The video time for this lesson is 58 minutes.

The recommended reading for this lesson is:
Data Processing
https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cse6242/recommended+reading/processing.pdf

In this lesson, the following datasets are referenced or used.

To install the movies dataset:
install.packages(“plyr”) //注意若將此句 copy 到 terminal, 則引號的格式會變, 要重新改引號.

To install player dataset:
install.packages(“plyr”)

To use the diamonds dataset:
install.packages(“ggplot2) #do this if you have not done so already
library(“ggplot2”)
diamonds

To use the MASS library:
library(MASS)

To use the robustHD library:
# first: install dependent packages
install.packages(c("MASS", "akima", "robustbase"))

# second: install suggested packages
install.packages(c("cobs", "robust", "mgcv", "scatterplot3d", "quantreg", "rrcov", "lars", "pwr", "trimcluster", "parallel", "mc2d", "psych", "Rfit"))

# third: install an additional package which provides some C functions
install.packages("devtools")
library("devtools")
install_github("mrxiaohe/WRScpp")

# fourth: install WRS
install_github("nicebread/WRS", subdir="pkg")

install.packages(“robustHD”)
library(“robustHD”)

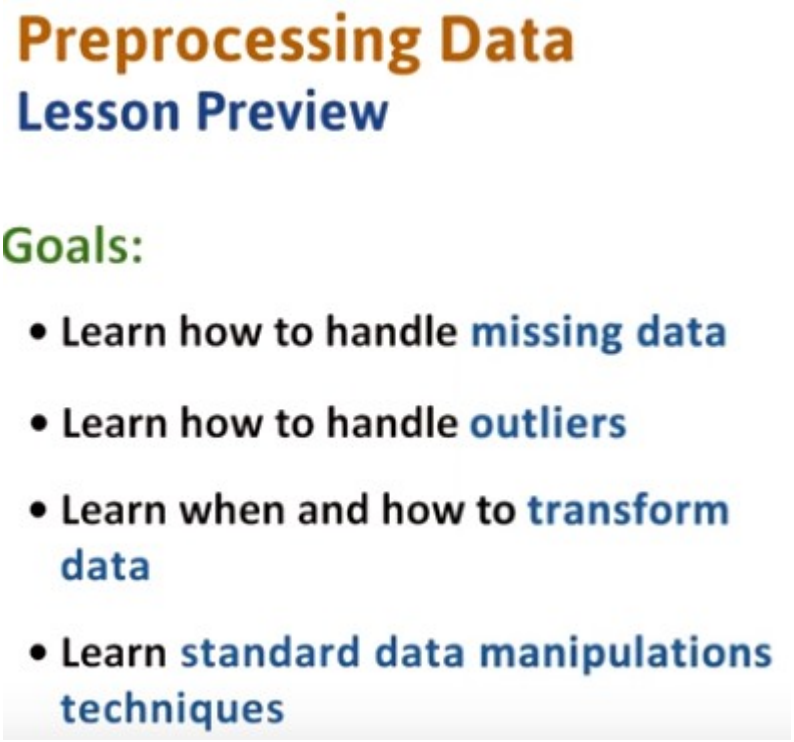
The RobustHD instructions can also be found at: RobustHD install instructions

To use the melt command:

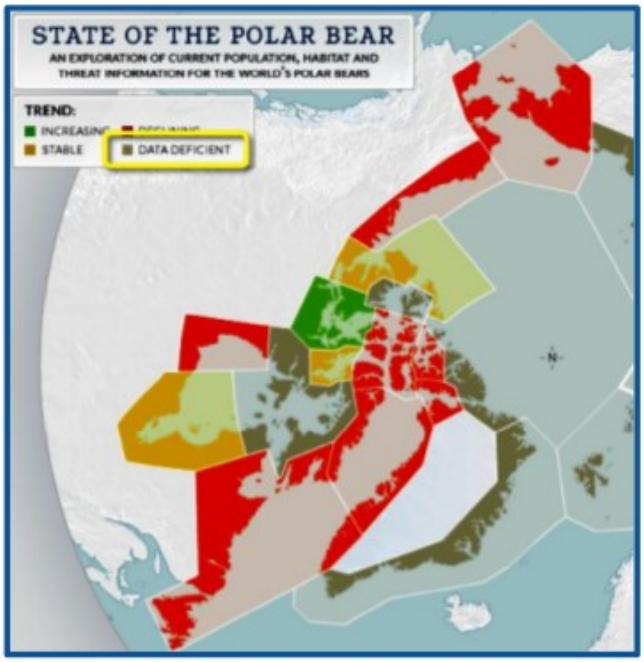
library(reshape2)
melt command can now be used

To use the tips dataset:
library(reshape2)
#The tips dataset can now be accessed.

To use the Ozone dataset
library(plyr)
The ozone dataset can now be accessed.

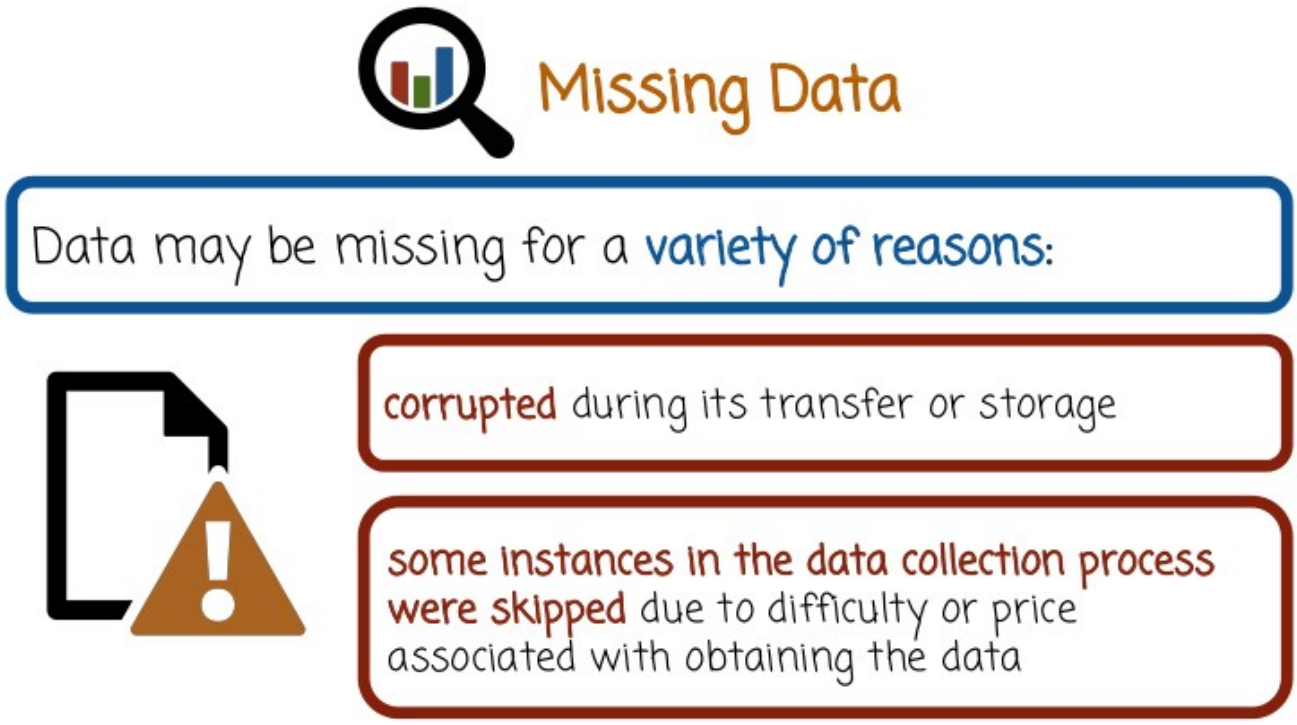


1. In this lesson we will learn about data preprocessing. In many data analysis tasks, the data preprocessing part takes the longest amount of time and is the most crucial. It can make the difference between getting the right insights or being misled by poorly prepared data. There are four goals in this lesson. Learn how to handle missing data. Learn how to handle outliers. Learn how to use data transformations. And learn standard data manipulation techniques.

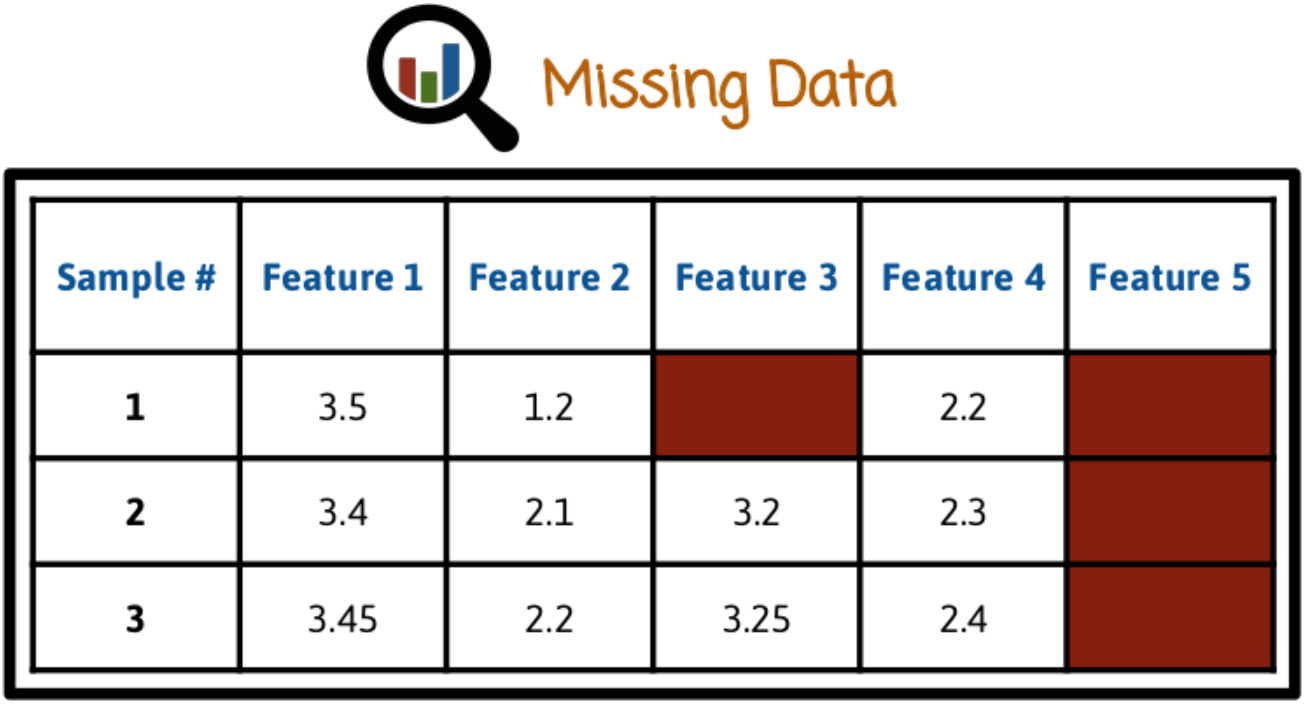


2. Let's start with a quick quiz about a study investigating polar bears in the Arctic region. The data of whether polar bear population is increasing or decreasing is used to color regions on the map with different colors. Green areas show where the polar bear population is growing, yellow areas show a stable population, red areas show a decline. Dark gray areas show where there's no information on the polar bear population. I wanted to take a guess as to why these areas do not have information. Here are a few options. There are no polar bears in these areas. There are too few polar bears to achieve a reliable count. These areas are controlled by a country that did not allow the collection of the data.

3. The label of data deficient is not the same as no polar bears. It means that the data or measurement is missing so we can discard the first answer. The second answer is technically possible though the data deficient label seems to suggest the third answer. In this specific case, different regions belong to different countries and it is quite likely that the lack of data has to do with the lack of being able to measure in specific areas. The author of the study in fact points to Russia providing limited access and so interestingly the presence of no data or missing data sometimes conveys some information about the data in itself which may be useful in the data analysis task.

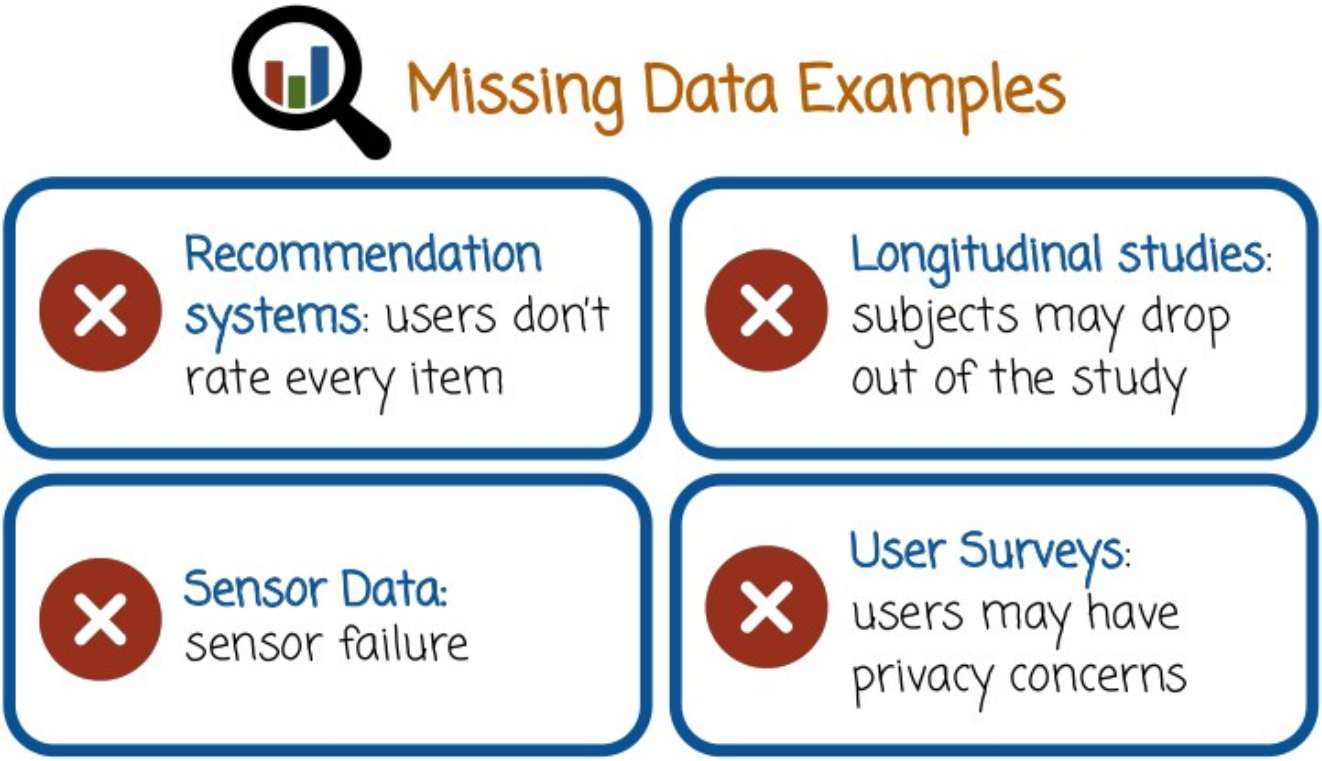


4. Data may be missing for a variety of reasons. Data could be corrupted during its transfer or storage. For example, a common problem with old records is that the media on which the data was stored was damaged over the years. Another example is data that is spread over multiple files where some files are erased or not kept for some reason. It could also be that during the data collection process some instances were skipped. For example, when collecting spatial measurements over a period of time, some areas may be harder to get to and so may be measured with lower frequency than other areas.



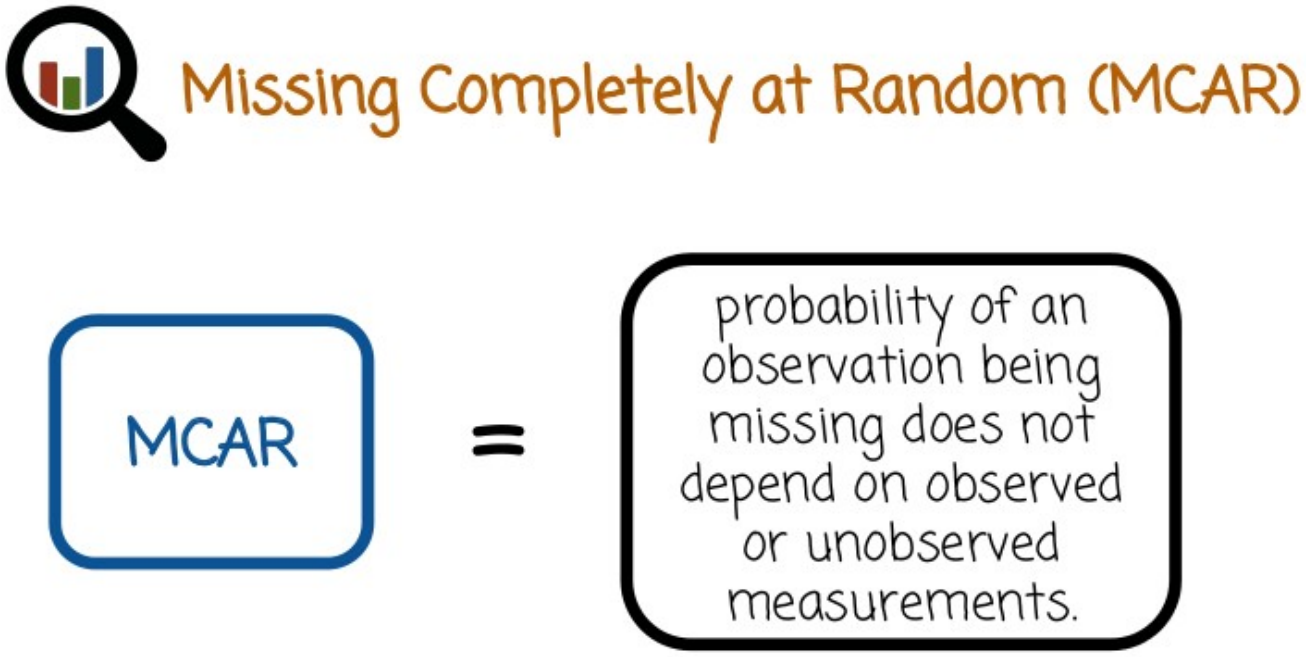
Sample #	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
1	3.5	1.2		2.2	
2	3.4	2.1	3.2	2.3	
3	3.45	2.2	3.25	2.4	

We identify samples as rows, and variables or features as columns. Different features in different samples may be missing. For example in this case, the first sample, which is the first row, may have its third feature or column missing, while the second and third samples may have the fifth feature missing. In this specific case, we have the first sample having two variables missing, the third variable and the fifth variable, while the second and third sample have the fifth variable missing.



上圖是 data missing 的一些例子

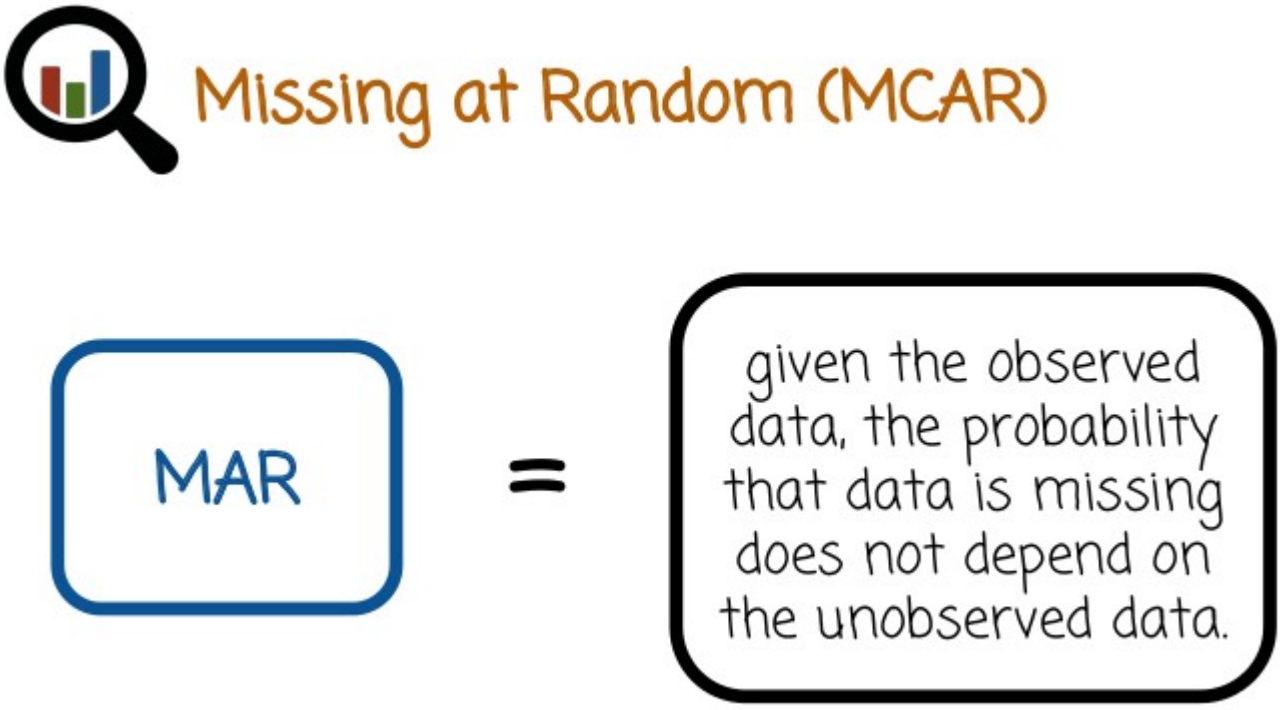
5. Recommendation systems, recommend to users items from a catalog based on historical user ratings. Often, there are a lot of items in the catalog, and each user typically indicates their star ratings for only a small subset of them. The movies that the user didn't watch, as well as the movies that the user watched but didn't rate are considered missing. In longitudinal studies, some of the subjects may not be able to attend each of the surveys throughout the study period. The study organizers may also have lost contact with some of the subjects in which case all measurements beyond a certain time point are missing. In sensor data some of the measurements may be missing due to sensor failure, battery discharge or electrical interference. In user surveys, users may choose to not respond to some of the questions for privacy reasons. These are all examples of why data may be missing.



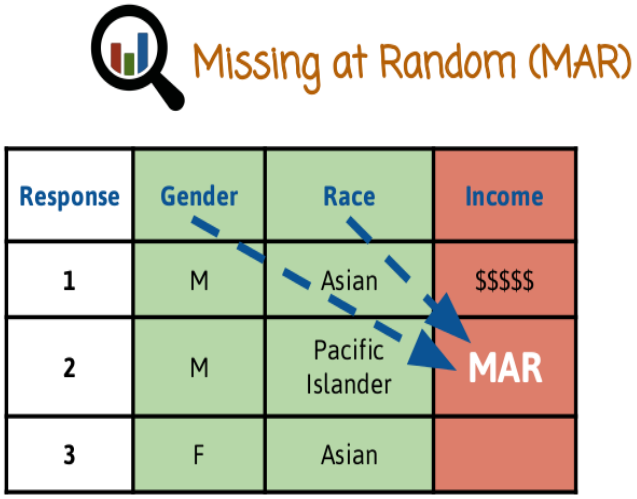
6. MCAR, or missing completely at random, states that the probability of an observation being missing does not depend on observed or unobserved measurements.

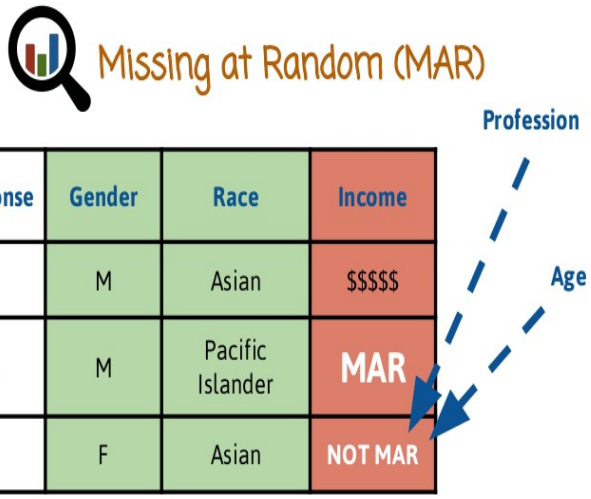
For example, in the case of users rating movies using star ratings of one to five stars, we consider ratings of specific movies as data frame columns. We also consider rows representing users. Since some movies are more popular than others, some of the columns are more likely to be more missing than others, depending on the popularity of the movie, violating the MCAR definition. In this specific

case, we see that the movie Casablanca has much more ratings while the movie Spies has fewer ratings.



7. Missing at Random, or MAR, states that given the observed data, the probability that data is missing does not depend on the unobserved data.

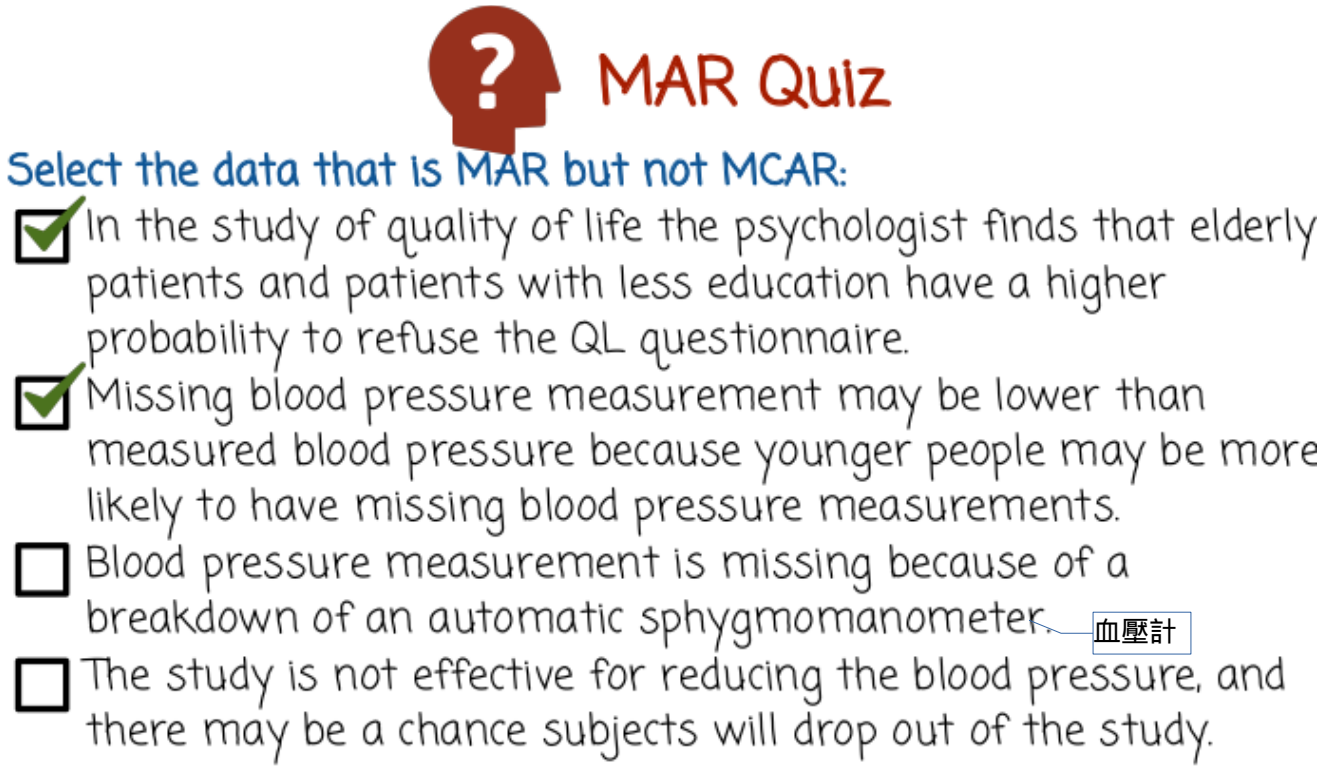




For example, in a survey recording gender, race, and income, gender and race are not very objectionable questions. So we may assume for now that the survey respondents answered these questions fully. The income question is more sensitive and users may choose to not respond for privacy reasons. The tendency to report income or to not report income, typically varies from person to person and may depend on factors such as gender or race. If it only depends on gender and race, then the data is Missing at Random, or MAR. If it is not, Missing Completely at Random, or MCAR, in this case. If the decision whether to report or not report income depends also on other variables that are not

in the data frame, such as age or profession, or under variable value itself, in this case the income level, the data is not MAR. This is likely the case for this survey.

8. In this quiz about data missing at random, let's select the data that is missing at random, but not missing completely at random. In the study of quality of life the psychologist finds that elderly patients and patients with less education have a higher probability to refuse the QL questionnaire. Missing blood pressure measurement may be lower than measured blood pressure because younger people may be more likely to have missing blood pressure measurements. Blood pressure measurement is missing because of a breakdown of the device. The study is not effective for reducing the blood pressure and there may be a chance subjects will drop out of the study.



9. The first statement, the data would be MAR, assuming the specified dependencies, age and education, are recorded in the survey, and there are no additional dependencies. Similarly, the second statement is also MAR, assuming that the missingness depends only on age and the age is recorded. The third statement is tricky. The missing data is missing completely at random, or MCAR, since the device breaking down does not depend on observed or unobserved data. So in this case, the data is MCAR. It's also MAR, but note that the question in the quiz asked for data that is MAR and not MCAR, so this question here is incorrect. The fourth statement is MCAR if subject drop out of study with fixed probabilities. Or it's not MAR if the likelihood of dropping out depends on patient's unobserved variables. In both of these cases, we do not select the bullet because the condition of the quiz is not satisfied.

Handling Missing Data

Response	Gender	Race	Income
1	M	Asian	$$$$$
2	M	Pacific Islander	
3	F	Asian	

10. Most methods are designed to work with fully observed data. There are some general ways to convert missing data to non-missing data.

Handling Missing Data

Remove all data instances (for example dataframe rows) containing missing values.

Response	Gender	Race	Income
1	M	Asian	$$$$$

The simplest way, though not necessarily the best, is to remove all instances or data frame rows with missing values. In this case, we will only keep the first row of the data frame, and remove the second

and the third row.

Handling Missing Data

Replace all missing entries with a substitute value, for example the mean of the observed instances of the missing variable.

Response	Gender	Race	Income
1	M	Asian	$$$$$
2	M	Pacific Islander	**mean of income**
3	F	Asian	**mean of income**

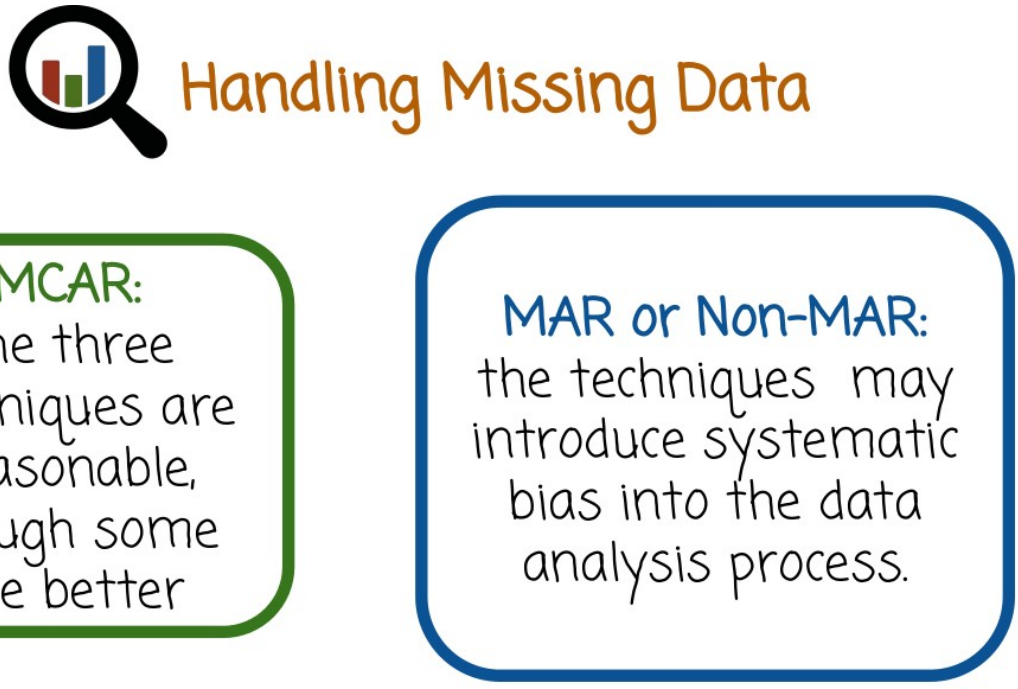
A more sophisticated method, rather than remove the rows with missing data, is to replace the missing entries with generic substitute values such as the median or mean of the variable if the variable is numeric or the most likely value if the variable is categorical. This allows us to retain the rows and use standard data analysis procedures that do not have an assumption that missing data exists.

Handling Missing Data

Estimate a probability model for the missing variable and replace the missing value with one or more samples from that probability model.

Response	Gender	Race	Income
1	M	Asian	$$$$$
2	M	Pacific Islander	**Probability Model Sample 1**
3	F	Asian	**Probability Model Sample 2**

A third method of handling missing data would be to build a probability model for the variable that is missing, in this case income, sample from that model values, and insert the randomly sampled values in place of the missing entries. This method is called imputation.



In the case of MCAR data, all three techniques are reasonable, in that they may not introduce systematic errors. However, some techniques are better than others. For example, imputation is considered better than substitution with mean or median. And substitution with mean or median is considered better than removing instances altogether. For non-MAR data, the three techniques may introduce systematic bias into the data analysis process. For MAR data, the first and second method introduce systematic bias, and the third one may or may not, depending on how carefully the probability model from which the samples are collected is modeled.

Missing Data and R

Response	Gender	Race	Income
1	M	Asian	$$$$$
2	M	Pacific Islander	NA
3	F	Asian	NA

R represents missing data using the NA keyword.

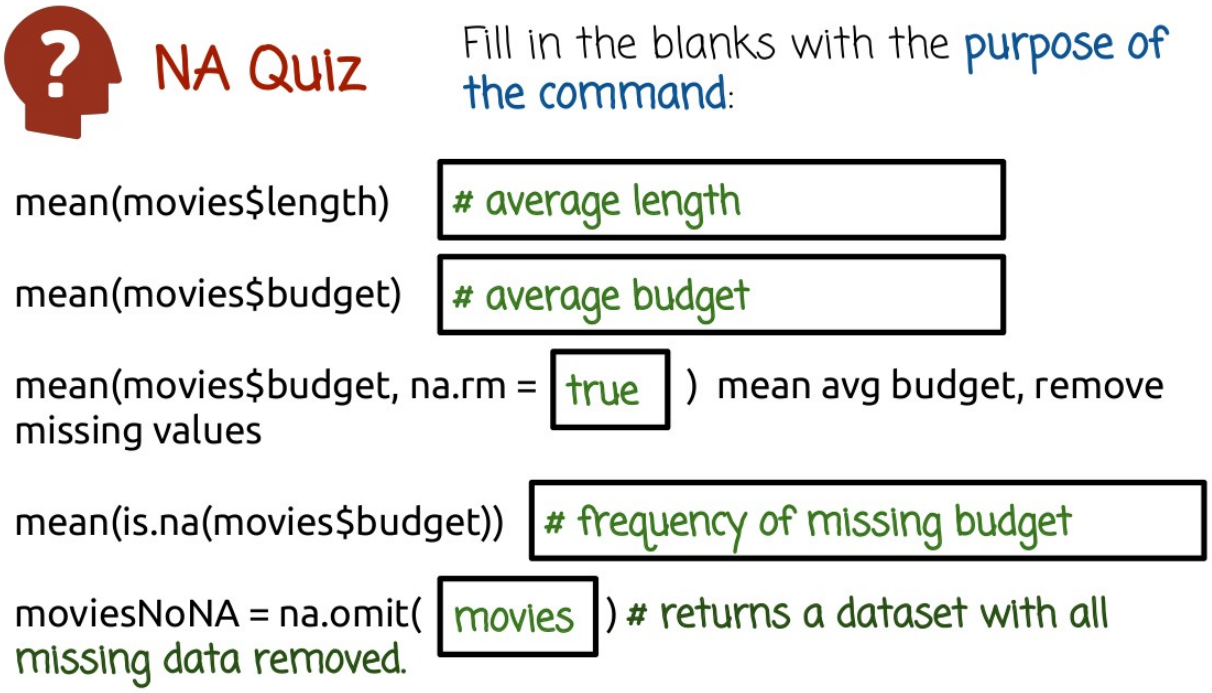
11. In R, we usually keep data in data frames where rows are instances and columns are variables. R uses the NA keyword to represent missing data. In this case, this could be a dataframe in R, and when the data is missing, R would represent it as NA symbols.

Missing Data and R

is.na	• Returns TRUE for missing data • Returns FALSE otherwise
complete.cases()	• Returns a vector whose components are FALSE for all samples • Returns TRUE otherwise
na.omit()	• Returns a new dataframe omitting all samples containing missing values
na.rm	• If set TRUE changes the function behavior so that it proceeds to operate on the supplied data after removing all dataframe rows with missing values

Here are a few functions and tools that R provides to handle missing values. The function is.na returns a data structure having true values where the corresponding data is missing, and FALSE otherwise. The function complete.cases returns a vector whose components are FALSE for all samples or

dataframe rows containing missing values, and TRUE otherwise. na.omit returns a new dataframe omitting all samples or dataframe rows containing missing values. Some functions have a na.rm argument which if set to TRUE changes the function behavior. So that it proceeds to operate on the supply data after removing all data frame rows with missing values.



12. The code below analyzes the data frame "movies" in the ggplot2 package, which contains 24 attributes, such as genre, year, budget, user ratings, etc. For 58,000 movies obtained from the website, IMDB.com with some missing values. Please fill in the blanks with code or comments.

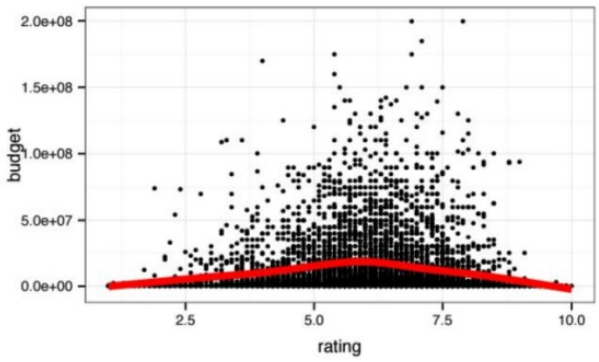
13. The first line computes the mean of the length variable. The second line computes the mean of the budget variable. The third line computes the mean of the budget variable excluding missing values because of the na.rm parameter being set to true. The fourth line computes the mean of an indicator vector capturing movies with missing budget. And therefore, it measures the frequency of movies with missing budgets. The last line creates a new dataset with missing values removed using the na.omit function.

14. Now that we have generated the desired data set, without missing values, let's plot it. Fill in the blanks to create the given plot.

R Plot Quiz

Fill in the blanks to create the given plot:

```
moviesNoNA = na.omit(movies)
qplot(rating, budget, data = [moviesNoNA] , size = I([1.2] )) +
   stat_smooth(color = " [red] ", size = I( [2] ), se = [F] )
```



15. In the first blank, we use the data frame that we created without the missing values, moviesNoNA. In the second blank, we selected an appropriate size for the scatter plot markers. The specific size does not matter. In the third blank, we want to choose the color of the smoother line, we can choose other colors as well. In the fourth blank, we select the width of the smoother line. In the last blank, we provide the letter F which stands for false, in order to not include standard error bars together with the smoother line, rather only display the smoother line

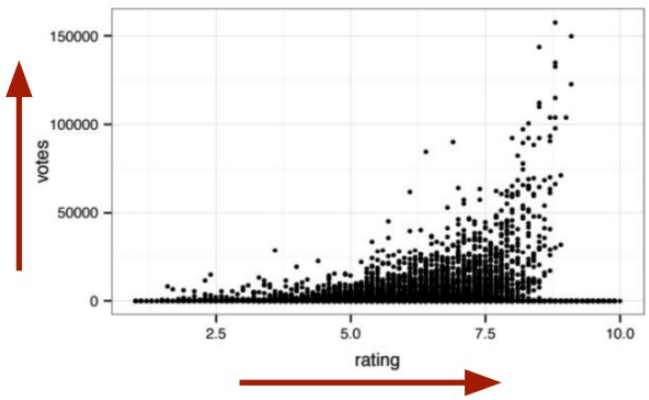
16. This is a plot graphing the number of votes for a movie versus the average rating of the movie on a scale of one to ten for IMDB movies. The code that generates the graph is listed above the graph. Which of the following statements can be derived from the plot? The number of votes tend to increase as the average ratings increase. The spread in the number of votes increases with the average rating. Movies featuring the highest average ratings have a very small number of votes. Observed ratings will tend to be higher than ratings gathered after showing users random movies.

Movie Data Analysis Quiz

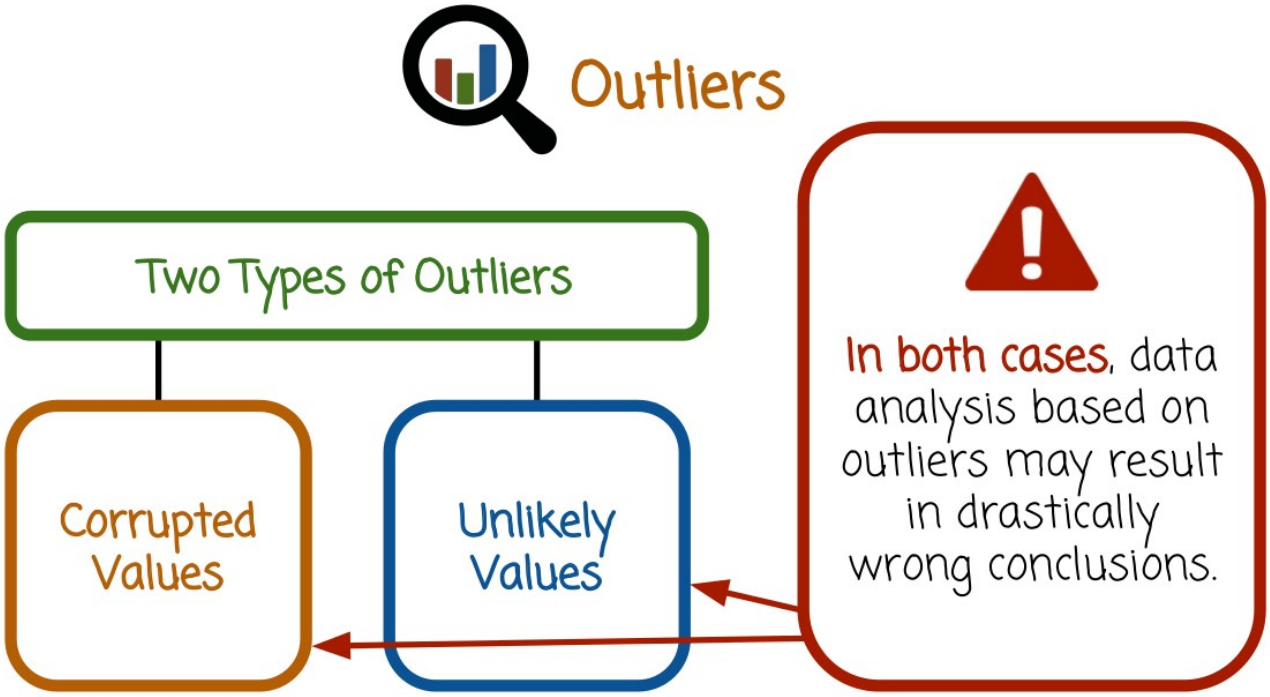
Select **which of the following statements can be derived** from the plot:

- [x] Number of votes tend to increase as the average ratings increase.
- [x] Spread in the number of votes increases with the average rating.
- [x] Movies featuring the highest average ratings have a very small number of votes.
- [x] Observed ratings will tend to be higher than ratings gathered after showing users random movies

```
moviesNoNA = na.omit(movies)
qplot(rating, votes, data = moviesNoNA, size = I(1.2))
```

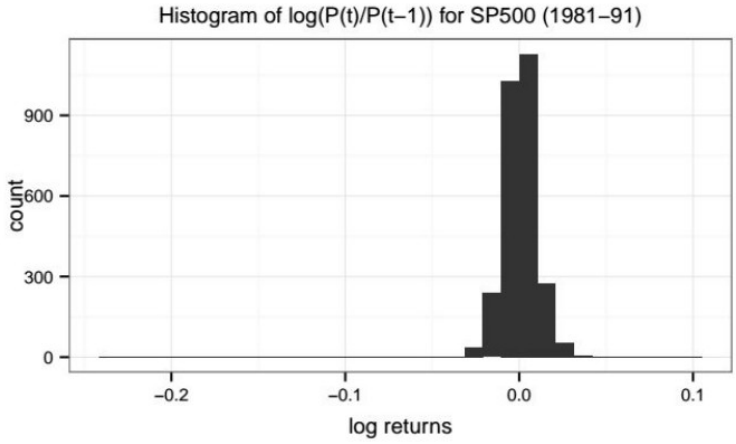


17. The first bullet is correct since the graph shows a positive correlation, or upward trend between the two variables, rating and votes. The second bullet is correct since the variance in the y axis increases for large values of the x axis. The third bullet captures the bottom right corner, where there are only a few movies with very high ratings but only a few votes. This is kind of surprising, because it seems to contradict the upward correlation trend of the first bullet. One possible explanation for this phenomenon that we see right here, is that there are some movies that only a few people watched them. And for these movies it's much easier to get a very high average ratings. The last bullet is a reasonable explanation for the first bullet, though not necessarily a proof of causality. It explains why there is an upward trend here. Movies that are popular are popular because viewers choose to view the movies. And because they choose to view the movies, they have some basic expectation that they would like the movies. That contributes to popular movies having relatively high ratings.



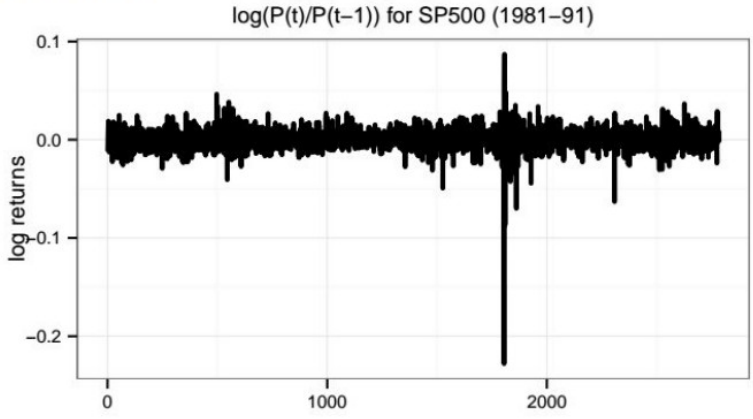
18. Outliers are extreme values that are much bigger or smaller than the rest of the data. There are two types of outliers, the first type is corrupted values. For example, the result of human error during a manual process of entering measurement in a spreadsheet. The second type of outliers are substantially unlikely values given our modeling assumption. One example is the Black Monday stock rush on October 19th, 1987 when the Dow Jones Industrial Average lost 22% in one day. In both cases, data analysis based on outliers may result in drastically wrong conclusions, that applies especially if they are corrupted values. But it can even happen if the values are correct and not corrupted. But they are highly unlikely and violate our modeling assumption.


```
library(Ecdat)
data(SP500, package = 'Ecdat')
qplot(r500,
  main = "Histogram of log(P(t)/P(t-1)) for SP500 (1981-91)",
  xlab = "log returns",
  data = SP500)
```



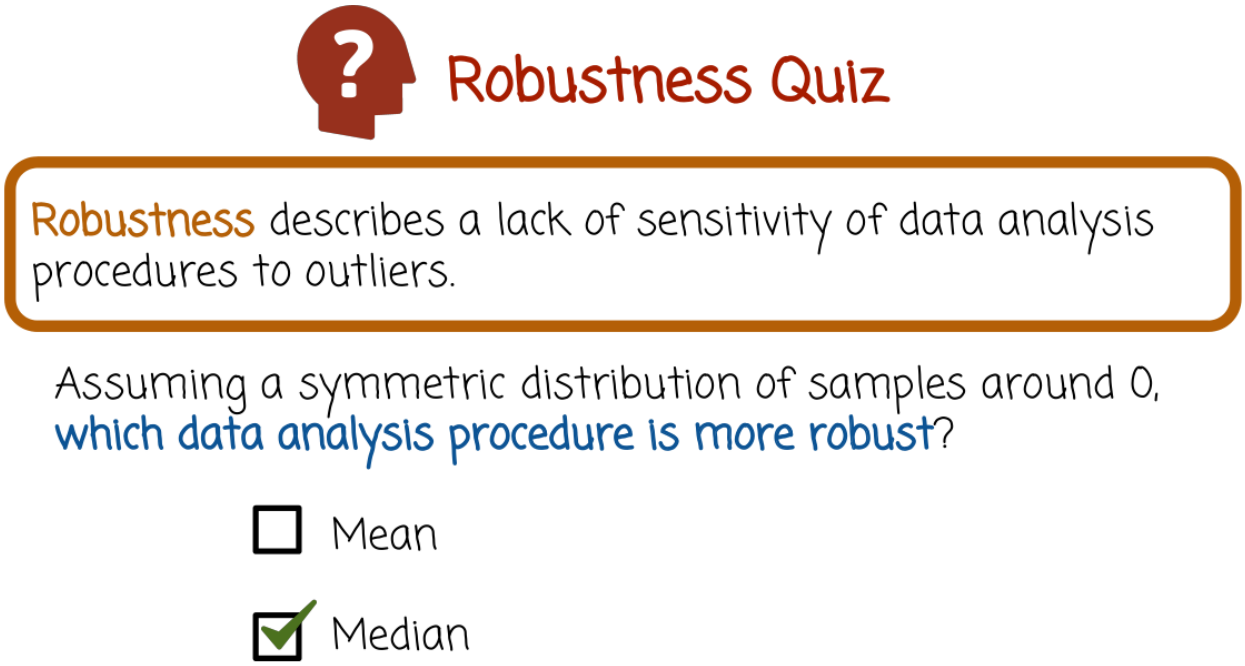
Here is a graph of the log daily returns in the S&P 500, the stock index, for a period of 10 years between 1981 and 1991. A daily return of one or a log daily return of zero indicates zero profit if you invest $1 in one day and you sell it the next day. We see here a histogram that is centered around zero. Value 0 means, that the investor did not get any return from one day to the next day. A positive value, shows that there is a positive revenue gain, or a negative value show that there is a loss. The code that generates the graph is written right here. The histogram seems nice and unimodel and does not show any suspicious properties, if we wanted to model it, we can perhaps chose the Gaussian distribution.


```
qplot(seq(along = r500),
  r500,
  data = SP500,
  geom = "line",
  xlab = "trading days since January 1981",
  ylab = "log returns",
  main = "log(P(t)/P(t-1)) for SP500 (1981-91)")
```

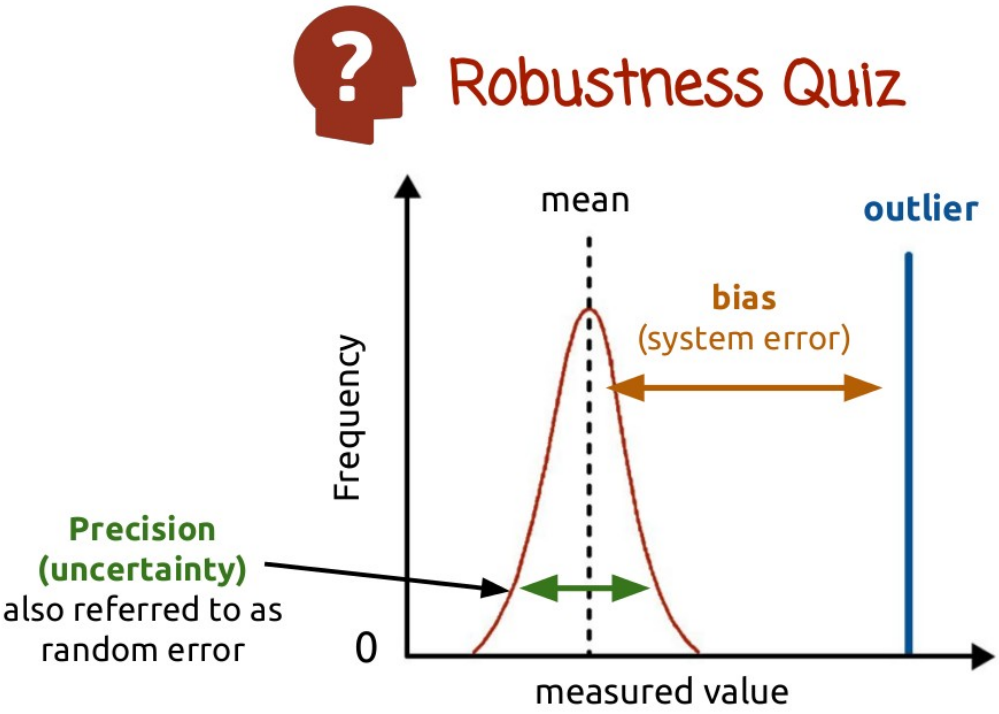


On the other hand, drawing a line graph of the log daily returns across the period show a huge variation or outlier right here. Again, the data that generates the graph is displayed right here. This graph exposes that we have a very, very significant outlier, and if we choose a Gaussian distribution to model the data, we may not be making or drawing the right conclusion later on.

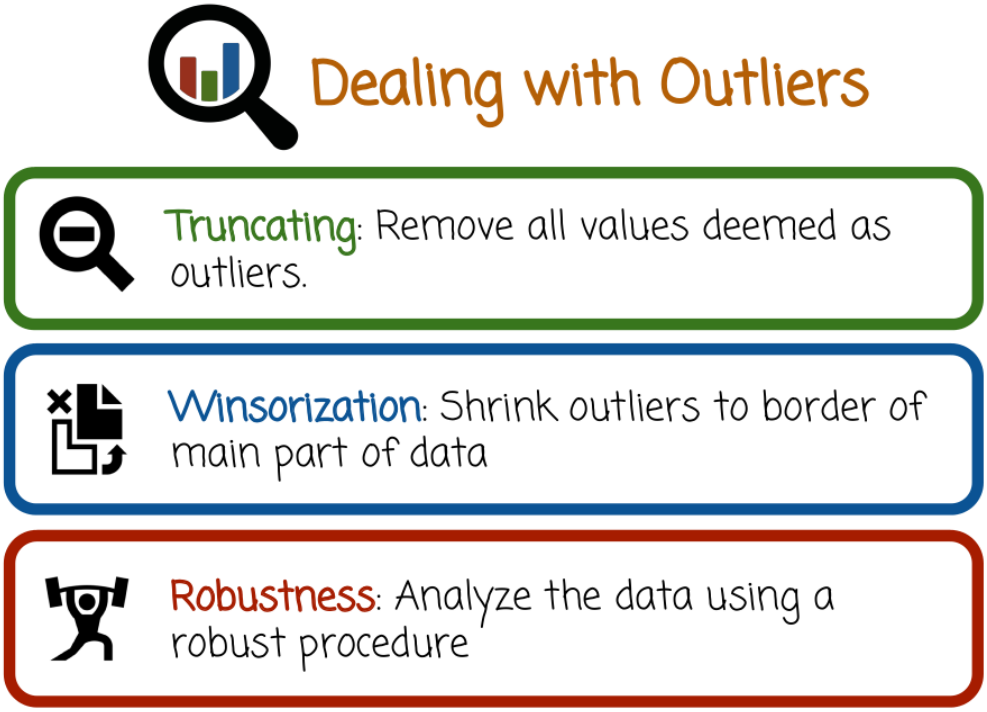
19. Robustness describes a lack of sensitivity of data analysis procedures to outliers. Assuming a symmetric distribution of samples around zero, which data analysis procedure is more robust, mean or median?



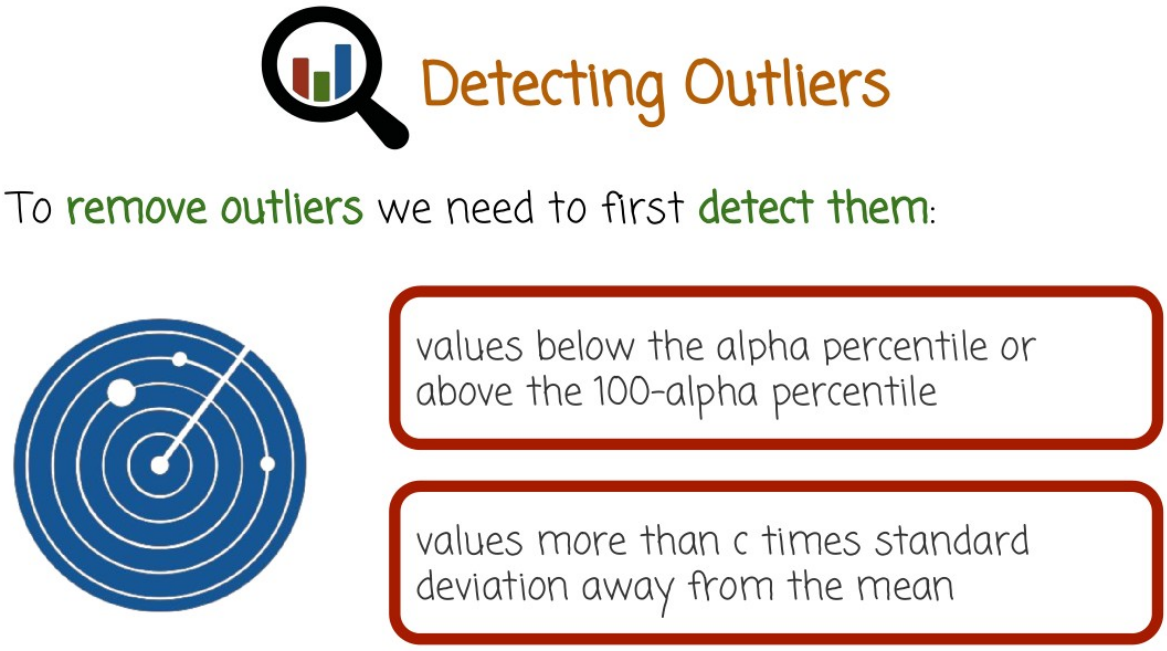
20. The mean of n numbers is a non-robust procedure, while the median is a robust procedure. Let's see why in the next slide.



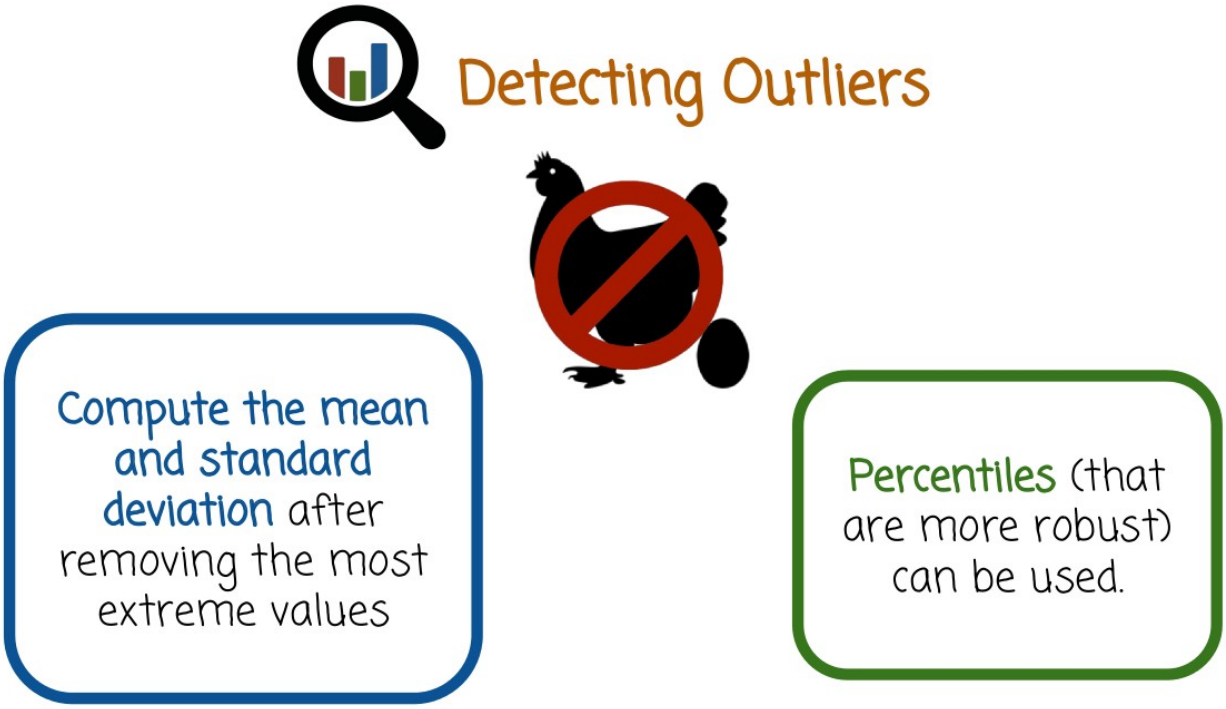
In this graph, we assume that the measured values are scattered very close to the mean. The mean is the dashed line and the outlier, perhaps a single outlier, is shown by the blue line right here. As the blue line gets further and further away from the red curve, or the rest of the data, the mean will move to the right. In the limit, when the outlier will go to infinity, the mean will go to infinity as well. This also happens if there's a single outlier in an extremely large data set. No matter how large the data set is, if the outlier moves to infinity, the mean will move to infinity as well. This is not so for the median. The median will stay in its place regardless of where the outlier goes. The outlier can go more and more to the right, but the median will stay fixed in its position without moving. In this case we refer to two type of uncertainty or bias. There's the precision uncertainty, which is the deviation of the data from the mean. And then there is also the bias of the outlier, it could be system error if the data is corrupted.



21. Let's see how we can deal with outliers. The first option is truncating or removing all values deemed as outliers. The second technique to deal with outliers is called winsorization. This technique shrinks outliers to the border of the main part of the data (後面 quiz 有個例子). In one special case, winsorization removes the outliers and replaces the outliers with 'the most extreme data points that remain after removing the outliers'. The third technique for dealing with outliers is not to remove the outliers or to modify them. Rather, we keep the outliers in the data as they are but we choose robust procedures to analyze the data. For example, if we want a summary of the value, we may want to choose the median rather than the mean. As we saw before, the median is much more robust than the mean to outliers.



And now, we are left with the question of how to detect outliers, because to remove outliers, or to winsorization the data, we need to first detect the outliers. Here is one way to detect outliers. We can define outliers to be values below the alpha percentile or above the 100-alpha (即 100 減去 alpha) percentile, for a small alpha such as 5 or 2, or even 1. A different strategy is to define outliers as values that are more than c times standard deviation away from the mean. Where c is a small integer such as 1 or 2, 3 or 5. The second technique is related to the first technique in that it follows from it if we assume that the data has a Gaussian distribution.



We do have a problem however, because to detect outliers, we need to compute the mean and the standard deviation. And the mean or the standard deviation may be affected by the outliers. So that raises a problematic issue in that the outliers will corrupt the calculation of the mean and the standard deviation. Which are needed to define what the outliers are. So we cannot remove the outliers based on corrupted measurements of mean and standard deviation. Here is one solution to this chicken and egg problem. We can compute the mean and standard deviation after removing the most extreme values. We'll see that in the next few slides. In this case, the mean and standard deviation are more robust in that they are not based on extreme values. A different strategy would be to not use the mean and standard deviation that are non-robust procedures. Rather use percentiles. Percentiles are robust procedures. Similar to median, in fact the median is the 50th percentile. So if we want to define outliers by percentiles, we can just look at the values that are smaller than the alpha percentile or larger than 100 minus alpha percentile for a small alpha.

22. Write code using R to first create samples from a normal distribution and create an outlier adding to the data, print the data, and then winsorize it.

Programming Quiz

Write code using 'R' to first create samples from a normal distribution and an outlier, print it, and then winsorize it

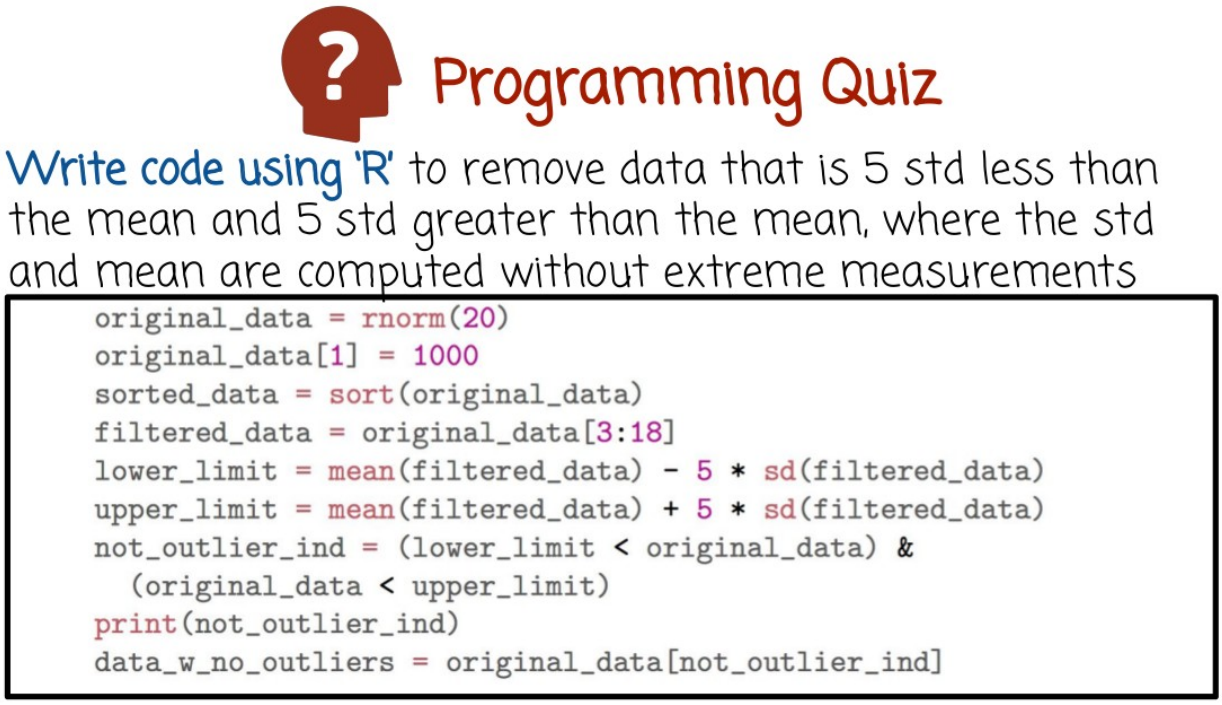
```
library(robustHD)
originalData = c(1000, rnorm(10))
print(originalData[1:5])

## [1] 1000.0000   -0.6265    0.1836   -0.8356    1.5953

print(winsorize(originalData[1:5]))

## [1]  3.2060 -0.6265  0.1836 -0.8356  1.5953
```

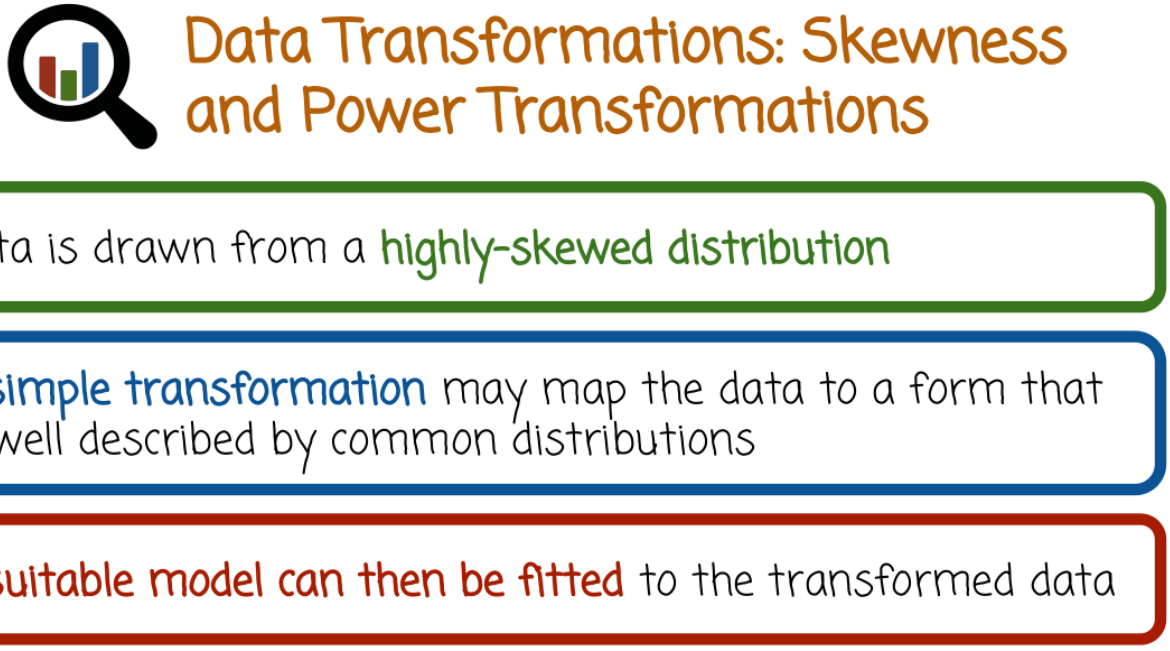
23. In this code here, we sample data from a normal distribution in the second line. And then we concatenate it with an outlier whose value is 1000. More specifically we create a vector whose first value is 1000 or an outlier. Followed by 10 samples from a normal distribution with mean 0 and standard deviation 1. The third line prints the data. We only print the first five values rather than the entire data to save space, but you can print the entire data as well. You can see the first value is the outlier and the next values are much smaller. To do winsorization we call the winsorize function in R and we pass this argument, the original data. Again, we use only the first five values to save space. As you can see, the outlier was removed by the winsorize function and replaced by a value that is much more similar to the rest of the data. If you want more details on the winsorize function in R, go to R and type help winsorize to see the details of exactly how this value 3.2060 is created.



上圖中 not_outlier_ind 是 a vector of indices

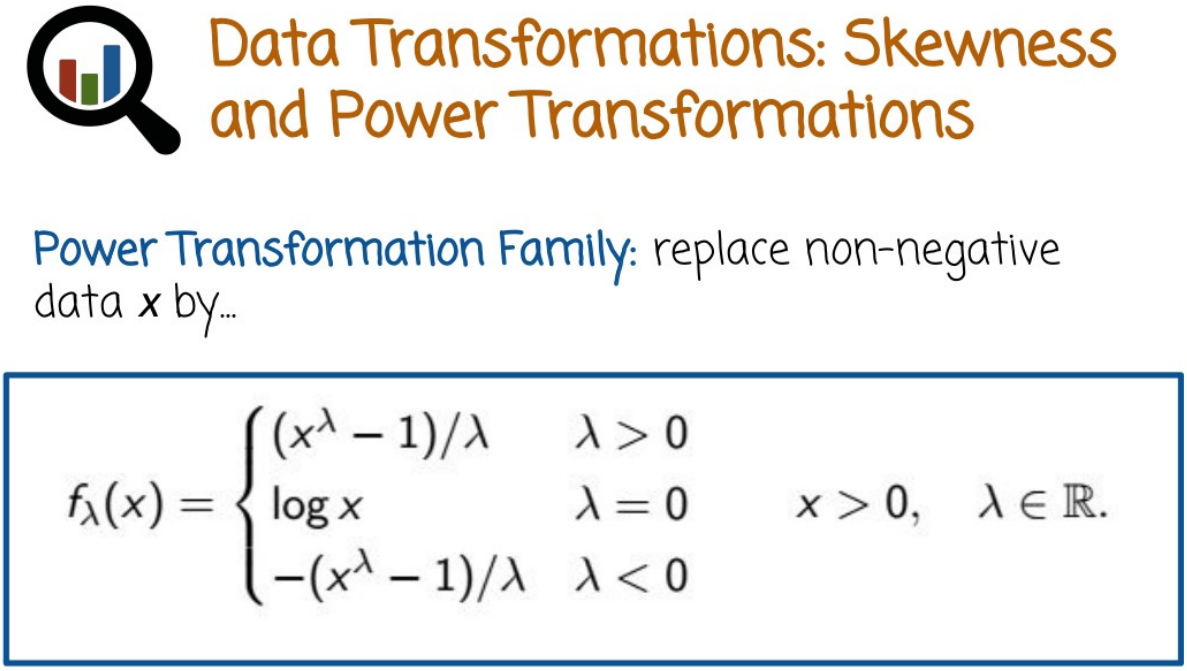
24. Write code using R to remove data that is 5 standard deviation less than than the mean and 5 standard deviation greater than the mean, where you compute the mean and standard deviation in a robust way by removing extreme values.

25. In this example, the first line generates data, random samples from a normal distribution. Then we replace the first value by an outlier whose value is 1000. The third line sorts the data that will make it easy to remove the extreme values, which is done by the fourth line. Specifically, we discard the smallest two values and the largest two values measurement. We only do that in order to compute the mean and the standard deviation that are used to define outliers. If we do not do that, then the mean and the standard deviation will be impacted a lot by the value of the 1000 here, 1000 outlier. The fifth and the sixth line define the lower and the upper limit of where we define the outliers. Notice that when we compute the mean and the standard deviation, we use the filtered data which was created by removing the extreme measurements, including the outlier of 1000. In this line here, we create a vector of indices (not_outlier_ind), of values that are not outliers. And then we convert the original data to data without outliers by picking all components or the indices that are present in the not_outlier_ind vector.



skewed(斜歪的)

26. In many cases, data is drawn from a highly skewed(斜歪的) distribution that is not well described by one of the common statistical distributions. A simple transformation may map the data to a form that is well described by common distributions such as the Gaussian or gamma distributions. A suitable model can then be fitted to the transformed data. If necessary predictions can be made on the original scale by inverting the transformation.



The most frequently used family of transformations are the power transformations or the power

transformation family. These transformations replace a non-negative data value x by f of x, parameterize by lambda, the definition of f of x, differs based on lambda, for lambda greater than zero, it is x to the power of lambda minus one divided by lambda, for lambda equals zero it's log of x, and for lambda smaller than zero, it is negative of x to the power lambda- 1 / lambda. What we have here is not a single transformation but a family of transformation. And for every selection of the parameter lambda, we define a different transformation.

Data Transformations: Skewness and Power Transformations

- The power transform maps x to x^{λ} up to multiplication by a constant and addition of a constant.
- Subtracting 1 and dividing by λ makes $f_{\lambda}(x)$ continuous in λ as well as in x

$\lambda > 1$	$\lambda < 1$
mapping is convex	mapping is concave
removes left skewness	removes right skewness

The formula in the previous slide may look somewhat complex and it's not clear why did we choose that specific definition. The transformation that we saw earlier basically maps x to x to the lambda up to multiplication by a constant and addition of a constant. That operation of subtracting one and dividing by lambda makes f of lambda of x continuous in both lambda and in x. But if you want to interpret them intuitively, this transformation basically amount to raising x to the power lambda with some small modifications of multiplication by a constant and addition of a constant. For lambda greater than one, the curve or the transformation is convex. It can be used to remove left skewness in the data. For lamda less than one, the mapping is concave, removes right skewness in the data. One example of lambda greater than one is lambda equals two. In this case, we raise x to the power two. Again, up to multiplication and addition by a constant. In the case of lambda less than one, we can think of an example being the square root, or lambda equals 0.5. This transformation replaces a value x by the square root of x. Again, up to multiplication and addition of the constant.

Data Transformations: Skewness and Power Transformations

To select λ :

Try different values, graph the histograms, and select one of them.

Use a method based on maximum likelihood.

How do we select the value lambda? One way is to just try different values, graph the histograms and use the value that shows the most insight. A second method is more principal using statistics, specifically using the maximum likelihood estimator to fit the best lambda to the data. In practice, many people just try different values of lambda, graph the histograms and use the best histogram or maybe a combination of several histograms to represent the data.

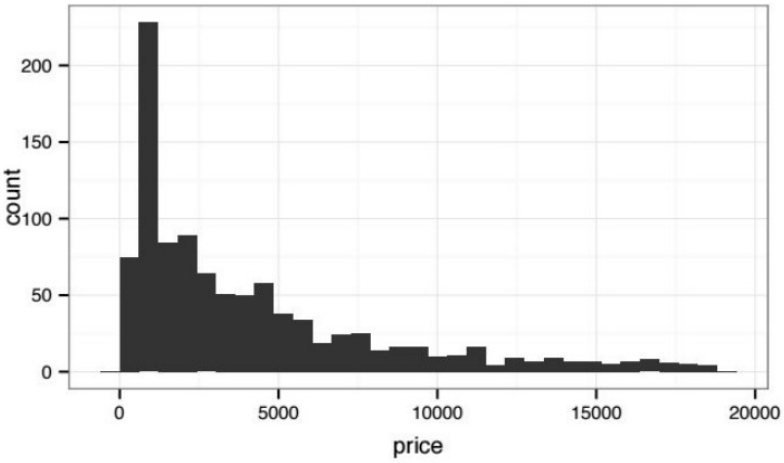
Diamonds Example

```
print(diamonds[1:10,1:8])
```

```
##    carat       cut color clarity depth table price    x
## 1   0.23     Ideal     E     SI2  61.5    55   326 3.95
## 2   0.21   Premium     E     SI1  59.8    61   326 3.89
## 3   0.23      Good     E     VS1  56.9    65   327 4.05
## 4   0.29   Premium     I     VS2  62.4    58   334 4.20
## 5   0.31      Good     J     SI2  63.3    58   335 4.34
## 6   0.24 Very Good     J    VVS2  62.8    57   336 3.94
## 7   0.24 Very Good     I    VVS1  62.3    57   336 3.95
## 8   0.26 Very Good     H     SI1  61.9    55   337 4.07
## 9   0.22      Fair     E     VS2  65.1    61   337 3.87
## 10  0.23 Very Good     H     VS1  59.4    61   338 4.00
```

27. The diamonds dataset contains many diamonds, their carats or weight, their cut, color, clarity, depth, table, price, and different measurements such as x, y and z. This print function here, we display the first 10 rows and the first 8 columns.


```
diamondsSubset = diamonds[sample(dim(diamonds)[1], 1000),]
qplot(price, data = diamondsSubset)
```

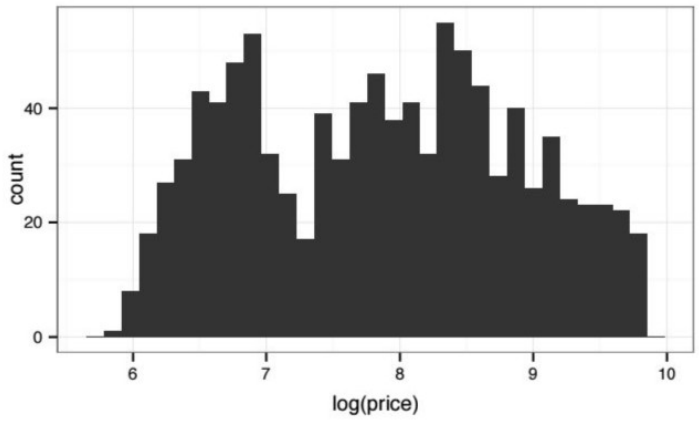


In this graph we sample 1,000 diamonds from the dataset, and then we plot a histogram of the price of that sample. The reason we sample 1,000 diamonds is because the dataset is very big, and computing the histogram of the entire dataset may take a while. The shape of the histogram is interesting. It shows that the histogram goes up to a certain maximum point, and then it starts to go down, meaning they're relatively few diamonds that are less expensive. The number of diamonds go up very quickly, up to a typical price. And then it starts to decay slowly with a long tail, until you reach very high prices with very low diamonds.

28. In this graph, we plot the histogram of the log of the price of the diamonds from the previous data set. Recall the previous histogram was for the price variable and here we use a power transformation, the log of the price. The code that generates the graph is displayed on top. Based on the given graph, what conclusions can we draw about the count price relationship?

Diamond Quiz

```
qplot(log(price), size = I(1), data = diamondsSubset)
```



Based on the given graph, what conclusions can we draw about the count-price relationship?

we see a bi-modal relationship here that was not visible on the original scale

29. This graph shows a bi-modal relationship that was not visible on the original scale before we transformed the values (上上圖). As the log of the price increases, we see an increase in the count of diamonds. And then a decrease, and then another increase and a decrease. This detail was not visible on the original scale.

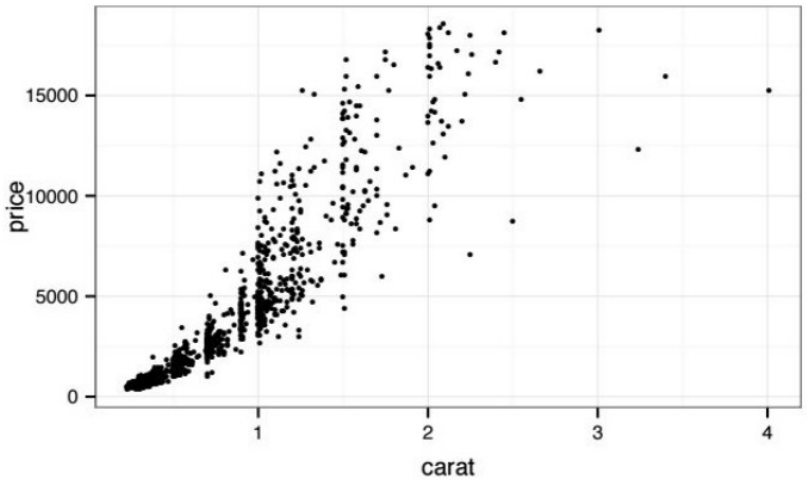
Power Transformation

Power Transformations can be used to examine the relationship between two or more data types.

30. We saw earlier how we can use power transformations to study histograms. Power transformation can also be used to examine relationships between two or more data types.

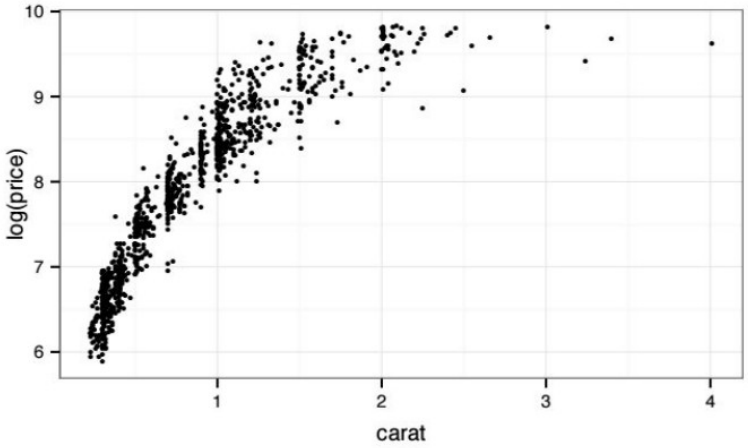
Power Transformation

```
qplot(carat,
      price,
      size = I(1),
      data = diamondsSubset)
```



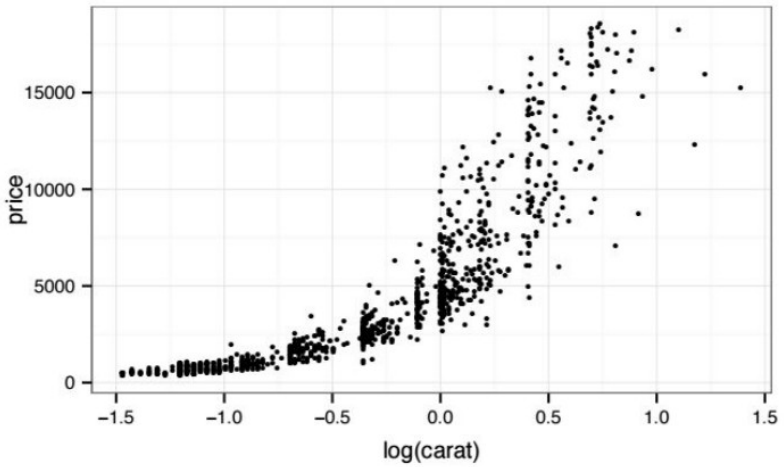
Power Transformation

```
qplot(carat,
      log(price),
      size = I(1),
      data = diamondsSubset)
```



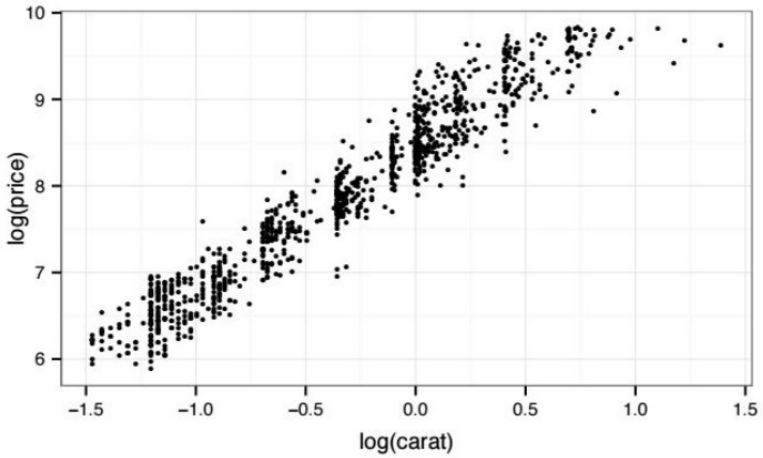
Power Transformation

```
qplot(log(carat),
      price,
      size = I(1),
      data = diamondsSubset)
```



Power Transformation

```
qplot(log(carat),
      log(price),
      size = I(1),
      data = diamondsSubset)
```



In this case we draw a scatter plot showing the relationship between price and carat from the diamonds dataset. Again we use the smaller dataset of 1000 samples rather than the full dataset. The graph shows an obvious positive correlation between price and carat. Let's see what happens if we transform the carat variable with the power transformation, or the price variable with the power transformation, or both of them. We'll see what happened with this relationship and we'll also see what additional insights plotting the transformed values would give, beyond the current graph in the original scale. In this case we transformed the price with a log transformation. In this case we transformed carat with a log

transformation. In this case we transformed both the price and the carat by a log transformation. Such a graph is also called a log log plot. This graph is very interesting, it shows in the log log scale, a linear relationship between carat and price that we could not see before when we look at the data in it's original scales. Why is that useful and insightful? It's insightful to know that there is a simple relationship between these two variables. The relationship is linear on the log log scale. If you want to build a statistical model to predict the price, based on the carat for example, a regression model. It makes more sense if you use linear regression to do the regression model on the log of the carat, rather than on the carat. And then interpret the result as the log of the price rather than the price. If you want to predict the price, you can then later use the exponential transformation to invert the log of the price transformation

31. We see here a data set of animals that exist in the MASS package. The first column in the data frame is the name of the animal. The second column is the body weight. The third column is the brain weight. The code here draws a scatter plot showing the relationship between the brain of an animal and the body of the animal within the animals data set. Given the following data and its plot, change the QPlot command displayed here to show the relationship between the body and brain mass on a log log scale.

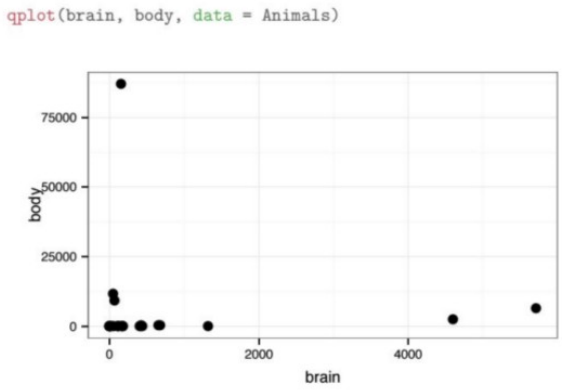
Power Quiz

Given the following data and its plot, change the qplot command to show the relationship between the body and brain mass on a log log scale

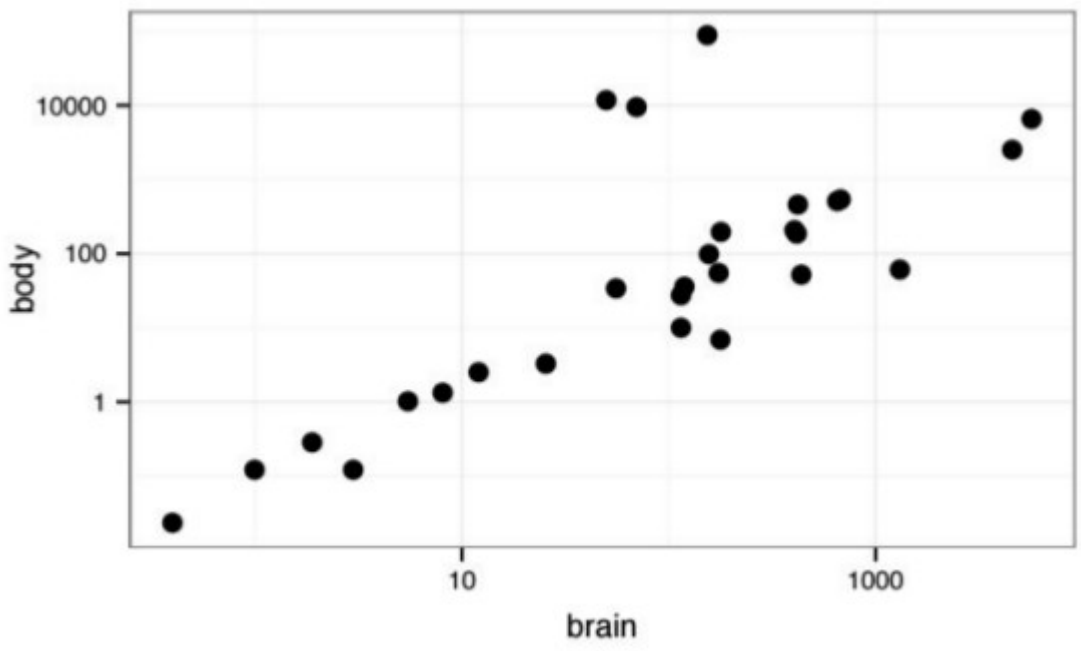
```
library(MASS)
print(Animals[1:12,])
```

```
##                    body  brain
## Mountain beaver     1.35    8.1
## Cow               465.00  423.0
## Grey wolf          36.33  119.5
## Goat               27.66  115.0
## Guinea pig          1.04    5.5
## Dipliodocus     11700.00   50.0
## Asian elephant   2547.00 4603.0
## Donkey            187.10  419.0
## Horse             521.00  655.0
## Potar monkey       10.00  115.0
## Cat                 3.30   25.6
## Giraffe           529.00  680.0
```

```
qplot(brain, body, log = "xy", data = Animals)
```



32. One way to do this would be to just transform brain and body with the log function, similar to what we did before with the diamonds data set. Here, we're showing a different way of doing that using an argument, log, that is passed to the Q plot function and assigning the value xy to that log argument, make sure that the Q plot displays the scatter plot on a log-log scale.



Here is the graph between the body and the brain mass on a log-log scale. As we see here, again, there is a very interesting additional information that is provided when we plot the transformed data, using a power transformation rather than the original data. The original data was not very informative. On the transform scale, we see a very nice, very clear linear relationship between the body weight and the brain weight, right here, and then we also see three additional points representing three additional animals that deviate significantly from this linear relationship on the log log scale. If you will investigate the animals date effect, you will find out that these three points right here, corresponding to animals, actually correspond to prehistoric animals or dinosaurs. This raises the question, why are the three points here deviating from the main line of contemporary animals. There could be several explanations, one explanation could be that in the past, prehistoric times, the relationship between body and brain was different than it is today. Perhaps the more likely explanation is that the brain weight and the body weight that we have of these three pre-historic animals are dinosaur, is just incorrect. That makes some sense, because our estimate of what was the body weight and the brain weight is based on indirect scientific analysis rather than direct measurement as it is for the other more contemporary animals. This kind of information, very nicely displayed on a log log scale, is very difficult to extract from the original plot on the original scale.

Data Transformations: Binning

Definitions:

Numeric variable: represents real valued measurements whose values are ordered in a manner consistent with the natural ordering of the real line.

Ordinal variable: represents measurements in a certain range R for which we have a well defined order relation.

Categorical variable: represents measurements that do not satisfy the ordinal or numeric assumption.

33. Numeric variables represent real valued measurements, whose values are ordered in a manner consistent with the natural ordering of the real line. This similarity between two measurements, a and b may be described by the Euclidean distance. The absolute value of B minus A. For example, height and weight are numeric variables. Ordinal variables represent measurements in a certain range for which we have a well defined order relation. Numeric variables are a special case of Ordinal variables. This means that all numeric variables are also ordinal variables, but not all ordinal variables are also numeric variables. For example, seasons of the year are ordinal measurements. The seasons of the year have a very clear ordering. But it does not make sense to subtract winter from spring, and measure the dissimilarity using that value. Categorical variables represent measurements that do not satisfy the ordinal or numeric assumption. For example, food items on the restaurants menu are categorical variables. Fish or meat, or pasta, there is no clear ordering on these variables and you cannot subtract one from the other.

Data Transformations: Binning

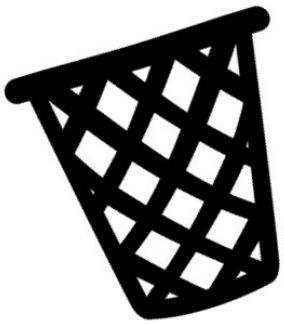
Binning (also known as discretization): taking a numeric variable x 2 R (typically a real value, though it may be an integer), dividing its range into several bins, and replacing it with a number representing the corresponding bin.

Binarization: a special case (replaces a variable with either 0 or 1 depending on whether the variable is greater or smaller than a certain threshold).

Discretization in R can be done via the function cut.

Binning is a data transformation also known as discretization. It takes a numeric variable, typically a real value though it may also be an integer, divide its range into several bins, and then replaces that variable With a number representing the corresponding bin. Binarization is a special case of Binning. It replaces a variable with either a 0 or 1, depending on whether the variable is greater or smaller than a certain threshold. Discretization in R can be done using the function cut. There are several reasons why we may want to bin values. One reason may be data reduction. Storing the bin values and processing them may be much faster, than storing and processing the original values and it also takes less space. Specifically, it can improve scalability of data analysis procedures that handle very big data. Another reason is that, it can help capture non linear effects when using linear models. This is a very common technique where a variable is binned, and then different nonlinear transformations are operated on the binned values with all the values being concatenated into a vector of measurement that is then inserted into a linear model. That provides a way to capture nonlinear relationships between the data, using linear modeling tools.

Data Transformations: Binning



Binning Example:
Suppose x represent the tenure of an employee (in years) and ranges from 0 to 50.

Binning would divide the range into (0,10], (11,20], ...,(41, 50] and pick a representative number for each bin (for example middle point)

Let's consider an example, where x represents the tenure of an employee in years and ranges from 0 to 50. In this specific example, we may want to bin the range into the bins 0-10, 11-20, and so on until 41-50, and pick a representative number for each bin. For example, the middle point of the bin. And then replace each value by the representative number of the bin in which the values resides. The notation that you see here, 0 to 10 has a parenthesis on the left and a bracket on the right. It represents all values that are greater than 0 and less than or equal to 10. Similarly, this notation represents all values that are greater than 11 and are less than or equal to 20.

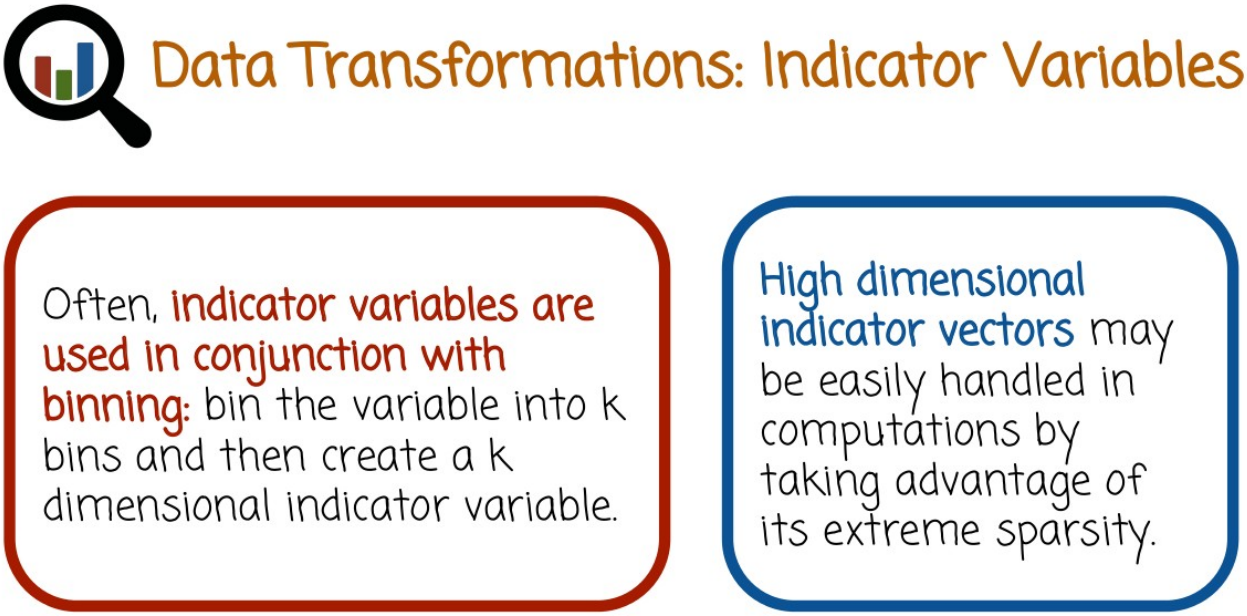
Data Transformations: Indicator Variables

Replace a variable x (numeric, ordinal, or categorical) taking k values with a binary k-dimensional vector v, such that v[i] (or v_i in mathematical notation) is one if and only if x takes on the i-value in its range.

Replace variable by vector that is all zeros, except for one component that equals one.

34. A transformation that is related to beanning is indicator variables (indicator variable 的意思見後面的 quiz 就明白了). Indicator variables replace the variable x that maybe numeric, ordinal or categorical,

taking k values with the binary k dimensional vector v, such that v[i] or vi in mathematical notation, meaning the ith component of the vector v, is one if and only if x takes on the i-value in its range. Otherwise, it is zero. In other words, we take a variable that may take multiple values and we replaced it with a binary vector whose components are all zeros. Except for one component that is one corresponding to the index representing the original value of the variable.



Often indicator variables are used in conjunction with binning. We first bin the variable into k bins and then create a k dimensional indicator variable. Indicator variables may be high dimensional. If the original variable has many different possible values but the high dimensionality of indicator vectors is not necessarily a problem, it can be easily handled both in storage and in computation by taking advantage of its extreme sparsity, these are vectors that are almost all zeros. And so, we can store them efficiently, and process them quickly. Even if their vector is a very high dimensionality. For example, 10,000, 1 million, 100 million, it doesn't matter so much, the dimensionality of the indicator vector in terms of computation as long as it's almost all zeros, we can store and compute with it efficiently.

Data Transformations: Indicator Variables

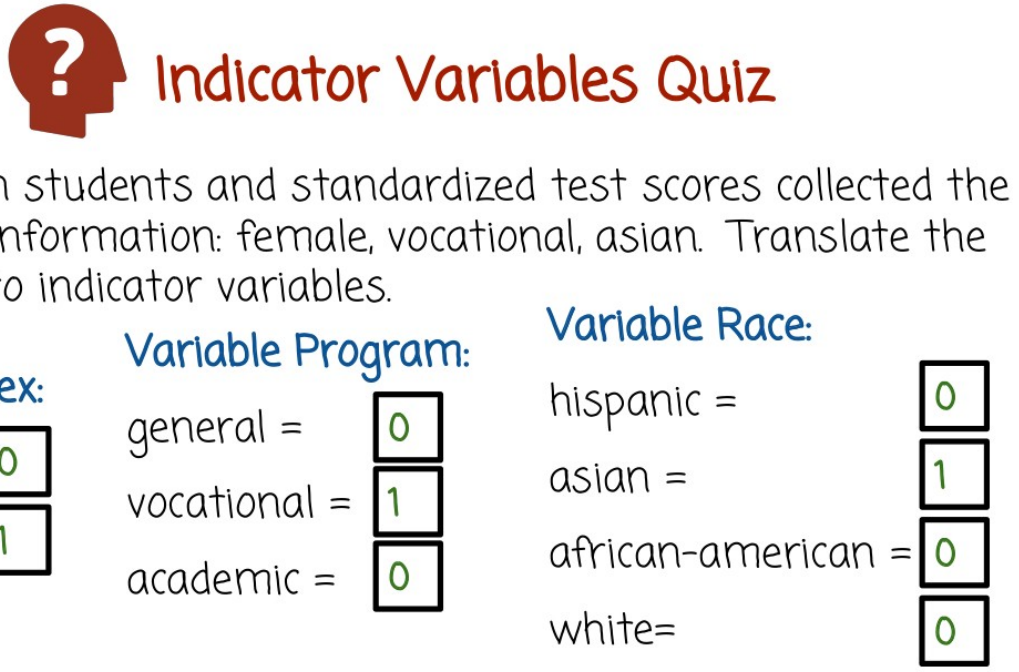
Models for numeric or binary data cannot directly model ordinal or categorical data.

Transform the data using several non-linear transformations bin the transformed data, and create indicator vectors.

It is often **much easier to compute with indicator functions since they are binary**, and thus replacing numeric variables with indicator vectors may improve scalability.

A very important use case for indicator variables is when we want to use a standard statistical model such as linear regression or logistic regression or other similar models, but our data is either ordinal or categorical. In this case, we can convert the variables whether they are ordinal or categorical into an indicator variable. Then we concatenate all of the indicators vectors into a single long vector and we feed that long binary vector of zeros and ones into the standard statistical model like linear regression or logistic regression. Computation would be relatively fast if we take care to exploit disparsity. And the model would be able to find interesting relationship in the data. If we do not use this process, we need to find out a different way to introduce the ordinal or categorical data into standard statistical models that rely on numeric data. To summarize, we can transform the data using several non-linear transformations. For example, multiple power transformations. Then we can bin the transformed data, and create indicator vectors. We can also transform existing ordinal or categorical variables into indicator vectors. We concatenate all the indicator vectors, whether they belong to categorical variables, ordinal variables, or if they belong to numeric variables that were transformed and then binned and then transformed into an indicator vector. We concatenate all of these vectors together, and then we use that vector in the standard statistical model. This practice is very popular as it is scalable computationally, it is effective, and it also is a easy way to handle both categorical and ordinal variables that are normally difficult to handle using standard models. It also can help capture non-linear effects using linear models because we use non-linear transformations earlier on on the numeric variables.

35. A study on students and standardized test scores collected three variables, sex, program, and race. Sex can take two values, male and female, it is a categorical variable. Program can take three values, general, vocational, or academic. It's also a category called variable. Race can take four values, hispanic, asian, african-american, or white. It's also a category called variable. Translate the variables to indicator vectors, or variables.



36. If you recall, an indicator variable or indicator vector is a binary vector that is all zeros except for one component that is one, corresponding to the index that matches its value. In this case of seeing a measurement of female vocational and Asian, we'll take female and convert it into a vector zero and one. Where the second component is one corresponding to measurement being female. The program vector corresponding to the measurement vocational will give a binary vector of three components that are all zeros, except for the middle component being 1, that matches the vocational measurement. The measurement Asian is converted into a binary vector of four components or four dimensions, that are all zeroes representing rates except for one component that is 1, that's the second component corresponding to the Asian measurement. The specific ordering of, in this case, Hispanic corresponding to the first component of the vector, Asian the second, African American the third component and white the fourth component is arbitrary. We could easily permute the ordering or use a different ordering and everything would be perfectly fine, as long as we keep the ordering consistent when we collect, store, and process the data.

Data Manipulations: Shuffling

A common operation in data analysis is to **select a random subset of the rows of a dataframe**, with or without replacement.

sample() accepts a vector of values from which to sample (typically a vector of row indices), the number of samples, whether the sampling is done with or without replacement, and the probability of sampling different values.

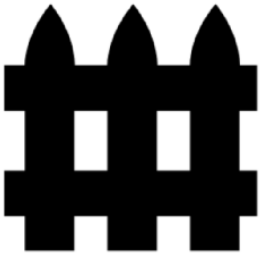
sample(k,k) generates a random permutation of order k.

```
D = array(data = seq(1, 20, length.out = 20), dim = c(4, 5))
D_shuffled = D[sample(4, 4),]
```

37. A common operation in data analysis is to select random subset of the rows, or the columns of a data frame with or without replacement. The R function sample, accepts a vector of values from which to sample. Typically, a vector of row indices or column indices, the number of samples whether the sampling is done with or without replacement and the probability of sampling different values. For example, the R function sample of K comma K generates a random permutation of order K. The reason being is that we sample four values from a vector of four values without replacement being the default of the function sample. This random permutation can be used to shuffle the rows of the data. Here is an example, the first line creates an array of the numbers 1 to 20, with 4 rows and 5 columns. Now suppose we want to shuffle the rows of that array. We first compute the permutation using the code sample 4,4 and then we insert it in square brackets to the array D living entire selection of columns. This would create a new array which we assign to D_shuffled that's equivalent to the old array but where the rows are completely shuffled with random ordering. This is a common operation in data analysis because we often want to take the data, randomly shuffle it and create training set and testing set. Train the data on the training set, evaluate it on the testing set, and perhaps repeat that several times. This process is called cross validation. The process of randomly selecting a training set and a testing set from a given data set can be done by this process here. It's important to first shuffle the data using a random permutation in order to avoid biasing the training data and the testing data, because earlier measurements in the data may be different from later measurements.

Data Manipulations: Partitioning

In some cases, we need to partition the dataset's rows into two or more collection of rows.



Generate a random permutation of k objects (using sample(k,k)), where k is the number of rows in the data, and then divide the permutation vector into two or more parts based on the prescribed sizes, and new dataframes whose rows correspond to the divided permutation vector.

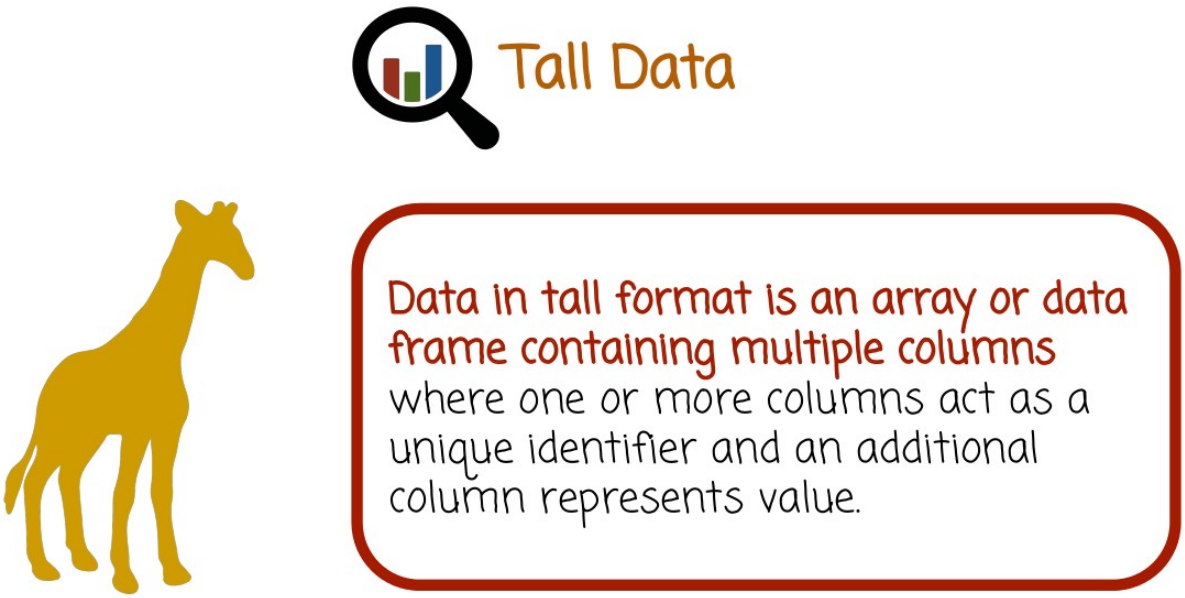
38. Continuing the previous example, let's see how we partition the dataset's rows into two or more collections of rows. For example, one may be used for training and machine learning model, and the other may be used for evaluating that machine learning model. Assuming we want to partition the data set's rows randomly into two collections, we need to first generate a permutation as we described in the previous video. We can do that using the function sample (k,k) where k is the number of rows in the data. And then, divide the permutation vector that the sample (k,k) function returns into two or more parts based on the prescribed sizes and create new data frames who's rows corresponding to the divided permutation vector.

Data Manipulations: Partitioning

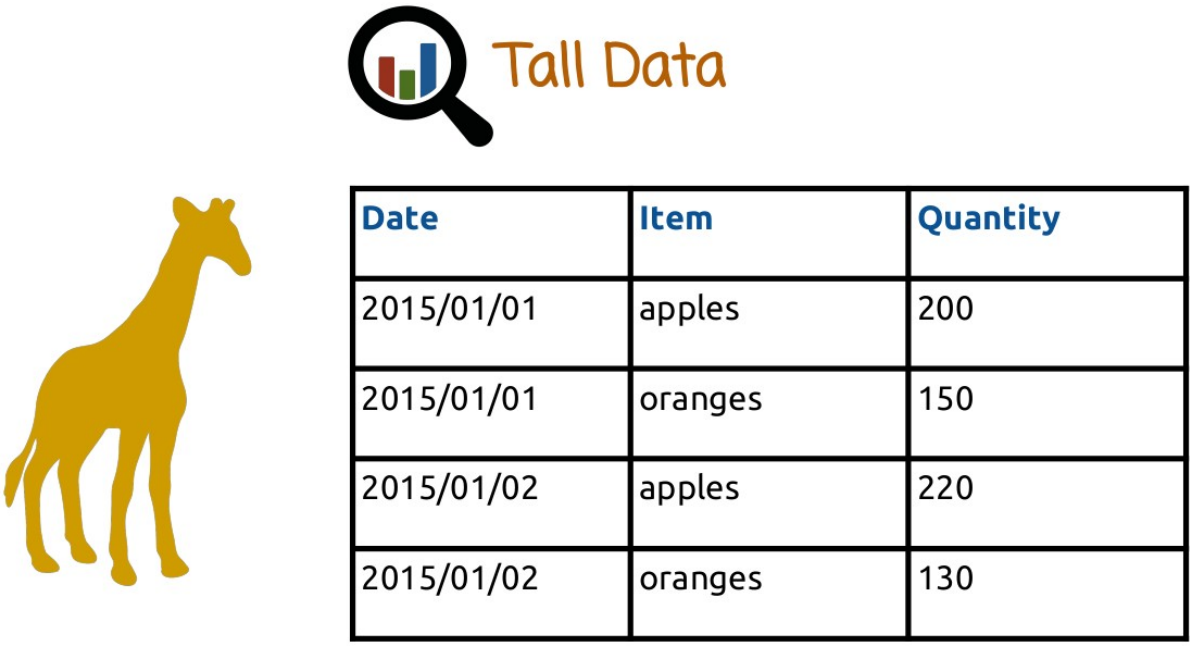
```
D = array(data = seq(1, 20, length.out = 20), dim = c(4, 5))
rand_perm = sample(4,4)
first_set_of_indices = rand_perm[1:floor(4*0.75)]
second_set_of_indices = rand_perm[(floor(4*0.75)+1):4]
D1 = D[first_set_of_indices,]
D2 = D[second_set_of_indices,]
```

In the first line here we create an array of number from 1 to 20 with four rows and five columns. The second line we generate a random permutation of the numbers one, two, three, four. That random permutation will allow us to divide the rows into two groups randomly. Now suppose we want the first group of rows to be 75% of the data set, and the second group of rows to be 25% of the data set. For

example, in the case of training a model, we may want to train the model on 75% of the data, the majority of the data, but hold out 25% of the data for evaluation. Here we create the first set of indices by taking the first 75% of indices in the random permutation vector. That we got from the sample for comma four function. The remaining values in the random permutation vector will go into a second vector of indices that will capture the second group of rose. This nine here we create the first data frame based on the old dataframe but we keep only 75% of the rows, chosen randomly. In this line here, we generate a second dataframe containing the remaining 25% of the rows again, chosen randomly



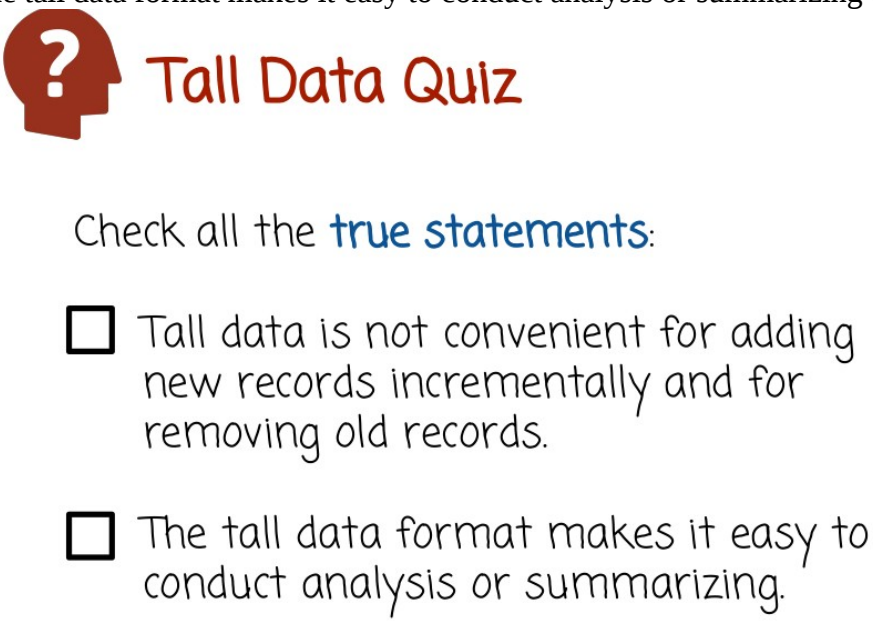
39. Data in tall format is an array or data frame containing multiple columns where one, or more columns act as a unique identifier and an additional column, or several additional columns represent value.



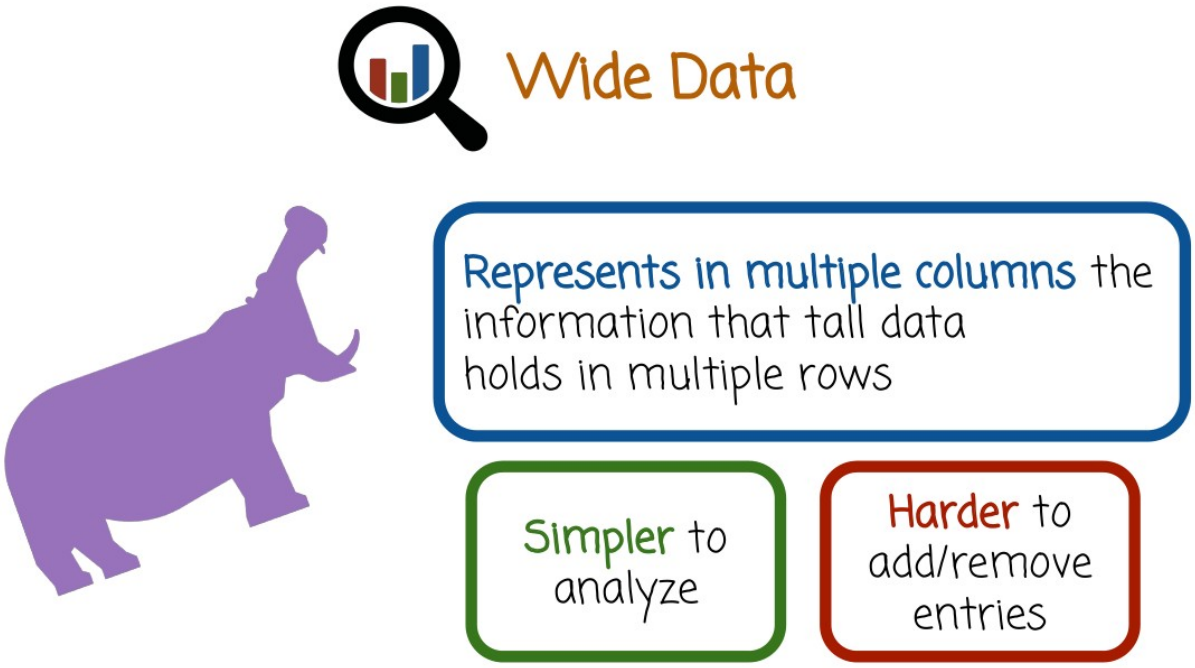
Date	Item	Quantity
2015/01/01	apples	200
2015/01/01	oranges	150
2015/01/02	apples	220
2015/01/02	oranges	130

Here is an example of a dataset representing orders from a grocery store. The first column is the date. Second column is the item and the third column is the quantity that's being ordered. The first two columns, the date and the item represent the unique identifier. Every combination of date and item should appear in only one row in the dataset. The quantity is a measurement describing the combination of date and item. The columns that are the unique identifier can be selected differently. As I mentioned, in this case, which shows the first two columns to be the unique identifier. So no two rows can have the same unique identifier, but that was one specific choice. We can use different choices, depending on our needs.

40. Check all the true statements. Tall data is not convenient for adding new records incrementally, and for removing old records. The tall data format makes it easy to conduct analysis or summarizing data.

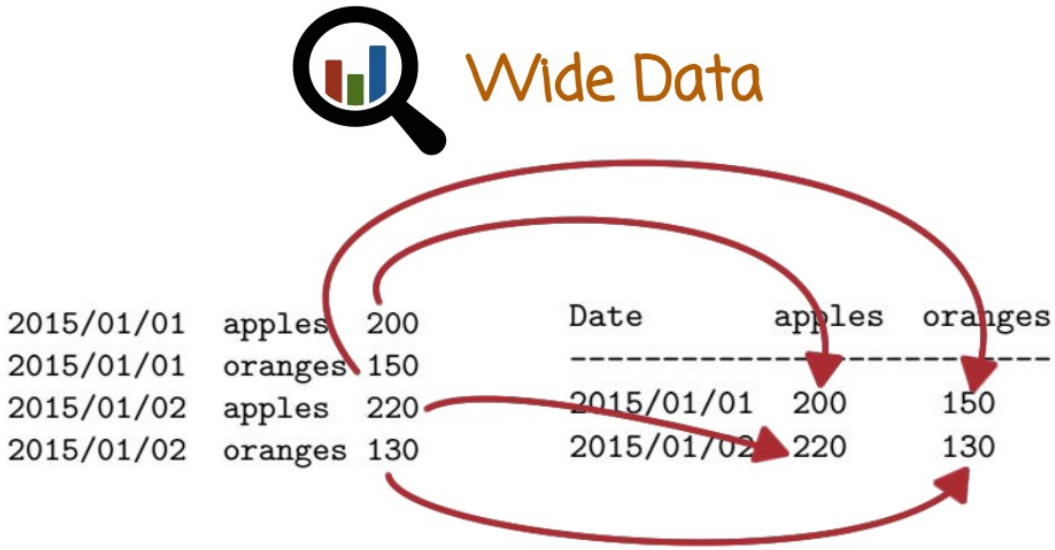


41. Both of these statements are false. Tall data is convenient for adding new records incrementally and for removing old records. It's simply a matter of adding more rows at the end of the table or removing specific rows in the middle or in the beginning. A disadvantage of tall data format is that it's not easy for conducting analysis or summarizing it. For example, if we want to compute average daily sales of the store, it would be relatively hard to do that if the data is in a tall data format.



Wide data 的例子見下圖

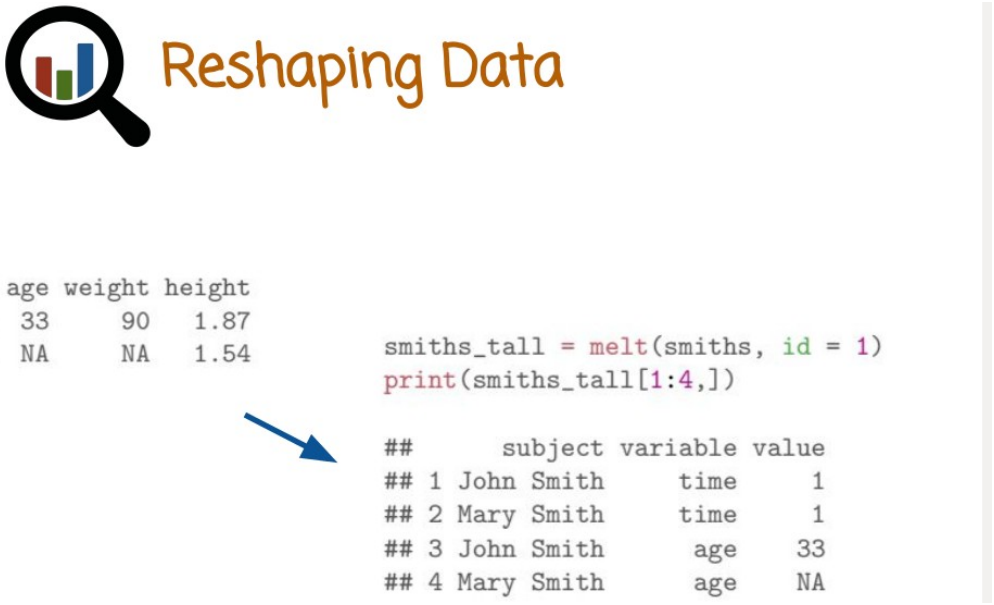
42. Wide data represents in multiple columns the information that tall data hold in multiple rows. It is simpler to analyze, for example if you want to compute the average daily sales we saw earlier, it's not convenient to do that using tall data. You would have to do a complete pass of the data, extract the daily measurements, and then compute the mean. Wide data is considerably easier to analyze. You may not need to do a full pass over the data, you just jump straight to the specific position and read the result. Or do some more elementary calculation in a very specific location in the table. The disadvantage of wide data is that it's harder to add and remove entries. Earlier, we saw that that's the advantage of tall data, for wide data, it's a disadvantage. Interestingly tall data and wide data have different pros and cons. If you need the data set format where you quickly add or remove entries, the better choice is probably tall data. If the data set is more stationary, you don't add or remove that frequently or maybe you're finished adding or removing and you want to focus on data analysis tasks. You want to make sure that data is in a convenient format for that, then you probably want to choose wide format.



- When converting tall data to wide data, we need to specify ID variables that define the row and column structure (date and item in the example above).

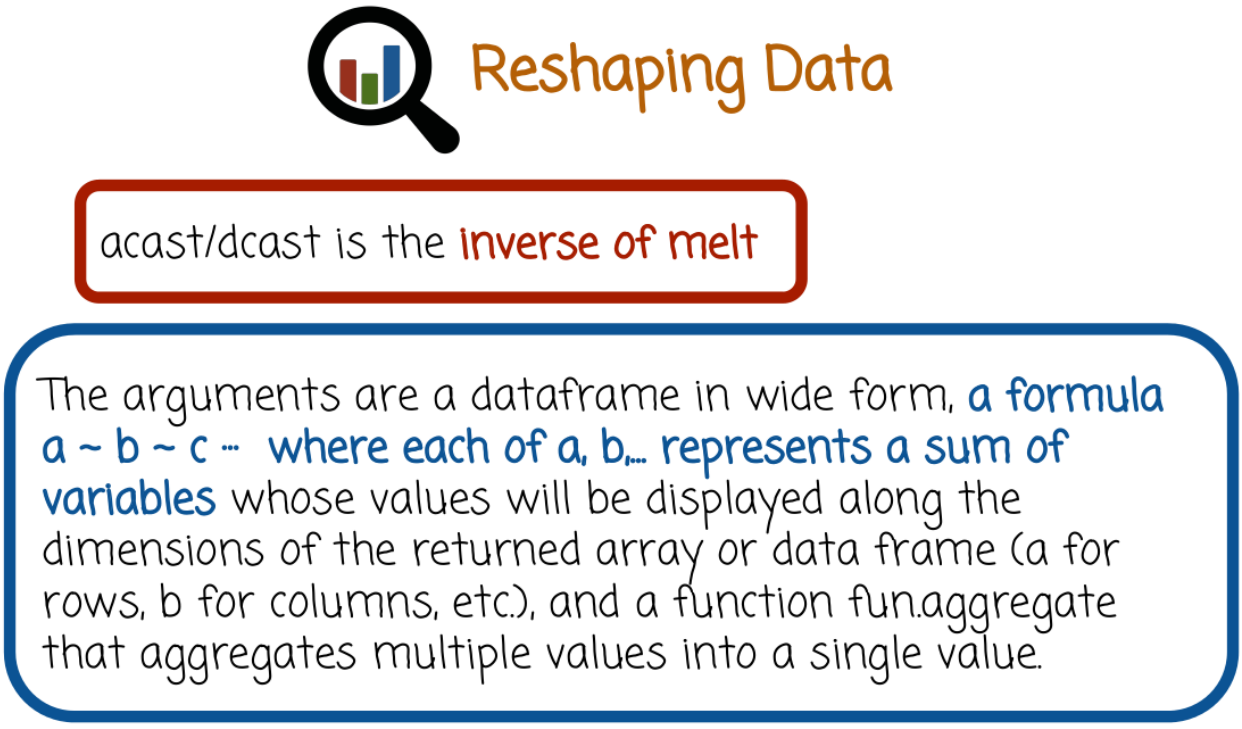
上圖左邊是 tall data, 右邊是 wide data.

Let's see an example. This is the same date as set from the tall data format earlier of orders in the grocery store. On the left you see the tall data format. The first two columns are the unique identifier, the day, and the item. And the third column is the number of items ordered in that specific day. On the right-hand side you'll see the same data in a wide data format, each row in this case corresponds to a date. We have one more column for apples, and one more column for oranges, and a combination of row columns will just hold the number of items, whether it's apples or oranges, that are sold in a specific date. As you see it's easier to add or remove specific orders from the tall data on the left side, but if you want to analyze the date this format is more convenient. For example, if you want to compute the average daily sales in this case, in the wide date format, you just compute the average of the specific row you're interested in. If you want the average order of a specific item, you just take the average of the corresponding column. This is much easier than similar operation on the tall data format. When converting tall date to wide data, we need to specify ID variables that define the row and column structure. In the example above, it's date and time.

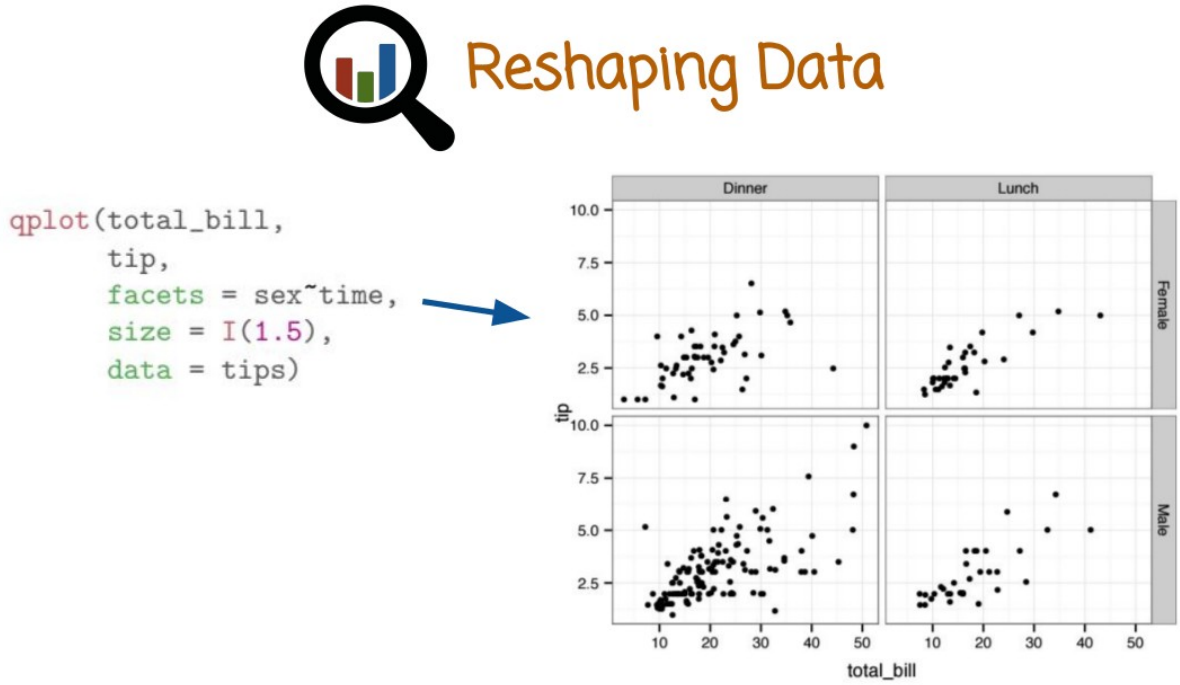


上圖左邊是 wide data, 右邊是 tall data. 由此可知, 對於 wide data, 一個 id 值只能有一行

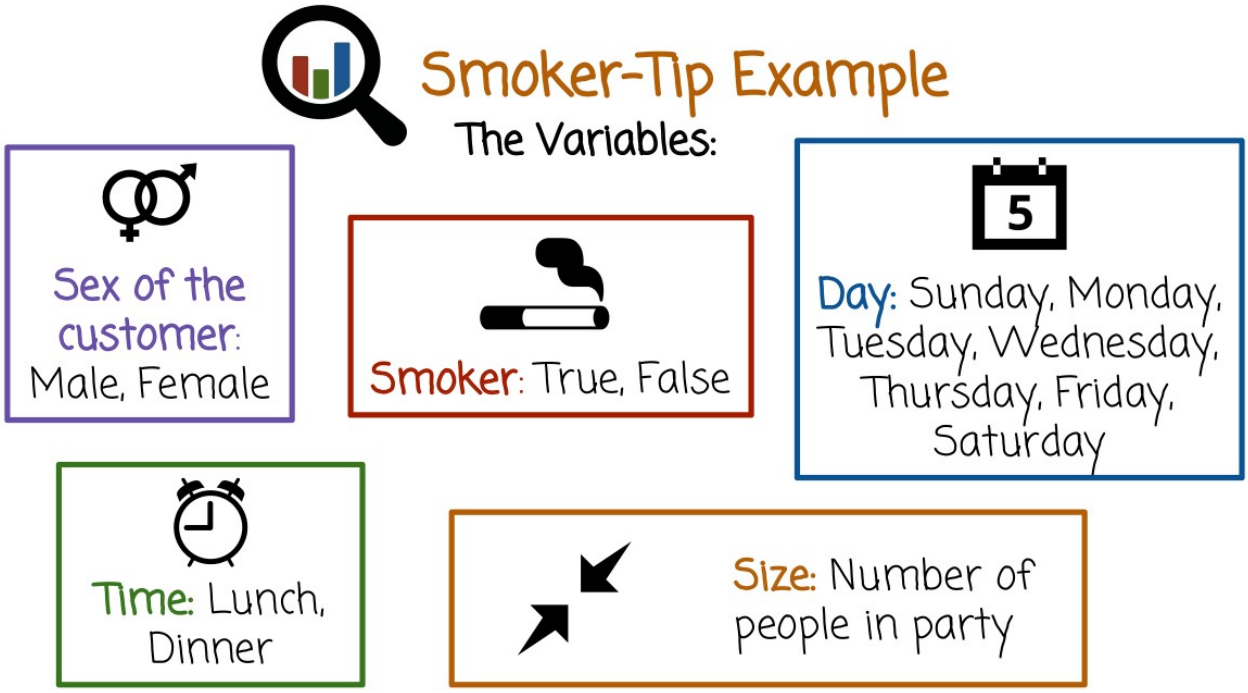
43. The R package reshaped2, converts data between tall and wide formats. Specifically, the melt function accepts the data framing in a wide format and the indices of the columns that act as unique identifiers with the remaining columns acting as measurement or values, and returns a tall version of the data. Here's an example. The Smith's data set is a data set within the reshape2 package. It's a pretty small data set with only two rows and five columns. This line here calls the melt function with the Smith's data set as the first argument and id equals 1. Conveying to the melt function that the first column is to be interpreted as the unique identifier. This line here prints the first four rows of the tall version of that data set.



To convert data from wide format to tall format, we used the function acast or dcast. The difference between them is that acast returns an array and dcast returns a data frame. Otherwise, they're pretty similar. The arguments to the acast or dcast functions are a data framing wide format to be transformed to tall. A formula a tilde, b tilde, c and so on where each of the a and b represents a sum of variables whose values will be displayed along the dimensions of the returned array or data frame (a for rows, b for columns, etc.), and a function that aggregates multiple values into a single value. This is needed in case there are multiple values that are mapped to a specific value or cell in the new array or data frame.



Let's see some examples. Before we move on with some examples, I want to introduce a new data called tips. Tips is a data frame in the reshape2 package describing restaurant visits of customers. The code here grasps the relationship between the total bill that was paid and the tip that was paid by the customer, and uses different facets so that we can concentrate individually on different sex or gender of customers and time of the restaurant visit. So the columns of the array of graphs or facets, the column facets, represents dinner versus lunch. This corresponds to time. And the rows correspond to the sex, or the gender of the person who paid the bill. And the X axis in each one of these graphs is a total bill, and right axis represents the tip amount. Interestingly, we see a positive correlation for each one of these plots, whether it's lunch, dinner, female, or a male. That's to be expected. Tips go up as the total bill goes up. But what's more interesting is to explore the relationship between tips and total bill, for lunch versus dinner, and male versus female. One direction of exploring this data would be with a sequence of graphs. Such as graphs we've seen at the second lesson, graphing data. That would involve drawing multiple graphs, looking at it, redrawing additional graphs and so on. Next video I want to emphasize that this can also be done using table exploration, using the reshape2 package very quickly. This is an alternative way of analyzing the data that is nongraphical, it just uses tables and numbers. But we will see how the mechanism of r using reshape functions, can allow us to use that mechanism to explore the data very efficiently, very quickly, similarly to how we would use graphs.



44. Let's return to the tips data set. The variables are the sex of the customer, whether the customer is a smoker, what day of the week the customer visited the restaurant, what time, is it lunch or dinner, and the size of the party.


```
tipsm = melt(tips, id = c("sex","smoker","day","time","size"))
dcast(tipsm,  # Mean of measurement variables broken by sex
      sex~variable,
      fun.aggregate = mean)
```

```
##      sex total_bill   tip
## 1 Female      18.06 2.833
## 2   Male      20.74 3.090
```

The first thing we want to do is we want to transform the data into a table that will allow us to investigate what we want to investigate. Remember, we're not going to draw graphs. Rather, we want to create many small tables that are going to show different insights from the data. In this case what we want to get is the table at the bottom where the rows are the sex, female or male, and the row are total bill average and tip average. So here's how we do it. We first melt the tips data set using unique identifier of sex, smoker, day, time, and size. And then we use dcast function to convert it back into a format that we can display as table, but where we aggregate multiple cells using the mean function or average. And we convey to the dcast function that we want the table to have sex as rows and the

variable as the columns. What we can draw from this table is the information that overall males pay on average higher total bills, as well as higher tips. Interestingly, computing this table and displaying it took only two lines, rather than a complicated routine where we have to go over the dataset with the for loop, process the dataset, and extract the numbers that we want.

Smoker-Tip Example

```
# Number of occurrences for measurement variables broken by sex
dcast(tipsm,
      sex~variable,
      fun.aggregate = length)
```

```
##      sex total_bill tip
## 1 Female         87  87
## 2   Male        157 157
```

This is similar case to the previous slide, but instead of computing the average total bill and tip, what I want to capture here is how many measurements we have for each combination of female, male, total bill and tip. So we see there are 87 cases of female customers paying and 157 cases of male customers paying. The way to accomplish that is similar to the previous example, but we use a length function aggregator rather than a mean.

Smoker-Tip Example

```
# Average total bill and tip for different times
dcast(tipsm,
      time~variable,
      fun.aggregate = mean)
```

```
##     time total_bill   tip
## 1 Dinner      20.80 3.103
## 2  Lunch      17.17 2.728
```

In this case, we generate a table that shows in rows instead of sex, rather the time, whether it's dinner or lunch. The difference is that we use the time variable to the left of the tilde instead of sex. What we see from this table is that customers generally pay higher total bill for dinner than for lunch, and also a higher total tip for dinner than for lunch on average.


```
# Similar to above with breakdown for sex and time:
dcast(tipsm,
      sex+time~variable,
      fun.aggregate = length)
```

```
##      sex   time total_bill tip
## 1 Female Dinner         52  52
## 2 Female  Lunch         35  35
## 3   Male Dinner        124 124
## 4   Male  Lunch         33  33
```

Here's a slightly more complicated example where we want to break down total bill and tip by combination of both sex and time. In the first example here, we're just going to look at the number of cases where we have female paying for dinner and female paying for lunch, male paying for dinner and male paying for lunch. We get that by using the length function aggregator.

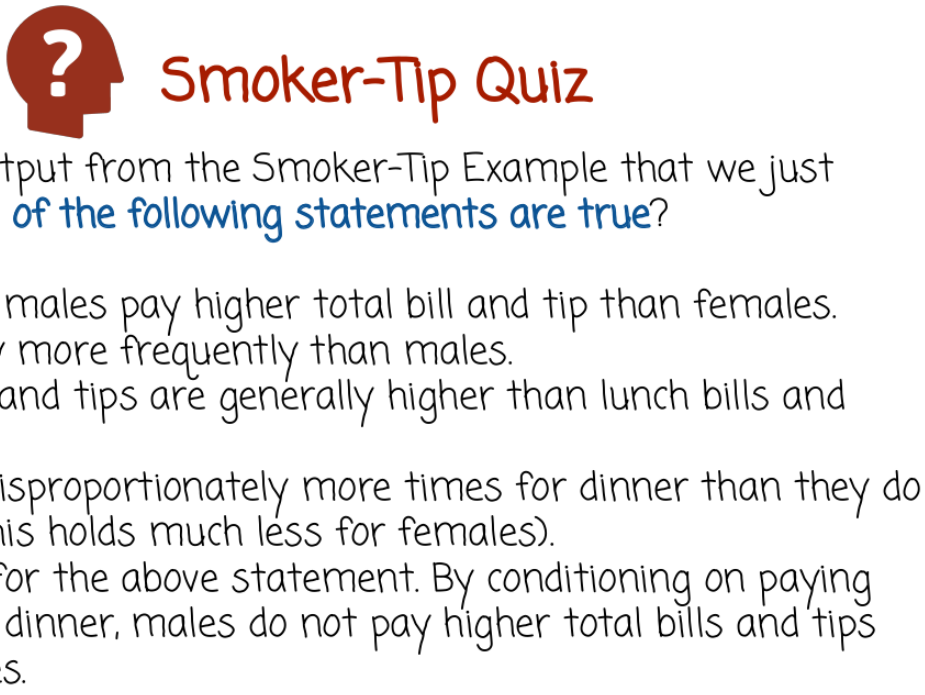

```
# Similar to above, but with mean and added margins
dcast(tipsm,
      sex+time~variable,
      fun.aggregate = mean,
      margins = TRUE)
```

```
##      sex   time total_bill   tip  (all)
## 1 Female Dinner      19.21 3.002 11.108
## 2 Female  Lunch      16.34 2.583  9.461
## 3 Female  (all)      18.06 2.833 10.445
## 4   Male Dinner      21.46 3.145 12.303
## 5   Male  Lunch      18.05 2.882 10.465
## 6   Male  (all)      20.74 3.090 11.917
## 7  (all)  (all)      19.79 2.998 11.392
```

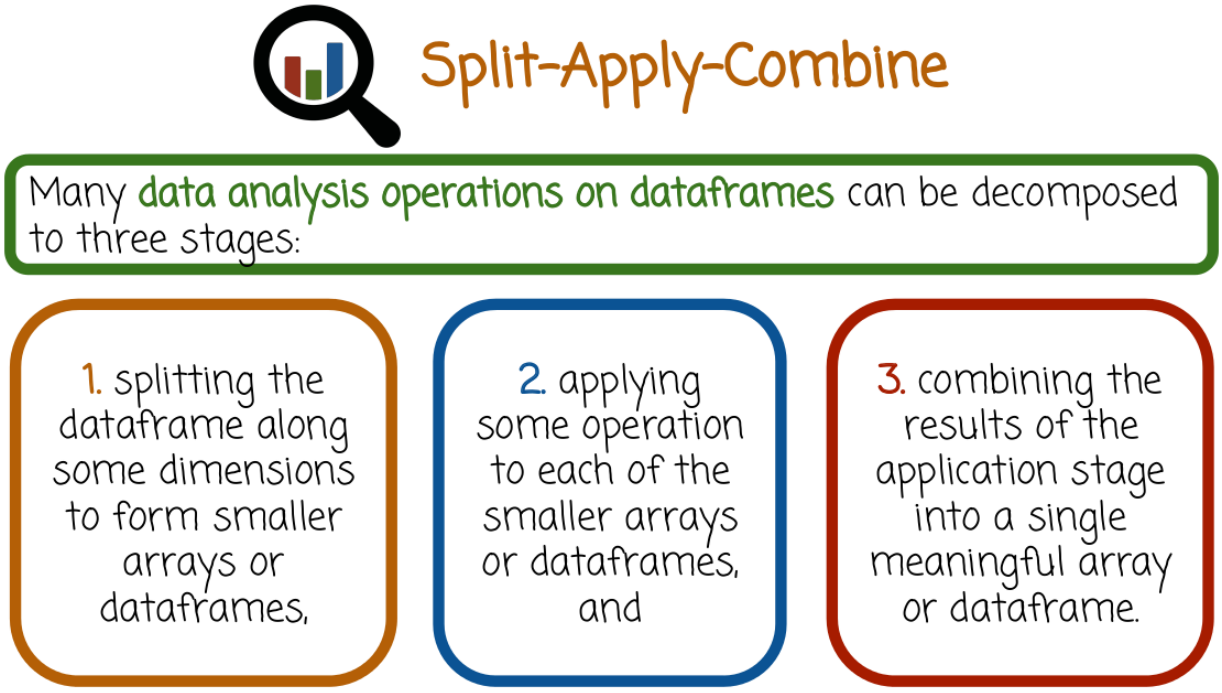
Now we move on to computing the mean here in function aggregator rather than computing the length. We also pass to the function margins = TRUE argument which includes an additional row and column corresponding to the overall mean of the entire rows and columns. Now we can study this table to extract information that we're interested in. For example, is the gap between male and female total bill or tip different from lunch to dinner, does it get bigger, does it get smaller, etc. That kind of

investigation again can be also done with a graph. But this alternative way of doing it is sometimes quicker and sometimes has an advantage because you can just look at directly the numbers rather than looking at the graphic representations.

45. Based on the output from the Smoker-Tip Example that we just discussed, which of the following statements are true? On average, males pay higher total bill and tip than females. Females pay more frequently than males. Dinner bills and tips are generally higher than lunch bills and tips. Males pay disproportionately more times for dinner than they do for lunch, this holds much less for females. Accounting for the above statement by conditioning on paying for lunch or dinner, males do not pay higher total bills and tips than females. If you need to explore the data set to answer the quiz, you can just download the reshaped tool package, bring it into scope using the library command, and start exploring it using the reshaped two tools.



46. The first statement is correct, on average males do pay a higher total bill and tip than females. At least for the day time the tips data frame. It is not correct that females pay more frequently than males rather the opposite is true. Dinner bills and tips are generally higher than lunch bills and tips, that is correct. The fourth statement is also correct. Males pay disproportionately more times for dinner than they do for lunch. Females also do that, but to a lesser extent. The last statement is incorrect. Conditioning on paying for lunch or dinner, males do pay higher total bills and tips than females.



47. May data analysis operations on data frames can be decomposed to three stages. Splitting the data frame along some dimensions to form smaller arrays or data frames. The second stage, applying some operation to each of the smaller arrays or dataframes. The third stage, combining the results of the application stage into a single meaningful array or dataframe. Performing these three stages again and again on the same data set or on different data sets can cause problems. It is very useful to automate these three stages so that the programmer will not introduce errors and will be able to save their time to the data analysis task rather to implementing these three steps repeatedly. We're going to next look at the package in R called plyr, that automates these three stages and allows the programmer to focus on the logic of how to split, how to apply an operation and how to combine in a simple way, rather than to implement every single stage from the beginning to the end repeatedly.

output input	array	dataframe	list	discarded
array	aaply	adply	alply	a_ply
dataframe	daply	ddply	dlply	d_ply
list	laply	ldply	llply	l_ply

- Arguments: data, dimensions/columns used to to split the data, function to execute in the apply stage.

aaply 就是一個 function

The package plyr has many different functions. This function typically ends with P-L-Y where the first two letter correspond to what is the input data and what is the output data. For example, if the input data is array and the output data is an array, the corresponding function is aaply. If the input data is a dataframe, the output data is a list, the corresponding function is dlply. The arguments to these functions are the data the dimensions or columns used to split the data, the function to execute in the apply state.


```
library(plyr)
names(baseball)
```

```
##  [1] "id"    "year"  "stint" "team"  "lg"    "g"     "ab"
##  [9] "h"     "X2b"   "X3b"   "hr"    "rbi"   "sb"    "cs"
## [17] "so"    "ibb"   "hbp"   "sh"    "sf"    "gidp"
```

```
# count number of players recorded for each year
bbPerYear = ddply(baseball, "year", "nrow")
head(bbPerYear)
```

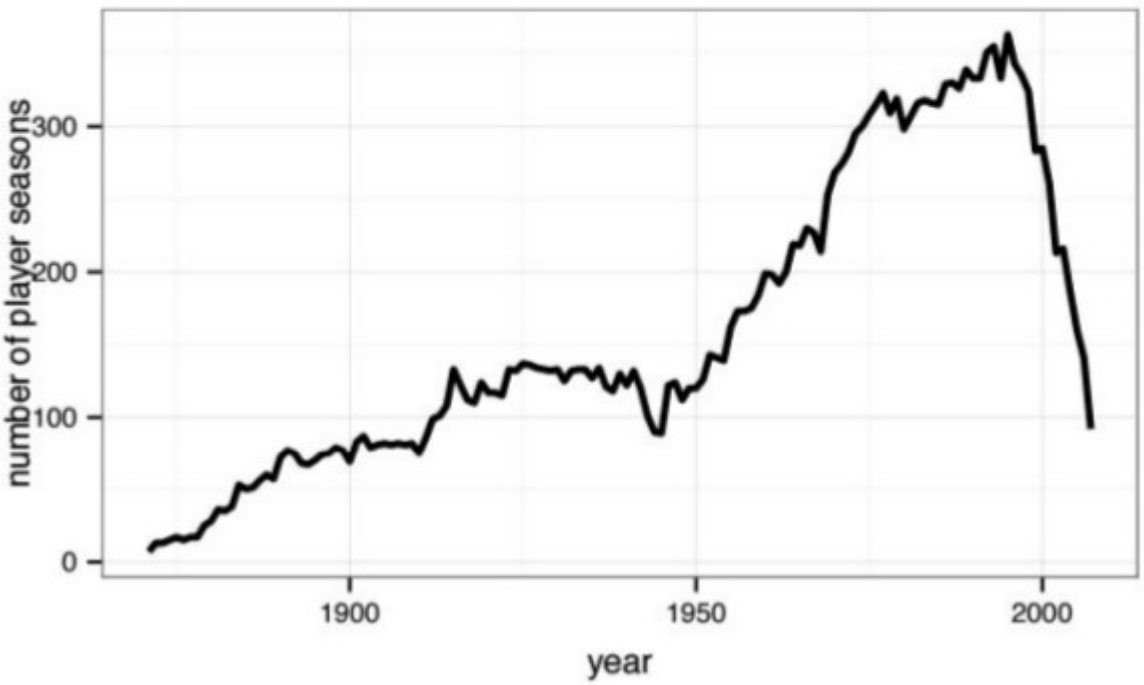
```
##   year nrow
## 1 1871    7
## 2 1872   13
## 3 1873   13
## 4 1874   15
## 5 1875   17
## 6 1876   15
```

We're going to see next some example of applying Plyr to some baseball data, specifically we're going to look at the baseball data frame in the Plyr package. We first have to call the library function to bring Plyr into scope. And let's look next at the different columns of the baseball data frame. The columns here correspond to measurements. For specific player in a specific team in a specific year having a specific RBI in other kind of baseball measurements. As our first application of Plyr, let's try to count the number of players recorded for each year in the baseball package. Baseball package may have the same player in multiple rows. What we want to do Is go over the years and count how many players are recorded for each year. We call the function ddply because the input is going to be a data frame baseball and we want the output to be also in a data frame shape. We're going to split across the year

dimension, and we're going to apply the function nrow which counts the number of rows. Then we're going to aggregate the results back into a data frame. So we split by year we call the function nrow and that counts the number of rows for each year which correspond to the number of players. And then we display the first several lines of the resulting data frame. We that in the year 1871, we have only seven players recorded. In 1872, we have 13, and so on. As you can see, this operation here, is very simple, we just use one line to create a meaningful statistic that otherwise we would need to program a loop to iterate through the data set or vectorized code that will not be as simple and we may introduce errors.

Baseball Example

```
qplot(x = year, y = nrow,
      data = bbPerYear, geom = "line",
      ylab="number of player seasons")
```



48. Let's graph the data from the previous example. This is for each year the number of players recorded in that season. We see interestingly that the data frame has much more players recorded in the years right before the year 2000. Much earlier years have fewer number of players recorded and that number increases gradually, although it decreases extremely steeply around the year 2000. This information can be very useful if we want to model the dataset and understand performances of players.

We want to understand the number of measurements for each year because that is going to indicate the accuracy of the model in different years. If we didn't look at this graph, we would not notice that the model would be much more accurate in some years than in other years.

Baseball Example

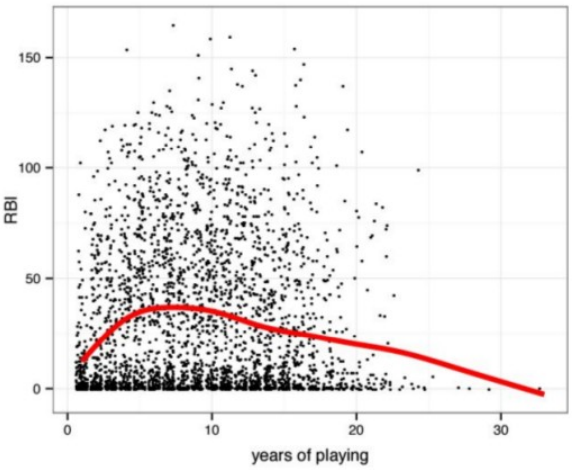
```
# compute mean rbi (batting attempt resulting in runs)
# for all years. Summarize is the apply function, which
# takes as argument a function that computes the rbi mean
bbMod=ddply(baseball, "year", summarise,
            mean.rbi = mean(rbi, na.rm = TRUE))
qplot(x = year, y = mean.rbi, data = bbMod,
      geom = "line", ylab = "mean RBI")
```

In this example, we want to compute the mean RBI, which is a batting attempt resulting in runs, for all years. We're going to call the ddply package with the baseball dataframe. We want to split the data by the year column, or dimension. The apply function is summarized, which takes as argument a function that computes the RBI mean. Notice here we remove missing values by passing to the function the argument na.rm = TRUE, remove missing values. After we computed that dataframe we want to graph it and we want to see how the mean RBI, which is some measure of performance of baseball players, changes as a function of the year. We see here the mean RBI is particularly high in the years around 1900, very early years, and then it goes down. As before, we see very interesting characteristic or pattern in the data that is exposed using very simple code that is relatively simple and easy to get right,

once you get used to it, without having to code loops and explicit code to compute it.

Baseball Example

```
# add a column career.year which measures the number of years
# passed since each player started batting
bbMod2 = ddply(baseball,
               "id",
               transform,
               career.year = year - min(year) + 1)
# sample a random subset 3000 rows to avoid over-plotting
bbSubset = bbMod2[sample(dim(bbMod2)[1], 3000),]
qplot(career.year,
      rbi, data = bbSubset,
      size = I(0.8),
      geom = "jitter",
      ylab = "RBI",
      xlab = "years of playing") +
  geom_smooth(color = "red", se = F, size = 1.5)
```



上圖左邊的代碼:

```
# add a column career.year which measures the number of years
# passed since each player started batting
bbMod2 = ddply(baseball,
               "id",
               transform,
               career.year = year - min(year) + 1)
# sample a random subset 3000 rows to avoid over-plotting
bbSubset = bbMod2[sample(dim(bbMod2)[1], 3000),]
qplot(career.year,
      rbi, data = bbSubset,
      size = I(0.8),
      geom = "jitter",
      ylab = "RBI",
      xlab = "years of playing") +
  geom_smooth(color = "red", se = F, size = 1.5)
```

In this case, we want to create the graph here, which is a scatterplot showing the RBI on the y-axis of different players. And the x-axis is the years of playing or tenure or amount of experience of the player. How many years has the player been playing before? That's going to show us the performance of baseball players and how they change based on the experience or number of years the player has played before. I'm going to start here by adding a column, career.year, which measures the number of years passed since each player started batting. We're going to then sample a random subset of 3,000 rows to avoid getting a scatterplot with too many points that is both time consuming to display and is going to

clutter the graph. Finally, the qplot command at the end displays the scatterplot and overlays with it a smoothing line, shown here by the bold red line, that averages using local averaging, which we've seen in the graphics module, to describe the overall trend. Because sometimes it's hard to see from the scatterplot points where are more points, at the top or at the bottom? Obviously we see more points at the bottom, but it's hard to understand precisely the quantitative trend here. Is the trend going up or down or flat? Well, having graphed that data, we see now here that the RBI of players as a function of how many years they played first goes up until we reach about five or six years of experience. And then it starts a long gradual decay. This is the average trend. Some players peak later and some players peak earlier. But this curve helps us understand the overall trend and how it applies generally. We can also get the variation by looking at the different points here and understanding that the number of players with very low RBI here close to the 0 y-axis is much higher than the players with the high RBI. Which is indicated here by relatively sparse distribution of black points.

Ozone Example

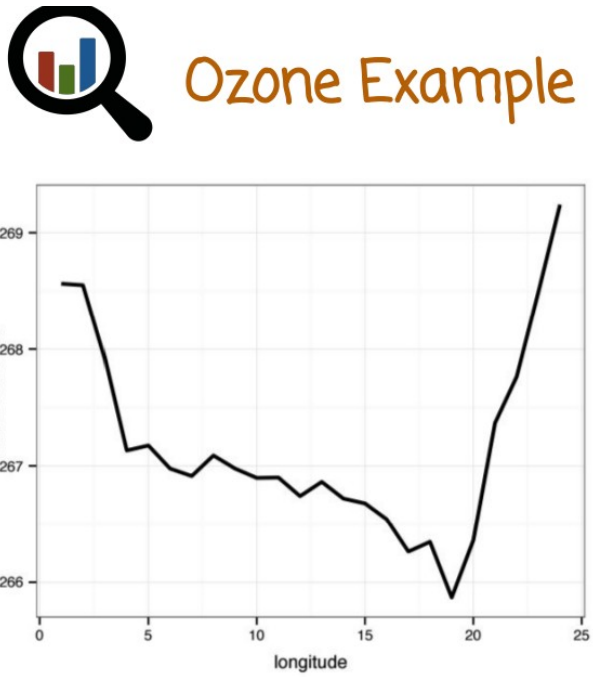
- The ozone dataset contains a **3-dimensional array of ozone measurements** varying by latitude, longitude, and time.

```
library(plyr)
latitude.mean = aaply(ozone, 1, mean)
longitude.mean = aaply(ozone, 2, mean)
time.mean = aaply(ozone, 3, mean)
longitude = seq(along = longitude.mean)
qplot(x = longitude,
      y = longitude.mean,
      ylab = "mean ozone level",
      geom="line")
```

ozone: 臭氧

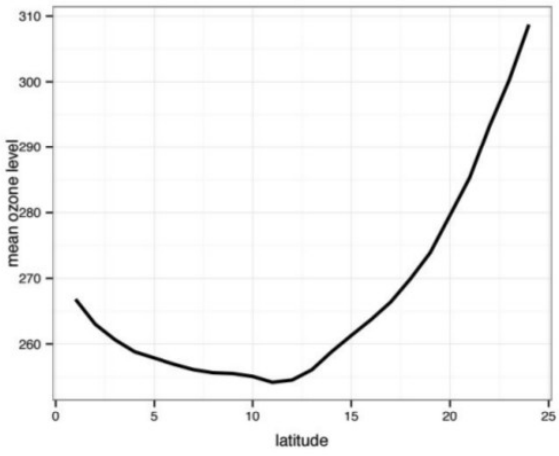
49. The next example will explore the ozone dataset, which contains a three dimensional array of ozone measurements varying by latitude, longitude and time. The ozone data set is inside the plyr package. So, we first need to call the plyr package into scope using the function library. This line here, we want to compute the mean of the latitude of the ozone. So, we call the aaply function. The input data is an array. The output data should be an array. The data is ozone. We want to split by the first column, which corresponds to latitude. We want to execute the function mean to aggregate. And the second case, we want to do the same thing, but for the longitude or the second dimension. In the third case, we want to do the same thing before the time, which is the third dimension. Next, we going to graph these results that we got from the aaply functions. But first, we want to compute a sequence of numbers that we call longitude that will be used to express the x-axis. We're going to use the function seq to create a sequence of values that's going to run along the longtitude.mean vector. The qplot is going to display

the longitude and the longitude.mean using a line plot.



Let's see the graph that we get, as a result of this code. We see here that the mean ozone layer where the mean is computed over all other variables latitude and time as a function of longitude goes down to a pretty significant minimum right here, and then goes steeply up.


```
latitude = seq(along = latitude.mean)
qplot(x = latitude,
      y = latitude.mean,
      ylab = "mean ozone level",
      geom = "line")
```

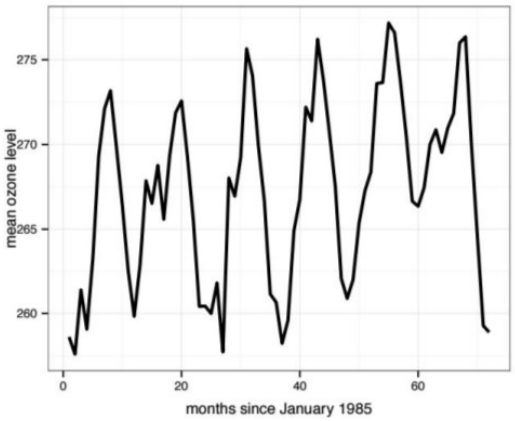


Let's do the same thing for latitude and see what kind of curve we get, and how the mean ozone changes as a function of the latitude. In this case, as before, the mean is computed over all other dimensions that we did not specify. Meaning, the mean of the ozone at that latitude where the mean is computed over all longitudes and all times. We see here a much smoother curve. In contrast to the

previous curve, there is indeed, again, a local minimum, but the qualitative shape of the curve is quite different from the behavior of the ozone as a function of the longitude.

Ozone Example

```
months = seq(along = time.mean)
qplot(x = months,
      y = time.mean,
      geom = "line",
      ylab = "mean ozone level",
      xlab = "months since January 1985")
```

Finally, we want to plot the mean ozone as a function of time. Remember, we already computed the mean when we varied all of the variables, except for time. In this case, longitude and latitude. That would be in some sense, the overall planetary ozone averaged over longitude and latitude as a function of time. The function here shows the mean ozone layer, mean over both longitude and latitude as a function of months since January 1985 or time. We see a very clear period here, periodicity corresponding to the seasons of the year. The ozone apparently changes very strongly with the seasonality of the year. Notice that this graph only shows a small number of years. Around 1985, it's hard to detect from this graph the long-term trend. For example, overall, the ozone has been trending down, but this graph shows a slight upward trend perhaps. But this is because we're looking at the very small measurement of time to detect actual trends in the ozone over time, we need to zoom out and look at a much bigger time frame. We can, however, from this graph observes the periodicity in the ozone as a function of the season of the year.

50. Based on the outputs from the ozone example that we just discussed, which of the following statements are true?

Ozone Quiz

Based on the outputs from the Ozone Example that we just discussed, **which of the following statements are true**?

- ☐ Ozone has a clear minimum mean ozone level at longitude 12 and latitude 19
- ☐ Ozone level has an interesting temporal periodicity superimposed with a slight decreasing trend.
- ☑ The periodicity coincides with the annual season cycle (each period is 12 months)
- ☑ The functions in the plyr package are very general and simplify the coding of many data analysis tasks.

51. The first statement is only partially true. It does have a clear minimum mean ozone level, but it does not appear at the longitude and the latitude that are described here. It appears at different values of longitude and latitude. The second statement is also not true. At least we cannot determine it from the graph that we see, ozone level has an interesting temporal periodicity superimposed with a slight decreasing trend. The graph that we saw does not show a decreasing trend, although it does show a temporal periodicity. If you look at a longer time frame, you will see a clear decreasing trend. But that's not available in the graph that we looked at. The third statement is true. The periodicity coincides with the annual season cycle. We can see that by looking at the graph of average ozone over time and measuring the length of the period and noticing that the length of the period is 12 months. The last statement, the functions in the plyr package are very general and simplify the coding of many data analysis tasks, is correct.

Preprocessing Data
Lesson Summary

- It is important to know how to handle missing data and outliers
- Transforming the data can reveal insights and improve modeling
- Standard data manipulation techniques can be automated by tools in the R language

52. In this lesson we learned what to do when we have missing values in our data and how to handle outliers. We also saw how transforming the data can reveal insights that were previously hard to recognize in how to process data using the R packages reshape and plyr.