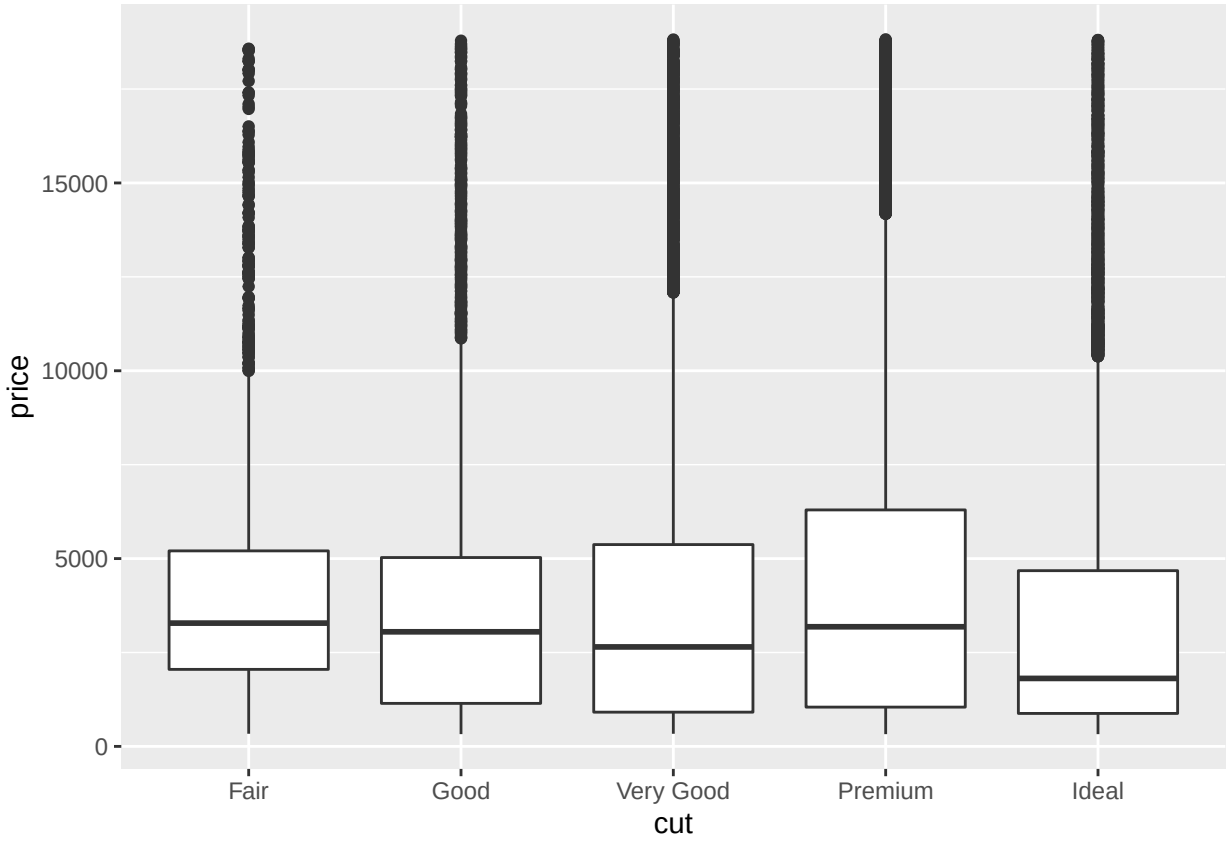



```
## Finished bar plot
p + labs(title = "Tooth length per dose", x = "Dose (mg)", y = "Length") +
    theme_classic() + scale_fill_manual(values = c("#999999",
    "#E69F00"))
```

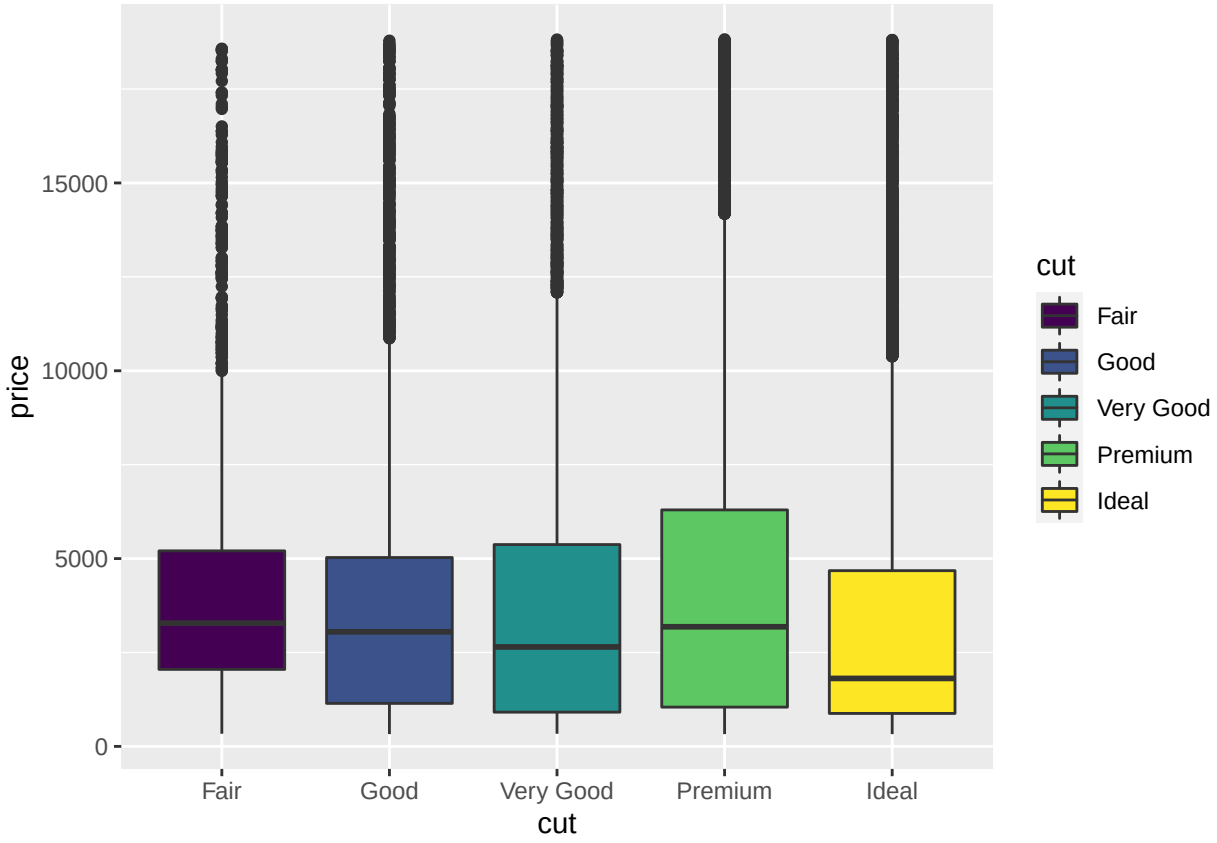


Boxplots

```
ggplot(diamonds, aes(x = cut, y = price)) + geom_boxplot()
```




```
ggplot(diamonds, aes(x = cut, y = price, fill = cut)) + geom_boxplot()
```



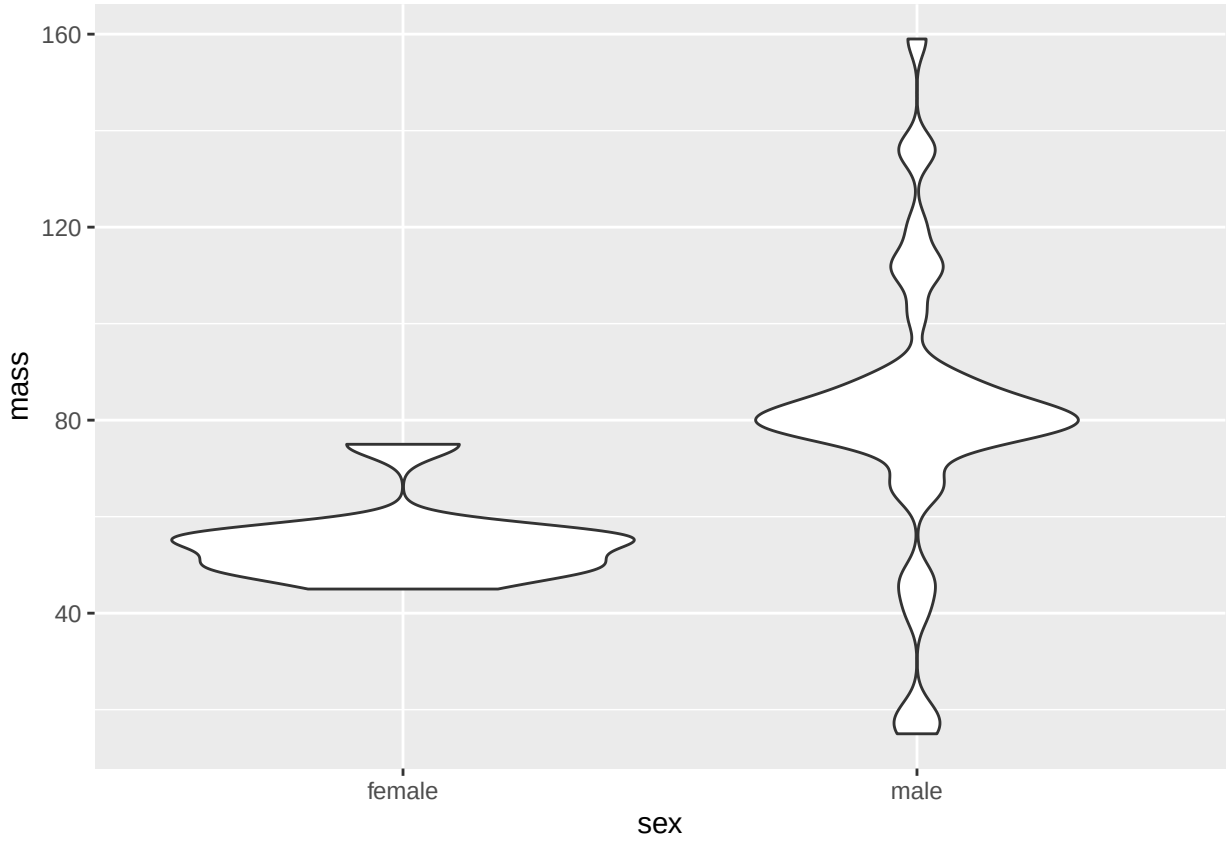
Violin plots

```
starwars
```

```
## # A tibble: 87 x 14
##    name    height  mass hair_color  skin_color eye_color birth_year sex   gender
##    <chr>    <int> <dbl> <chr>       <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke S~    172    77 blond       fair       blue            19   male  mascu~
##  2 C-3PO      167    75 <NA>        gold       yellow         112   none  mascu~
##  3 R2-D2       96    32 <NA>        white, bl~ red             33   none  mascu~
##  4 Darth ~    202   136 none        white      yellow          41.9 male  mascu~
##  5 Leia O~    150    49 brown       light      brown           19   fema~ femin~
##  6 Owen L~    178   120 brown, grey light      blue            52   male  mascu~
##  7 Beru W~    165    75 brown       light      blue            47   fema~ femin~
##  8 R5-D4       97    32 <NA>        white, red red             NA   none  mascu~
##  9 Biggs ~    183    84 black       light      brown           24   male  mascu~
## 10 Obi-Wa~    182    77 auburn, wh~ fair       blue-gray       57   male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

```
starwars %>%
  filter (sex %in% c('male', 'female')) %>%
  ggplot(aes(x=sex, y=mass)) + geom_violin()
```

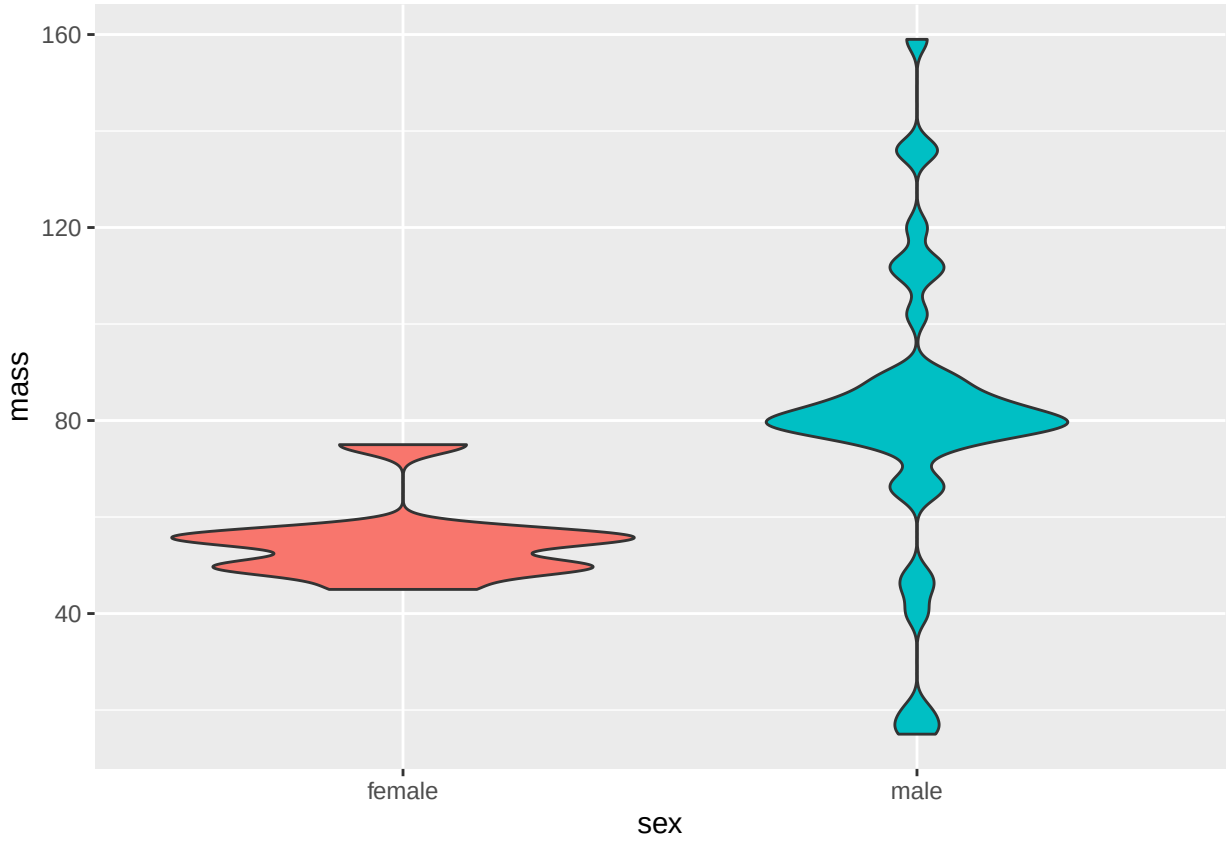
```
## Warning: Removed 23 rows containing non-finite values (stat_ydensity).
```




```
starwars %>%
  filter (sex %in% c('male', 'female')) %>%
  ggplot(aes(x=sex, y=mass, fill=sex)) +
  geom_violin(adjust=0.7) + # default adjust is 1. Lower means finer resolution
  guides(fill=FALSE) # make the entries coloured but delete the legend
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

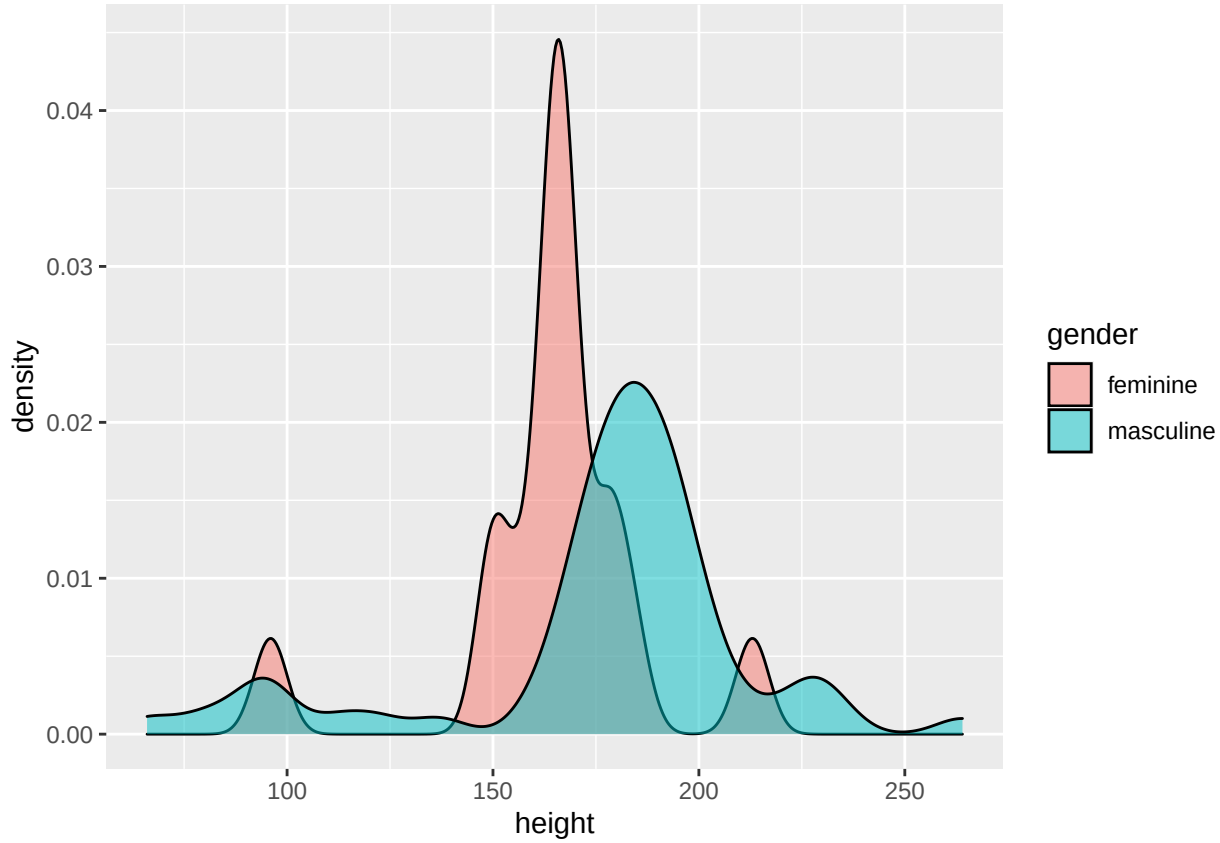
```
## Warning: Removed 23 rows containing non-finite values (stat_ydensity).
```



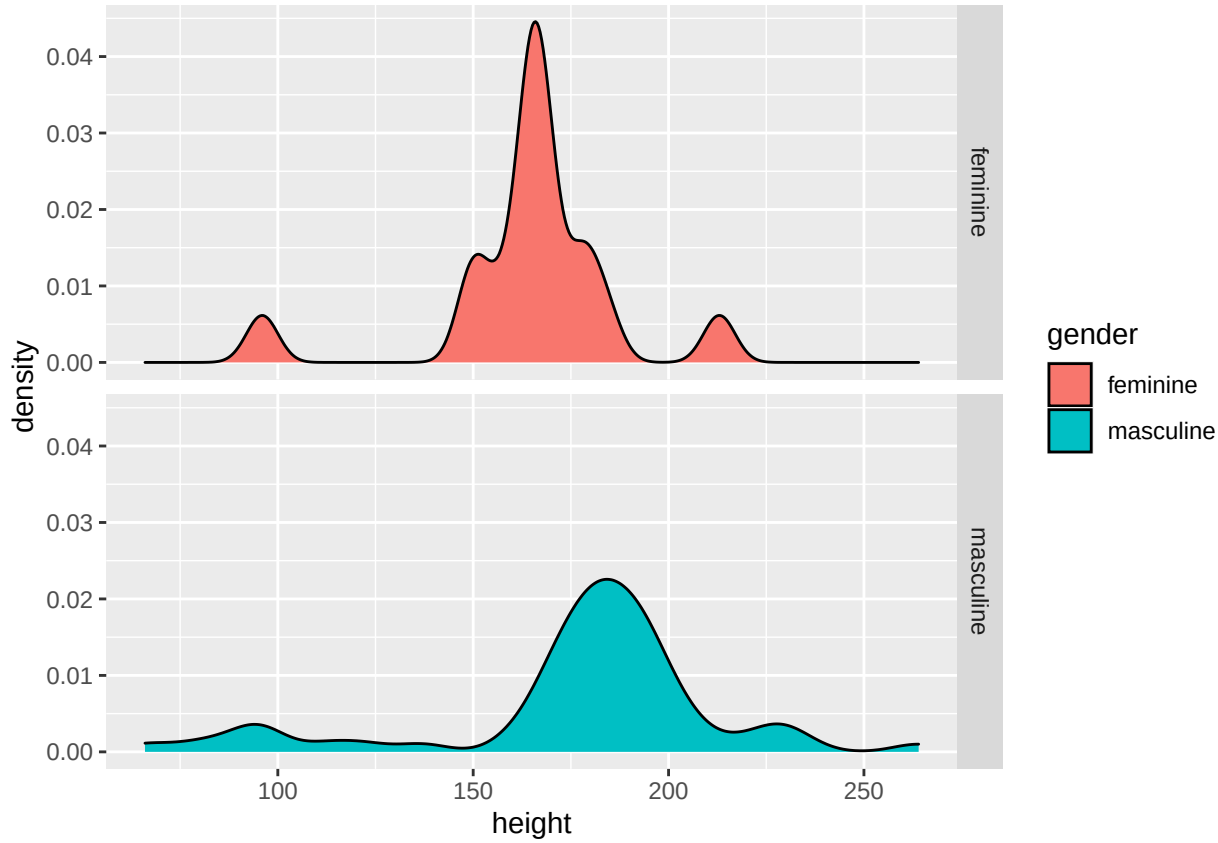
Density plots

They are like histograms, but with probability instead od count

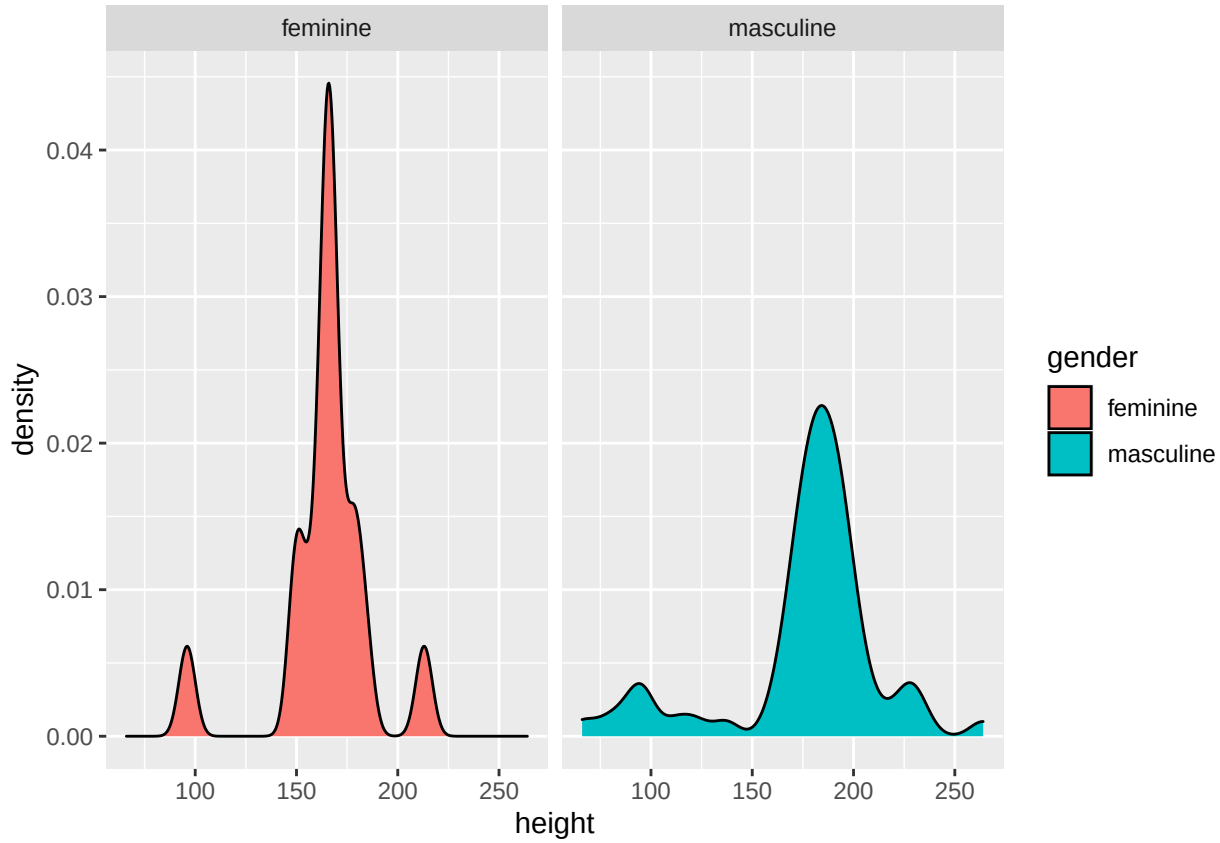
```
starwars %>%
    filter(gender != "NA") %>%
    ggplot(aes(x = height, fill = gender)) + geom_density(alpha = 0.5)
```



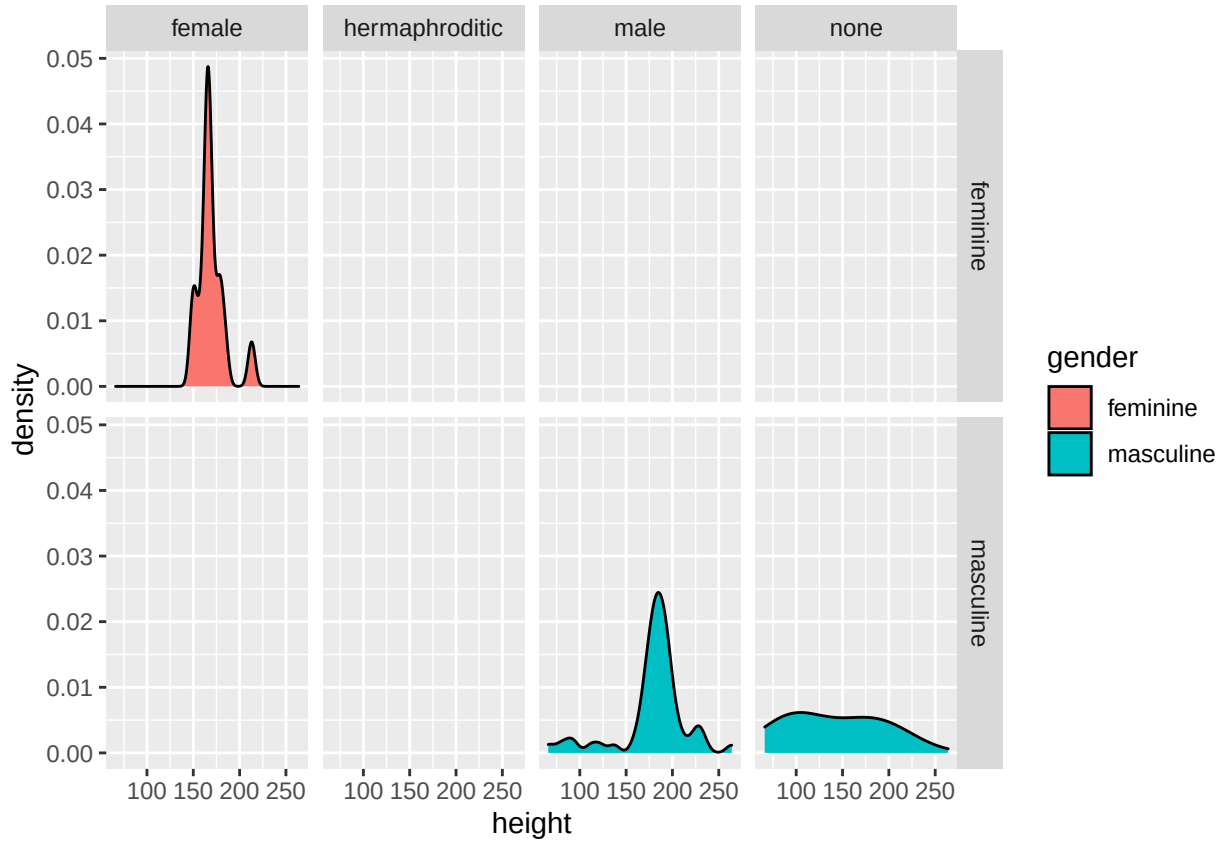

```
starwars %>%
    filter(gender != "NA") %>%
    ggplot(aes(x = height, fill = gender)) + geom_density() +
    facet_grid(gender ~ .)
```




```
starwars %>%
    filter(gender != "NA") %>%
    ggplot(aes(x = height, fill = gender)) + geom_density() +
    facet_grid(. ~ gender)
```



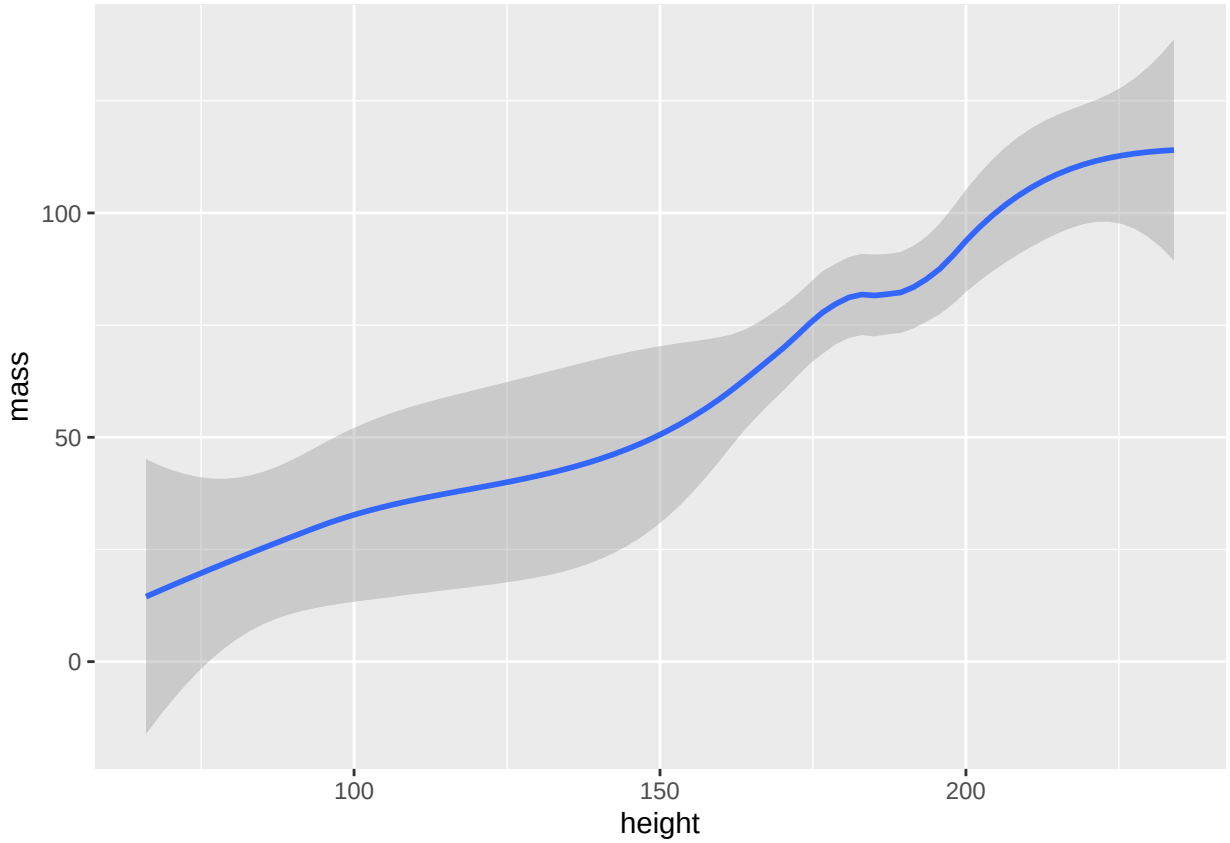

```
starwars %>%
    filter(gender != "NA") %>%
    ggplot(aes(x = height, fill = gender)) + geom_density() +
    facet_grid(gender ~ sex)
```



Line graphs

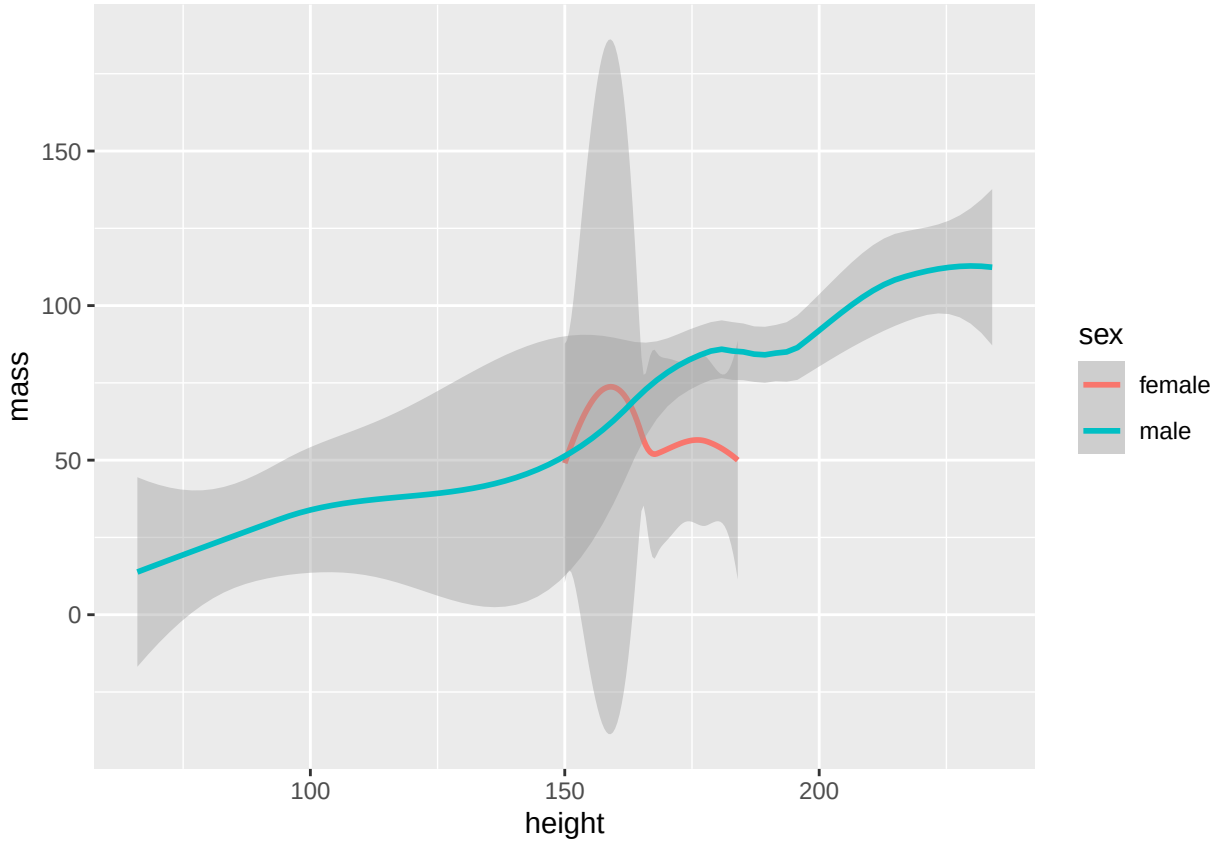
```
starwars2 <- starwars %>%
    filter(sex %in% c("male", "female"), gender %in% c("masculine",
        "feminine"))


ggplot(starwars2, aes(x = height, y = mass)) + geom_smooth()
```

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

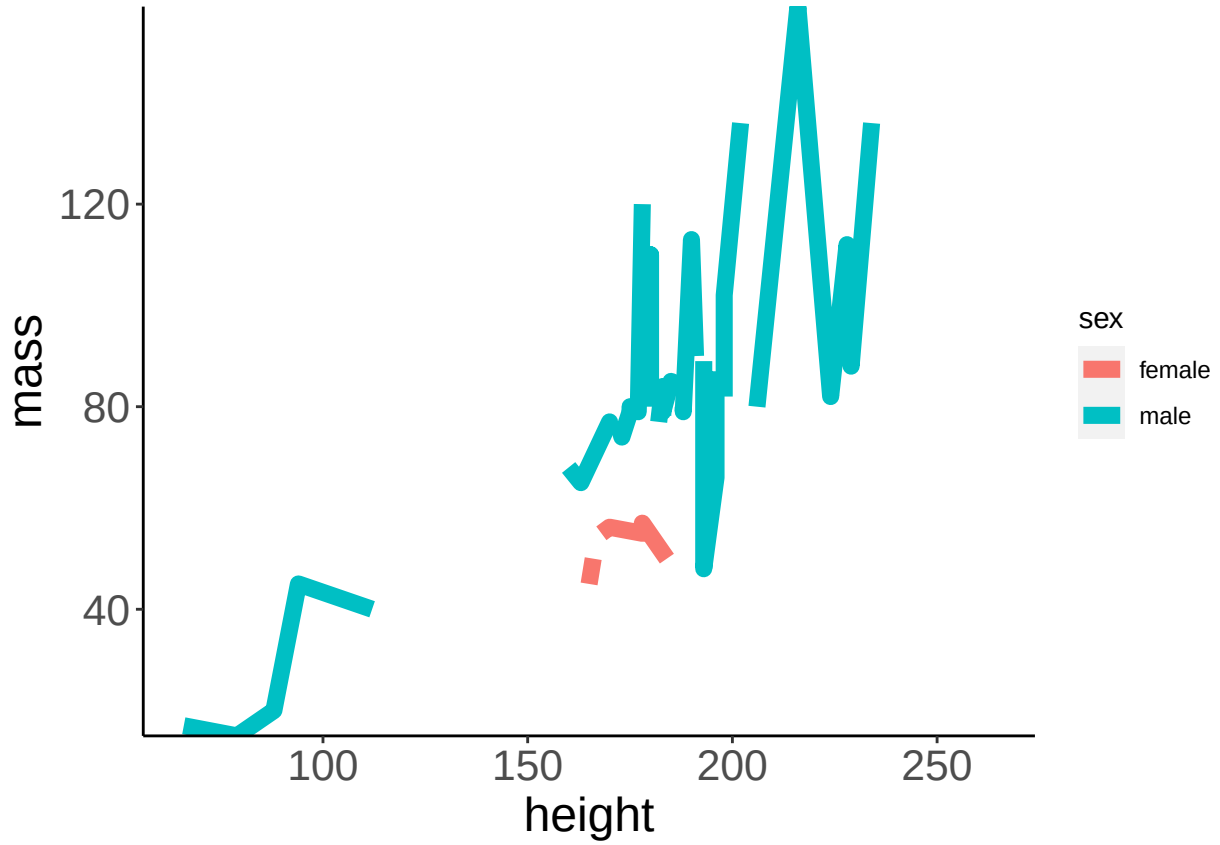



```
ggplot(starwars2, aes(x = height, y = mass, color = sex)) + geom_smooth()
```

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




```
ggplot(starwars2, aes(x = height, y = mass, colour = sex)) +
    geom_line(size = 3) + theme(axis.line = element_line(size = 0.5),
    panel.background = element_rect(fill = NA, size = rel(20)),
    panel.grid.minor = element_line(colour = NA), axis.text = element_text(size = 16),
    axis.title = element_text(size = 18)) + scale_y_continuous(expand = c(0,
    0))
```

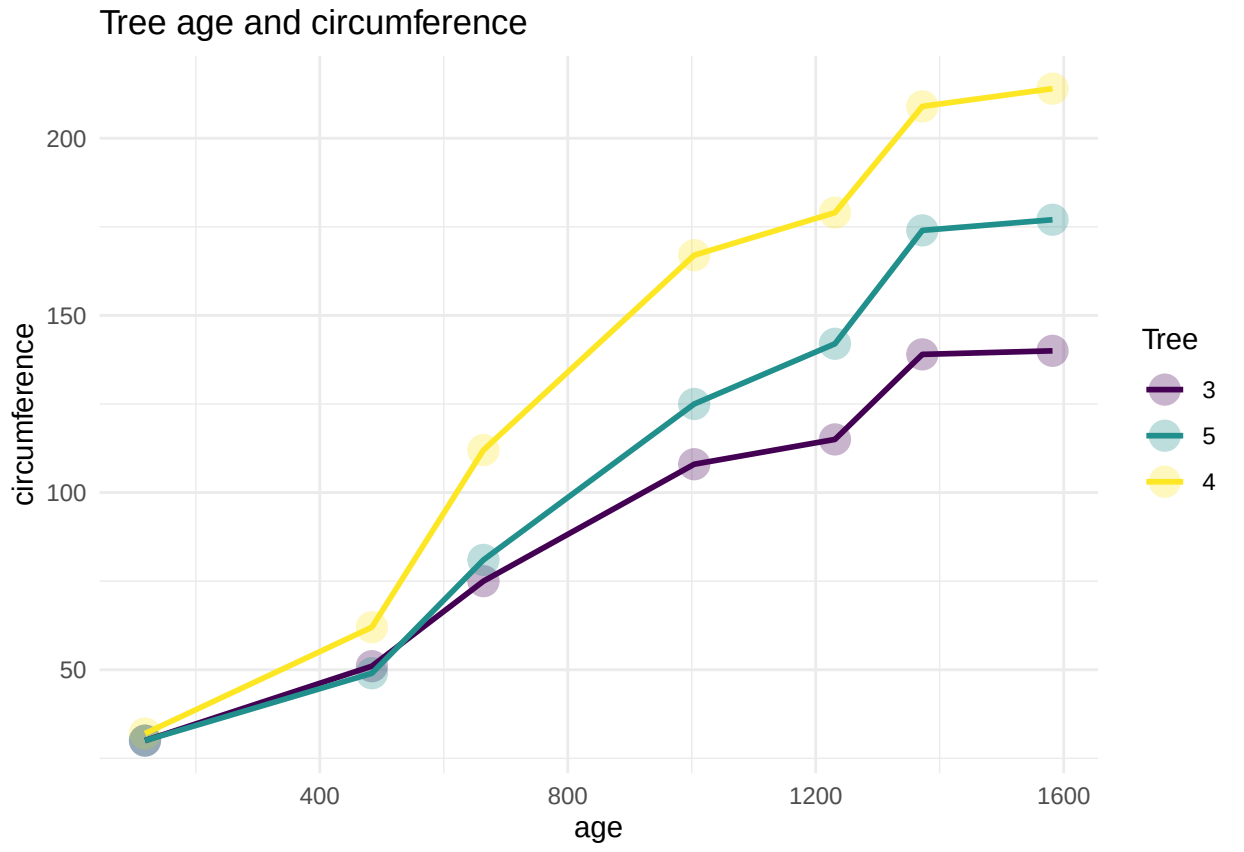



```
Orange
```

```
## Grouped Data: circumference ~ age | Tree
##    Tree  age circumference
## 1     1  118            30
## 2     1  484            58
## 3     1  664            87
## 4     1 1004           115
## 5     1 1231           120
## 6     1 1372           142
## 7     1 1582           145
## 8     2  118            33
## 9     2  484            69
## 10    2  664           111
## 11    2 1004           156
## 12    2 1231           172
## 13    2 1372           203
## 14    2 1582           203
## 15    3  118            30
## 16    3  484            51
## 17    3  664            75
## 18    3 1004           108
## 19    3 1231           115
## 20    3 1372           139
## 21    3 1582           140
## 22    4  118            32
```

```
## 23     4  484            62
## 24     4  664           112
## 25     4 1004           167
## 26     4 1231           179
## 27     4 1372           209
## 28     4 1582           214
## 29     5  118            30
## 30     5  484            49
## 31     5  664            81
## 32     5 1004           125
## 33     5 1231           142
## 34     5 1372           174
## 35     5 1582           177
```

```
Orange %>%
    filter(Tree != "1" & Tree != "2") %>%
    ggplot(aes(x = age, y = circumference, colour = Tree)) +
    geom_point(size = 5, alpha = 0.3) + geom_line(size = 1) +
    theme_minimal() + labs(title = "Tree age and circumference")
```




```
# line graphs with error bars (SEM) + customly-coloured
# SEM, mean, custom colours
library(ggplot2)

sem <- function(x, na.rm = FALSE) {
```

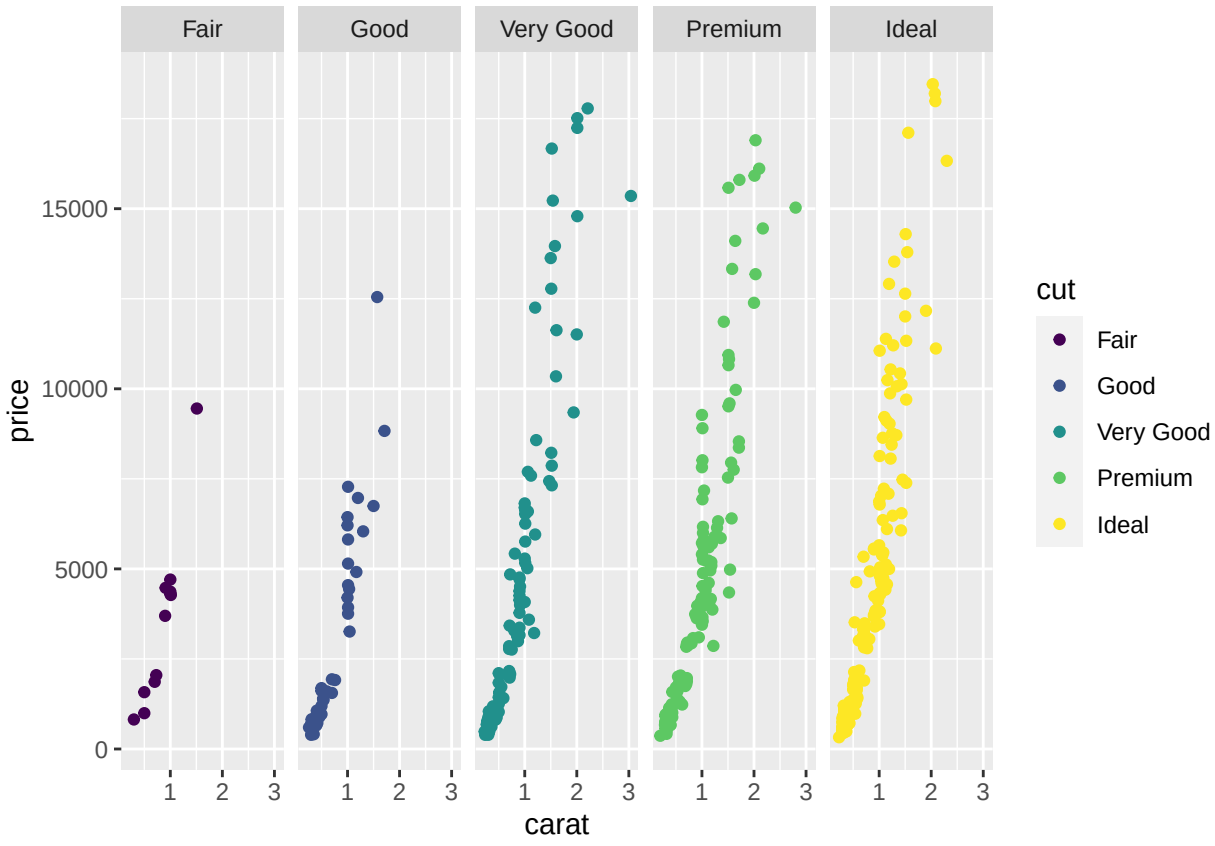
```
    out <- sd(x, na.rm = na.rm)/sqrt(length(x))
    return(out)
}
pointcolour <- c(Fair = "yellow", Good = "red", `Very Good` = "pink",
    Premium = "blue", Ideal = "black")
# write.csv(diamonds, file='diamonds.csv', row.names=TRUE)
```

Pie charts

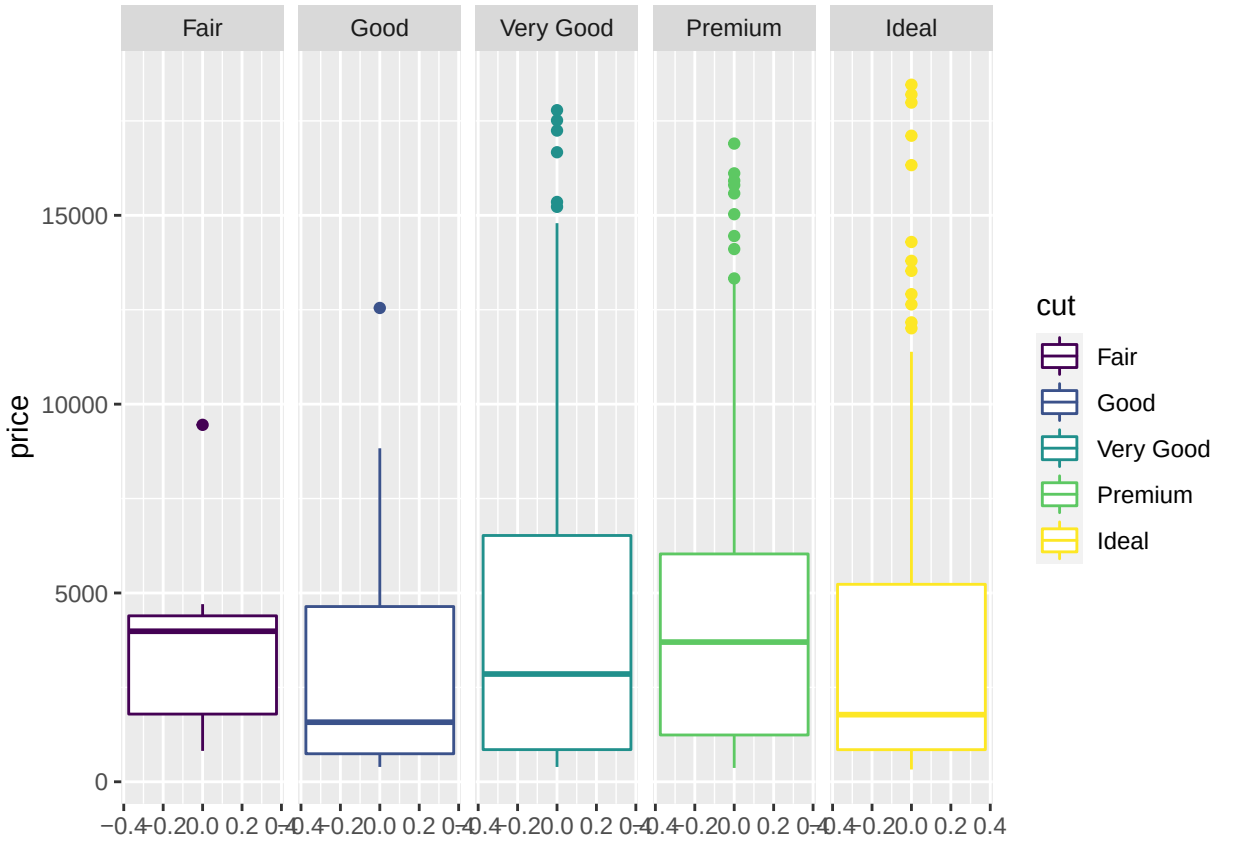
Nothing here right now...

Tiered faceted graphs

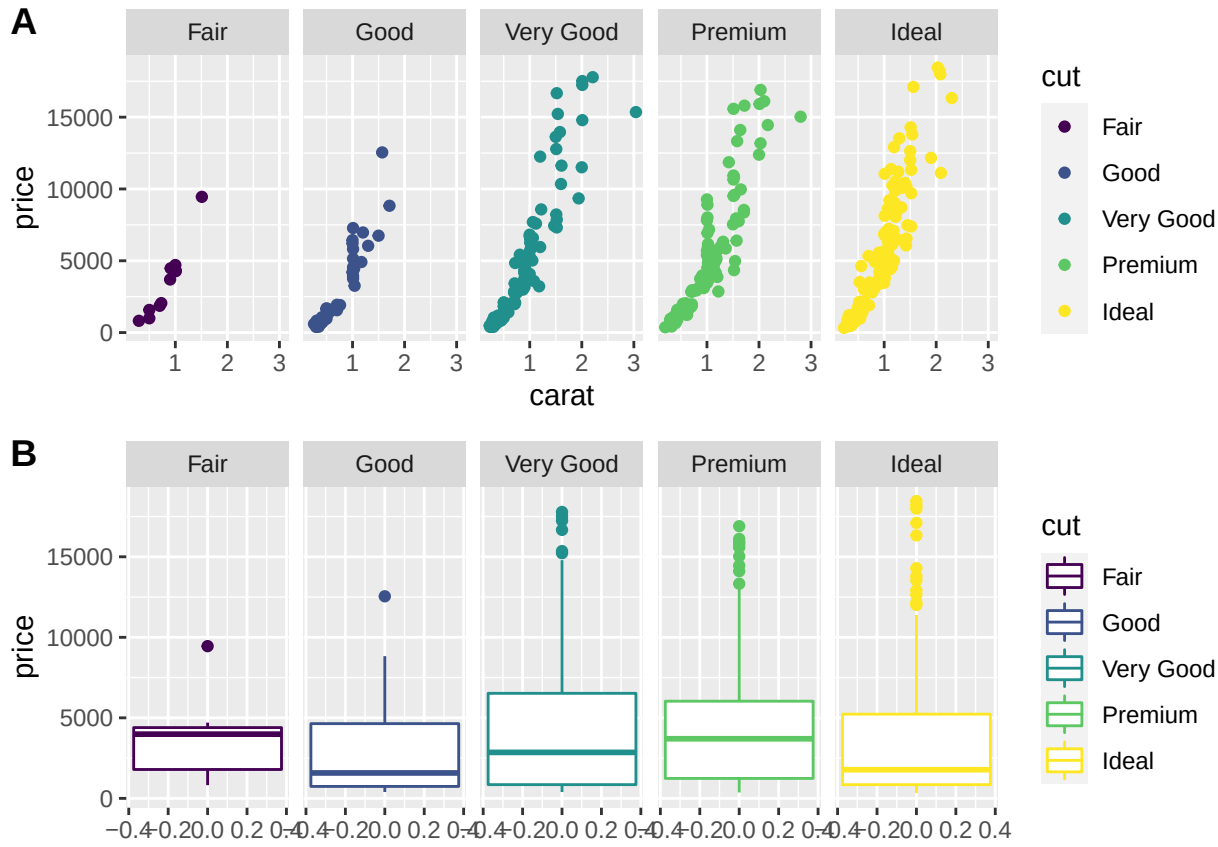
```
library(ggpubr)

p1 <- ggplot(diamonds[seq(1, 50000, by = 100),], aes(x = carat, y = price, colour = cut)) + geom_point()
p1
```



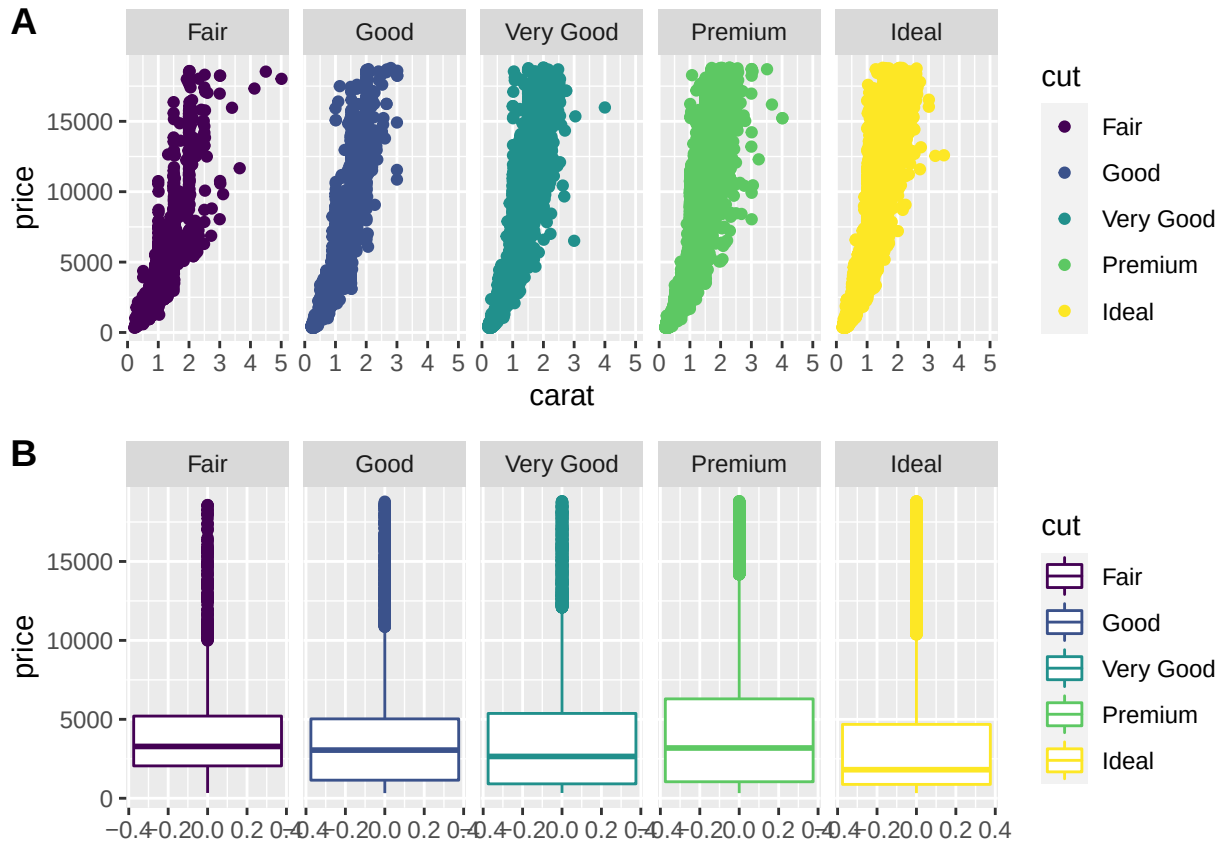

```
p2 <- ggplot(diamonds[seq(1, 50000, by = 100),], aes(y = price, colour = cut)) + geom_boxplot() + facet_
p2
```




```
ggarrange(p1, p2, nrow = 2, labels = "AUTO")
```




```
# OR the same, just the entire dataset & without creating new variables #
ggarrange(
  # graph 1
  ggplot(diamonds, aes(x = carat, y = price, colour = cut)) + geom_point() + facet_wrap( ~ cut, nrow = 
  # graph 2
  ggplot(diamonds, aes(y = price, colour = cut)) + geom_boxplot() + facet_wrap( ~ cut, nrow = 1),
  # ggarrange settings
  nrow = 2, labels = "AUTO"
)
```

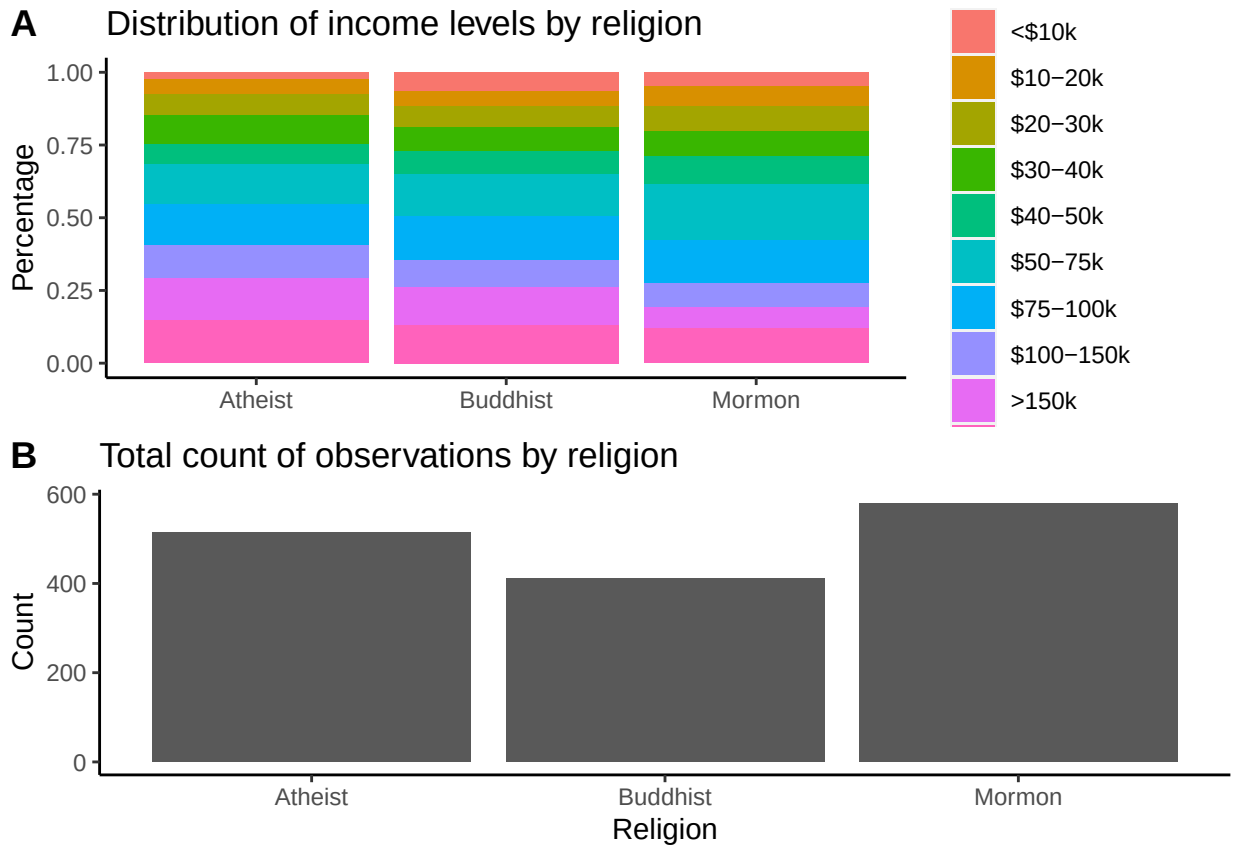



```
summary(relig_income)
```

religion <$10k $10-20k $20-30k
Length:18 Min. : 1.00 Min. : 2.00 Min. : 3.0
Class :character 1st Qu.: 12.25 1st Qu.: 14.75 1st Qu.: 17.0
Mode :character Median : 20.00 Median : 27.00 Median : 33.5
Mean :107.22 Mean :154.50 Mean : 186.5
3rd Qu.:170.00 3rd Qu.:193.00 3rd Qu.: 192.0
Max. :575.00 Max. :869.00 Max. :1064.0
$30-40k $40-50k $50-75k $75-100k
Min. : 4.00 Min. : 2.0 Min. : 7.00 Min. : 3.00
1st Qu.: 15.75 1st Qu.: 15.0 1st Qu.: 34.25 1st Qu.: 25.25
Median : 40.00 Median : 34.0 Median : 66.50 Median : 65.50
Mean :183.44 Mean :171.4 Mean : 288.06 Mean :221.67
3rd Qu.:198.75 3rd Qu.:166.8 3rd Qu.: 201.50 3rd Qu.:128.75
Max. :982.00 Max. :881.0 Max. :1486.00 Max. :949.00
$100-150k >150k Don't know/refused Min. : 4.0 Min. : 4.00 Min. : 8.00
1st Qu.: 22.5 1st Qu.: 23.75 1st Qu.: 41.25
Median : 48.5 Median : 53.50 Median : 74.50
Mean :177.6 Mean :144.89 Mean : 340.06
3rd Qu.:103.5 3rd Qu.:134.25 3rd Qu.: 294.75
Max. :792.0 Max. :634.00 Max. :1529.00

```
p1 <- relig_income %>%
  filter(religion == "Atheist" |
```

```
          religion == "Mormon" |
          religion == "Buddhist") %>%
  melt(id.vars = "religion") %>%
  group_by(religion, variable) %>%
  ggplot(aes(x = religion, y = value, fill = variable)) +
  geom_bar(stat="identity", position = "fill") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title.x = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(size = 0.5)) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Distribution of income levels by religion",
       y = "Percentage")
p2 <- relig_income %>%
  filter(religion == "Atheist" |
          religion == "Mormon" |
          religion == "Buddhist") %>%
  melt(id.vars = "religion") %>%
  group_by(religion, variable) %>%
  ggplot(aes(x = religion, y = value)) +
  geom_bar(stat="identity") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title = element_text(),
        panel.background = element_blank(),
        axis.line = element_line(size = 0.5)) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Total count of observations by religion",
       x = "Religion",
       y = "Count")
ggarrange(p1, p2, nrow = 2, labels = "AUTO")
```



Graph customisation

Theme settings

- Flip coordinates: + coord_flip()
- Log scale: + scale_y_log10()
- Show fonts: fonts()
- Globally change theme settings: theme_set(theme_gray(base_size=20))

Colour changes

- Globally reset theme settings to default: theme_set(theme_gray())
- Black and white theme: mainplot + theme_bw()
- Manually assign colours: scale_color_manual(values = c("#A6611A", "#DFC27D", "#6e6c6b", "#80CDC1", "#018571"))

Colour palettes

- library(RColorBrewer)
- display.brewer.all()
- colours() # gives names of different colours
- mainplot + scale_fill_brewer(palette="Set1")

- Setting colour values for each of the 5 variables in my mainplot graph manually: mainplot + scale_fill_manual(values = c("green", "blue", "red", "orange", "grey"))

Changing theme

`+ theme()` argument

- Number size on axis: axis.text = element_text(size = 20)
- Title size on axis: axis.title = element_text(size = 20)
- Legend: legend.title = element_text(size = 20)
- Background: panel.background = element_rect(fill = "pink")
- Change major gridlines: panel.grid.major = element_line(colour = "blue")
  - size = 0.2
  - NA
- Change minor gridlines: panel.grid.minor = element_line(colour = "red")
- Change tick marks: axis.ticks = element_line(size=2, colour="blue")
  - As options: axis.ticks.x, axis.ticks.y
- Change legend position: legend.position="top"
  - Other arguments:
  - "bottom"
  - Bottom left: c(0,0)
  - Near the top right: c(0.8,0.8)

Changing labels

`+ labs()` argument

- title = "Title"
- y = "Title y"
- x = "Title x"

Publication style

Clear background, axis lines, no box, no grid lines, basic colors, no legend

```
# Just for one plot
mainplot + theme(axis.line = element_line(size = 0.5), panel.background = element_rect(fill = NA,
    size = rel(20)), panel.grid.minor = element_line(colour = NA),
    axis.text = element_text(size = 16), axis.title = element_text(size = 18)) +
    scale_y_continuous(expand = c(0, 0))  # to stop the bars from floaing above the x-axis

# Globally apply this theme
theme_set(theme_gray() + theme(axis.line = element_line(size = 0.5),
    panel.background = element_rect(fill = NA, size = rel(20)),
    panel.grid.minor = element_line(colour = NA), axis.text = element_text(size = 16),
    axis.title = element_text(size = 18)))
```