

Unit 3: Unsupervised Learning (Clustering, Inference & Bootstrapping)

Clustering (Ch. 9)

- ☐ Explain the K-means clustering algorithm.
- ☐ Interpret the output of a K-means analysis.
- ☐ Differentiate between clustering and classification.
- ☐ Identify when it is necessary to scale variables before clustering, and do this using R.
- ☐ Perform K-means clustering in R using kmeans.
- ☐ Use the elbow method to choose the number of clusters for K-means.
- ☐ Visualize the output of K-means clustering in R using colored scatter plots.
- ☐ Describe the advantages, limitations and assumptions of the K-means clustering algorithm.

Supervised Learning: We are given data that are labeled with a predictive target, and we will make predictions for future data

- classical and regression.
- There is a response variable (a category label or value), and we have past data to help predict the future.

Unsupervised Learning: We are given unlabelled data, and the task is to find structure or patterns of data

- wishy wash and less concrete, the job is to find some structure of pattern
- Label-less

Clustering: separate data into the group by similarity

- Use or exploratory analysis, developing new questions and subgrouping to improve predictive models
- No target that you are trying to predict
- Requires no annotation or input data (can't label everything)
- Not really to evaluate the quality of clustering
- Uses **Euclidean distance**

Examples:

- Movies, use genre to find out what they are interested in
- Amazon, you know that someone buys a grill. Online sellers are not categorizing; you can cluster the grills to avoid advertising it.

K-Means:

*****only quantitative data should be used with this algorithm*****

- a procedure that groups data into K clusters. It starts with an initial clustering of the data, and then iteratively improves it by making adjustments to the assignment of data to clusters until it cannot improve any further.

Searching into K clusters.

1. Specify the number of clusters, $K=3$ and each one as a centroid (center)
2. The K centroid gets randomly placed in your space. We need to set a seed for reproducibility
3. Each observation gets temporarily assigned to its closest centroid (by Euclidean distance) All data points have one color (group)
4. To improve, the centroid could be moved to be in the middle of the centroid. Move centroid to be in the center of the cluster
5. Points are now closer to different centroids. You need to modify the centroid again. You go back and forth between these 2 steps.
6. Continue when nothing is moving or being reassigned anymore. Iteration is finished and alg is done.

K-Means clustering summary

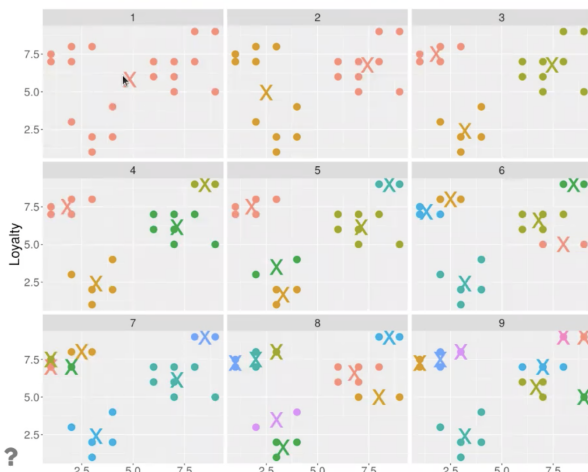
1. **Pick K:** number of clusters to group the observations
2. **Initialize:** assign data to K clusters randomly
3. **Iterate:**
 - A. **Center Update:** calculate average coordinates of each cluster
 - B. **Label Update:** re-assign the data to the closest cluster center
 - C. If no labels changed, terminate.

How do we pick K

- We can determine based on visualization
- What you see as cluster can be depend on how you look at it (bias)
- Look at the points in reference to other data points.

Choosing K

- K too small misses the clusters entirely
- K too large "subdivides" the clusters
- K = 3 looks just right; captures clusters, does not subdivide



How do we measure the quality of the k Clusters:

- **within-cluster sum-of-squared-distances (WSSD)** and tightly packed around centroid
 1. we find the cluster centers by computing the mean of each variable over data points in the cluster
 2. add up the squared distance between each point in the cluster and the cluster center. We use the straight-line / Euclidean distance formula
 3. we sum them together to get the total WSSD, adding up all the squared distances for the 18 observations

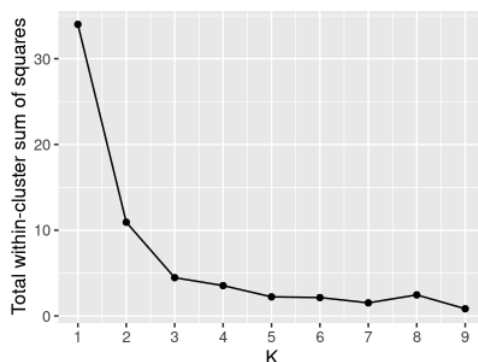
*The larger the value of S^2 , the more spread out the cluster is, since large S^2 means that points are far from the cluster center. Note, however, that “large” is relative to both the scale of the variables for clustering and the number of points in the cluster

if K is chosen too small, then multiple clusters get grouped together; if K is too large, then clusters get subdivided

- If we set K less than 3, then the clustering merges separate groups of data; this causes a large total WSSD, since the cluster center is not close to any of the data in the cluster
- Picking the best value of K is the elbow
- We pick where the big decrease happens. Where the flattening happens is the elbow

Random initialization :

- Kmeans can get stuck with an unlucky random initialization
- K means an iterative algorithm. Depending on where you start, you can get different centroids.
- Use nstart tells you the mount of times you should start the algorithm
- How do we pick the right cluster at the end of the 10 initialization.



K-means can get “stuck” in a bad solution. Unfortunately, for K = 8 we had an unlucky initialization and found a bad clustering

Pros and cons

Pro: easy to implement and interpret and computationally more efficient

Cons: need to specify K, depends on initialization and is sensitive to scale of features.

Q1: Predict the score a student will get on the MCAT exam based on their past grade, the school they attended etc.

- We have a target “predict” and have predictor variables to do that. This would be a supervised learning problem, this would be a regression problem because it's numerical.

Q2: Determine if there are pattern in the past grades, the school they attend, ad the self-reported study habits of students that take the MCAT, LSAT, and DAT exams

- This could be a clustering problem because we don't know what we are trying to figure out, we are trying to find a pattern based on the data we have. We aren't trying to predict anything
- This is an unsupervised learning problem but this is not a clustering problem, the end result should be grouping and not general patterns. Clustering is specific, you are separating it into groups. This is an exploratory problem

Q3: we want to separate photos into groups based on the labeled location the photo was taken

- It depends on what we mean by labeled location. If there is lat and long, then it would be clustering. If he labels it a categorization, urban, rural etc. You know the label, and therefore it's a wrangles problem.

Q4: separate into 5 distinct piles based on photo images

- We don't have a target variable, we group them into 5 distinct files. Therefore it's a clustering problem. No labels and is an unsupervised problem.

Data pre-processing for K-means

- the scale of each of the variables in the data will influence which cluster data points end up being assigned. Variables with a large scale will have a much larger effect on deciding cluster assignment than variables with a small scale
- We need to standardize

1.	Read data	<pre>not_standardized_data <- read_csv("data/penguins_not_standardized.csv") not_standardized_data</pre>
2.	Scale	<pre>standardized_data <- not_standardized_data > mutate(across(everything(), scale)) standardized_data</pre>
3.	K-means clustering in R	<pre>penguin_clust <- kmeans(standardized_data, centers = 3) penguin_clust</pre>

4.	returns a data frame with the data and the cluster assignments for each point:	<pre>library(broom) clustered_data <- augment(penguin_clust, standardized_data) clustered_data</pre>
5.	Visualize cluster	<pre>cluster_plot <- ggplot(clustered_data, aes(x = flipper_length_mm, y = bill_length_mm, color = .cluster), size = 2) + geom_point() + labs(x = "Flipper Length (standardized)", y = "Bill Length (standardized)", color = "Cluster") + scale_color_manual(values = c("dodgerblue3", "darkorange3", "goldenrod1")) + theme(text = element_text(size = 12)) cluster_plot</pre>
6	obtain the total WSSD, elect K by finding where the “elbow”	<pre>glance(penguin_clust)</pre>
7	calculate the total WSSD for a variety of K, create a data frame with a column names K	<pre>penguin_clust_ks <- tibble(k = 1:9) penguin_clust_ks</pre>
8	Calculate the WSSD for each K, complex data frame with 3 columns, one for K, one for the K-means	<pre>penguin_clust_ks <- tibble(k = 1:9) > rowwise() > mutate(penguin_clusters = list(kmeans(standardized_data, k))) penguin_clust_ks > pull(penguin_clusters) > pluck(1) penguin_clust_ks <- tibble(k = 1:9) ></pre>

	clustering objects, and one for the clustering statistics	<pre> rowwise() > mutate(penguin_clusts = list(kmeans(standardized_data, k)), glanced = list(glance(penguin_clusts))) penguin_clust_ks </pre>
9	extract the total WSSD from the column named glanced	<pre> clustering_statistics <- penguin_clust_ks > unnest(glanced) clustering_statistics </pre>
10	Line plot	<pre> elbow_plot <- ggplot(clustering_statistics, aes(x = k, y = tot.withinss)) + geom_point() + geom_line() + xlab("K") + ylab("Total within-cluster sum of squares") + scale_x_continuous(breaks = 1:9) + theme(text = element_text(size = 12)) elbow_plot </pre>
	Nstart to pick the best initialization - R will return to us the best clustering from this	<pre> penguin_clust_ks <- tibble(k = 1:9) > rowwise() > mutate(penguin_clusts = list(kmeans(standardized_data, nstart = 10, k)), glanced = list(glance(penguin_clusts))) clustering_statistics <- penguin_clust_ks > unnest(glanced) elbow_plot <- ggplot(clustering_statistics, aes(x = k, y = tot.withinss)) + geom_point() + geom_line() + xlab("K") + ylab("Total within-cluster sum of squares") + scale_x_continuous(breaks = 1:9) + theme(text = element_text(size = 12)) elbow_plot </pre>

Statistical Inference (ch.11)

Inferential Questions: Data analysis questions regarding how summaries, patterns, trends, or relationships in a data set extend to the wider population are called inferential questions.

- Using a sample to make an inference about the wider population the sample came from.
- Use the information from the sample to make a conclusion about the wider population
- Ensure the sample is unbiased

Statistics is something you compute using a sample, and then you infer (**inference**) of the population.

Something you compute using your sample. Example mean, proportion, max, min etc..

- Problems?
- When you get more samples, you might get a different number. You ended up worrying about the uncertainty of what your sample determines about the population
- Bias, when you pick the samples you are randomizing

Modes of inference: Many other advance of inference problem :

A/B Testing: See which website they will most likely click on

Estimation: a particular inferential problem where we try to estimate a quantitative property of the population. Take a statistic and appropriate the population.

Special kind of statistics that is trying to approximate a population parameter

Step 1: randomly select sample

Step 2: calculate the proportion in our sample and use it as a estimate of the true population proportion

> **population:** the population is the complete collection of individuals or cases we are interested in studying. Total set of items you can observed

> **population parameter:** computing a quantity—the proportion of iPhone owners—based on the entire population

> **sample:** a subset of individuals collected from the population.

> **sample estimate:** a numerical characteristic of the sample—that estimates the population parameter

> **statistical inference:** the process of using a sample to make a conclusion about the broader population from which it is taken. since the sample was random, if we were to take another random sample of size 40 and compute the proportion for that sample, we would not get the same

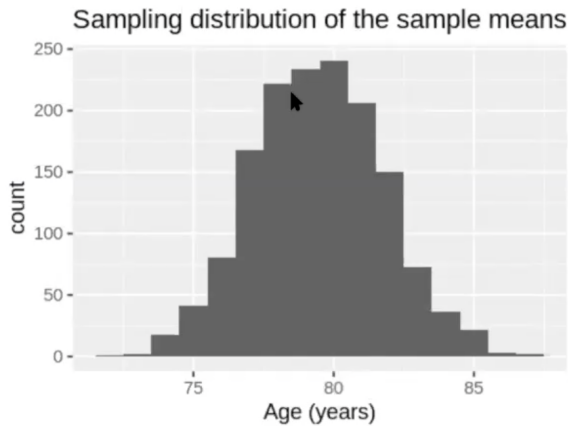
point estimate: our best guess of our population parameter using **this/ONE** sample

sampling variability: different value for our estimate this time. That means that our point estimate might be unreliable. Indeed, estimates vary from sample to sample

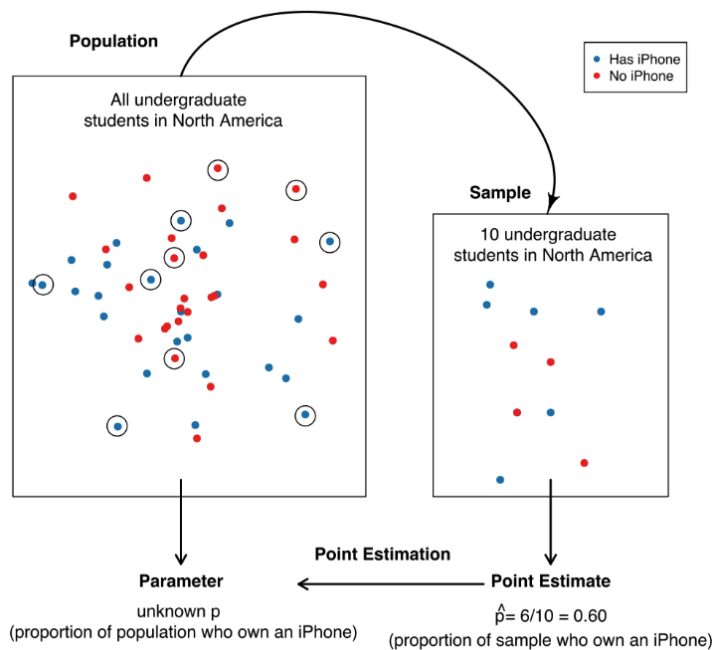
sampling distribution: The distribution of the estimate for all possible samples of a given size (which we commonly refer to as n) from a population. The sampling distribution will help us see how much we would expect our sample proportions from this population to vary for samples of size 40.

- The width of the bell curve tells you how certain you are.

Sample size is 40



- This shows the distribution of sample means you get with the sample size of 40. The peak of the distribution should be centered around the mean age.
- If the sample is smaller, there will be more of a spread because as we increase the sample size, you can more accurately predict the population parameter
- If the sample was larger, there would be less of a spread
- You might not be able to create the distributions from the total number of samples. You can really see the sampling distribution in practice
- the mean of the sampling distribution should be equal to the population proportion



- ☐ Describe real-world examples of questions that can be answered with statistical inference.

Ex. population of UBC that has an iPhone

Ex2. average price of a studio apartment rentals in Vancouver (population parameter is the average price per month)

- ☐ Define common population parameters (e.g., mean, proportion, standard deviation) that are often estimated using sampled data, and estimate these from a sample.
- ☐ Define the following statistical sampling terms: population, sample, population parameter, point estimate, and sampling distribution.
- ☐ Explain the difference between a population parameter and a sample point estimate.
- ☐ Use R to draw random samples from a finite population.
- ☐ Use R to create a sampling distribution from a finite population.
- ☐ Describe how sample size influences the sampling distribution.
- ☐ Define bootstrapping.
- ☐ Use R to create a bootstrap distribution to approximate a sampling distribution.
- ☐ Contrast the bootstrap and sampling distributions.

To calculate mean, med and sd

```
pop_parameters <- can_seniors|>
  summarize(pop_mean = mean(age),
            pop_med = median(age),
            pop_sd = sd(age))
```

To draw samples, group by and calculate mean

```
sample_estimates_100 <- rep_sample_n(can_seniors, size = 100, reps = 1500) |>
  group_by(replicate) |>
  summarise(sample_mean = mean(age))
```

Sampling distribution for proportions (interest—room_type—was categorical, and the population parameter was a proportion)

1.	Load the data and set seed	<pre>library(tidyverse) set.seed(123) airbnb <- read_csv("data/listings.csv") airbnb</pre>
2.	To estimate the proportion of the listing	<pre>airbnb > summarize(n = sum(room_type == "Entire home/apt"),</pre>

		<pre>proportion = sum(room_type == "Entire home/apt") / nrow(airbnb))</pre> <p>Entire home/apt listings in the data set is 0.747. This value, 0.747, is the population parameter. Remember, this parameter value is usually unknown in real data</p>
3.	<p>approximate it with a small subset of data! Randomly select, and compute the proportion for that sample.</p> <p>If we take another sample, the answer would be different</p>	<pre>library(infer) sample_1 <- rep_sample_n(tbl = airbnb, size = 40) airbnb_sample_1 <- summarize(sample_1, n = sum(room_type == "Entire home/apt"), prop = sum(room_type == "Entire home/apt") / 40)</pre> <p>airbnb_sample_1</p>
4.	we will simulate many samples	<pre>samples <- rep_sample_n(airbnb, size = 40, reps = 20000) samples</pre>
5.	compute the proportion of entire home/apartment listings in each sample	<pre>sample_estimates <- samples > group_by(replicate) > summarize(sample_proportion = sum(room_type == "Entire home/apt") / 40)</pre> <p>sample_estimates</p>
6.	samples of size 40 using a histogram	<pre>sampling_distribution <- ggplot(sample_estimates, aes(x = sample_proportion)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey", bins = 12) + labs(x = "Sample proportions", y = "Count") + theme(text = element_text(size = 12))</pre> <p>sampling_distribution</p>
7.	calculate the mean of the sample proportions	<pre>sample_estimates > summarize(mean = mean(sample_proportion))</pre>

- By increasing to 20,000 samples of size 40 the sample proportion is neither an overestimate or an underestimate of the population proportion.

Quantitative variables - estimate the population mean (or average)

1.	Visualize	<pre>population_distribution <- ggplot(airbnb, aes(x = price)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Price per night (Canadian dollars)", y = "Count") + theme(text = element_text(size = 12))</pre> <p>population_distribution</p>
2.	Calculate mean	<pre>population_parameters <- airbnb > summarize(pop_mean = mean(price))</pre> <p>population_parameters</p>
3.	Random sampling if the population is unknown	<pre>one_sample <- airbnb > rep_sample_n(40)</pre>
4.	Visualize again	<pre>sample_distribution <- ggplot(one_sample, aes(price)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Price per night (Canadian dollars)", y = "Count") + theme(text = element_text(size = 12))</pre> <p>sample_distribution</p>
5.	Calculate mean for sample	<pre>estimates <- one_sample > summarize(sample_mean = mean(price))</pre> <p>estimates</p>
6.	Increase reps	<pre>samples <- rep_sample_n(airbnb, size = 40, reps =</pre>

	of samples	20000) samples
7.	Calculate The mean for each rep	<pre>sample_estimates <- samples > group_by(replicate) > summarize(sample_mean = mean(price))</pre> <pre>sample_estimates</pre>
8.	Visualize	<pre>sampling_distribution_40 <- ggplot(sample_estimates, aes(x = sample_mean)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Sample mean price per night (Canadian dollars)", y = "Count") + theme(text = element_text(size = 12))</pre> <pre>sampling_distribution_40</pre>

- One way to improve a point estimate is to take a larger sample.
- First, the mean of the sample mean (across samples) is equal to the population mean. In other words, the sampling distribution is centered at the population mean. Second, increasing the size of the sample decreases the spread (i.e., the variability) of the sampling distribution. Therefore, a larger sample size results in a more reliable point estimate of the population parameter. And third, the distribution of the sample mean is roughly bell-shaped.

Interpreting example:

1. Where is the mean and what is the mean?
2. What shape is the graph (bell and one peak?)
3. Long right tail? Lower spread?

Bootstrapping (ch.11)

- But in real data analysis settings, we usually have just one sample from our population and do not have access to the population itself.
- Our sample estimate's value can vary significantly from the population parameter. So reporting the point estimate from a single sample alone may not be enough

Our sample estimate's value can vary significantly from the population parameter. We also need to report some notion of uncertainty in the value of the point estimate.

- approximate what the sampling distribution would look like for a sample, we could use that approximation to then report how uncertain our sample point estimate is (as we did above with the exact sampling distribution).

Bootstrapping : if our sample is big enough that it looks like our population, we can pretend that our sample is the population, and take more samples (with replacement) of the same size from it instead!

- **the bootstrap distribution** is the approx of sampling distribution
 - Pretend that the one sample is the population
 - Take a sample big enough so the chances of being unlucky is slimmer
1. Randomly select an observation from the original sample, which was drawn from the population.
 2. Record the observation's value.
 3. Return observation to the original sample. If you don't you'll have the exact same bootstrap sample each time
 4. Repeat steps 1–3 (sampling with replacement) until you have n observations, which form a bootstrap sample.
 5. Calculate the bootstrap point estimate (e.g., mean, median, proportion, slope, etc.) of the n observations in your bootstrap sample.
 6. Repeat steps 1–5 many times to create a distribution of point estimates (the bootstrap distribution).
 7. Calculate the plausible range of values around our observed point estimate.

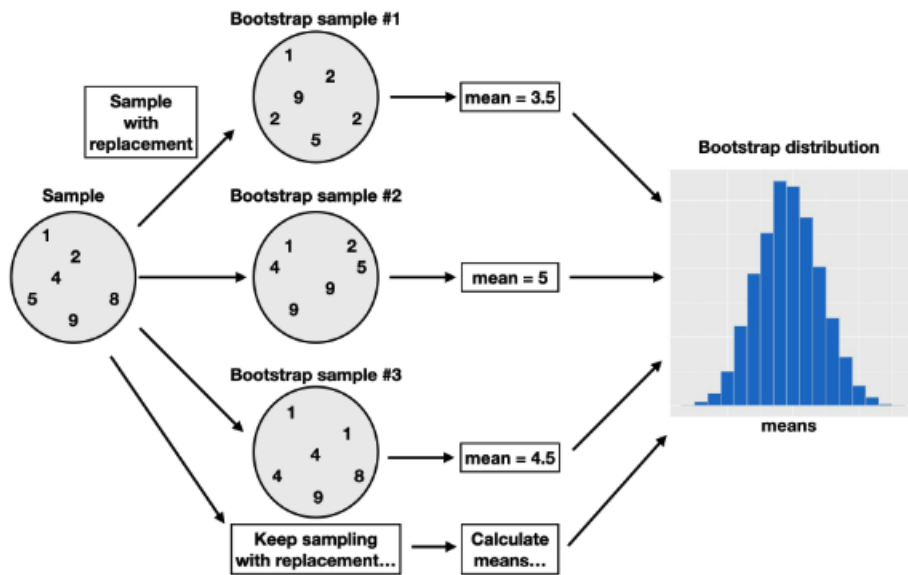
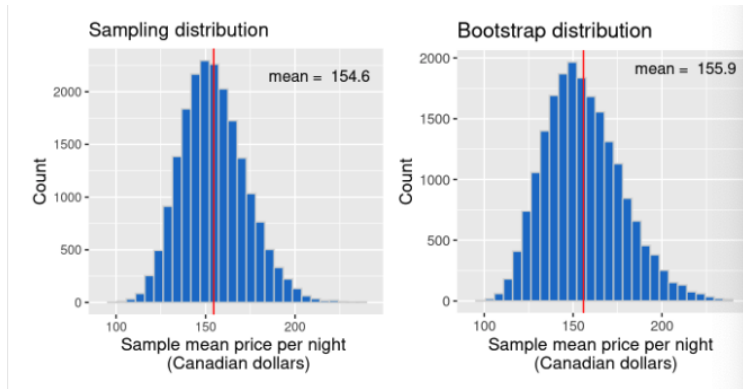


Figure 10.9: Overview of the bootstrap process.

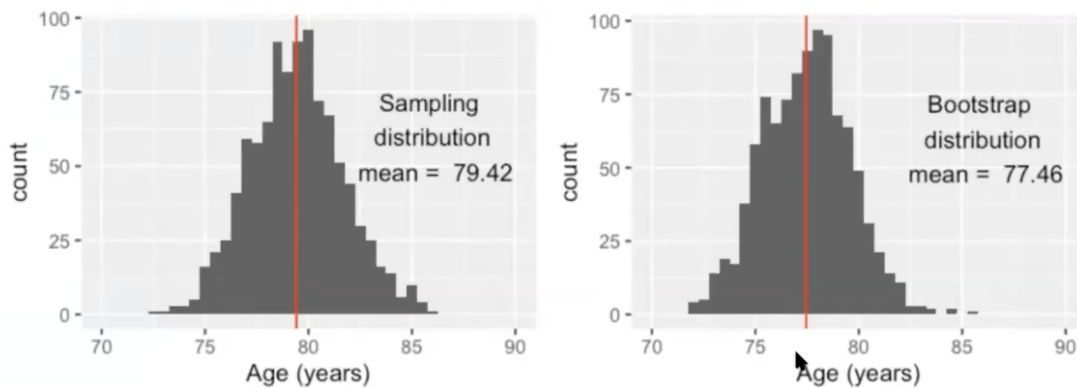
1.	Get sample	<code>one_sample</code>
2.	Visualize the sample	<pre>one_sample_dist <- ggplot(one_sample, aes(price)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Price per night (Canadian dollars)", y = "Count") + theme(text = element_text(size = 12))</pre>
3.	Summarize the sample	<code>summarize(boot1, mean = mean(price))</code>
4.	Create sample with higher reps	<pre>boot20000 <- one_sample > rep_sample_n(size = 40, replace = TRUE, reps = 20000)</pre>
5.	See the first 6 replicates	<pre>tail(boot20000)</pre>
6.	Create a histogram of the	<pre>six_bootstrap_samples <- boot20000 > filter(replicate <= 6)</pre>

	first 6 replicates	<pre>ggplot(six_bootstrap_samples, aes(price)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Price per night (Canadian dollars)", y = "Count") + facet_wrap(~replicate) + theme(text = element_text(size = 12))</pre>
7.	calculate the sample mean for each of these six replicates. bootstrap sample distributions and the sample means are different. They are different because we are sampling with replacement	<pre>six_bootstrap_samples > group_by(replicate) > summarize(mean = mean(price))</pre>
8.	calculate point estimates for our 20,000 bootstrap samples and generate a bootstrap distribution of our point estimates	<pre>boot20000_means <- boot20000 > group_by(replicate) > summarize(mean = mean(price))</pre> <p>boot20000_means</p>
9.	Visualize	<pre>boot_est_dist <- ggplot(boot20000_means, aes(x = mean)) + geom_histogram(fill = "dodgerblue3", color = "lightgrey") + labs(x = "Sample mean price per night \n (Canadian dollars)", y = "Count") + theme(text = element_text(size = 12))</pre> <p>boot_est_dist</p>
10.	Compare bootstrap distribution	

Sampling vs bootstrap distribution / Using the bootstrap to calculate a plausible range

Sampling vs bootstrap distribution

Reminder - true population mean = **79.3** years. The mean of our sample is 77.46 years.



- The spreads look roughly the same
- The bootstrap distribution lets us get a sense of the point estimate's variability.
- Because we are resampling from the original sample repeatedly, we see that the bootstrap distribution is centered at the original sample's mean value
- We can use a bootstrap distribution to calculate the possible range of values for the population
- Look at the range of bootstrap

confidence interval is a range of plausible values for the population parameter.

The confidence interval would be around 72 - 84 which is the range of the bootstrap distribution

- We will find the range of values covering the middle 95% of the bootstrap distribution, giving us a 95% confidence interval
 - A higher confidence level corresponds to a wider range of the interval, and a lower confidence level corresponds to a narrower range.
1. Arrange the observations in the bootstrap distribution in ascending order.
 2. Find the value such that 2.5% of observations fall below it (the 2.5% percentile). Use that value as the lower bound of the interval.
 3. Find the value such that 97.5% of observations fall below it (the 97.5% percentile). Use that value as the upper bound of the interval.

1.	<pre>bounds <- boot20000_means > select(mean) > pull() > quantile(c(0.025, 0.975))</pre>
----	---

		bounds
--	--	--------

Our sample mean age for Canadian seniors was measured to be 77.8 years, and we're 95% "confident" that the true population mean for Canadian seniors is between (73.7, 82.0).

Here our 95% confidence interval does contain the true population mean for Canadian seniors, 79.3 years - pretty neat! However, in real life we would never be able to know this because we only have observations from a single sample, not the whole population.