ETC3250 2018 - Lab 1 and 2 solutions

Souhaib Ben Taieb

27 February 2018

Getting up and running with the computer:

- R and RStudio
- RStudio Projects
- RMarkdown
- R syntax and basic functions

What is R?

From Wikipedia: "R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis."

R is free to use and has more than 11,000 user contributed add-on packages on the Comprehensive R Archive Network (CRAN).

What is RStudio?

From Julie Lowndes:

If R were an airplane, RStudio would be the airport, providing many, many supporting services that make it easier for you, the pilot, to take off and go to awesome places. Sure, you can fly an airplane without an airport, but having those runways and supporting infrastructure is a game-changer.

The RStudio integrated development environment (IDE) has multiple components including:

1. Source editor (to edit your scripts):

- Docking station for multiple files,
- Useful shortcuts ("Knit"),
- Highlighting/Tab-completion,
- Code-checking (R, HTML, JS),
- Debugging features

2. Console window (to run your scripts, to test small pieces of code):

- Highlighting/Tab-completion,
- Search recent commands

3. Other tabs/panes:

- Graphics,
- R documentation,
- Environment pane,
- File system navigation/access,
- Tools for package development, git, etc

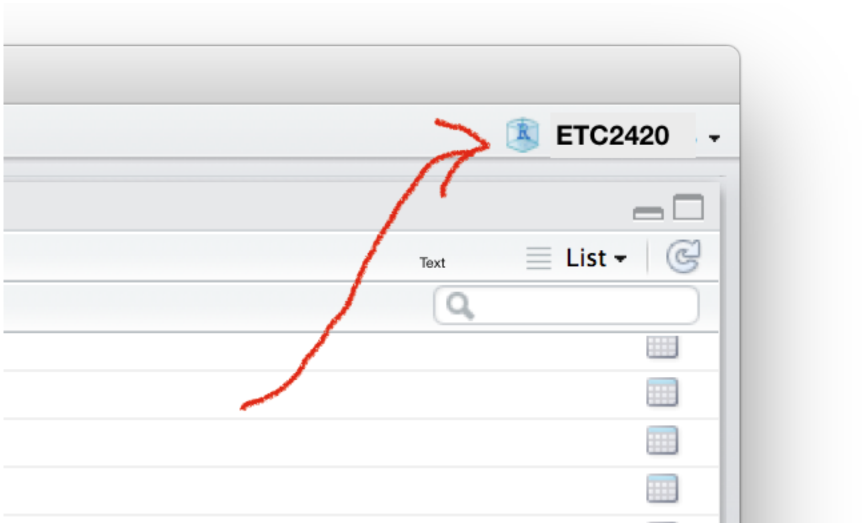


Figure 1: Using projects to organise your work

RStudio Projects

- Project directories keep your work organized since you will keep your data, your code, your results all located in one place.
- For the unit ETC2420, I have created a project on my laptop called `ETC2420`. Note that the name of the current project can be seen at the top right of the RStudio window.
- Each time you start RStudio for this class, be sure to open the right project.

Exercise 1

Create a project for this unit, in the directory.

- File -> New Project -> Existing Directory -> Empty Project

Exercise 2

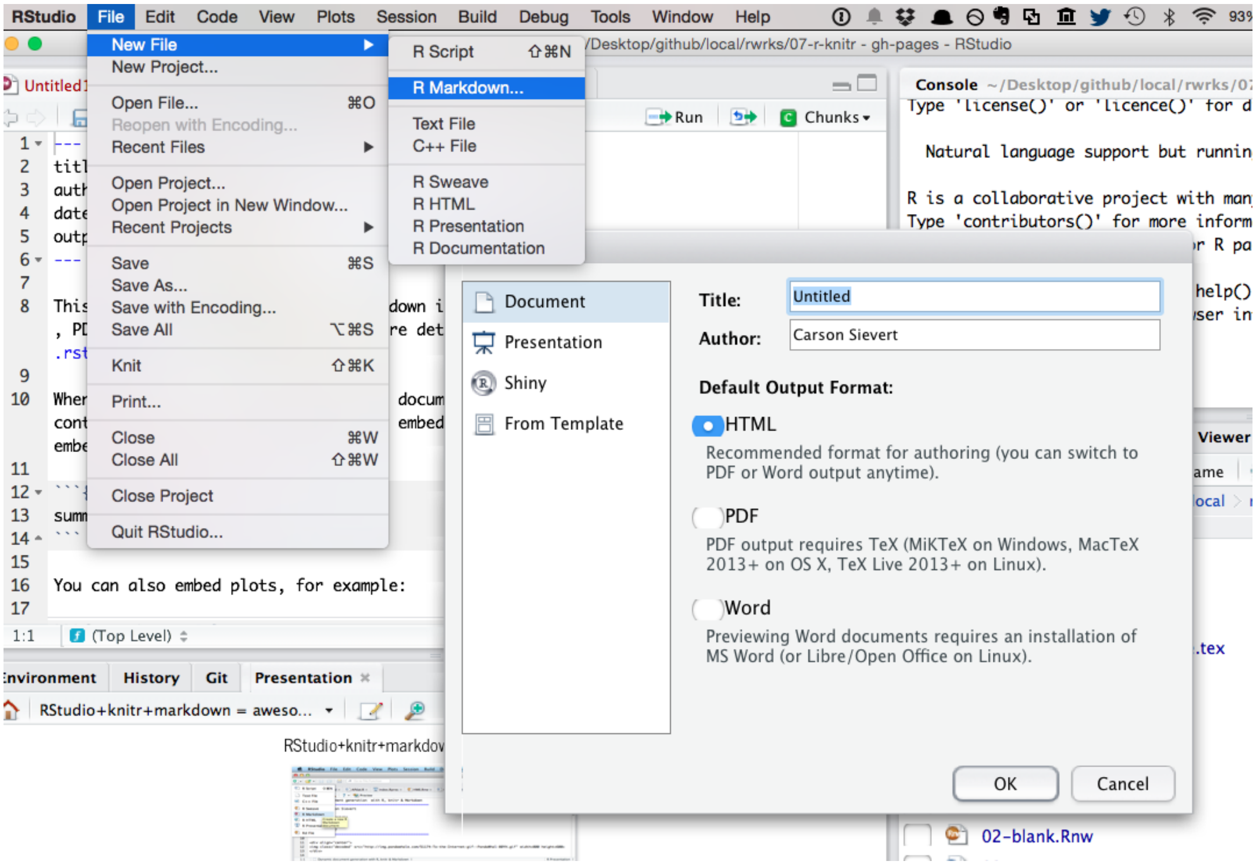
Open a new Rmarkdown document. You are going to want to call it `Lab1` (it will automatically get the file extension `.Rmd` when you save it).

- File -> New File -> R Markdown -> OK -> Knit HTML

What is RMarkdown?

- R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R.
- It combines the core syntax of **markdown** (an easy-to-write plain text format) **with embedded R code chunks** that are run so their output can be included in the final document.
- R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.



Title: Untitled

Author: Carson Sievert

Default Output Format:

● HTML

Recommended format for authoring (you can switch to PDF or Word output anytime).

○ PDF

PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word

Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

```
Type 'license()' or 'licence()' for d

 Natural language support but runnin

R is a collaborative project with man
Type 'contributors()' for more inform
```

```
You can also embed plots, for example:
```

Figure 2: Writing and computing with the one document

Equations can be included using LaTeX (https://latex-project.org/) commands like this:

`$$s^2 = \frac{1}{n-1}\sum_{i=1}^n (x_i-\bar{x})^2.$$`

produce

$$s^2 = \frac{1}{n-1}\sum_{i=1}^n (x_i-\bar{x})^2.$$

We can also use inline mathematical symbols such as `$\alpha$` and `$\infty$`, which produce α and ∞, respectively.

For more details on using R Markdown see http://rmarkdown.rstudio.com. Spend a few minutes looking over that website before continuing with this document.

Exercise 3

Look at the text in the `lab1.Rmd` document.

- What is R code?
- How does `knitr` know that this is code to be run?
- Using the RStudio IDE, work out how to run a chunk of code. Run this chunk, and then run the next chunk.
- Using the RStudio IDE, how do you run just one line of R code?
- Using the RStudio IDE, how do you highlight and run multiple lines of code?
- What happens if you try to run a line that starts with “““{r}”? Or try to run a line of regular text from the document?
- Using the RStudio IDE, `knit` the document into a Word document.

Some R Basics

- Type and Figure out what each of the following command is doing:

  ```
  (100+2)/3
  5*10^2
  1/0
  0/0
  (0i-9)^(1/2)
  sqrt(2*max(-10,0.2,4.5))+100
  x <- sqrt(2*max(-10,0.2,4.5))+100
  x
  log(100)
  log(100,base=10)
  ```

- Check that these are equivalent: `y <- 100`, `y = 100` and `100 -> y`

- Find the help page for the `mean` command, either from the help menu, or by typing one of these: `help(mean)` and `?mean`. Most help pages have examples at the bottom.

- The `summary` command can be applied to almost anything to get a summary of the object. Try `summary(c(1, 3, 3, 4, 8, 8, 6, 7))`

Data Types

- `list`'s are heterogeneous (elements can have different types)
- `data.frame`'s are heterogeneous but elements have same length (`dim` reports the dimensions and `colnames` shows the column names)
- `vector`'s and `matrix`'s are homogeneous (elements have the same type), which would be why `c(1, "2")` ends up being a character string.
- `function`'s can be written to save repeating code again and again
- Try to understand these commands: `class`, `typeof`, `is.numeric`, `is.vector` and `length`
- See Hadley Wickham's online chapters on data structures (http://adv-r.had.co.nz/Data-structures.html) for more

Operations

- Use built-in *vectorized* functions to avoid loops

```
set.seed(1000)
x <- rnorm(6)
x
# [1] -0.44577826 -1.20585657  0.04112631  0.63938841 -0.78655436 -0.38548930
sum(x + 10)
# [1] 57.85684
```

- `R` has rich support for documentation, see `?sum`

- Use `[` to extract elements of a vector.

```
x[1]
# [1] -0.4457783
x[c(T, F, T, T, F, F)]
# [1] -0.44577826  0.04112631  0.63938841
```

- Extract *named* elements with `$`, `[[`, and/or `[`

```
x <- list(
  a = 10,
  b = c(1, "2")
)
x$a
# [1] 10
x[["a"]]
# [1] 10
x["a"]
# $a
# [1] 10
```

Examining ‘structure’

- `str()` is a very useful `R` function. It shows you the “structure” of (almost) *any* R object (and *everything* in R is an object!!!)

```
str(x)
# List of 2
#  $ a: num 10
#  $ b: chr [1:2] "1" "2"
```

Missing Values

- `NA` is the indicator of a missing value in R
- Most functions have options for handling missings

```
x <- c(50, 12, NA, 20)
mean(x)
# [1] NA
mean(x, na.rm=TRUE)
# [1] 27.33333
```

Counting Categories

- the `table` function can be used to tabulate numbers

```
table(c(1, 2, 3, 1, 2, 8, 1, 4, 2))
#
# 1 2 3 4 8
# 3 3 1 1 1
```

Functions

One of the powerful aspects of R is to build on the reproducibility. If you are going to do the same analysis over and over again, compile these operations into a function that you can then apply to different data sets.

```
average <- function(x)
{
  return(sum(x)/length(x))
}

y1 <- c(1,2,3,4,5,6)
average(y1)
# [1] 3.5

y2 <- c(1, 9, 4, 4, 0, 1, 15)
average(y2)
# [1] 4.857143
```

Now write a function to compute the mode of some vector, and confirm that it returns `4` when applied on `y <- c(1, 1, 2, 4, 4, 4, 9, 4, 4, 8)`

Exercise 4

- What's an R package?
- How do you install a package?
- How does the library() function relates to a package?
- How often do you load a package?
- Install and load the package ISLR

Getting data

Data can be found in R packages

```
library(dplyr)
data(economics, package = "ggplot2")
# data frames are essentially a list of vectors
glimpse(economics)
# Observations: 574
# Variables: 6
# $ date     <date> 1967-07-01, 1967-08-01, 1967-09-01, 1967-10-01, 1967...
# $ pce      <dbl> 507.4, 510.5, 516.3, 512.9, 518.1, 525.8, 531.5, 534....
# $ pop      <int> 198712, 198911, 199113, 199311, 199498, 199657, 19980...
# $ psavert  <dbl> 12.5, 12.5, 11.7, 12.5, 12.5, 12.1, 11.7, 12.2, 11.6,...
# $ uempmed  <dbl> 4.5, 4.7, 4.6, 4.9, 4.7, 4.8, 5.1, 4.5, 4.1, 4.6, 4.4...
# $ unemploy <int> 2944, 2945, 2958, 3143, 3066, 3018, 2878, 3001, 2877,...
```

These are not usually kept up to date but are good for practicing your analysis skills on.

Or in their own packages

```
library(gapminder)
glimpse(gapminder)
# Observations: 1,704
# Variables: 6
# $ country   <fctr> Afghanistan, Afghanistan, Afghanistan, Afghanistan,...
# $ continent <fctr> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asi...
# $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992...
# $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.8...
# $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 1488...
# $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 78...
```

I primarily use the readr package for reading data now. It mimics the base R reading functions but is implemented in C so reads large files quickly, and it also attempts to identify the types of variables.

```
library(readr)
ped <- read_csv("https://raw.githubusercontent.com/bsouhaib/BA2018/master/data/Pedestrian_Counts.csv")
glimpse(ped)
# Observations: 1,392,618
# Variables: 4
# $ Date_Time     <chr> "01-MAY-2009 00:00", "01-MAY-2009 00:00", "01-MA...
# $ Sensor_ID     <int> 4, 17, 18, 16, 2, 1, 13, 15, 9, 10, 12, 11, 5, 6...
# $ Sensor_Name   <chr> "Town Hall (West)", "Collins Place (South)", "Co...
# $ Hourly_Counts <int> 209, 28, 36, 22, 52, 53, 17, 124, 5, 8, 2, 5, 15...
```

You can pull data together yourself, or look at data compiled by someone else.

Question 1

- Look at the `economics` data in the `ggplot2` package. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Solution:

There could be many possible questions that might be answered by this data. Examples include these ones:

- Does the personal savings rate dip when unemployment is high?
- Is there a seasonal effect in unemployment?
- Is population increasing?

Question 2

- Read the documentation for `gapminder` data. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Solution:

There could be many possible questions that might be answered by this data. Examples include these ones:

- Is life expectancy positively associated with gdp percapita?
- Is life expectancy increasing over time?
- Is the trend in life expectancy similar across all countries?

Question 3

- Read the documentation for `pedestrian sensor` data. Can you think of two questions you could answer using these variables?
- Write these into your `.Rmd` file.

Solution:

There could be many possible questions that might be answered by this data. Examples include these ones:

- What places in the city see the most pedestrians?
- What times would be rush hours on week days?
- Can you see the Wednesday night markets location and time based on pedestrian traffic?
- Is White Night visible in terms of pedestrian traffic?
- Are more people out and about in summer than in winter?

Question 4

1. Read in the OECD PISA data (available at https://github.com/bsouhaib/BA2018/raw/master/data/student_sub.rds)
2. Tabulate the countries (CNT)
3. Extract the values for Australia (AUS) and Shanghai (QCN)
4. Compute the average and standard deviation of the reading scores (PV1READ), for each country
5. Write a few sentences explaining what you learn about reading in these two countries.

Solution

```
student2012.sub <- readRDS("../../data/student_sub.rds")
student2012.sub <- readRDS(gzcon(url("https://github.com/bsouhaib/BA2018/raw/master/data/student_sub.rds
table(student2012.sub$CNT)
#
#   ARE   AUS   AUT   BEL   BGR   BRA   CAN   CHL   COL   CZE   DEU   DNK
# 11500 14481  4755  8597  5282  5506 21544  6856  9073  5327  5001  7481
#   ESP   EST   FIN   FRA   GBR   HKG   HRV   HUN   IRL   ISR   ITA   JPN
# 10175  4779  8829  4613  4185  4670  5008  4810  5016  5055  5495  6351
#   KOR   MAC   MNE   MYS   NLD   NOR   POL   PRT   QCN   RUS   SGP   SRB
#  5033  5335  4744  5197  4460  4686  4607  5722  5177  5231  5546  4684
#   SVK   SVN   SWE   TAP   TUR   URY   USA
#  4678  5911  4736  6046  4848  5315  4978
australia <- student2012.sub[student2012.sub$CNT=="AUS",]
shanghai <- student2012.sub[student2012.sub$CNT=="QCN",]
mean(australia$PV1READ)
# [1] 500.8453
sd(australia$PV1READ)
# [1] 100.7817
mean(shanghai$PV1READ)
# [1] 567.4197
sd(shanghai$PV1READ)
# [1] 79.91869
```

The reading scores are higher in Shanghai than in Australia by about 67 points. The variation in scores in Australia is higher, with a standard deviation of 100 as opposed to 80 for Shanghai.

Alternative code:

```
library(dplyr)
library(knitr)
library(tidyr)
student2012.sub %>% select(CNT) %>% group_by(CNT) %>% tally()
# # A tibble: 43 x 2
#      CNT     n
#    <chr> <int>
#  1   ARE 11500
#  2   AUS 14481
#  3   AUT  4755
#  4   BEL  8597
#  5   BGR  5282
#  6   BRA  5506
#  7   CAN 21544
#  8   CHL  6856
#  9   COL  9073
# 10   CZE  5327
# # ... with 33 more rows
student2012.sub %>% filter(CNT %in% c("AUS", "QCN")) %>%
  group_by(CNT) %>%
  summarise(m=mean(PV1READ), s=sd(PV1READ)) %>% kable(digits=1)
```

CNT	m	s
AUS	500.8	100.8
QCN	567.4	79.9

Resources

- RStudio cheat sheet
- Q/A site: http://stackoverflow.com
- Dynamic Documents with R and knitr, Yihui Xie,

More exercises

- R is great for matrix calculations:

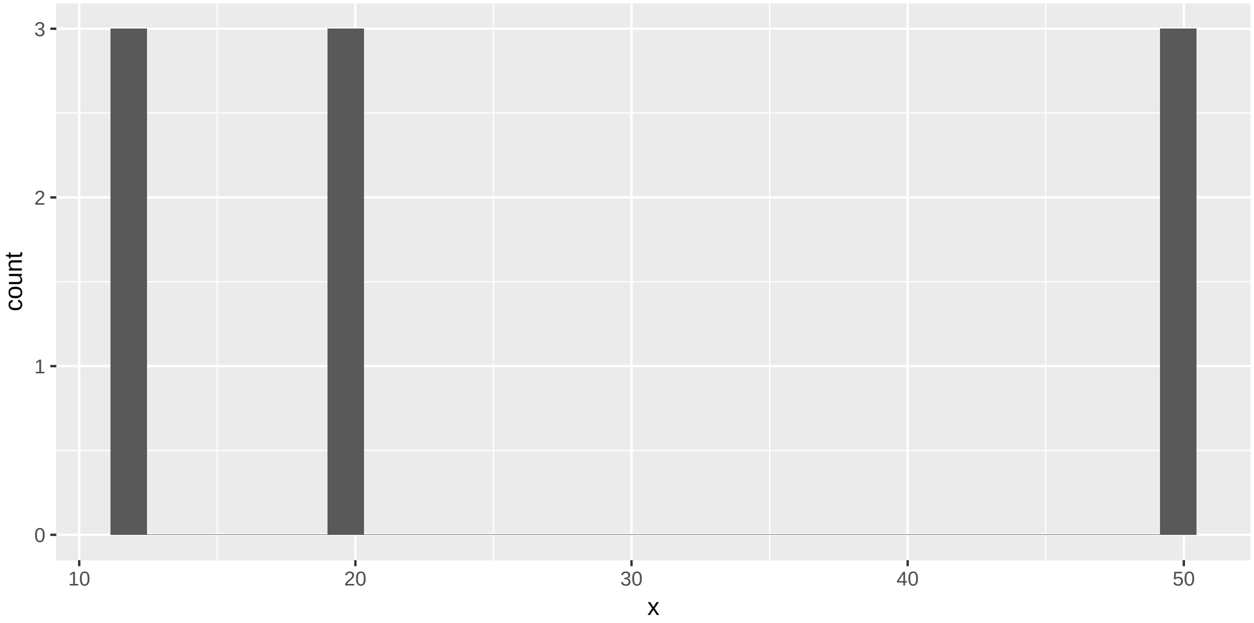
```
X <- matrix(c(3,4,5,2), nrow=2, ncol=2)
t(X)
#      [,1] [,2]
# [1,]    3    4
# [2,]    5    2
Xinv <- solve(X)
X %*% Xinv
#      [,1] [,2]
# [1,]    1    0
# [2,]    0    1

A <- matrix(rnorm(200), nrow=5, ncol=40)
B <- A %*% t(A)
dim(B)
# [1] 5 5
diag(B)
# [1] 32.40283 44.06336 37.12973 33.43034 36.98610
eigen(B)
# $values
# [1] 55.59994 38.03823 34.03742 28.19920 28.13756
#
# $vectors
#             [,1]       [,2]       [,3]       [,4]        [,5]
# [1,]  0.07823187  0.4147787  0.6359504 -0.3550793  0.53974460
# [2,] -0.71630943  0.4208819  0.1030362  0.5466234  0.01859059
# [3,] -0.29577628 -0.7898528  0.2640844  0.1557843  0.44117983
# [4,]  0.29027001 -0.1105043  0.6950648  0.3529438 -0.54391882
# [5,]  0.55591392  0.1214021 -0.1791507  0.6529038  0.46673636
svd(A)
```

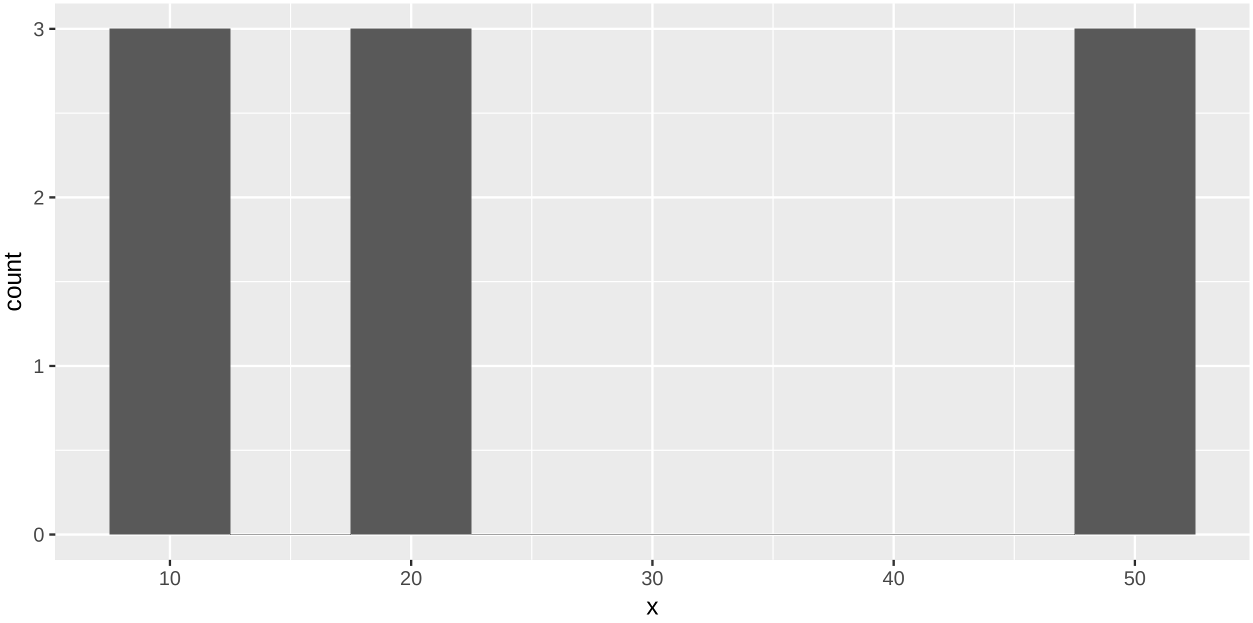
```
# $d
# [1] 7.456537 6.167514 5.834160 5.310292 5.304485
#
# $u
#             [,1]       [,2]       [,3]       [,4]        [,5]
# [1,] -0.07823187  0.4147787 -0.6359504  0.3550793 -0.53974460
# [2,]  0.71630943  0.4208819 -0.1030362 -0.5466234 -0.01859059
# [3,]  0.29577628 -0.7898528 -0.2640844 -0.1557843 -0.44117983
# [4,] -0.29027001 -0.1105043 -0.6950648 -0.3529438  0.54391882
# [5,] -0.55591392  0.1214021  0.1791507 -0.6529038 -0.46673636
#
# $v
#               [,1]        [,2]        [,3]          [,4]         [,5]
#  [1,]  0.200098106  0.02474797  0.17341971  0.1066878804 -0.006918478
#  [2,]  0.052014674  0.01375789  0.22816349  0.0218648906 -0.085911769
#  [3,] -0.287775257  0.32296899  0.13189034 -0.2746117793 -0.058730521
#  [4,]  0.094352275 -0.07033170  0.19049218 -0.2147126769 -0.041763028
#  [5,]  0.041379531 -0.19499058  0.01118734 -0.2253637901  0.277479370
#  [6,] -0.039009965  0.09377122  0.17352266  0.2272478976 -0.227766027
#  [7,]  0.167078726  0.15649910  0.19956145  0.0416260706  0.002492023
#  [8,] -0.099453028  0.14587584 -0.08451175 -0.0890335611 -0.291599520
#  [9,] -0.027811958  0.22158076  0.15852049  0.0781083047 -0.030552785
# [10,] -0.064851307  0.09807866  0.17470405 -0.2088427044  0.119257556
# [11,]  0.142994249 -0.40008219  0.19330202 -0.2363837791 -0.167470087
# [12,]  0.211486227 -0.09216712  0.01683007 -0.1193563082 -0.088823094
# [13,]  0.233707080  0.16424741  0.02631102  0.0045267029  0.268331956
# [14,]  0.116145811 -0.12015418 -0.13512613  0.0413285356 -0.159551415
# [15,]  0.060965991  0.03028393 -0.07999491 -0.3197885956 -0.151137319
# [16,]  0.011001116 -0.19474818 -0.13619836  0.0005968125 -0.085966019
# [17,] -0.490987033 -0.02709454 -0.08906072 -0.1138595058  0.139387145
# [18,]  0.027350631 -0.06599522  0.23529252 -0.2218992380  0.155363523
# [19,]  0.083085393  0.07283543 -0.32808370 -0.0100672612  0.311631458
# [20,]  0.002120825 -0.10537633 -0.06409014 -0.0349441748 -0.091483399
# [21,] -0.280149377 -0.26408612 -0.16993959  0.0529700283 -0.059870396
# [22,] -0.057798128 -0.06051053 -0.18989548 -0.0519699464  0.270620215
# [23,] -0.133983109 -0.28353064 -0.08907073  0.0399100396 -0.184805218
# [24,]  0.127239586 -0.14620115 -0.08060288 -0.0389676745  0.034782853
# [25,] -0.046152222 -0.17362359  0.16913197  0.1827168119  0.046604092
# [26,]  0.137260149  0.09121796 -0.12809752  0.0045107477 -0.055126611
# [27,] -0.232411554  0.15544798 -0.03649120  0.2303091485 -0.005728744
# [28,]  0.024661472 -0.12310216  0.05022506  0.2122909568 -0.211978427
# [29,]  0.195397993  0.16271411 -0.27890203 -0.0246315515  0.023520471
# [30,]  0.191241482 -0.14595482 -0.20168331  0.2295189432 -0.035366274
# [31,]  0.073611291 -0.03978370  0.23168329 -0.0536905204 -0.037372871
# [32,]  0.209867159  0.23825555 -0.08056727 -0.0687235080 -0.093530618
# [33,] -0.011606650  0.14629597 -0.01366956 -0.0467913070 -0.022681892
# [34,]  0.098599535  0.04065092  0.24180351  0.2247937664  0.118699510
# [35,] -0.011570401  0.21838288 -0.16578060 -0.1999870672 -0.384697844
# [36,]  0.153937887  0.02068612 -0.24701883 -0.1572902824 -0.155070929
# [37,]  0.170385517 -0.12010283  0.02774610 -0.0813152968 -0.049957110
# [38,] -0.182767641 -0.14634955  0.12351262 -0.2473099491 -0.010612227
# [39,] -0.010903643  0.03970494  0.05770148 -0.0369430869 -0.264550784
# [40,] -0.147163035 -0.02549114  0.03688128  0.2820582986 -0.109943400
```

- Visualizing your data is one of the essential elements of data analysis. We are going to primarily use the `ggplot2` package for making data plots. The reason is that it provides elegant graphics in a concise conceptual framework. We will learn more about this later in the semester, but let's get started using the quick plot function `qplot`:

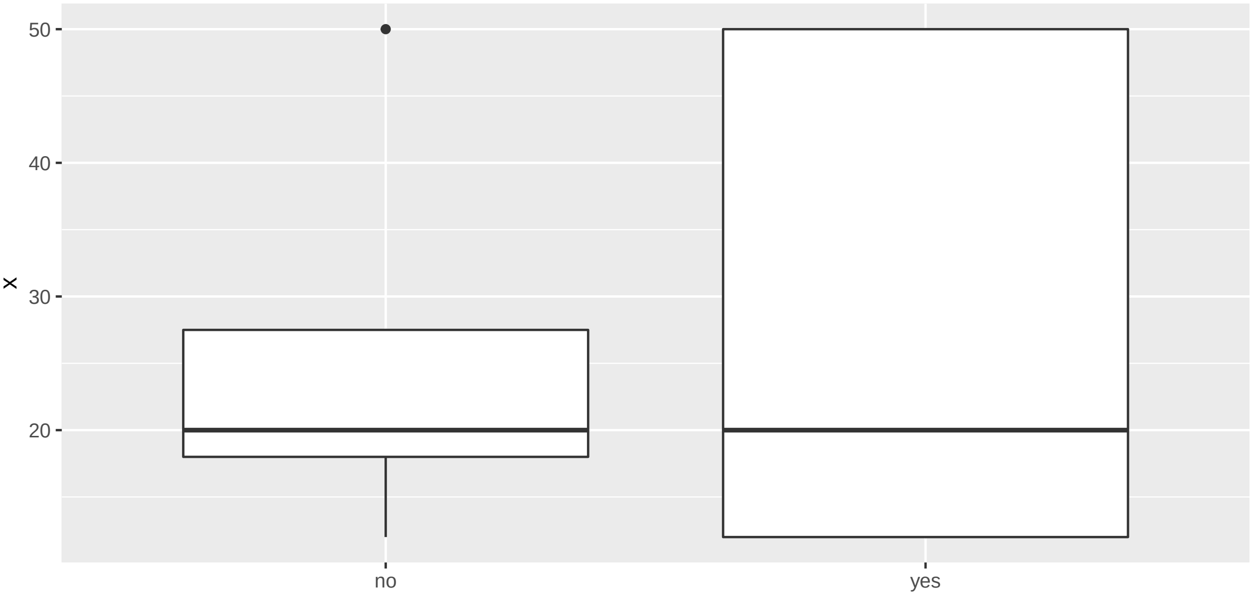
```
library(ggplot2)
df <- data.frame(x=x, y=c(rep("yes", 7), rep("no", 5)))
df
#     x   y
# 1  50 yes
# 2  12 yes
# 3  NA yes
# 4  20 yes
# 5  50 yes
# 6  12 yes
# 7  NA yes
# 8  20  no
# 9  50  no
# 10 12  no
# 11 NA  no
# 12 20  no
qplot(x, data=df)
```



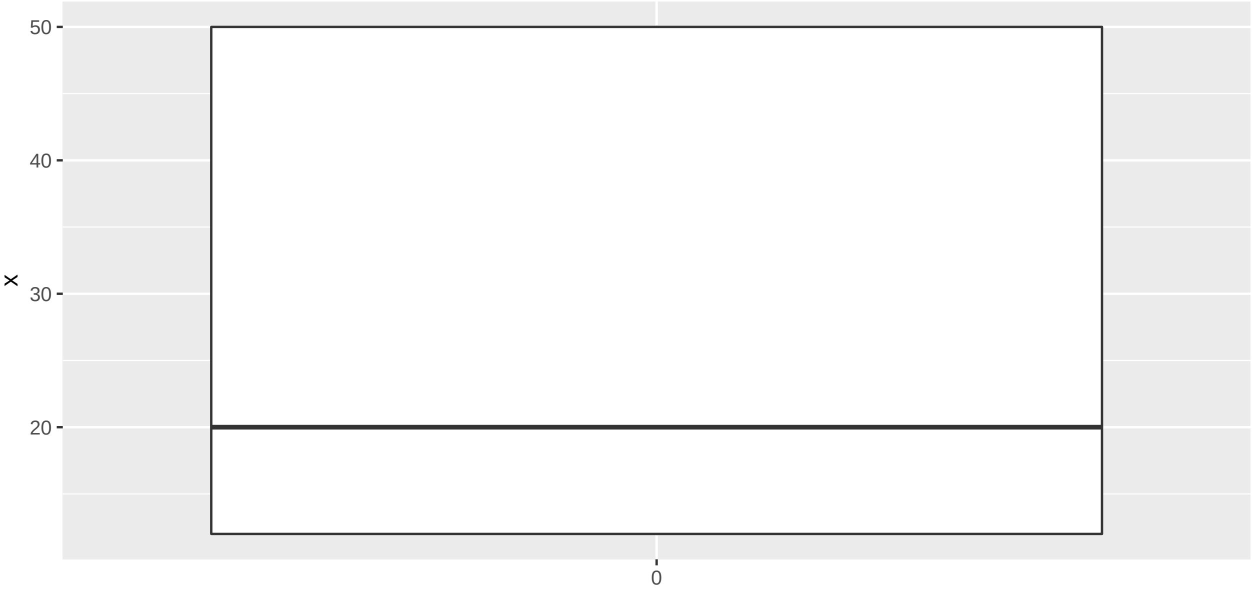
```
qplot(x, data=df, binwidth=5)
```



```
qplot(y, x, data=df,  geom="boxplot", xlab="")
```



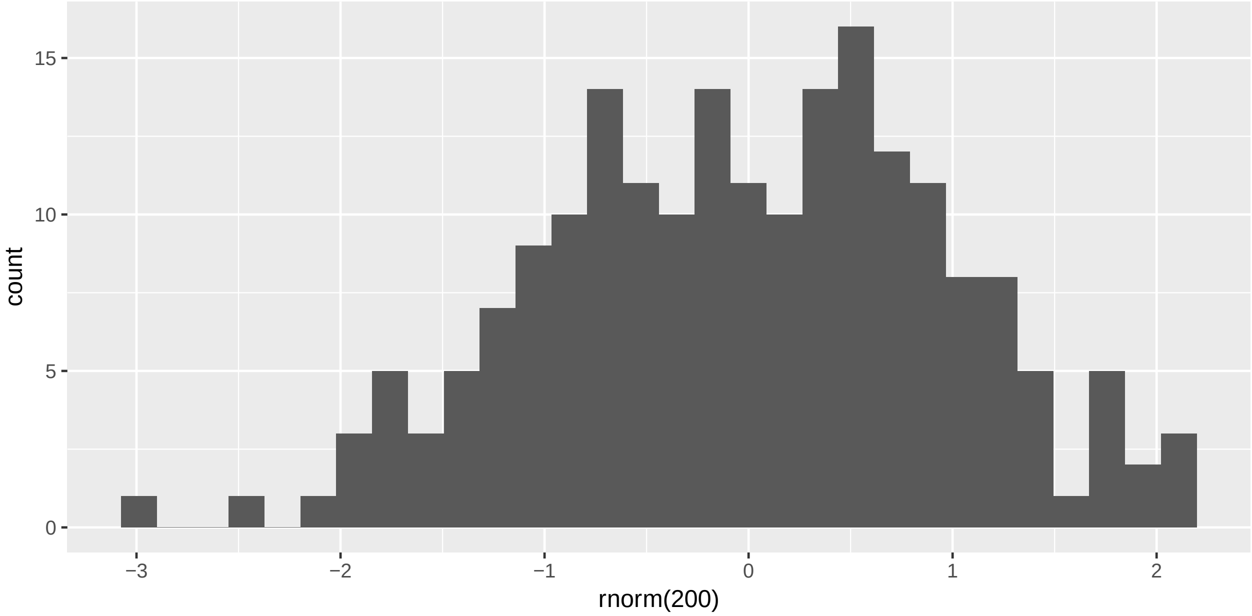
```
qplot(factor(0), x, data=df,  geom="boxplot", xlab="")
```



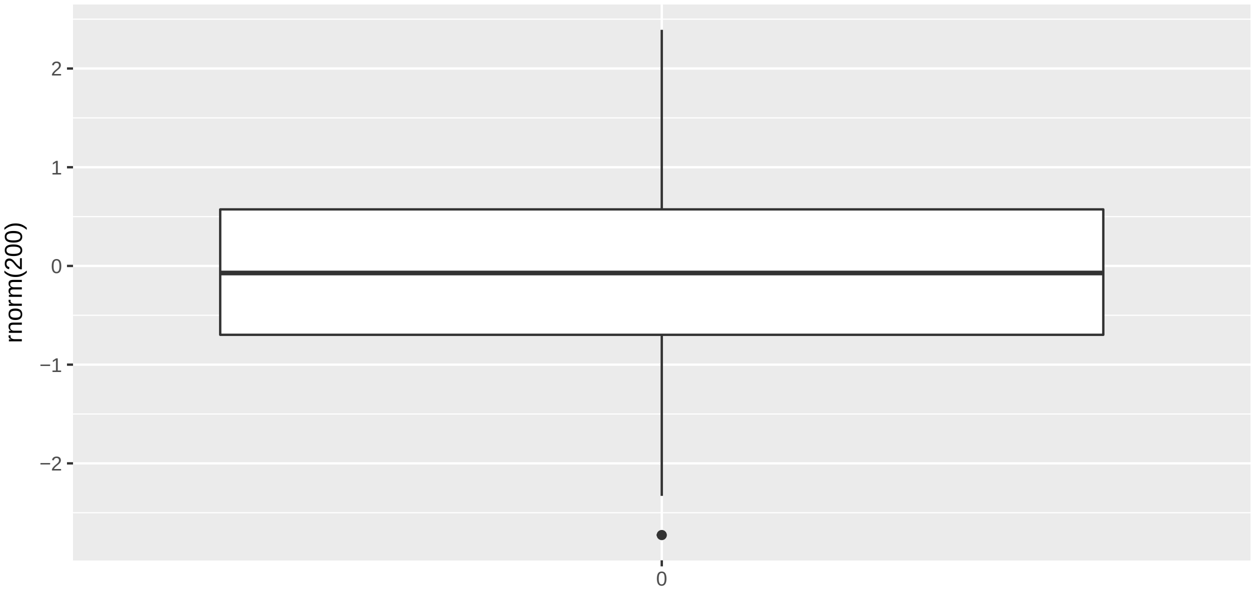
Different R functions can require different data input types. Many of the original functions operate on matrices, but more recently written functions require data frames as input. The package `ggplot2` likes to have data frames.

- The function `rnorm` generates random numbers from a standard normal distribution. Produce a histogram of 200 random numbers from N(0,1)

```
# 200 random numbers from N(0,1)
qplot(rnorm(200))
```



```
qplot(factor(0), rnorm(200), geom="boxplot", xlab="")
```



Modify these commands so that the boxplot uses the same numbers as the histogram. (Hint: save the output of rnorm for re-use.) Notice in these commands that the qplot will also take the output of rnorm directly, which is a numeric vector.

- Let's look at some data sets from the ISLR package. First we need to make sure it is installed.

```
library(ISLR)
```

If that returns an error, you can use `install.packages("ISLR")` at the command line.

A package only needs to be installed once, but you have to load it via the *library* command in each session.

Once the package is installed, try again with

```
library(ISLR)
```

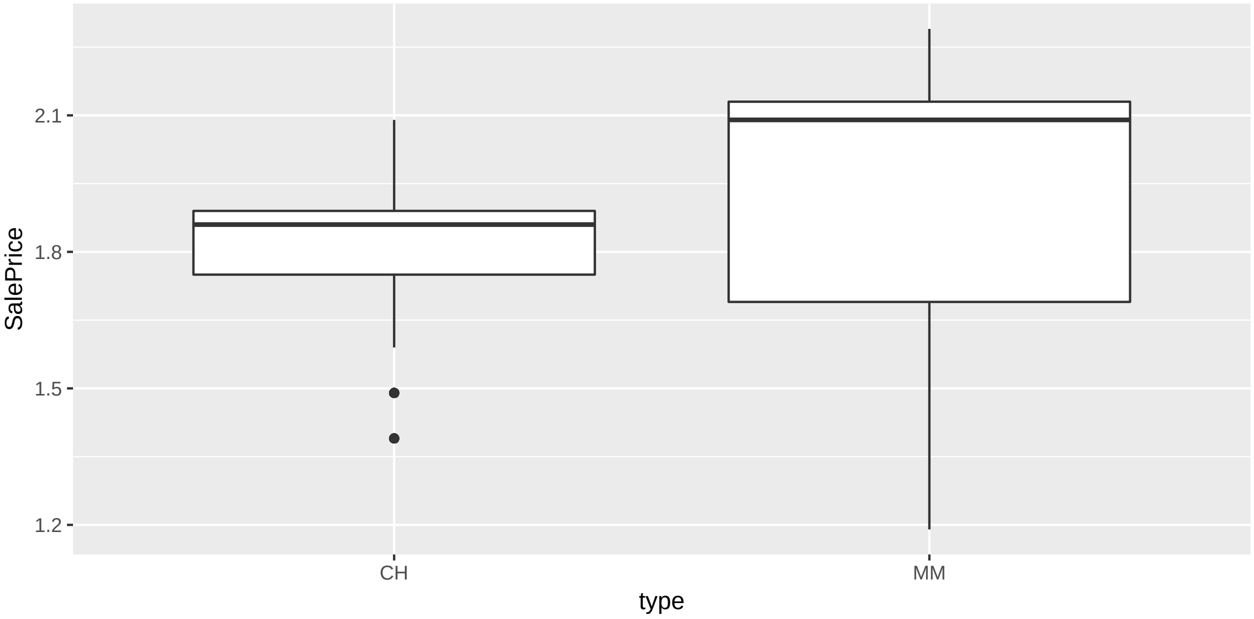
- Then look at the `OJ` data set:

```
help(OJ)
head(OJ)
#   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
# 1       CH            237       1    1.75    1.99   0.00    0.0         0
# 2       CH            239       1    1.75    1.99   0.00    0.3         0
# 3       CH            245       1    1.86    2.09   0.17    0.0         0
# 4       MM            227       1    1.69    1.69   0.00    0.0         0
# 5       CH            228       7    1.69    1.69   0.00    0.0         0
# 6       CH            230       7    1.69    1.99   0.00    0.0         0
#   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
# 1         0 0.500000        1.99        1.75      0.24     No  0.000000
# 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
# 3         0 0.680000        2.09        1.69      0.40     No  0.000000
# 4         0 0.400000        1.69        1.69      0.00     No  0.000000
# 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
# 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
#   PctDiscCH ListPriceDiff STORE
# 1  0.000000          0.24     1
# 2  0.000000          0.24     1
# 3  0.091398          0.23     1
# 4  0.000000          0.00     1
# 5  0.000000          0.00     0
# 6  0.000000          0.30     0
#View(OJ)
summary(OJ)
#  Purchase WeekofPurchase     StoreID        PriceCH         PriceMM     
#  CH:653   Min.   :227.0   Min.   :1.00   Min.   :1.690   Min.   :1.690  
#  MM:417   1st Qu.:240.0   1st Qu.:2.00   1st Qu.:1.790   1st Qu.:1.990  
#           Median :257.0   Median :3.00   Median :1.860   Median :2.090  
#           Mean   :254.4   Mean   :3.96   Mean   :1.867   Mean   :2.085  
#           3rd Qu.:268.0   3rd Qu.:7.00   3rd Qu.:1.990   3rd Qu.:2.180  
#           Max.   :278.0   Max.   :7.00   Max.   :2.090   Max.   :2.290  
#      DiscCH            DiscMM         SpecialCH        SpecialMM     
#  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
#  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
#  Median :0.00000   Median :0.0000   Median :0.0000   Median :0.0000  
#  Mean   :0.05186   Mean   :0.1234   Mean   :0.1477   Mean   :0.1617  
#  3rd Qu.:0.00000   3rd Qu.:0.2300   3rd Qu.:0.0000   3rd Qu.:0.0000  
#  Max.   :0.50000   Max.   :0.8000   Max.   :1.0000   Max.   :1.0000  
#     LoyalCH           SalePriceMM     SalePriceCH      PriceDiff      
#  Min.   :0.000011   Min.   :1.190   Min.   :1.390   Min.   :-0.6700  
#  1st Qu.:0.325257   1st Qu.:1.690   1st Qu.:1.750   1st Qu.: 0.0000  
#  Median :0.600000   Median :2.090   Median :1.860   Median : 0.2300  
#  Mean   :0.565782   Mean   :1.962   Mean   :1.816   Mean   : 0.1465  
#  3rd Qu.:0.850873   3rd Qu.:2.130   3rd Qu.:1.890   3rd Qu.: 0.3200  
#  Max.   :0.999947   Max.   :2.290   Max.   :2.090   Max.   : 0.6400  
#  Store7      PctDiscMM        PctDiscCH       ListPriceDiff  
#  No :714   Min.   :0.0000   Min.   :0.00000   Min.   :0.000  
#  Yes:356   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.140  
#            Median :0.0000   Median :0.00000   Median :0.240  
#            Mean   :0.0593   Mean   :0.02731   Mean   :0.218  
#            3rd Qu.:0.1127   3rd Qu.:0.00000   3rd Qu.:0.300  
#            Max.   :0.4020   Max.   :0.25269   Max.   :0.440  
#      STORE      
#  Min.   :0.000  
#  1st Qu.:0.000  
#  Median :2.000  
#  Mean   :1.631  
#  3rd Qu.:3.000  
#  Max.   :4.000  
OJ[,"PriceCH"]
#    [1] 1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 1.86 1.86 1.99
#   [14] 1.86 2.06 2.06 1.75 1.86 1.86 1.86 1.86 1.76 1.86 1.86 1.86 1.86
#   [27] 1.86 1.86 1.86 2.06 1.89 1.86 1.96 1.69 1.99 1.99 1.76 1.99 1.99
#   [40] 1.79 1.79 1.79 1.79 1.79 1.79 1.79 1.99 1.99 1.99 1.99 1.99 1.99
#   [53] 1.99 1.99 1.99 1.99 1.99 1.99 1.86 1.99 1.99 1.99 1.99 1.99 1.99
#   [66] 1.99 1.99 1.99 1.99 1.99 1.99 2.09 2.09 2.09 1.69 1.79 1.79 1.99
#   [79] 1.89 1.99 1.99 1.99 1.99 1.99 2.09 1.99 2.09 1.99 2.09 2.09 1.79
#   [92] 1.99 1.99 1.86 1.75 1.99 1.99 1.69 1.69 1.69 1.75 1.75 1.75 1.75
#  [105] 1.86 1.89 1.86 1.89 1.89 1.89 1.86 1.99 1.89 1.89 1.89 1.99 1.86
#  [118] 1.86 1.99 1.86 1.86 1.76 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [131] 1.86 1.86 1.86 1.86 1.86 1.96 1.96 1.99 1.99 1.99 1.75 1.99 1.99
#  [144] 1.86 2.09 1.75 1.75 1.86 1.86 1.86 1.86 1.99 1.79 1.79 1.79 1.69
#  [157] 1.99 1.99 1.99 1.99 1.99 1.99 1.86 1.99 1.99 1.99 2.09 2.09 2.09
#  [170] 1.86 1.69 1.69 1.79 1.79 1.79 1.79 1.86 1.99 1.86 1.99 1.99 1.86
#  [183] 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.86 1.86 1.86 1.99
#  [196] 1.99 1.99 1.99 2.09 1.99 2.09 1.69 1.69 1.69 1.75 1.75 1.86 1.86
#  [209] 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.96
#  [222] 1.69 1.69 1.86 1.76 1.76 1.99 1.99 1.99 1.86 1.99 1.69 1.75 1.75
#  [235] 1.75 1.79 1.79 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [248] 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.99 1.69 1.69 1.86 1.99 1.86
#  [261] 1.86 1.86 1.86 1.86 1.99 1.99 1.86 1.79 1.79 1.86 1.86 2.09 2.09
#  [274] 1.79 1.79 1.79 1.79 1.79 1.79 1.99 1.99 1.99 1.99 1.99 1.99 1.99
#  [287] 1.99 1.99 1.99 1.99 1.99 1.99 1.86 1.86 1.96 2.09 1.69 1.69 1.69
#  [300] 1.69 1.79 1.75 1.75 1.89 1.99 1.89 1.86 1.86 1.99 1.86 1.86 1.86
#  [313] 1.96 1.99 1.99 1.99 1.79 1.79 1.99 1.99 1.99 1.99 1.99 1.99 1.69
#  [326] 1.69 1.76 1.86 1.99 1.86 1.76 1.86 1.69 1.75 1.75 1.75 1.86 1.86
#  [339] 1.86 1.75 1.86 1.86 1.76 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.99
#  [352] 2.06 1.79 1.99 1.79 1.79 1.79 1.99 1.99 1.99 1.99 1.86 1.86 1.79
#  [365] 1.86 1.75 1.99 1.86 1.86 1.86 1.86 1.96 1.86 1.69 1.75 1.76 1.69
#  [378] 1.69 1.75 1.75 1.89 1.86 1.86 1.69 1.69 1.69 1.69 1.69 1.69 1.75
#  [391] 1.75 1.75 1.75 1.79 1.86 1.75 1.86 1.86 1.86 1.86 1.99 1.86 1.86
#  [404] 1.96 1.99 1.99 1.69 1.75 1.86 1.86 1.86 1.79 1.75 1.79 1.79 1.79
#  [417] 1.86 1.99 1.86 1.99 1.99 1.99 1.69 1.69 1.99 2.09 1.99 1.86 1.86
#  [430] 1.99 1.99 1.86 1.75 1.86 1.99 1.76 1.76 1.96 1.86 1.76 1.76 1.76
#  [443] 1.76 1.76 1.86 1.96 1.75 1.86 1.99 1.86 1.86 1.86 1.86 1.86 1.99
#  [456] 1.86 1.86 1.99 1.96 2.09 1.89 1.79 1.69 1.86 1.86 1.86 1.86 1.86
#  [469] 1.86 1.86 1.86 1.96 1.99 1.99 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [482] 1.89 1.89 1.86 1.79 1.79 1.75 1.75 1.79 1.75 1.79 1.75 1.99 1.99
#  [495] 1.99 1.69 1.86 1.86 1.96 1.99 1.79 1.79 1.79 1.79 1.99 1.99 1.99
#  [508] 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 2.09 2.09 1.69 1.86
#  [521] 1.86 1.96 1.86 1.99 1.76 1.76 1.99 1.89 1.89 1.75 1.75 1.75 1.79
#  [534] 1.75 1.86 1.86 1.86 1.99 1.86 1.86 1.86 1.86 1.86 1.79 1.69 1.69
#  [547] 1.69 1.69 1.75 1.75 1.75 1.75 1.86 1.89 1.89 1.86 1.96 1.99 1.75
#  [560] 1.86 1.89 2.06 1.79 1.76 1.86 1.79 1.75 1.99 1.86 1.76 1.79 1.79
#  [573] 1.79 1.99 1.99 1.99 1.69 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [586] 1.86 1.86 1.69 1.75 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [599] 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.99 1.86
#  [612] 1.75 1.86 1.86 1.86 1.86 1.86 1.76 1.76 1.86 1.86 1.86 1.86 1.86
#  [625] 1.86 1.86 1.86 1.75 1.89 1.86 1.99 1.86 1.86 1.79 1.79 1.99 1.99
#  [638] 1.79 1.99 1.86 1.79 1.75 1.75 1.75 1.79 1.75 1.79 1.99 1.99 1.89
#  [651] 1.89 1.89 1.86 1.86 1.99 1.86 1.86 1.86 1.86 1.86 1.76 1.86 1.86
#  [664] 1.86 1.69 1.86 1.89 1.86 1.86 1.99 1.76 1.86 1.86 1.86 1.86 1.86
#  [677] 1.96 1.96 1.99 1.75 1.86 1.99 1.86 1.86 1.86 1.86 1.86 2.06 1.79
#  [690] 1.79 1.79 1.79 1.79 1.79 1.79 1.79 1.79 1.79 1.79 1.99 1.99 1.99
#  [703] 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99
#  [716] 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 2.09 2.09 1.86
#  [729] 1.86 1.96 1.99 2.09 2.09 1.69 1.86 1.86 1.79 1.99 1.99 1.99 1.99
#  [742] 1.99 1.99 1.99 2.09 1.75 1.76 1.79 1.75 1.69 1.69 1.79 1.69 1.86
#  [755] 1.86 1.89 1.86 1.89 1.86 1.75 1.99 2.09 1.89 1.89 1.89 1.86 1.86
#  [768] 1.86 2.06 1.69 1.69 1.69 1.86 1.89 1.89 1.69 1.86 1.76 1.99 1.99
#  [781] 1.69 1.86 1.86 1.79 1.79 1.79 1.79 1.99 1.86 1.96 1.79 1.86 1.75
#  [794] 1.76 1.76 1.76 1.76 1.96 1.86 1.99 1.99 1.79 1.79 1.86 1.76 1.86
#  [807] 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [820] 1.86 1.99 1.79 1.79 1.79 1.79 1.79 1.99 1.86 1.99 1.99 1.99 1.69
#  [833] 1.79 1.79 1.99 1.79 1.99 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.96
#  [846] 1.96 1.96 1.96 1.75 1.86 1.69 1.69 1.69 1.75 1.86 1.86 1.86 1.86
#  [859] 1.86 1.76 1.86 1.75 1.86 1.69 1.69 1.75 1.75 1.99 1.86 1.86 1.86
#  [872] 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 1.99 2.09 1.75
#  [885] 1.75 1.76 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86 1.86
#  [898] 1.86 1.86 1.86 1.86 1.86 1.86 1.99 1.99 1.69 1.75 1.75 1.86 1.86
#  [911] 1.96 1.99 1.75 1.86 1.89 1.89 1.89 1.99 1.99 1.86 1.86 1.86 1.86
#  [924] 1.99 1.99 1.79 1.79 1.79 1.79 1.79 1.99 1.79 1.99 1.99 1.69 1.69
#  [937] 1.69 1.69 1.75 1.89 1.89 1.89 1.89 1.99 1.89 1.86 1.86 1.99 1.86
#  [950] 1.86 1.86 1.86 1.86 1.99 1.86 1.86 1.99 1.99 1.96 1.99 1.99 1.69
#  [963] 1.69 1.69 1.69 1.69 1.75 1.75 1.75 1.75 1.75 1.86 1.86 1.86 1.86
#  [976] 1.86 1.96 1.99 1.99 1.99 1.75 1.86 1.76 1.86 1.86 1.86 1.96 1.86
#  [989] 1.99 1.99 1.86 1.96 1.69 1.79 1.99 1.86 1.69 1.69 1.86 1.99 1.69
# [1002] 1.86 1.79 1.79 1.79 1.99 1.99 1.99 1.69 1.69 1.69 1.69 1.75 1.75
# [1015] 1.69 1.75 1.75 1.75 1.86 1.86 1.86 1.89 1.99 1.89 1.99 1.86 1.86
# [1028] 1.76 1.86 1.86 1.86 1.99 1.86 1.99 1.99 1.69 1.69 1.69 1.69 1.75
# [1041] 1.75 1.75 1.75 1.75 1.86 1.76 1.76 1.76 1.76 1.69 1.69 1.69 1.75
# [1054] 1.75 1.86 1.69 1.69 1.69 1.75 1.69 1.75 1.86 1.86 1.76 1.86 1.86
# [1067] 1.86 1.86 1.86 1.86
```

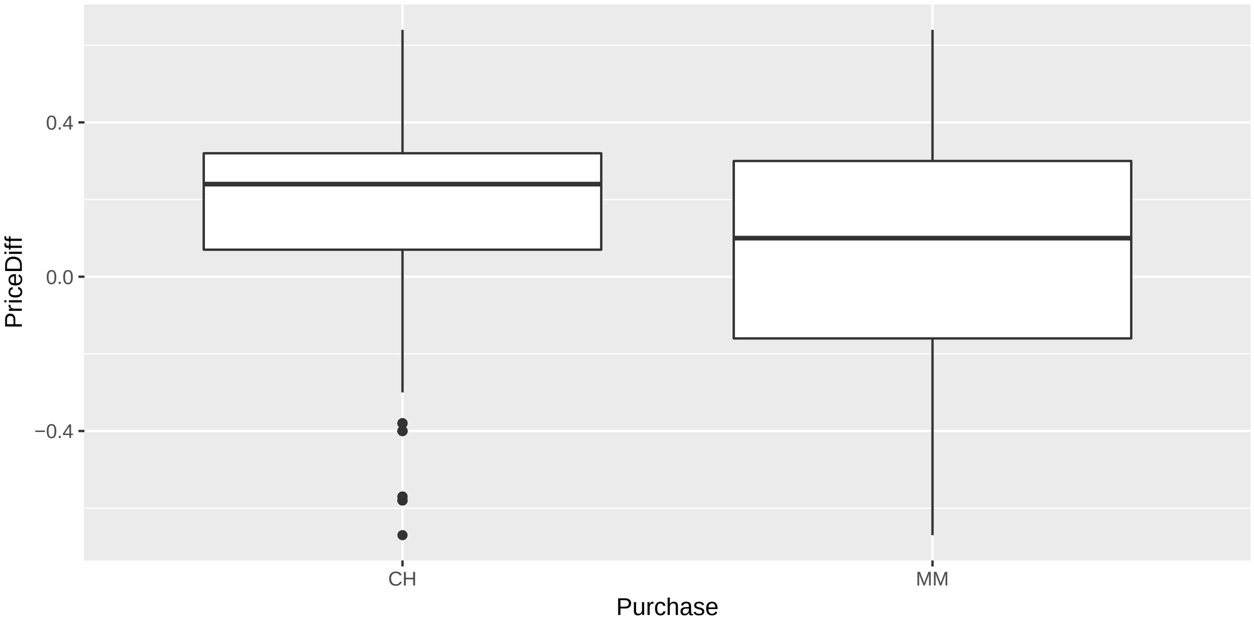
Can you figure out what the square brackets mean in the output from the last command?

Now lets make some plots of the data

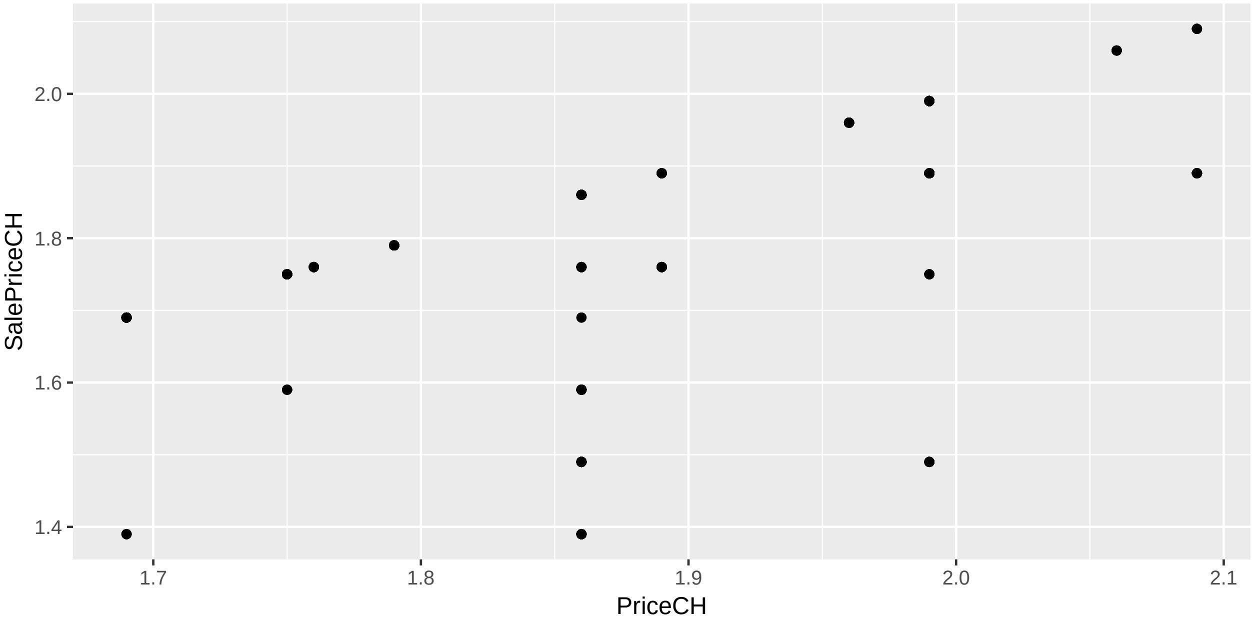
```
saleprice <- data.frame(SalePrice=c(OJ$SalePriceMM,OJ$SalePriceCH),
                type=c(rep("MM",nrow(OJ)),rep("CH",nrow(OJ))))
qplot(type, SalePrice, data=saleprice, geom="boxplot")
```



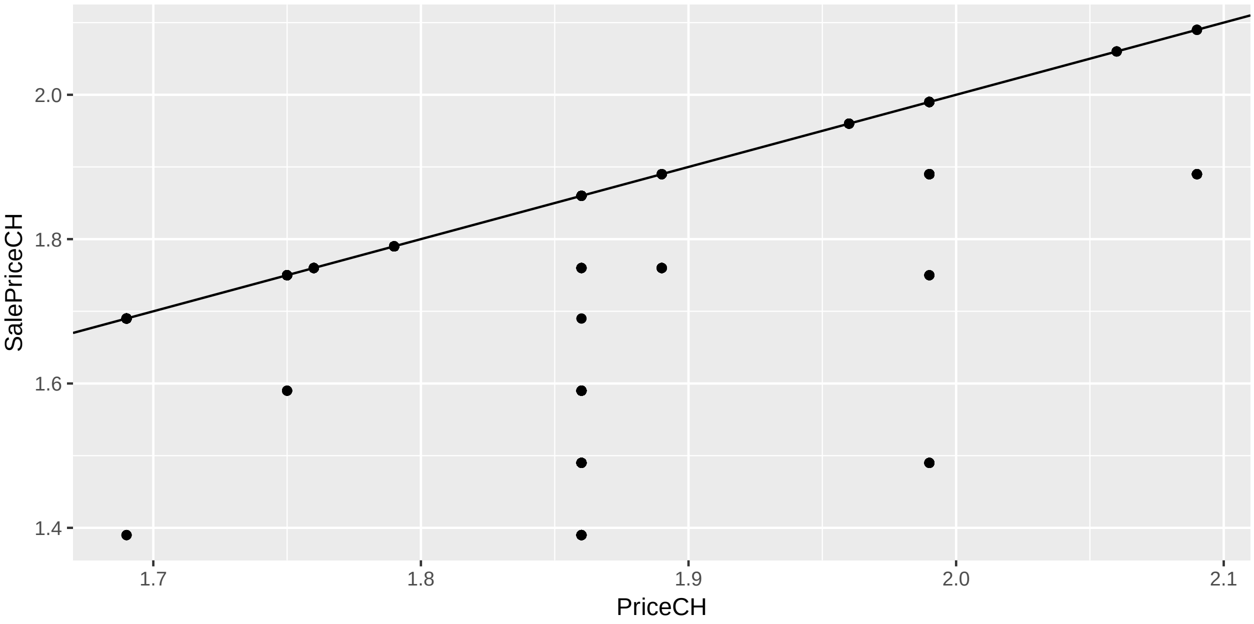
```
qplot(Purchase, PriceDiff, data=OJ, geom="boxplot")
```



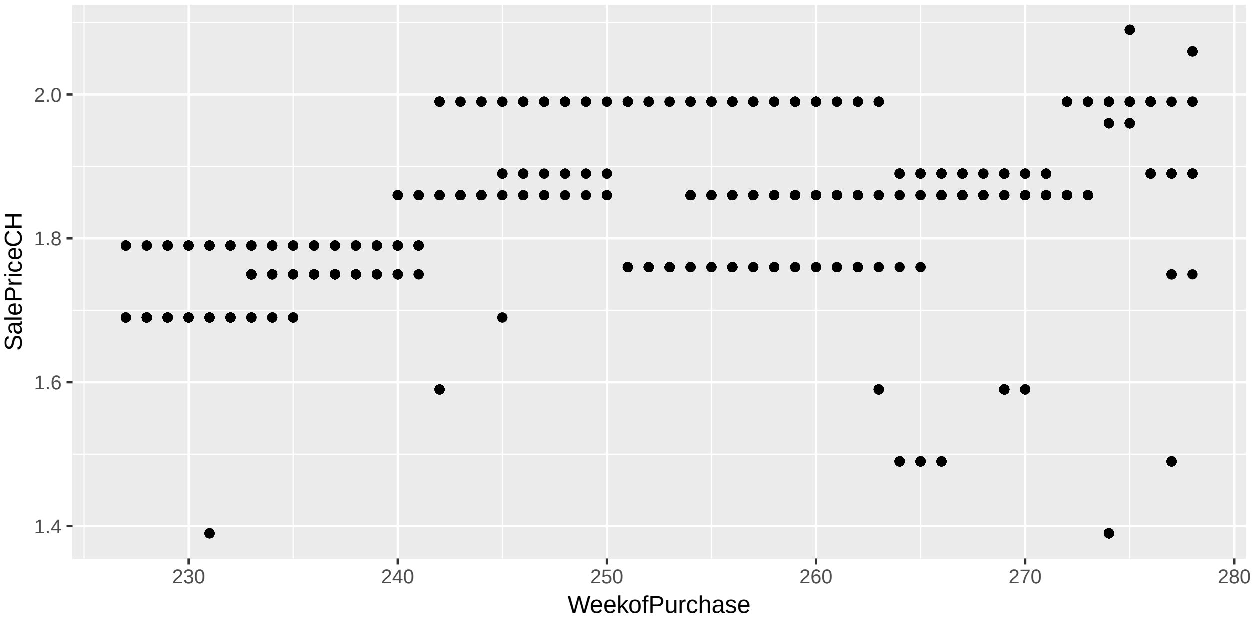
```
qplot(PriceCH, SalePriceCH, data=OJ)
```



```
qplot(PriceCH, SalePriceCH, data=OJ, position="jitter") +
  geom_abline(intercept=0,slope=1)
```



```
qplot(WeekofPurchase, SalePriceCH, data=OJ, position="jitter")
```



Make sure you understand what is being plotted in each case, and what the graphs are telling you about the data.

- Tabulating variables:

```
table(OJ$StoreID)
#
#   1   2   3   4   7
# 157 222 196 139 356
table(OJ$Purchase, OJ$SpecialCH)
#
#        0   1
#   CH 532 121
#   MM 380  37
```

What do these tables tell you?

- The *summary* command can be applied to almost anything to get a summary of the object. Try it on some other data sets in the *ISLR* package. Note that the *summary* is just what R thinks should be the summary, and it may not always be the best summary. If someone asks you to “summarise” this data set, you may need to think about what is important and use different functions that are appropriate for the situation.

- Check the PISA data:

```
library(readr)
pisa <- read_csv("https://github.com/bsouhaib/BA2018/raw/master/data/PISA-oz.csv")
dim(pisa)
# [1] 14481    80
colnames(pisa)
#  [1] "name"     "SCHOOLID" "ST03Q01"  "ST03Q02"  "ST04Q01"  "ST08Q01" 
#  [7] "ST09Q01"  "ST115Q01" "ST11Q01"  "ST11Q02"  "ST11Q03"  "ST11Q04" 
# [13] "ST11Q05"  "ST11Q06"  "ST13Q01"  "ST17Q01"  "ST27Q01"  "ST27Q02" 
# [19] "ST27Q03"  "ST27Q04"  "ST27Q05"  "ST28Q01"  "ST85Q01"  "ST85Q02" 
# [25] "IC05Q01"  "IC06Q01"  "IC07Q01"  "IC08Q03"  "IC08Q04"  "IC08Q05" 
# [31] "PV1MATH"  "PV2MATH"  "PV3MATH"  "PV4MATH"  "PV5MATH"  "PV1MACC" 
# [37] "PV2MACC"  "PV3MACC"  "PV4MACC"  "PV5MACC"  "PV1MACQ"  "PV2MACQ" 
# [43] "PV3MACQ"  "PV4MACQ"  "PV5MACQ"  "PV1MACS"  "PV2MACS"  "PV3MACS" 
# [49] "PV4MACS"  "PV5MACS"  "PV1MACU"  "PV2MACU"  "PV3MACU"  "PV4MACU" 
# [55] "PV5MACU"  "PV1MAPE"  "PV2MAPE"  "PV3MAPE"  "PV4MAPE"  "PV5MAPE" 
# [61] "PV1MAPF"  "PV2MAPF"  "PV3MAPF"  "PV4MAPF"  "PV5MAPF"  "PV1MAPI" 
# [67] "PV2MAPI"  "PV3MAPI"  "PV4MAPI"  "PV5MAPI"  "PV1READ"  "PV2READ" 
# [73] "PV3READ"  "PV4READ"  "PV5READ"  "PV1SCIE"  "PV2SCIE"  "PV3SCIE" 
# [79] "PV4SCIE"  "PV5SCIE"
#View(pisa)
str(pisa)
# Classes 'tbl_df', 'tbl' and 'data.frame':  14481 obs. of  80 variables:
#  $ name    : chr  "Australia" "Australia" "Australia" "Australia" ...
#  $ SCHOOLID: chr  "0000001" "0000001" "0000001" "0000001" ...
#  $ ST03Q01 : chr  "11" "12" "07" "09" ...
#  $ ST03Q02 : int  1996 1996 1996 1996 1996 1996 1996 1996 1996 1996 ...
#  $ ST04Q01 : chr  "Female" "Female" "Female" "Male" ...
#  $ ST08Q01 : chr  "One or two times" "None" "One or two times" "One or two times" ...
#  $ ST09Q01 : chr  "One or two times" "None" "None" "None" ...
#  $ ST115Q01: int  2 1 1 1 1 1 1 1 1 1 ...
#  $ ST11Q01 : chr  "Yes" "Yes" "Yes" "Yes" ...
#  $ ST11Q02 : chr  "Yes" "Yes" "Yes" "Yes" ...
#  $ ST11Q03 : chr  "Yes" "Yes" "No" "No" ...
#  $ ST11Q04 : chr  "Yes" NA "No" "Yes" ...
#  $ ST11Q05 : chr  "No" NA "No" "No" ...
#  $ ST11Q06 : chr  "No" NA "No" "No" ...
#  $ ST13Q01 : chr  "<ISCED level 3A>" "<ISCED level 2>" NA "<ISCED level 3A>" ...
#  $ ST17Q01 : chr  "<ISCED level 2>" "<ISCED level 2>" "<ISCED level 3B, 3C>" "<ISCED level 3A>" ...
#  $ ST27Q01 : chr  "Three or more" "Three or more" "Three or more" "Three or more" ...
#  $ ST27Q02 : chr  "Three or more" "Three or more" "Two" "Two" ...
#  $ ST27Q03 : chr  "Three or more" "Three or more" "Three or more" "Three or more" ...
#  $ ST27Q04 : chr  "Three or more" "Two" "Three or more" "Two" ...
#  $ ST27Q05 : chr  "Three or more" "Two" "One" "Two" ...
#  $ ST28Q01 : chr  "201-500 books" "26-100 books" "201-500 books" "201-500 books" ...
#  $ ST85Q01 : chr  "Agree" "Agree" NA NA ...
#  $ ST85Q02 : chr  "Agree" "Agree" NA NA ...
#  $ IC05Q01 : chr  "03" "03" "03" "05" ...
#  $ IC06Q01 : chr  "04" "04" "06" "07" ...
#  $ IC07Q01 : chr  "04" "04" "06" "07" ...
#  $ IC08Q03 : chr  "Once or twice a week" "Once or twice a month" "Almost every day" "Once or twice a 
#  $ IC08Q04 : chr  "Once or twice a month" "Once or twice a week" "Almost every day" "Every day" ...
#  $ IC08Q05 : chr  "Almost every day" "Every day" "Almost every day" "Every day" ...
#  $ PV1MATH : num  562 565 507 602 520 ...
#  $ PV2MATH : num  569 557 547 594 507 ...
#  $ PV3MATH : num  555 553 511 552 501 ...
#  $ PV4MATH : num  579 538 454 526 521 ...
#  $ PV5MATH : num  548 573 546 619 547 ...
#  $ PV1MACC : num  549 588 509 570 485 ...
#  $ PV2MACC : num  534 538 501 578 518 ...
#  $ PV3MACC : num  502 524 502 564 537 ...
#  $ PV4MACC : num  509 589 494 539 533 ...
#  $ PV5MACC : num  583 559 453 575 486 ...
#  $ PV1MACQ : num  583 595 501 584 530 ...
#  $ PV2MACQ : num  593 559 494 581 585 ...
#  $ PV3MACQ : num  543 560 496 580 582 ...
#  $ PV4MACQ : num  537 594 482 540 571 ...
#  $ PV5MACQ : num  622 590 443 562 549 ...
#  $ PV1MACS : num  572 556 516 588 469 ...
#  $ PV2MACS : num  578 529 493 617 536 ...
#  $ PV3MACS : num  537 506 508 602 514 ...
#  $ PV4MACS : num  523 555 477 588 518 ...
#  $ PV5MACS : num  621 528 444 603 469 ...
#  $ PV1MACU : num  598 573 471 601 522 ...
#  $ PV2MACU : num  601 545 500 573 550 ...
#  $ PV3MACU : num  577 545 471 617 555 ...
#  $ PV4MACU : num  558 561 440 560 550 ...
#  $ PV5MACU : num  612 573 437 564 554 ...
#  $ PV1MAPE : num  566 559 496 578 526 ...
#  $ PV2MAPE : num  588 548 503 567 531 ...
#  $ PV3MAPE : num  571 573 501 537 522 ...
#  $ PV4MAPE : num  553 584 506 631 459 ...
#  $ PV5MAPE : num  587 567 484 593 493 ...
#  $ PV1MAPF : num  569 585 493 586 487 ...
#  $ PV2MAPF : num  595 520 497 589 523 ...
#  $ PV3MAPF : num  577 546 536 530 498 ...
#  $ PV4MAPF : num  562 569 516 641 455 ...
#  $ PV5MAPF : num  595 609 464 601 461 ...
#  $ PV1MAPI : num  555 617 471 576 545 ...
#  $ PV2MAPI : num  590 619 497 587 571 ...
#  $ PV3MAPI : num  589 612 501 574 524 ...
#  $ PV4MAPI : num  530 617 502 661 495 ...
#  $ PV5MAPI : num  567 603 459 616 488 ...
#  $ PV1READ : num  582 617 584 650 554 ...
#  $ PV2READ : num  571 572 551 608 560 ...
#  $ PV3READ : num  602 560 588 594 517 ...
#  $ PV4READ : num  572 564 575 575 564 ...
#  $ PV5READ : num  585 565 624 620 572 ...
#  $ PV1SCIE : num  583 627 556 668 574 ...
#  $ PV2SCIE : num  579 600 573 665 612 ...
#  $ PV3SCIE : num  593 574 584 620 571 ...
#  $ PV4SCIE : num  567 582 532 592 598 ...
#  $ PV5SCIE : num  587 625 589 656 662 ...
#  - attr(*, "spec")=List of 2
#   ..$ cols   :List of 80
#   .. ..$ name    : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ SCHOOLID: list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST03Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST03Q02 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
#   .. ..$ ST04Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST08Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST09Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST115Q01: list()
#   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
#   .. ..$ ST11Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST11Q02 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST11Q03 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST11Q04 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST11Q05 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST11Q06 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST13Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST17Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST27Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST27Q02 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST27Q03 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST27Q04 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST27Q05 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST28Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST85Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ ST85Q02 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC05Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC06Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC07Q01 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC08Q03 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC08Q04 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ IC08Q05 : list()
#   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
#   .. ..$ PV1MATH : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MATH : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MATH : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MATH : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MATH : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MACC : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MACC : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MACC : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MACC : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MACC : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MACQ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MACQ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MACQ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MACQ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MACQ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MACS : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MACS : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MACS : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MACS : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MACS : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MACU : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MACU : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MACU : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MACU : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MACU : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MAPE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MAPE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MAPE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MAPE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MAPE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MAPF : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MAPF : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MAPF : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MAPF : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MAPF : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1MAPI : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2MAPI : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3MAPI : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4MAPI : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5MAPI : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1READ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2READ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3READ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4READ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5READ : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV1SCIE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV2SCIE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV3SCIE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV4SCIE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   .. ..$ PV5SCIE : list()
#   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
#   ..$ default: list()
#   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
#   ..- attr(*, "class")= chr "col_spec"
head(pisa)
# # A tibble: 6 x 80
#        name SCHOOLID ST03Q01 ST03Q02 ST04Q01           ST08Q01
#       <chr>    <chr>   <chr>   <int>   <chr>             <chr>
# 1 Australia  0000001      11    1996  Female One or two times
# 2 Australia  0000001      12    1996  Female              None
# 3 Australia  0000001      07    1996  Female One or two times
# 4 Australia  0000001      09    1996    Male One or two times
# 5 Australia  0000001      06    1996    Male One or two times
# 6 Australia  0000001      07    1996  Female              None
# # ... with 74 more variables: ST09Q01 <chr>, ST115Q01 <int>,
# #   ST11Q01 <chr>, ST11Q02 <chr>, ST11Q03 <chr>, ST11Q04 <chr>,
# #   ST11Q05 <chr>, ST11Q06 <chr>, ST13Q01 <chr>, ST17Q01 <chr>,
# #   ST27Q01 <chr>, ST27Q02 <chr>, ST27Q03 <chr>, ST27Q04 <chr>,
# #   ST27Q05 <chr>, ST28Q01 <chr>, ST85Q01 <chr>, ST85Q02 <chr>,
# #   IC05Q01 <chr>, IC06Q01 <chr>, IC07Q01 <chr>, IC08Q03 <chr>,
# #   IC08Q04 <chr>, IC08Q05 <chr>, PV1MATH <dbl>, PV2MATH <dbl>,
# #   PV3MATH <dbl>, PV4MATH <dbl>, PV5MATH <dbl>, PV1MACC <dbl>,
# #   PV2MACC <dbl>, PV3MACC <dbl>, PV4MACC <dbl>, PV5MACC <dbl>,
# #   PV1MACQ <dbl>, PV2MACQ <dbl>, PV3MACQ <dbl>, PV4MACQ <dbl>,
# #   PV5MACQ <dbl>, PV1MACS <dbl>, PV2MACS <dbl>, PV3MACS <dbl>,
# #   PV4MACS <dbl>, PV5MACS <dbl>, PV1MACU <dbl>, PV2MACU <dbl>,
# #   PV3MACU <dbl>, PV4MACU <dbl>, PV5MACU <dbl>, PV1MAPE <dbl>,
# #   PV2MAPE <dbl>, PV3MAPE <dbl>, PV4MAPE <dbl>, PV5MAPE <dbl>,
# #   PV1MAPF <dbl>, PV2MAPF <dbl>, PV3MAPF <dbl>, PV4MAPF <dbl>,
# #   PV5MAPF <dbl>, PV1MAPI <dbl>, PV2MAPI <dbl>, PV3MAPI <dbl>,
# #   PV4MAPI <dbl>, PV5MAPI <dbl>, PV1READ <dbl>, PV2READ <dbl>,
# #   PV3READ <dbl>, PV4READ <dbl>, PV5READ <dbl>, PV1SCIE <dbl>,
# #   PV2SCIE <dbl>, PV3SCIE <dbl>, PV4SCIE <dbl>, PV5SCIE <dbl>
```

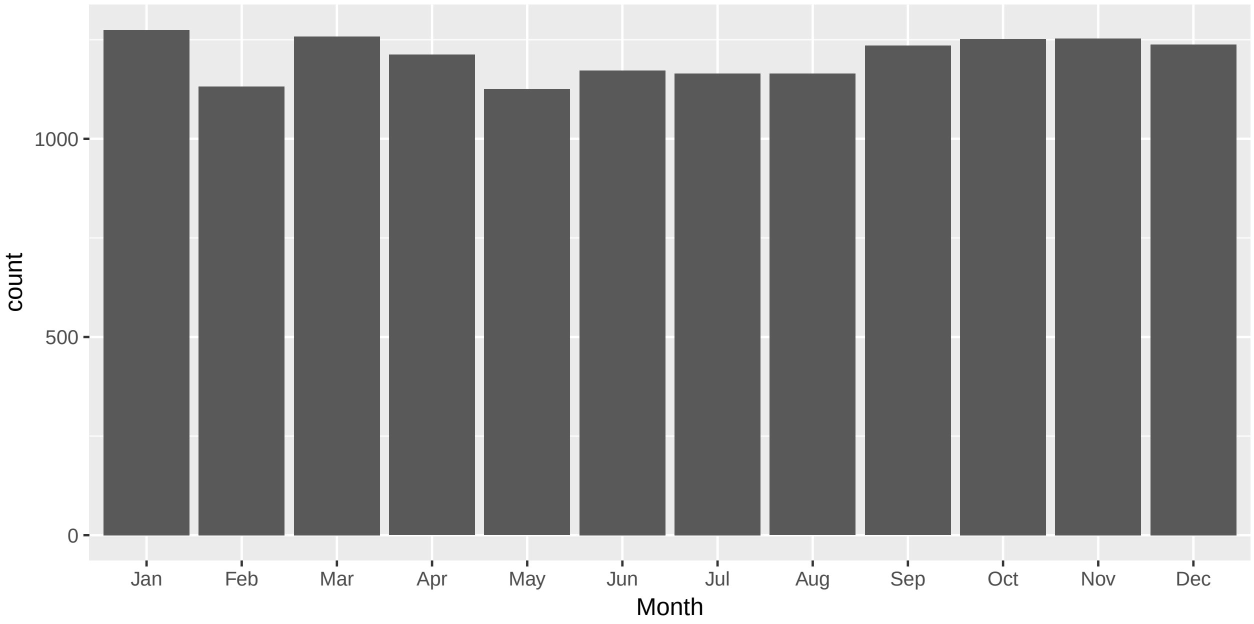
Which columns of pisa are numeric? Which columns are character?

- How many different schools were sampled (according to the variable *SCHOOLID*)?

There are several ways of answering this question. First use the *table* command. Then try using a combination of *length* and *unique*.

- Look at the distribution of birth months amongst the Australian students:

```
qplot(factor(ST03Q01, labels=month.abb), data=pisa, xlab="Month")
```

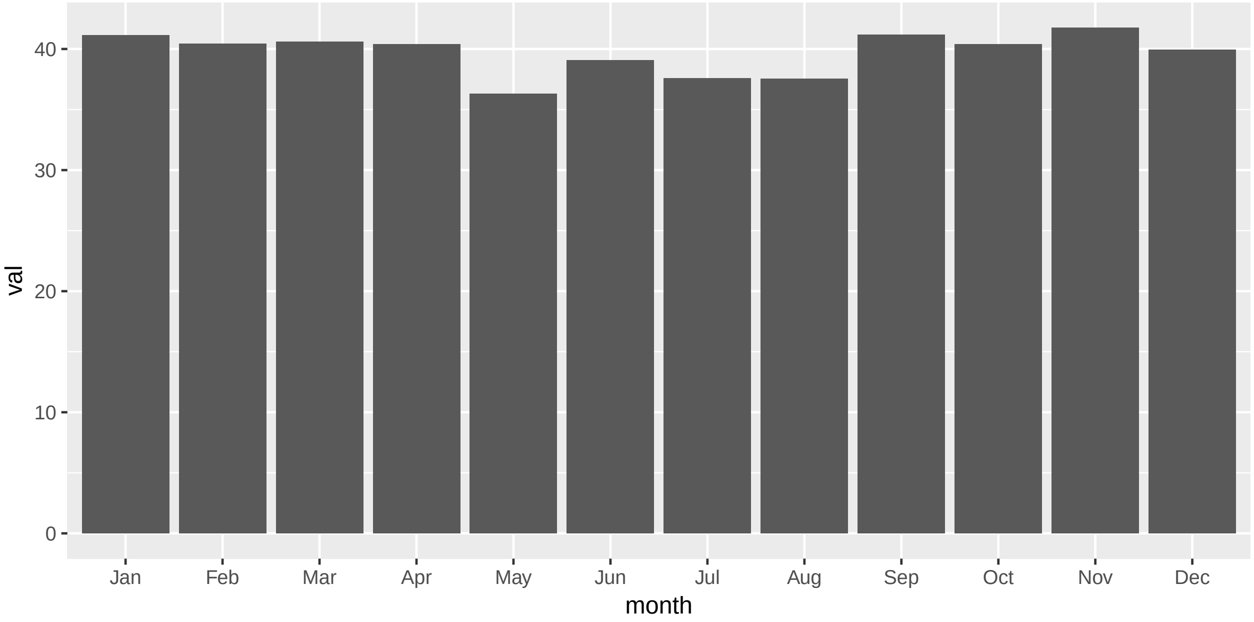


Can you explain the variation? Why are February and May the smallest?

- Perhaps we should adjust for month length:

```
monthdays <- c(31,28,31,30,31,30,31,31,30,31,30,31)
monthtot <- table(pisa$ST03Q01)

y <- data.frame(month=factor(1:12, ordered=TRUE, labels=month.abb),
                val=c(monthtot)/monthdays)
ggplot(y, aes(x=month,y=val)) + geom_bar(stat="identity")
```



- Check if the differences are statistically significant after adjusting for month length:

```
chisq.test(monthtot, p=monthdays/365)
#
```

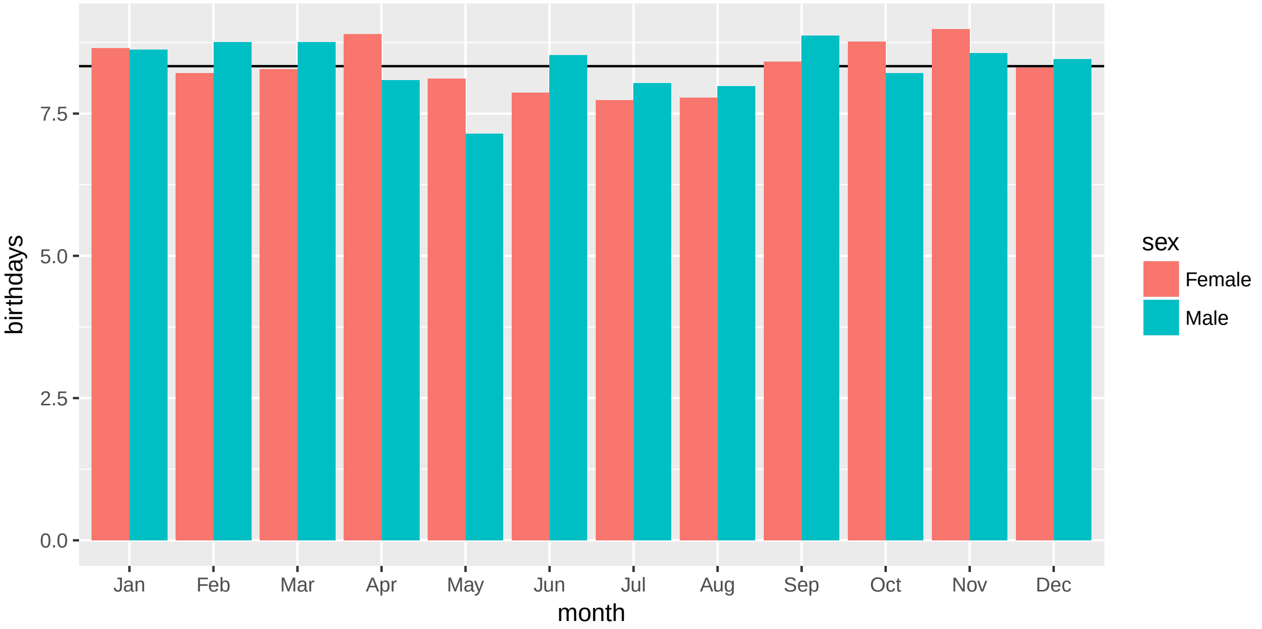
```
#   Chi-squared test for given probabilities
#
# data:  monthtot
# X-squared = 24.723, df = 11, p-value = 0.01001
```

Why is it so?

- What if we split by sex and turn into percentages:

```
male <- table(subset(pisa, ST04Q01=="Male")$ST03Q01)/monthdays
male <- male / sum(male) * 100
female <- table(subset(pisa, ST04Q01!="Male")$ST03Q01)/monthdays
female <- female / sum(female) * 100
sextot <- data.frame(birthdays=c(male,female),
            sex=c(rep("Male",12),rep("Female",12)),
            month=factor(rep(month.abb,2),levels=month.abb,ordered=TRUE))

ggplot(sextot, aes(x=month, y=birthdays, fill=sex)) + geom_hline(aes(yintercept=100/12)) + geom_bar(sta
```



The largest deviation from what you would expect is for males born in May. Why?

- It is easy to create your own data:

```
mynumbers <- 5:12
```

Names can be almost anything, except for special characters `^`, `!`, `$`, `@`, `+`, `-`, `/`, `*`. It is good practice to name your objects with some meaning for what they contain, be reasonably short (less typing). They should not be the same as common R functions; for example, don't use `data` because it is also used to load stored data from packages, or `c` because this is an R function that allows you to collect a bunch of objects together. You won't get errors by using these names but you may get confused when you come back and look at your code later.

- Objects can be of different types. The object `mynumbers` is a vector of numbers. Numbers can be various types also: integer or double.

```
typeof(mynumbers)
# [1] "integer"
```

```
is.numeric(mynumbers)
# [1] TRUE
is.vector(mynumbers)
# [1] TRUE
length(mynumbers)
# [1] 8
```

- Other common types of objects for data analysis are characters, logicals, factors, dates. Factors store categorical data. Dates have a special format that enables it to be treated similarly to how we use dates in real life.

```
mytext <- c("hello", "class")
length(mytext)
# [1] 2
mylogic <- c(TRUE, FALSE, TRUE, TRUE)
gender <- factor(c("male", "female", "female", "female", "male"))
levels(gender)
# [1] "female" "male"
summary(gender)
# female   male
#      3      2
```

- One of the powerful aspects of R is to build on the reproducibility. If you are going to do the same analysis over and over again, compile these operations into a function that you can then apply to different data sets.

```
y <- c(1,2,3,4,5,6)

average = function(x)
{
  return(sum(x)/length(x))
}

ybar <- average(y)
```

Try your function on other data.

- Now write a function to compute the mode of some data:

```
mymode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
mymode(c(rep("A", 5), rep("B", 3)))
# [1] "A"
```

- Write an R function `stats` to compute the mean, min, max, and the deciles, from a vector of data. You will need to search the R help facilities to find functions to compute each of the statistics. The function should return the statistics as a single vector in numerical order with appropriate names for the elements. Your function should be robust to missing values (i.e., the statistics should be computed on the non-missing values).