

Chapter 1: data cleaning, wrangling & visualization

Rafal Urbaniak and Nikodem Lewandowski

1 Data, tabulation and visualisation

Before you dive in, first load the libraries (if you need to install any, do this first):

```
library(modelr)
library(kableExtra)
library(ggplot2)
library(ggthemes)
library(dplyr)
library(knitr)
library(nycflights13)
library(forcats)
library(tidyverse)
```

1.1 Data: reading, cleaning, summarizing

The first thing to do when we start a data analysis, is to load the data (there are many easily googlable options for this, depending on what file format you're using). For now, let's use a longitudinal set connecting heights to earnings coming from the `modelr` package. It's already loaded, and we start with reading about the data set (sometimes, such information is available) and inspecting the first few lines (in our case, 10 of them). Then, it might be worthwhile to look at the structure of the dataset.

```
##?heights # find the information about the dataset
# (this will not show in PDF, check the help tab in RStudio)
heights <- as.data.frame(heights) # convert to a standard data frame format

# str() function displays the structure of an R object
# with that we can check some basic information about an object
str(heights) # number of variables, data types, column names etc.
```

```
## 'data.frame':    7006 obs. of  8 variables:
## $ income      : int  19000 35000 105000 40000 75000 102000 0 70000 60000 150000 ...
## $ height      : num  60 70 65 63 66 68 74 64 69 69 ...
## $ weight      : int  155 156 195 197 190 200 225 160 162 194 ...
## $ age         : int   53 51 52 54 49 49 48 54 55 54 ...
## $ marital     : Factor w/ 5 levels "single","married",...: 2 2 2 2 2 4 2 4 4 4 ...
## $ sex         : Factor w/ 2 levels "male","female": 2 2 1 2 1 2 1 2 1 1 ...
## $ education   : int   13 10 16 14 14 18 16 12 12 13 ...
## $ afqt        : num   6.84 49.44 99.39 44.02 59.68 ...
```

```
# now a more elegant tool
# inspect the first few lines to get a feel for the dataset,
# note the pipe "%>%" operator (google it and learn to use it)
# note also I used kable to plot a nice table in the pdf file
# you don't have to do this if you're working using an R script.
head(heights, n = 6) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)
```

| income | height | weight | age | marital | sex | education | afqt |
|--------|--------|--------|-----|----------|--------|-----------|--------|
| 19000 | 60 | 155 | 53 | married | female | 13 | 6.841 |
| 35000 | 70 | 156 | 51 | married | female | 10 | 49.444 |
| 105000 | 65 | 195 | 52 | married | male | 16 | 99.393 |
| 40000 | 63 | 197 | 54 | married | female | 14 | 44.022 |
| 75000 | 66 | 190 | 49 | married | male | 14 | 59.683 |
| 102000 | 68 | 200 | 49 | divorced | female | 18 | 98.798 |

We can view the means and medians of variables of interest as follows:

```
mean(heights$height)
```

```
## [1] 67.10434
```

```
#if a variable has NAs, you might choose or need to ignore them with na.rm
median(heights$weight, na.rm=TRUE)
```

```
## [1] 184
```

Note the use of the dollar sign between the data name and variable name. For data frames dollar sign enables a choice of a particular column by its name.

We can select parts of the data by a logical condition, using square brackets appropriately. For instance, the mean of weight for people with height>68 and the median of weight of people with height=69 can be obtained as follows:

```
mean(heights$weight[heights$height > 68], na.rm=TRUE)
```

```
## [1] 210.8534
```

```
#note double identity symbol for comparison
median(heights$weight[heights$height == 69], na.rm=TRUE)
```

```
## [1] 190
```

Exercise 1

I now load the DrunkAndDriving dataset for you. View 5 first entries and data structure, this should give you a feel for what the dataset is about. Note that you have to set your working directory to the directory containing the teachingData folder (you can use setwd() function for that purpose). Also, what is the purpose of "sep=" parameter?

```
drunkDriving <- read.csv("DrunkAndDrivingPoland.csv", sep = ";")
```

Exercise 2

Display the head of the dataset.

Exercise 3

Note that the first column name is informative but ugly. We can make it simpler, complete the following code to do so and view the dataset structure. Are you sure you know why we used [1]?

```
names(drunkDriving)[1] <- "___"
```

Exercise 4

Proportions given in percentages are not very informative in this case, complete the code to remove this column and view the head to make sure your move worked. Are you sure you know why we used [-4]?

```
<- _____[, -4]
```

Exercise 5

View the mean and median of the `cases` variable.

Exercise 6

What are the mean and median occurrences of known drunk and driving cases in years 2001-2006? Use `&` for conjunction in the logical condition.

Exercise 7

What are the means and medians for periods 2001-2010 and 2011-2021? Is, according to this data, drunk driving increasing or decreasing problem?

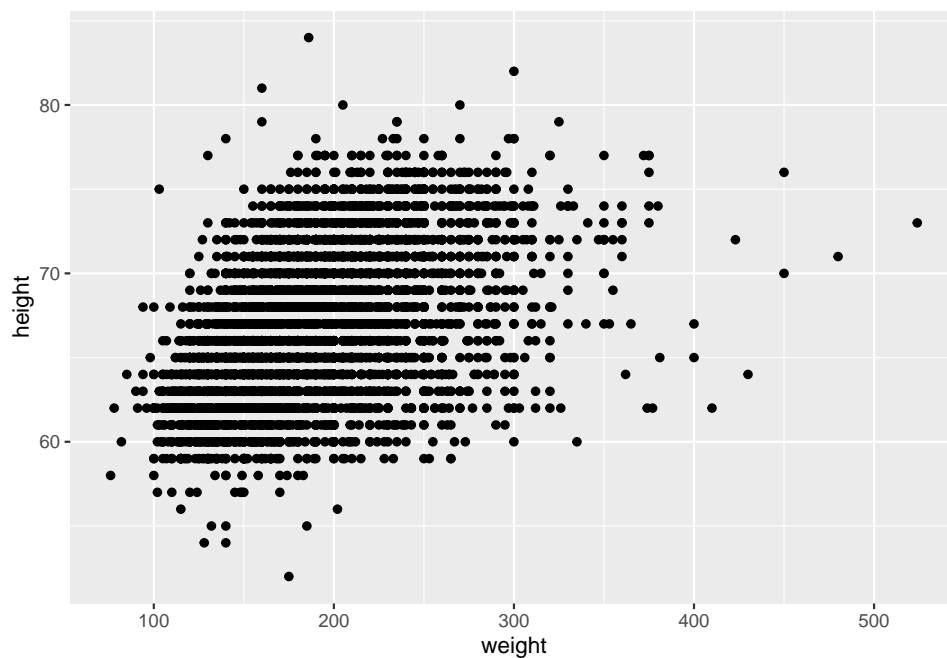
1.2 Basic plots with ggplot2

It is almost always useful to visualize the data, for various reasons. This action should precede any statistical analysis or inference. Here is one simple method: **scatterplot**. It works for two variables. We will use one of the most popular visualization packages in R, **ggplot2**. Let's do this for height and weight in the heights dataset:

```
# You can find a cheatsheet for using ggplot2, check out:  
# https://www.rstudio.com/resources/cheatsheets/
```

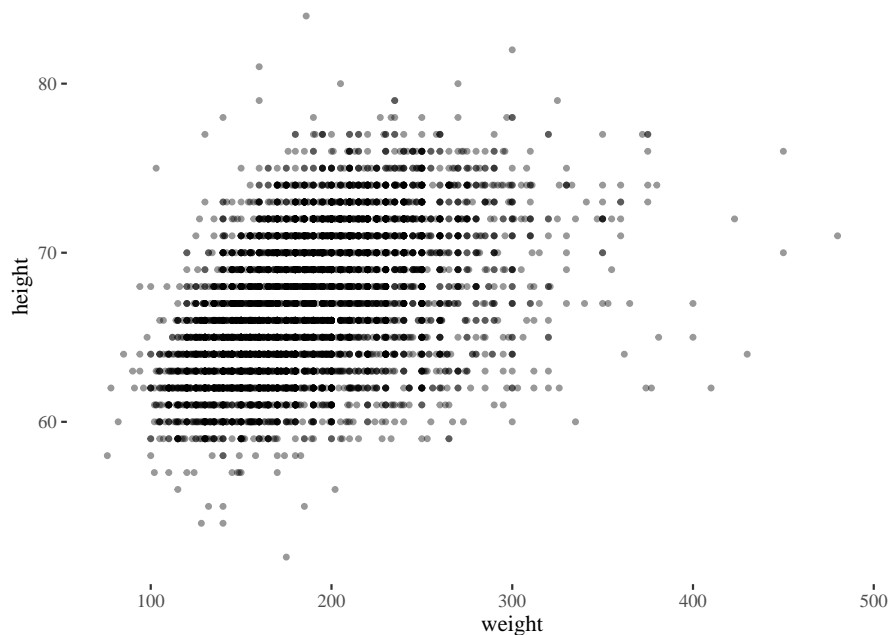
```
#specify the basics  
ggplot(heights, aes(x = weight, y = height)) +  
  geom_point() # add a layer of points
```

```
## Warning: Removed 95 rows containing missing values ('geom_point()').
```



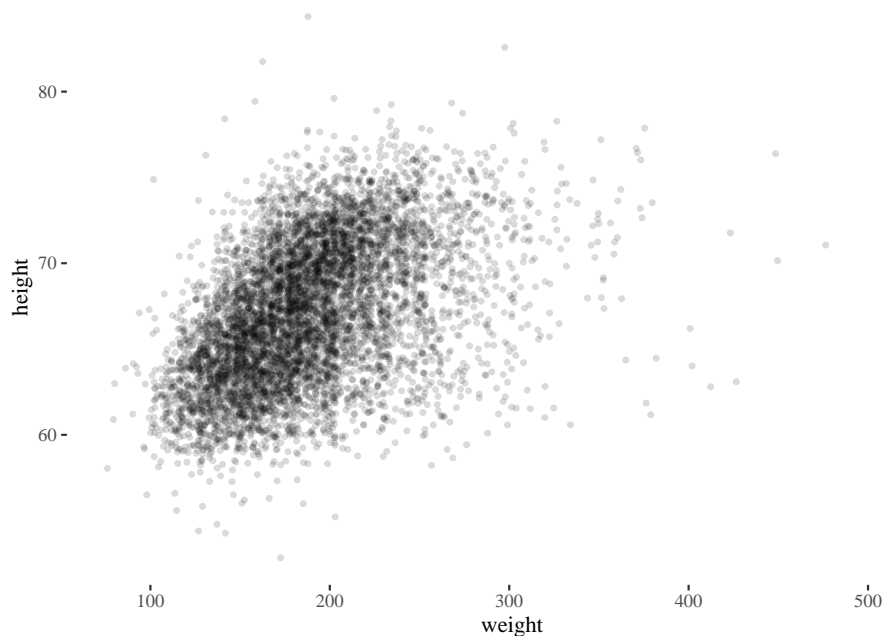
Notice also that you got a warning: the dataset contained NAs. The points are a bit large, we will make them a bit smaller and slightly transparent (`alpha` parameter) to better see how densely they occur. We also use the `theme_tufte` instead of the default `ggplot` theme:

```
#we first specify the basics
ggplot(heights, aes(x = weight, y = height)) +
  geom_point(size = .9, alpha = .4)+
  theme_tufte() # we add a layer of points
```



To get more clarity, we can “shake” the points a bit with a jitter. Sometimes you need to get creative and play around with the parameters to make the plot look nice.

```
ggplot(heights, aes(x = weight, y = height)) +
  geom_jitter(size = .9, alpha = .15, width = 4, height = 1)+
  theme_tufte()
```



We can add lines to the plot. Sometimes, we add exact values, as when we already have some summaries. To show you how to work with vectors I’ll give you a simple example, make sure you understand how the vector constructions worked, observe the use of three different functions (“:” operator, rep(), seq()). I also use length to make sure I created vectors of equal weight. Remember that you can always use “?” (e.g ?rep()) to check the details about the R function in question.

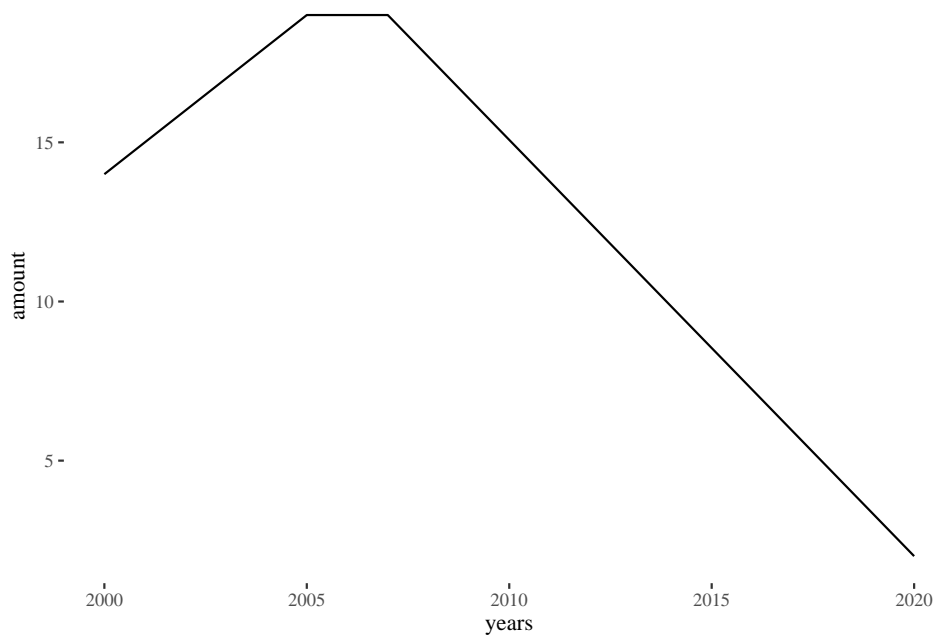
```
years <- 2000:2020
amount <- c(14:18, rep(19,2), seq(19,2, length.out = 14))
length(years)
```

```
## [1] 21
```

```
length(amount)
```

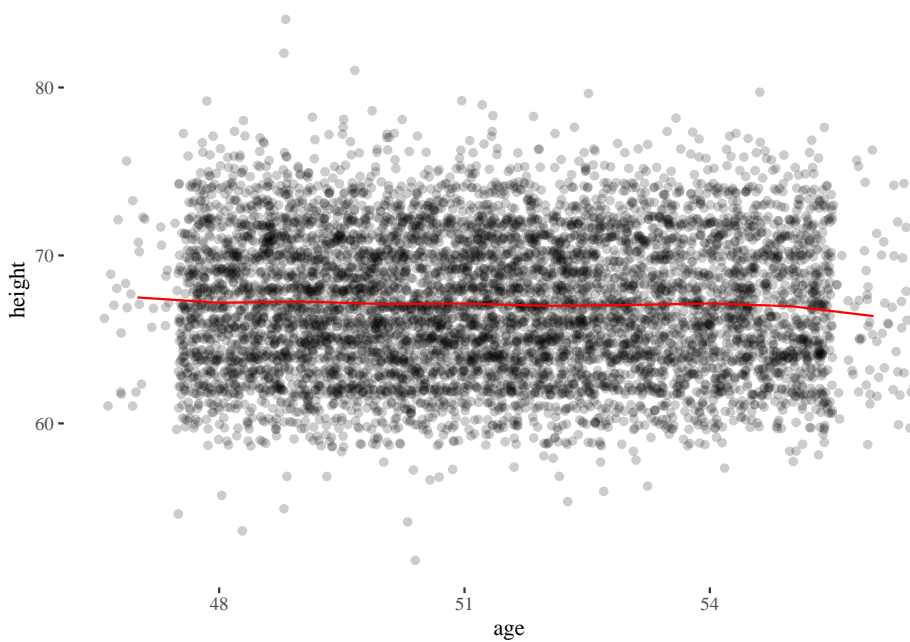
```
## [1] 21
```

```
ggplot() + geom_line(aes(x= years, y = amount))+  
  theme_tufte()
```



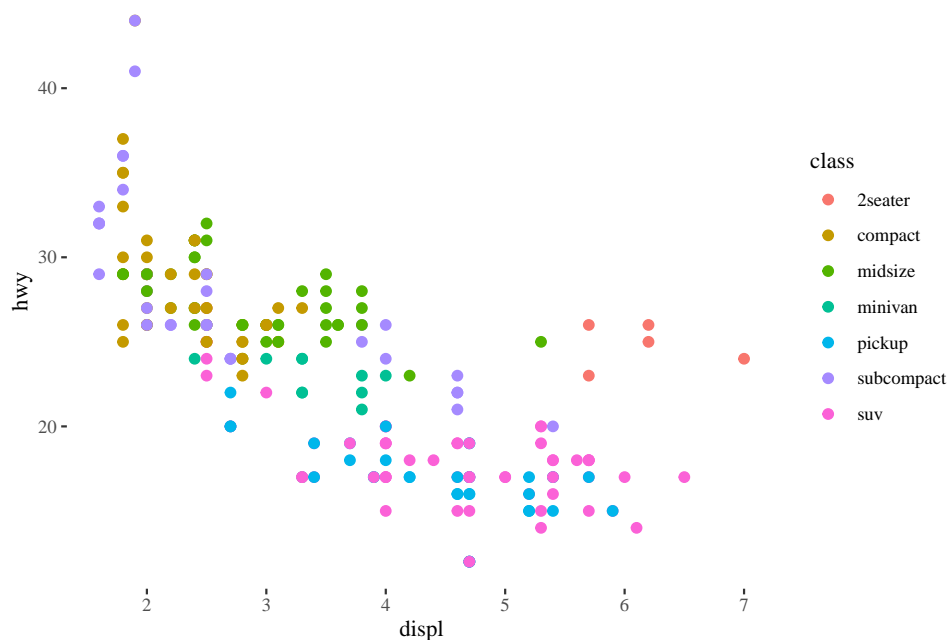
On some other occasions we want to focus on some statistics, such as mean. For instance, for heights we can do this as follows:

```
ggplot(heights, aes(x= age, y = height))+  
  geom_jitter(width = 0.5, alpha = 0.2)+  
  stat_summary(fun=mean, colour="red", geom="line", aes(group = 1))+  
  theme_tufte()
```



Now we look at `mpg` dataset to illustrate how we can also color the points by grouping. Feel free to take a look at its help description to see what data and variables it contains.

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy,
                             color = class), size = 2) +
  theme_tufte()
```



Observe the extreme outliers when it comes to fuel efficiency. Grouping by color makes it easy to see that these are subcompact cars.

Exercise 8

Drunk driving again, we all like to enjoy a cold-one once in a while, but maybe a car and alcohol is not a good connection. Plot cases against years with a line. Use the tufte theme and color "skyblue".

Now we will work on an interesting dataset containing data on shooting incidents in New York. The chunk below loads the data and does some basic wrangling (be sure to execute all of the code). View the data set using head() function. You can check what the variables mean here: <https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8>

```
nypd <- read.csv("NYPD_ShootingMOD.csv")
nypd$OCCUR_DATE <- as.Date(nypd$OCCUR_DATE, "%Y-%m-%d")
nypd <- nypd %>% arrange(OCCUR_DATE)
```

Exercise 9

Create a scatterplot of the number of occurrences versus the month of the year using the NYPD dataset. Add a line representing the mean. Note that the set is slightly more complicated, I will wrangle the data for you, so don't stress (be sure to execute the given code first!).

```
# just run the code below
nypd$MONTH <- as.integer(format(nypd$OCCUR_DATE, "%m"))
nypd_monthly_counts <- nypd %>%
  mutate(MONTH = format(OCCUR_DATE, "%Y-%m")) %>%
  group_by(MONTH) %>%
  summarise(OCCURRENCES = n()) %>%
  ungroup()

# use the nypd_monthly_counts as your table to create the requested plot:
_____ <- ggplot(nypd_monthly_counts, _____) +
```

Exercise 10

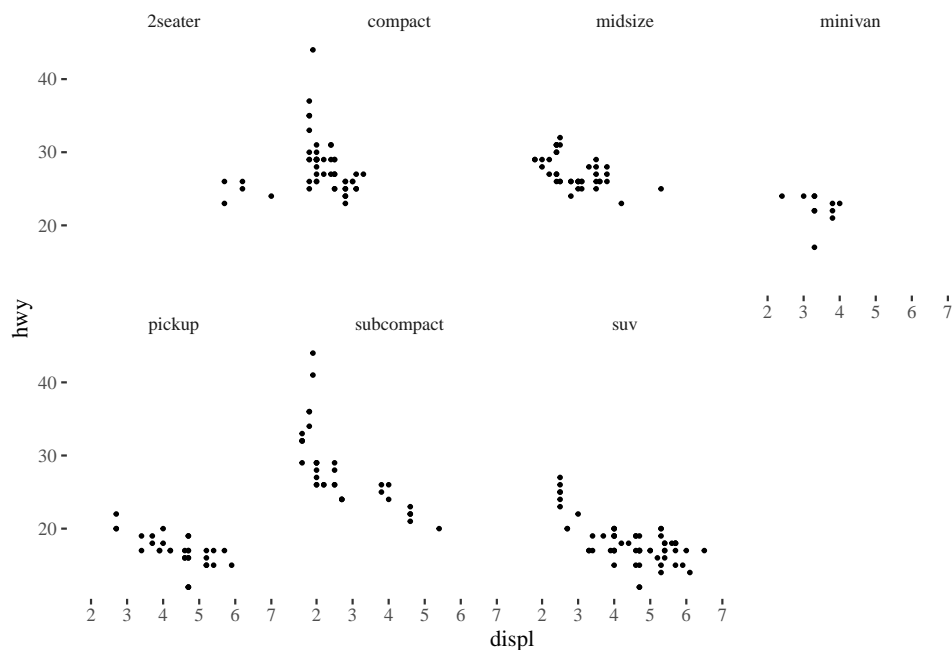
Make a similar plot (months against occurrences), use the table below, and add points colored by borough (it's a district, there are 6 of them). Add mean and median line (mean should be red, median green), but this time specify size = 1. Is there any borough that stands out in terms of occurrence rates?

```
# run this code
nypd_monthly_counts_boro <- nypd %>%
  mutate(MONTH = format(OCCUR_DATE, "%Y-%m")) %>%
  group_by(MONTH, BORO) %>%
  summarise(OCCURRENCES = n()) %>%
  ungroup()

# use nypd_monthly_counts_boro to proceed with your visualization
```

Another useful feature is splitting the plots by a categorical variable. For instance, for the mpg database we can do:

```
ggplot(mpg) + geom_point(aes(x = displ, y = hwy), size = 0.6) +
  facet_wrap(~ class, nrow = 2) + theme_tufte()
```



Exercise 11

Again, on the same plot (months against occurrences) represent the age of victims, create mean and median lines. Compare your findings with the earlier plot, do you see any pattern?

```
nypd_monthly_counts_age <- nypd %>%
  mutate(MONTH = format(OCCUR_DATE, "%Y-%m")) %>%
  group_by(MONTH, VIC_AGE_GROUP) %>%
  summarise(OCCURRENCES = n()) %>%
  ungroup()
```

Exercise 12

Create a scatterplot of the number of occurrences versus the month of the year using the NYPD dataset, with points colored by the borough and smoothed with a line of best fit. Is there any trend in the data over the year, in terms of the occurrence rates in each borough?

```
nypd_monthly_counts_boro <- nypd %>%
  mutate(MONTH = format(OCCUR_DATE, "%Y-%m")) %>%
  group_by(MONTH, BORO) %>%
  summarise(OCCURRENCES = n()) %>%
  ungroup()
```

Exercise 13

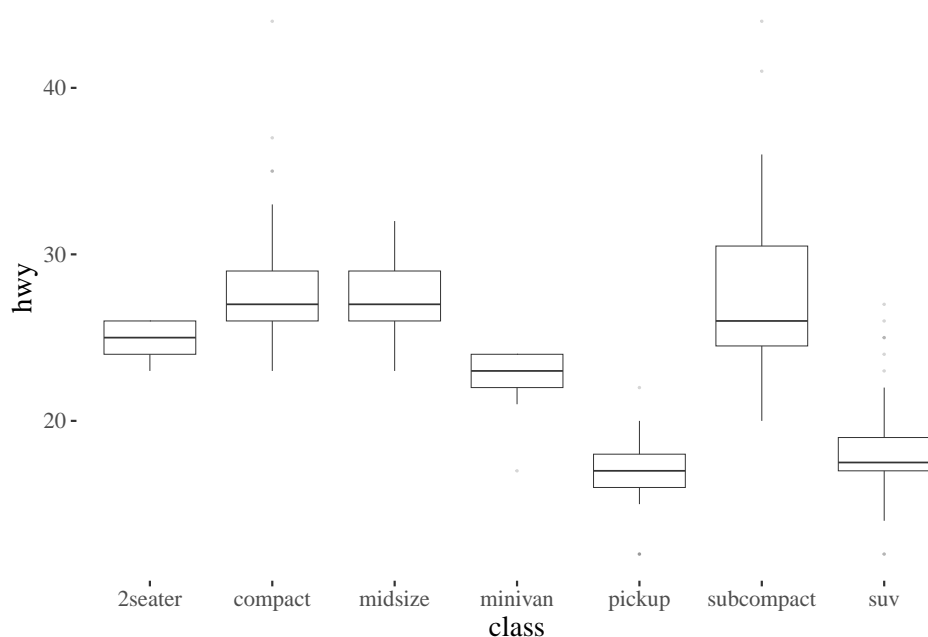
Look at the dataset on the eruptions of Old Faithful (google if you don't know what that is). Prepare a scatterplot of eruption times vs. waiting times. What strikes you?

```
old_faithful <- faithful
```

1.3 Boxplots

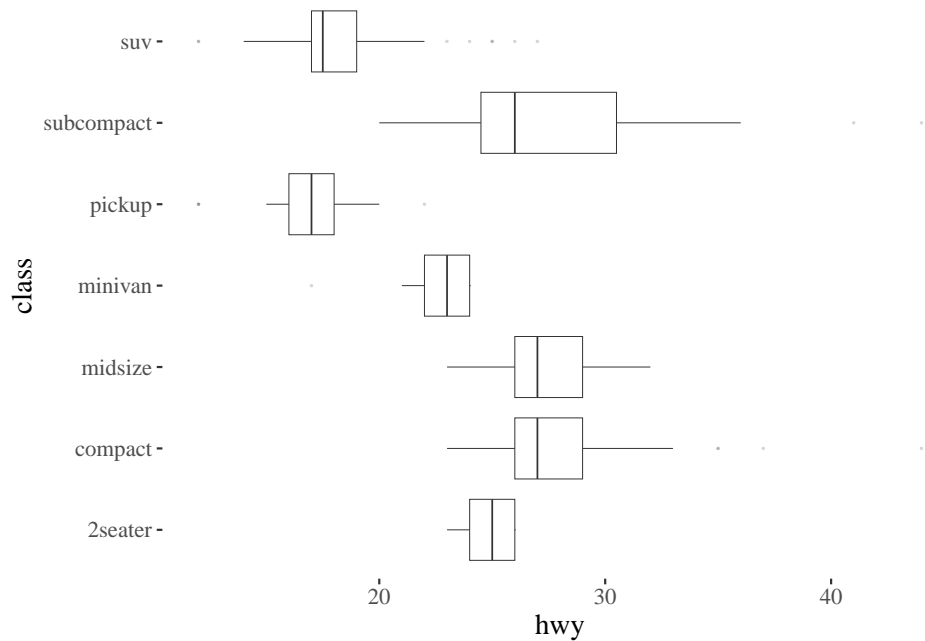
Another way to represent data is with boxplots, which nicely show the median, quartiles, IQR and can be used to detect potential outliers. Here's an example.

```
ggplot(mpg, aes(x = class, y = hwy)) +  
  geom_boxplot(size=0.2,alpha=0.2,outlier.size = 0.1) +  
  theme_tufte() +  
  theme(text = element_text(size=14))  
# In the last line I have increased the font size for sake of better visibility
```



Observe we can flip the coordinates if we find this useful:

```
ggplot(mpg, aes(x = class, y = hwy)) +  
  geom_boxplot(size=0.2,alpha=0.2,outlier.size = 0.1) +  
  theme_tufte() + coord_flip() +  
  theme(text = element_text(size=14))
```

Exercise 14

Create a box plot of the number of incidents per month against borough.

1.4 Using dplyr

The `dplyr` package has quite a few useful tools, we'll see them in action looking at the NYC flights dataset available in the `nycflights13` package. The key ones are:

- `filter`, which pick observations by their values.
- `arrange`, which reorders the rows.
- `select`, which picks variables by their names.
- `mutate`, which creates new variables with functions of existing variables.
- `summarize`, which collapse many values down to a single summary.

First, let's inspect the head of the dataset (there are quite a few variables, so it's not going to look user-friendly):

```
head(flights) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  kable_styling(font_size = 14)
```

| year | month | day | dep_time | sched_dep_time | dep_delay | arr_time | sched_arr_time | arr_delay | carrier | flight | tailnum | origin | dest | air_time | distance | hour | minute | time_hour |
|------|-------|-----|----------|----------------|-----------|----------|----------------|-----------|---------|--------|---------|--------|------|----------|----------|------|--------|---------------------|
| 2013 | 1 | 1 | 517 | 515 | 2 | 830 | 819 | 11 | UA | 1545 | N14228 | EWB | IAH | 227 | 1400 | 5 | 15 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 533 | 529 | 4 | 850 | 830 | 20 | UA | 1714 | N24211 | LGA | IAH | 227 | 1416 | 5 | 29 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 542 | 540 | 2 | 923 | 850 | 33 | AA | 1141 | N619AA | JFK | MIA | 160 | 1089 | 5 | 40 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 544 | 545 | -1 | 1004 | 1022 | -18 | B6 | 725 | N804JB | JFK | BQN | 183 | 1576 | 5 | 45 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 554 | 600 | -6 | 812 | 837 | -25 | DL | 461 | N668DN | LGA | ATL | 116 | 762 | 6 | 0 | 2013-01-01 06:00:00 |
| 2013 | 1 | 1 | 554 | 558 | -4 | 740 | 728 | 12 | UA | 1696 | N39463 | EWB | ORD | 150 | 719 | 5 | 58 | 2013-01-01 05:00:00 |

Another way to inspect the variables is to look at column names:

```
names(flights)
```

```
## [1] "year"          "month"         "day"           "dep_time"
## [5] "sched_dep_time" "dep_delay"     "arr_time"      "sched_arr_time"
## [9] "arr_delay"     "carrier"       "flight"        "tailnum"
## [13] "origin"        "dest"          "air_time"      "distance"
## [17] "hour"          "minute"        "time_hour"
```

For instance, we can pick only the flights that took place on Jan 1 by saying:

```
Jan1 <- filter(flights, month == 1, day == 1)
head(Jan1) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  kable_styling(font_size = 14)
```

| year | month | day | dep_time | sched_dep_time | dep_delay | arr_time | sched_arr_time | arr_delay | carrier | flight | tailnum | origin | dest | air_time | distance | hour | minute | time_hour |
|------|-------|-----|----------|----------------|-----------|----------|----------------|-----------|---------|--------|---------|--------|------|----------|----------|------|--------|---------------------|
| 2013 | 1 | 1 | 517 | 515 | 2 | 830 | 819 | 11 | UA | 1545 | N14228 | EWB | IAH | 227 | 1400 | 5 | 15 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 533 | 529 | 4 | 850 | 830 | 20 | UA | 1714 | N24211 | LGA | IAH | 227 | 1416 | 5 | 29 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 542 | 540 | 2 | 923 | 850 | 33 | AA | 1141 | N619AA | JFK | MIA | 160 | 1089 | 5 | 40 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 544 | 545 | -1 | 1004 | 1022 | -18 | B6 | 725 | N804JB | JFK | BQN | 183 | 1576 | 5 | 45 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 554 | 600 | -6 | 812 | 837 | -25 | DL | 461 | N668DN | LGA | ATL | 116 | 762 | 6 | 0 | 2013-01-01 06:00:00 |
| 2013 | 1 | 1 | 554 | 558 | -4 | 740 | 728 | 12 | UA | 1696 | N39463 | EWB | ORD | 150 | 719 | 5 | 58 | 2013-01-01 05:00:00 |

Exercise 15

Use filter to pick the incidents from the last two months of 2021 from the dataset NYPD shootings. Remember that the date format is YYYY-MM-DD, display the result and save as "twoLast". Hint, use ">=" and "<=" operators.

```
# firstly just run this code to change the format
nypd$OCCUR_DATE <- as.Date(nypd$OCCUR_DATE, "%Y-%m-%d")

# fill the blank places to complete the task
twoLast <- _____ %>% _____
```

We can arrange flights by variables. For instance, by year, month and day of the flight:

```
YearMonthDay <- arrange(flights, year, month, day)
head(YearMonthDay) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  kable_styling(font_size = 14)
```

| year | month | day | dep_time | sched_dep_time | dep_delay | arr_time | sched_arr_time | arr_delay | carrier | flight | tailnum | origin | dest | air_time | distance | hour | minute | time_hour |
|------|-------|-----|----------|----------------|-----------|----------|----------------|-----------|---------|--------|---------|--------|------|----------|----------|------|--------|---------------------|
| 2013 | 1 | 1 | 517 | 515 | 2 | 830 | 819 | 11 | UA | 1545 | N14228 | EWB | IAH | 227 | 1400 | 5 | 15 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 533 | 529 | 4 | 850 | 830 | 20 | UA | 1714 | N24211 | LGA | IAH | 227 | 1416 | 5 | 29 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 542 | 540 | 2 | 923 | 850 | 33 | AA | 1141 | N619AA | JFK | MIA | 160 | 1089 | 5 | 40 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 544 | 545 | -1 | 1004 | 1022 | -18 | B6 | 725 | N804JB | JFK | BQN | 183 | 1576 | 5 | 45 | 2013-01-01 05:00:00 |
| 2013 | 1 | 1 | 554 | 600 | -6 | 812 | 837 | -25 | DL | 461 | N668DN | LGA | ATL | 116 | 762 | 6 | 0 | 2013-01-01 06:00:00 |
| 2013 | 1 | 1 | 554 | 558 | -4 | 740 | 728 | 12 | UA | 1696 | N39463 | EWB | ORD | 150 | 719 | 5 | 58 | 2013-01-01 05:00:00 |

Exercise 16

Order your twoLast set by Borough.

Notice also how we can select columns to focus on:

```
head(select(flights, year:day, dep_time)) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), font_size = 9)
```

| year | month | day | dep_time |
|------|-------|-----|----------|
| 2013 | 1 | 1 | 517 |
| 2013 | 1 | 1 | 533 |
| 2013 | 1 | 1 | 542 |
| 2013 | 1 | 1 | 544 |
| 2013 | 1 | 1 | 554 |
| 2013 | 1 | 1 | 554 |

Now we put some things together. To do so, we first select certain columns from the flights dataset, and then use the pipe to pass the result to create new columns with mutate. The final result is called flightsMutated.

```
flightsMutated <- select(flights, year:day, ends_with("delay"),
  distance, air_time) %>%
mutate(gain = arr_delay - dep_delay,
  hours = air_time/60,
  gain_per_hour = gain/hours)
head(flightsMutated) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)
```

| year | month | day | dep_delay | arr_delay | distance | air_time | gain | hours | gain_per_hour |
|------|-------|-----|-----------|-----------|----------|----------|------|----------|---------------|
| 2013 | 1 | 1 | 2 | 11 | 1400 | 227 | 9 | 3.783333 | 2.378855 |
| 2013 | 1 | 1 | 4 | 20 | 1416 | 227 | 16 | 3.783333 | 4.229075 |
| 2013 | 1 | 1 | 2 | 33 | 1089 | 160 | 31 | 2.666667 | 11.625000 |
| 2013 | 1 | 1 | -1 | -18 | 1576 | 183 | -17 | 3.050000 | -5.573771 |
| 2013 | 1 | 1 | -6 | -25 | 762 | 116 | -19 | 1.933333 | -9.827586 |
| 2013 | 1 | 1 | -4 | 12 | 719 | 150 | 16 | 2.500000 | 6.400000 |

Quite usefully we can first group data and then summarize. In this case we first group by year, month and day, and then order by mean delay on a given day.

```
byDay <- group_by(flights, year, month, day)
delayByDay <- summarize(byDay, delay = mean(dep_delay, na.rm = TRUE))
```

```
## 'summarise()' has grouped output by 'year', 'month'. You can override using the
## '.groups' argument.
```

```
head(delayByDay) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)
```

| year | month | day | delay |
|------|-------|-----|-----------|
| 2013 | 1 | 1 | 11.548926 |
| 2013 | 1 | 2 | 13.858823 |
| 2013 | 1 | 3 | 10.987832 |
| 2013 | 1 | 4 | 8.951595 |
| 2013 | 1 | 5 | 5.732218 |
| 2013 | 1 | 6 | 7.148014 |

Exercise 17

Use this dataset to create a line plot of mean daily delay against day of the year (1:365). Use alpha. Note some increase in the holiday season and in winter.

Now, notice how we can do a few things in a step-wise manner. This time we group by destination first, then add three columns: the number of flights, mean distance, and mean delay. Then we eliminate those which had at most 20 flights, and "HNL" because flights to Honolulu are unusual and we don't care about them.

```
byDest <- group_by(flights, dest)
delay <- summarize(byDest, count=n(),
  dist=mean(distance, na.rm=TRUE),
  delay = mean(arr_delay, na.rm=TRUE))
delay <- filter(delay, count>20, dest != "HNL")
head(delay) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)
```

| dest | count | dist | delay |
|------|-------|-----------|-----------|
| ABQ | 254 | 1826.0000 | 4.381890 |
| ACK | 265 | 199.0000 | 4.852273 |
| ALB | 439 | 143.0000 | 14.397129 |
| ATL | 17215 | 757.1082 | 11.300113 |
| AUS | 2439 | 1514.2530 | 6.019909 |
| AVL | 275 | 583.5818 | 8.003831 |

The same things can be achieved by means of a pipe, which makes this shorter and more transparent:

```

delay2 <- flights %>%
  group_by(dest) %>%
  summarize(count = n(),
            dist = mean(distance, na.rm = TRUE),
            delay = mean(arr_delay, na.rm = TRUE) ) %>%
  filter(count > 20, dest != "HNL")
head(delay2) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)

```

| dest | count | dist | delay |
|------|-------|-----------|-----------|
| ABQ | 254 | 1826.0000 | 4.381890 |
| ACK | 265 | 199.0000 | 4.852273 |
| ALB | 439 | 143.0000 | 14.397129 |
| ATL | 17215 | 757.1082 | 11.300113 |
| AUS | 2439 | 1514.2530 | 6.019909 |
| AVL | 275 | 583.5818 | 8.003831 |

Exercise 18

Use the `delay` dataset to create a scatterplot of delay against distance, with smoothing. Specify `size = count` in the aesthetic. Remove standard error from smoothing by setting `se = FALSE`, make the line skyblue.

Now, we can pick flights that haven't been canceled, (think `!is.na(dep_delay)`, `!is.na(arr_delay)`), group by tail number, and summarize adding counts and mean delays.

```

not_cancelled <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay))
delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarize(
    n=n(),
    delay = mean(arr_delay)
  )
head(delays) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  kable_styling(font_size = 9)

```

| tailnum | n | delay |
|---------|-----|-----------|
| D942DN | 4 | 31.500000 |
| N0EGMQ | 352 | 9.982954 |
| N10156 | 145 | 12.717241 |
| N102UW | 48 | 2.937500 |
| N103US | 46 | -6.934783 |
| N104UW | 46 | 1.804348 |

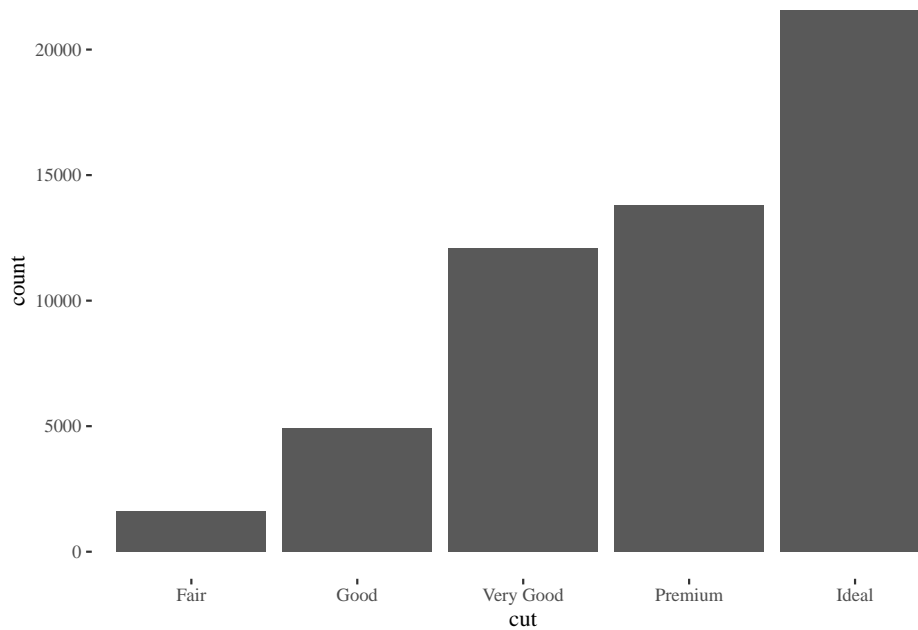
1.5 Counts and barplots

These are used usually to display counts or statistics against a categorical variable. We'll take a look at the `diamonds` dataset, which is built in. Feel free to check the help file and look at the head of the dataset. Let's prepare a barplot picturing the counts per cut type.

```

ggplot(diamonds) +
  geom_bar(aes(x = cut)) +
  theme_tufte()

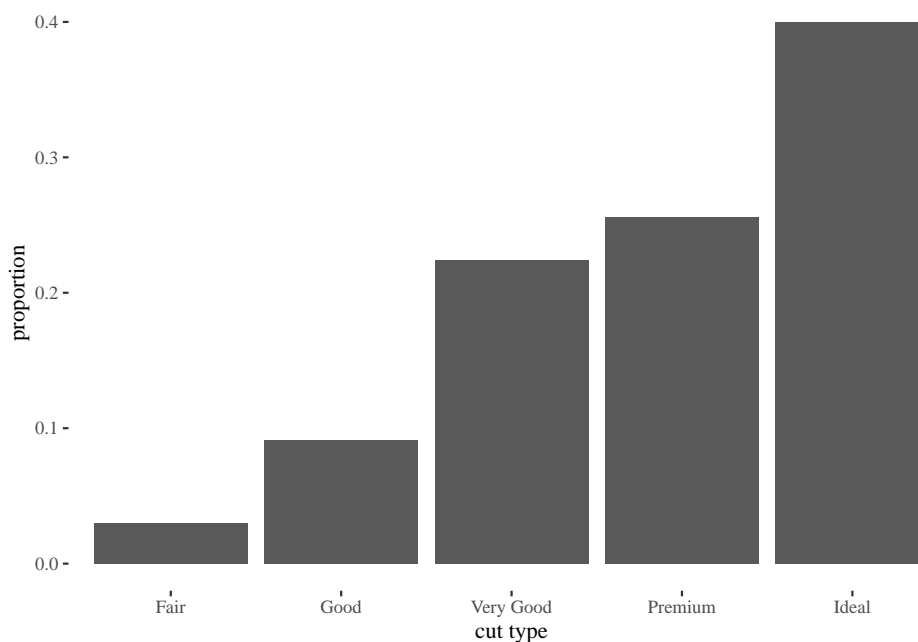
```



Alternatively, we can focus on proportions instead. Note how we can manually edit axis labels.

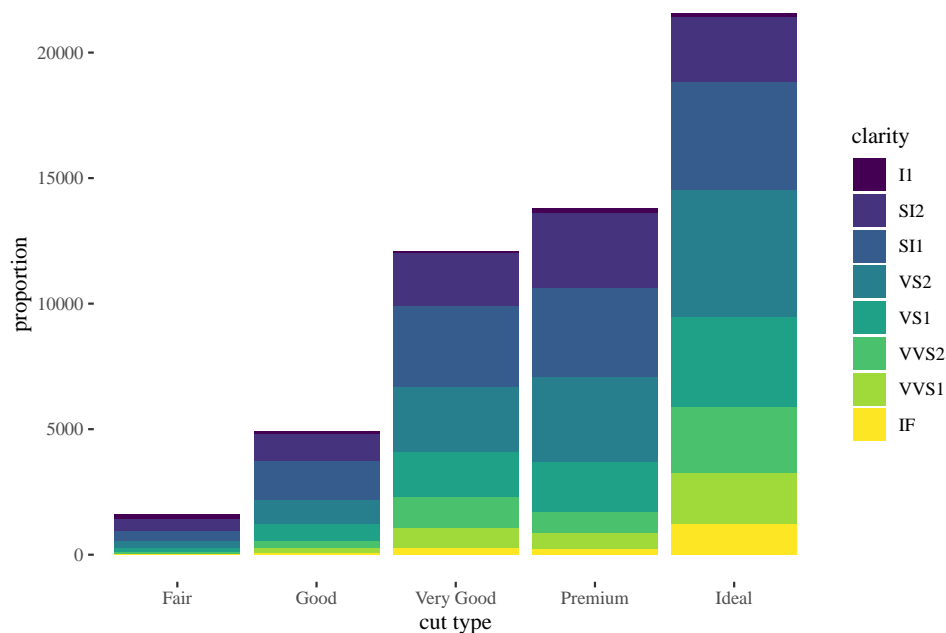
```
ggplot(diamonds) +
  geom_bar(aes(x = cut, y = ..prop.., group = 1)) +
  theme_tufte() +
  xlab("cut type") +
  ylab("proportion")
```

Warning: The dot-dot notation ('..prop..') was deprecated in ggplot2 3.4.0.
i Please use 'after_stat(prop)' instead.



We can also fill with color by clarity:

```
ggplot(diamonds) +
  geom_bar(aes(x = cut, fill = clarity)) +
  theme_tufte() +
  xlab("cut type") +
  ylab("proportion")
```



Now, let's look at data from General Social Survey (google if you don't know what this is), this is in the library **forcats**. Feel free to check the help file for this dataset.

```
head(gss_cat) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  kable_styling(font_size = 14)
```

| year | marital | age | race | rincome | partyid | relig | denom | tvhours |
|------|---------------|-----|-------|-----------------|--------------------|--------------------|------------------|---------|
| 2000 | Never married | 26 | White | \$8000 to 9999 | Ind,near rep | Protestant | Southern baptist | 12 |
| 2000 | Divorced | 48 | White | \$8000 to 9999 | Not str republican | Protestant | Baptist-dk which | NA |
| 2000 | Widowed | 67 | White | Not applicable | Independent | Protestant | No denomination | 2 |
| 2000 | Never married | 39 | White | Not applicable | Ind,near rep | Orthodox-christian | Not applicable | 4 |
| 2000 | Divorced | 25 | White | Not applicable | Not str democrat | None | Not applicable | 1 |
| 2000 | Married | 25 | White | \$20000 - 24999 | Strong democrat | Protestant | Southern baptist | NA |

One thing we might be interested in is counts by a single categorical variable. For instance, we can use **count** from package **tidyverse**:

```
gss_cat %>% count(race) %>%
  kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), font_size = 9)
```

| race | n |
|-------|-------|
| Other | 1959 |
| Black | 3129 |
| White | 16395 |

Alternatively, we can use the native **table** method, but it's a bit more messy:

```
table(gss_cat$race) %>%
  kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), font_size = 9)
```

| Var1 | Freq |
|----------------|-------|
| Other | 1959 |
| Black | 3129 |
| White | 16395 |
| Not applicable | 0 |

Exercise 19

Use the dataset to prepare a barplot by race.

Exercise 20

Group `gss_cat` by religion, and summarize creating mean age, mean tvhours and count. Use pipe, `group_by`, and `summarize`, create a new object called `relig`.

Exercise 21

Use the object you created and `geom_point` to plot religion vs. tvhours.

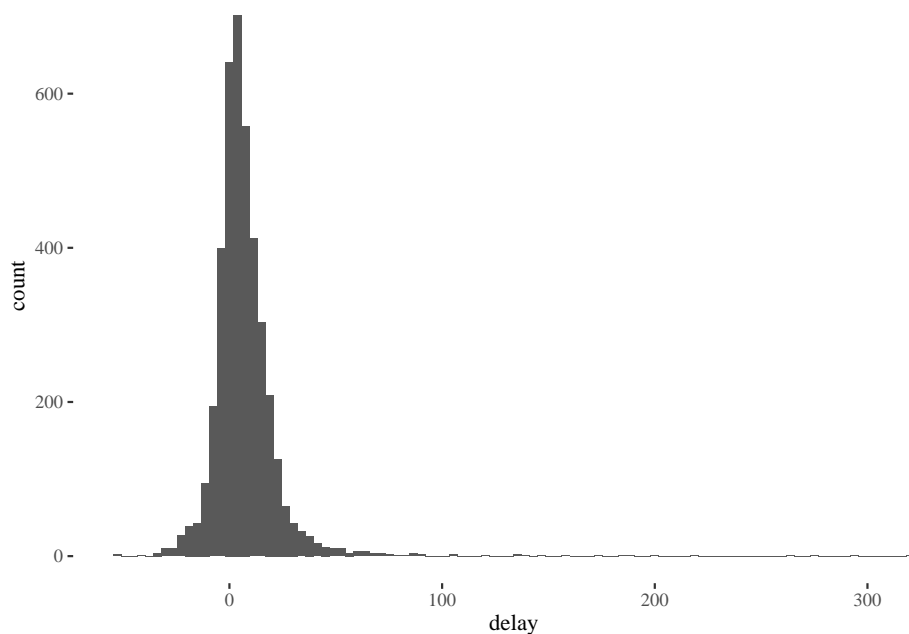
Exercise 22

Do the same for reported income level and tvhours.

1.6 Histograms

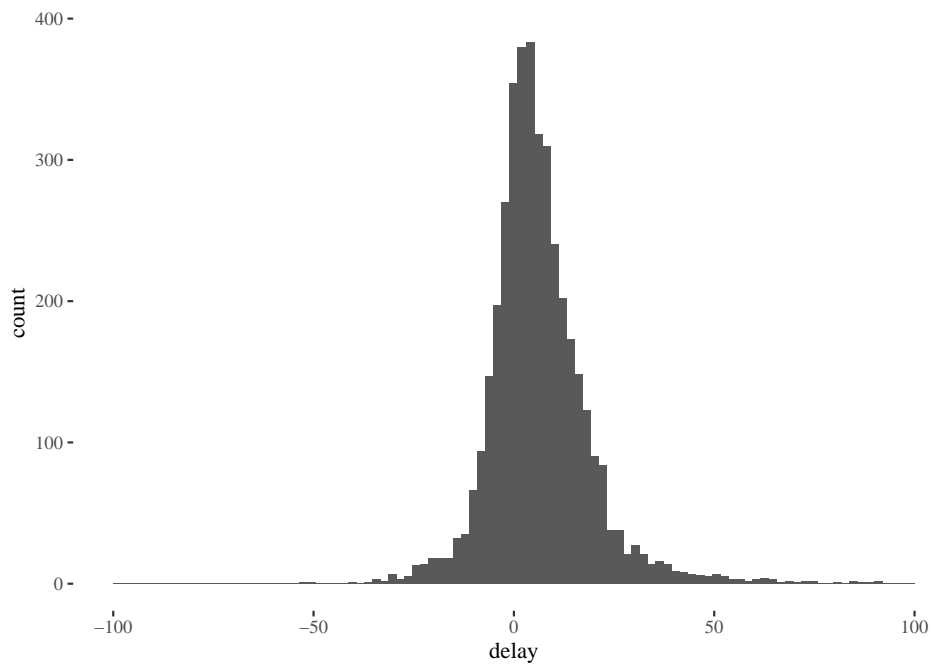
For a continuous variable we can first bin values in a certain number of intervals, and then plot a histogram. Like this (we're using mean delays by tail number, 10 bins):

```
ggplot(delays, aes(x = delay)) +  
  geom_histogram(bins = 100) +  
  theme_tufte()
```



We can focus on a certain range along the x axis. Like this:

```
ggplot(delays, aes(x = delay)) +  
  geom_histogram(bins = 100) +  
  theme_tufte() +  
  xlim(c(-100, 100))
```



1.7 Miniproject: water distribution in New Thariyal

This is a dataset based on interviews with the inhabitants of a small village in India, New Thariyal, who have been experiences problems with water distribution.



The project has been run by Live In Labs:

Identify
ent Challenges

Admissions Open for Fully Funded Ph. D. in Sustainable Development

APPLY NOW

ABOUT

Live-in-Labs® is a multidisciplinary experiential learning program that breaks classroom and lab barriers by applying learned theory in real-world settings. This credit-based academic program draws on principles of lean research for the development and refinement of sustainable solutions for

PROJECTS

With over 150 projects in 21 states across India, Live-in-Labs® participants have touched the lives of approximately 60,000 rural residents during the past several years. With participation from over 30 institutions around the world and nearly 50 departments, schools and centers at Amrita, students and

within the Data Science for Social Good project.

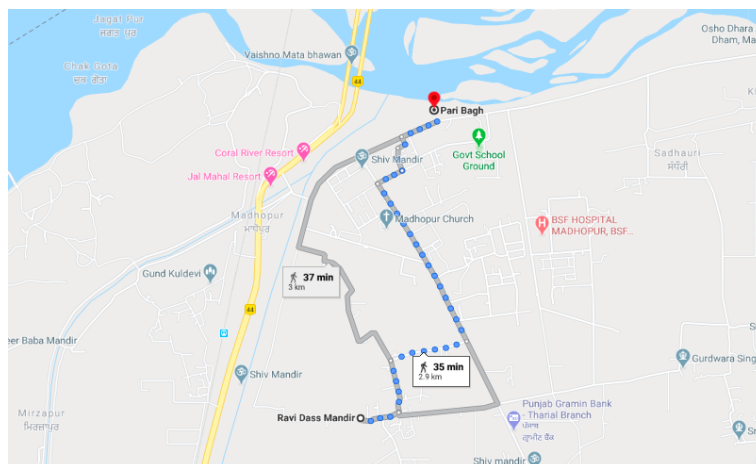


Life there looks like this:

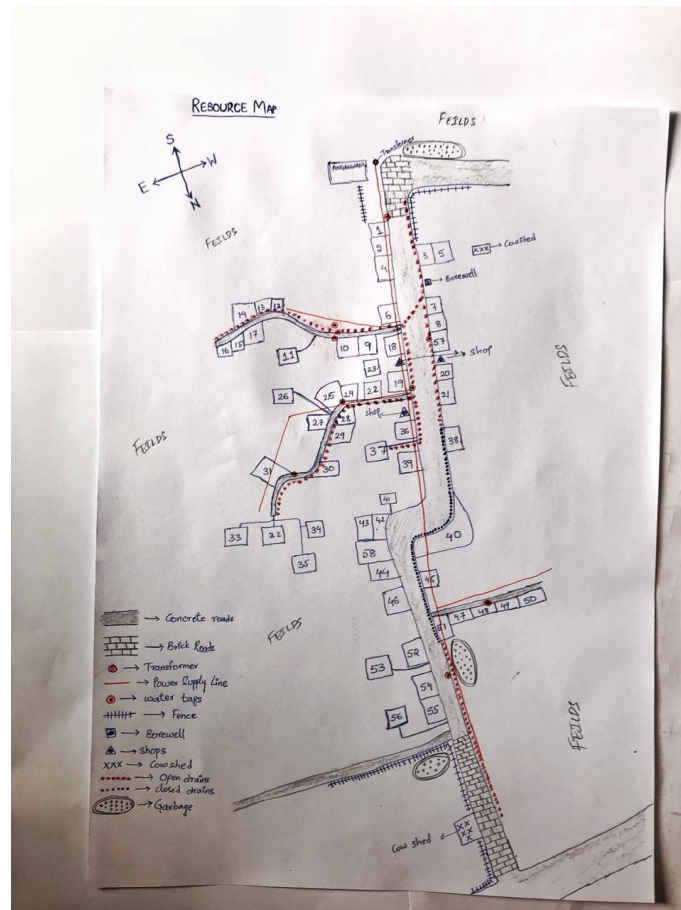




Quite importantly, if you want to bring water from the river, it's a bit of a walk:



And the plan of the village is:



The important thing to notice is that the borewell is in the south. Houses in the north have difficulty getting water if the pressure is low, and some houses illegally use their own pumps for assistance.

First, import and inspect the column names from dataset called `WaterSurveyData.csv` in the `teachingData` folder, using `read.csv()` (with `header = TRUE`) and `names()`.

```
WaterSurvey <- read.csv("teachingData/CsvFiles/WaterSurveyDataset.csv",
                        header = TRUE)
```

```
names(WaterSurvey)
```

```
## [1] "House.No."
## [2] "Sector.No."
## [3] "Choice.of.Storage"
## [4] "Capacity.ltrs.number..For..Cans...Buckets...we.have.encoded.count.of.both"
## [5] "Motor.."
## [6] "No..of.people"
## [7] "Borewell.."
## [8] "Thariyal.."
## [9] "Major.Source"
```

One issue is that names are a bit ugly and long. Let's change them. Use `names(...) <- c("name",...)` Our wishlist is:

- House.No. -> House
- Sector.No. -> Sector
- Choice.of.Storage -> Storage
- Capacity... -> Capacity
- Motor.. -> Pump
- No..of..people -> Inhabitants
- Borewell.. -> Borewell
- Thariyal.. -> Thariyal
- Major.Source -> MajorSource

Make the changes and inspect the head of the dataset now.

```
names(WaterSurvey) <- c("House", "Sector", "Storage", "Capacity",
                        "Pump", "Inhabitants", "Borewell", "Thariyal",
                        "MajorSource")
head(WaterSurvey) %>%
  kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), font_size = 9)
```

| House | Sector | Storage | Capacity | Pump | Inhabitants | Borewell | Thariyal | MajorSource |
|-------|--------|-----------------------|----------|------|-------------|----------|----------|-------------|
| 1 | 3 | Tank | 500 | 1 | 13 | 1 | 1 | B |
| 2 | 3 | Buckets | 12 | 0 | 12 | 1 | 1 | B |
| 3 | 3 | Controls the borewell | - | 0 | 17 | 1 | 1 | B |
| 4 | 3 | Buckets | 15 | 0 | 7 | 1 | 0 | B |
| 6 | 3 | Buckets | 15 | 0 | 5 | 1 | 0 | B |
| 7 | 3 | Buckets | 10 | 0 | 2 | 1 | 0 | B |

Things to note:

- There are “controls the borewell” and NAs in capacity
- Capacity is a factor, In fact, it has a non-uniform because it mixes litres with numbers of buckets and tanks. The measure is litres for tank or “control”, but counts of units for other sources/containers).
- Cans (metal barrels) have 50l, buckets 20l.

We need to clean the structure of the dataset. Check it with `str`. Notice that Pump, Borewell, and Thariyal should be logical values, make them so using `as.logical()` and assignment in **R**. Sector should be a factor, make it so using `as.factor()`. Check the structure again.

```
#check structure
str(WaterSurvey)
```

```
## 'data.frame': 47 obs. of 9 variables:
## $ House : int 1 2 3 4 6 7 10 11 12 13 ...
## $ Sector : int 3 3 3 3 3 3 3 3 3 3 ...
## $ Storage : chr "Tank" "Buckets" "Controls the borewell" "Buckets" ...
## $ Capacity : chr "500" "12" "-" "15" ...
## $ Pump : int 1 0 0 0 0 0 1 0 0 0 ...
## $ Inhabitants: int 13 12 17 7 5 2 4 7 6 10 ...
## $ Borewell : int 1 1 1 1 1 1 0 1 0 1 ...
## $ Thariyal : int 1 1 1 0 0 0 1 1 1 0 ...
## $ MajorSource: chr "B" "B" "B" "B" ...
```

```
#Notice the integer vectors which should be logical
WaterSurvey$Pump <- as.logical(WaterSurvey$Pump)
WaterSurvey$Borewell <- as.logical(WaterSurvey$Borewell)
WaterSurvey$Thariyal <- as.logical(WaterSurvey$Thariyal)
#Notice sectors are a factor
WaterSurvey$Sector <- as.factor(WaterSurvey$Sector)
WaterSurvey$Storage <- as.factor(WaterSurvey$Storage)
str(WaterSurvey)
```

```
## 'data.frame': 47 obs. of 9 variables:
## $ House : int 1 2 3 4 6 7 10 11 12 13 ...
## $ Sector : Factor w/ 2 levels "2","3": 2 2 2 2 2 2 2 2 2 2 ...
## $ Storage : Factor w/ 6 levels "Buckets","Cans",...: 6 1 4 1 1 1 6 3 2 1 ...
## $ Capacity : chr "500" "12" "-" "15" ...
## $ Pump : logi TRUE FALSE FALSE FALSE FALSE FALSE ...
## $ Inhabitants: int 13 12 17 7 5 2 4 7 6 10 ...
## $ Borewell : logi TRUE TRUE TRUE TRUE TRUE TRUE ...
## $ Thariyal : logi TRUE TRUE TRUE FALSE FALSE FALSE ...
## $ MajorSource: chr "B" "B" "B" "B" ...
```

Now, we need to have a uniform measure of storage. We have transformed this variable into a factor, let's check its levels with `levels`. Notice that “Tank” contains a redundant space.


```
levels(WaterSurvey$Storage)
```

```
## [1] "Buckets"          "Cans"              "Cans + Buckets"
## [4] "Controls the borewell" "Sump"              "Tank "
```

Now I'll walk you through cleaning up the storage variable.

```
#this will be our final vector,
#for now it's empty
Amount <- numeric(nrow(WaterSurvey))

#logical vector locating Tanks
tanks <- WaterSurvey$Storage == "Tank "

#for tanks we extract their capacities
WaterSurvey$Capacity <- as.factor(WaterSurvey$Capacity)

TankCapacities <- as.numeric(levels(WaterSurvey$Capacity[tanks])
                             ) [WaterSurvey$Capacity[tanks]]

#and we write them down to our target vector
Amount[tanks] <- TankCapacities

#locate cans
cans <- WaterSurvey$Storage == "Cans"

#for cans/barrels we multiply their counts by 50 liters

CansCapacities <- as.numeric(levels(
  WaterSurvey$Capacity[cans])) [WaterSurvey$Capacity[cans]]*50

#save to our target vector
Amount[cans] <- CansCapacities

#locate buckets
buckets <- WaterSurvey$Storage == "Buckets"

#for bucket we multiply counts by 20 liters
BucketsCapacities <- as.numeric(levels(
  WaterSurvey$Capacity[buckets])) [WaterSurvey$Capacity[buckets]]*20

#add to our target vector
Amount[buckets] <- BucketsCapacities

#For sumps (large underground tanks under the houses)
# we use their capacity in liters

#locate sumps
sumps <- WaterSurvey$Storage == "Sump"

#extract their capacities
SumpCapacities <- as.numeric(levels(
  WaterSurvey$Capacity[sumps])) [WaterSurvey$Capacity[sumps]]

#write in our targed vector
Amount[sumps] <- SumpCapacities

#check what we missed
WaterSurvey[Amount==0,] %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"),font_size = 9)
```

| | House | Sector | Storage | Capacity | Pump | Inhabitants | Borewell | Thariyal | MajorSource |
|----|-------|--------|-----------------------|----------|-------|-------------|----------|----------|-------------|
| 3 | 3 | 3 | Controls the borewell | - | FALSE | 17 | TRUE | TRUE | B |
| 8 | 11 | 3 | Cans + Buckets | 1,10 | FALSE | 7 | TRUE | TRUE | T |
| 20 | 23 | 3 | Cans + Buckets | 2,10 | FALSE | 5 | TRUE | FALSE | B |
| 23 | 26 | 3 | Cans + Buckets | 3,3 | FALSE | 3 | TRUE | TRUE | B |
| 28 | 32 | 3 | Cans + Buckets | 1,2 | FALSE | 4 | TRUE | FALSE | B |
| 37 | 43 | 2 | Cans + Buckets | 7,1 | TRUE | 11 | FALSE | TRUE | T |

```
#finish manually: Controls the borewell is, say 10000
Amount[WaterSurvey$Storage == "Controls the borewell"] <- 10000
Amount[8] <- 50 + 10 *20
Amount[20] <- 2*50 + 10*20
Amount[23] <- 3*50 + 3*20
Amount[28] <- 50 + 2*20
Amount[37] <- 7*50 + 20
```

```
#inspect the vector:
#head(WaterSurvey$Amount)
Amount
```

```
## [1] 500 240 10000 300 300 200 500 250 150 100 500 360
## [13] 200 80 140 100 60 200 80 252 120 140 210 350
## [25] 80 60 12743 90 1000 120 80 80 4247 100 100 120
## [37] 370 500 100 80 80 350 300 200 500 350 500
```

```
# And finally let's add the Amount vector as a column to the data frame
head(cbind(WaterSurvey, Amount)) %>% kable("latex", booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), font_size = 9)
```

| House | Sector | Storage | Capacity | Pump | Inhabitants | Borewell | Thariyal | MajorSource | Amount |
|-------|--------|-----------------------|----------|-------|-------------|----------|----------|-------------|--------|
| 1 | 3 | Tank | 500 | TRUE | 13 | TRUE | TRUE | B | 500 |
| 2 | 3 | Buckets | 12 | FALSE | 12 | TRUE | TRUE | B | 240 |
| 3 | 3 | Controls the borewell | - | FALSE | 17 | TRUE | TRUE | B | 10000 |
| 4 | 3 | Buckets | 15 | FALSE | 7 | TRUE | FALSE | B | 300 |
| 6 | 3 | Buckets | 15 | FALSE | 5 | TRUE | FALSE | B | 300 |
| 7 | 3 | Buckets | 10 | FALSE | 2 | TRUE | FALSE | B | 200 |

The result of this process is a cleaned up data frame with the column that fairly represents the amount of water for a household. Data in such form can be used for a further analysis.