R for Business 2, w/ Excel

1201 Data Science: Tim Raiswell

2018/11/13

Quick Update

Some tweaks to our format:

- 90 minutes instead of 120 minutes.
- More hands-on work in R in the session.
- Boost the takeaway value of the slides.

B is for Bootstrapping

Definition in a Machine Learning Context

Sampling a dataset "with replacement" n times.

> "The bootstrap method is a statistical technique for estimating quantities about a population by averaging estimates from multiple small data samples. Importantly, samples are constructed by drawing observations from a large data sample one at a time and returning them to the data sample after they have been chosen. This allows a given observation to be included in a given small sample more than once. This approach to sampling is called sampling with replacement."

Source: A Gentle Introduction to the Bootstrap Method, Jason Brownlee

Why Bootstrap?

To quantify the uncertainty associated with a given estimator or statistical learning method, e.g.

1. To make more robust inferences about a sample in the absence of infinite data.
2. To test the efficacy of machine learning models multiple times using the same dataset.

Scenario

Imagine we have a random sample (N = 1,000) of the heights of people in a country. Normally, we would use that sample to make inferences - like the average height - about the broader (unknowable), real population. But we would never know the true error in our inferences.

To make our inference more robust, we draw 1,000 random samples from the original sample. Each time we take an observation from the bag, we log it, and return it to the bag. We repeat this exercise 10,000 times. Now we have 10,000 representative samples and averages to work with.

If we wanted to, we could also test a machine learning model on the data 10,000 times to truly gauge its effectiveness.

From Data to Inference, Example #1

Comma-Separated Values (CSV)

Open up R-Studio and start a new Rmarkdown file. Download this file: https://bit.ly/2OEIYoF and place it into your 1201 project directory. Run `getwd()` if you forgot where that is.

Type this code into a cell:

```
library(tidyverse)
data_from_csv <- read_csv("Sidewalk_Cafes.csv") #Import data on New York sidewalk cafe
# businesses and assign it to the data_from_csv object.
```

What message does R return?

What Type and Shape is the Dataset?

Type and run these commands separately into your markdown document or console. What are the outputs?

```
class(data_from_csv)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(data_from_csv)
```

```
## [1] 1008   12
```

What Data Does the Data Frame Contain?

Let's look.

```
glimpse(data_from_csv)
```

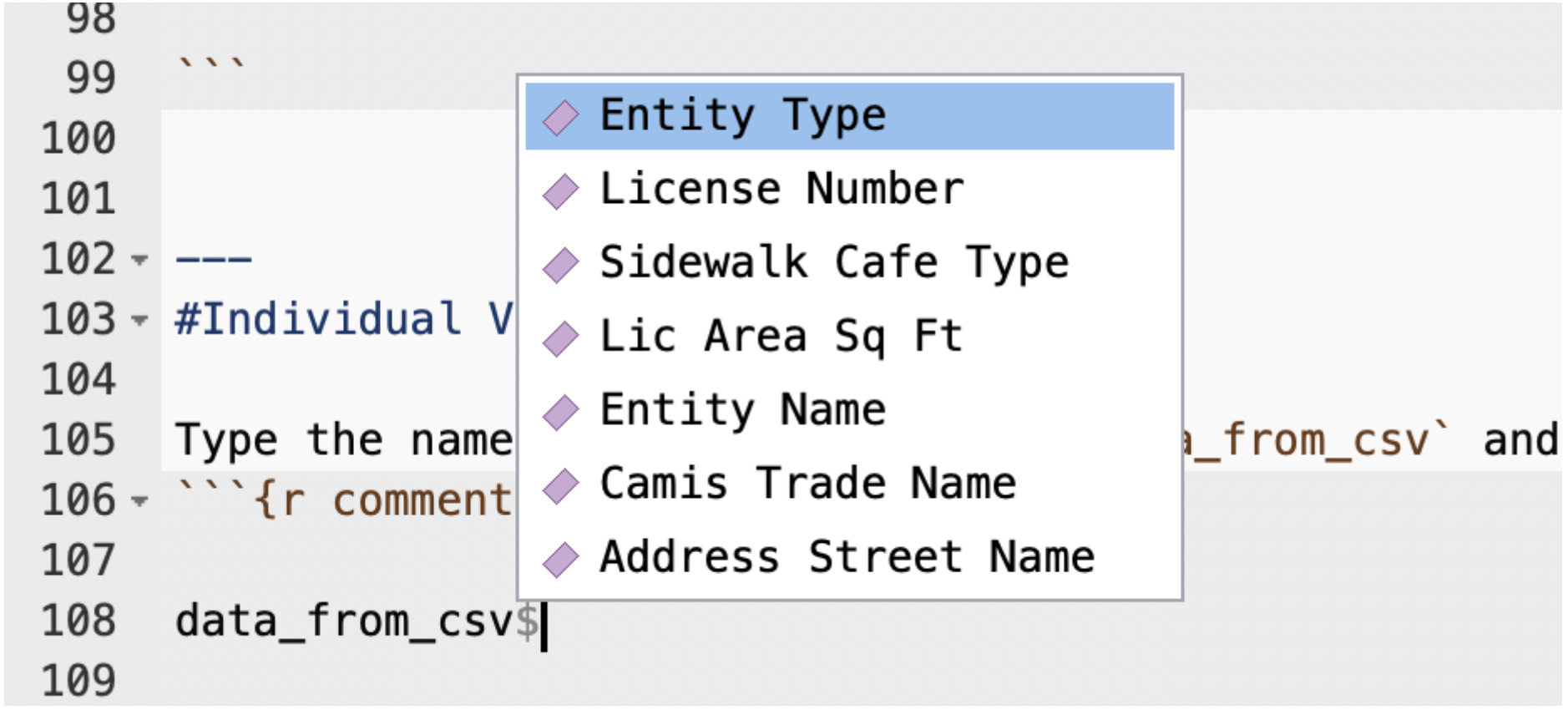
```
## Observations: 1,008
## Variables: 12
## $ `Entity Type`         <chr> "SIDEWALK CAFE", "SIDEWALK CAFE", "SIDEW...
## $ `License Number`      <int> 293907, 571720, 578810, 623018, 629616, ...
## $ `Sidewalk Cafe Type`  <chr> "Unenclosed", "Enclosed", "Enclosed", "E...
## $ `Lic Area Sq Ft`      <int> 611, 435, 407, 479, 366, 752, 129, 416, ...
## $ `Entity Name`         <chr> "DMF GRAMERCY ENTERPRISES INC", "1616 SE...
## $ `Camis Trade Name`    <chr> "PETE'S TAVERN       S/C #111", "DORRIAN...
## $ `Address Street Name` <chr> "EAST   18 STREET", "2 AVENUE", "3 AVENU...
## $ `Street Address`      <chr> "129 EAST   18 STREET", "1616 2 AVENUE",...
## $ `Address Location`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ `Address Zip Code`    <int> 10003, 10028, 10065, 11201, 10014, 10014...
## $ `Camis Phone Number`  <dbl> 2124737676, 2127726660, 2127581828, 7188...
## $ `Location 1`          <chr> "129 18 STREET\nNEW YORK, NY 10003\n(40....
```

Individual Variables

Type the name of your data object `data_from_csv` and then a `$` with no space. What happens?

Individual Variables

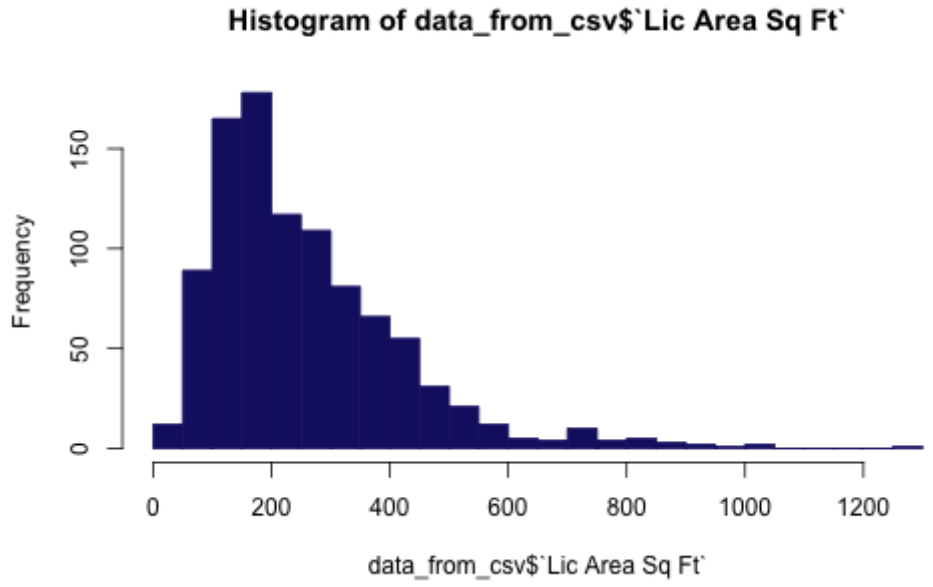
Type the name of your data object `data_from_csv` and then a `$` with no space. What happens?



Individual Variables Continued

Each variable in the data frame is a vector that can be accessed, analyzed and manipulated independently.

```
hist(data_from_csv$`Lic Area Sq Ft`,
     col = "midnightblue",
     border = "midnightblue",
     breaks = 20)
```



Individual Variables Continued

Try to get the mean average of the `Lic Area` variable using the `mean()` function. What does R output? Why?

```
mean(data_from_csv$`Lic Area Sq Ft`) # Take the mean of the area variable
```

Individual Variables Continued

Try to get the mean average of the `Lic Area` variable using the `mean()` function. What does R output? Why?

```
mean(data_from_csv$`Lic Area Sq Ft`) # Take the mean of the area variable
```

```
## [1] NA
```

Individual Variables Continued

Try to get the mean average of the `Lic Area` variable using the `mean()` function. What does R output? Why?

```
mean(data_from_csv$`Lic Area Sq Ft`) # Take the mean of the area variable
```

```
## [1] NA
```

Try this. Use any numbers you like. What is the output?

```
mean(c(2,7,3,5,9,9,2)) # Take the mean of a vector of numbers.
```

Individual Variables Continued

Try to get the mean average of the `Lic Area` variable using the `mean()` function. What does R output? Why?

```
mean(data_from_csv$`Lic Area Sq Ft`) # Take the mean of the area variable
```

```
## [1] NA
```

Try this. Use any numbers you like. What is the output?

```
mean(c(2,7,3,5,9,9,2)) # Take the mean of a vector of numbers.
```

```
## [1] 5.285714
```

So what happened with our data frame variable?

Individual Variables Continued

What do we learn if we run this code?

```
summary(data_from_csv$`Lic Area Sq Ft`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.0   145.0   214.0   258.6   337.0  1300.0      35
```

Individual Variables Continued

What do we learn if we run this code?

```
summary(data_from_csv$`Lic Area Sq Ft`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.0   145.0   214.0   258.6   337.0  1300.0      35
```

It gave us our missing mean average! But also some other important data.

Individual Variables Continued

What do we learn if we run this code?

```
summary(data_from_csv$`Lic Area Sq Ft`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.0   145.0   214.0   258.6   337.0  1300.0      35
```

It gave us our missing mean average! But also some other important data.

Try this.

```
mean(data_from_csv$`Lic Area Sq Ft`, na.rm = TRUE) # na.rm tells R to ignore the NA values.
```

```
## [1] 258.6475
```

Individual Variables Continued

- Now run this.
- Try it again.

What's happening? If we gathered the results and placed them into a histogram, what would happen?

```
mean(
  sample(data_from_csv$`Lic Area Sq Ft`, 50),
  na.rm = TRUE) # takes the mean of an n=50 sample.
```

```
## [1] 275.7083
```

Individual Variables Continued

We can do that!

```
sample_means <- replicate(1000,
                          mean(sample(data_from_csv$`Lic Area Sq Ft`, 50, replace=TRUE),
                               na.rm = TRUE)) # performs the same task as above, 1,000 times usi
```

Individual Variables Continued

Draw a histogram of the sample means.

```
hist(sample_means, breaks = 20,
     col = "darkgreen",
     border = "darkgreen")
abline(v=mean(sample_means),col="pink", lwd = 8) # adds a visual of the the mean average.
```

Calculate the Confidence Interval for the Mean

```
sd(sample_means) # calculate the standard deviation of the means
```

```
## [1] 22.65653
```

```
a <- 258.6475 # original sample mean
s <- 23.26 # standard deviation of the boot-strapped means
n <- 1000 # n =
error <- qnorm(0.95)*s/sqrt(n) # quantiles left = 0.05,
# right = 0.05 for a normal distro with standard dev of 23.11
left <- a-error
right <- a+error
left
```

```
## [1] 257.4376
```

```
right
```

```
## [1] 259.8574
```

Inference

We are 95% confident that mean cafe square footage in New York City falls between 257 and 260 square feet.

Working with Data from Excel

Install the `readxl` Package and Import the Data

Download the anonymous Gartner spreadsheet from here or use one of your own: https://bit.ly/2AYjaAb. I will remove this after this session to protect the data.

```
# install.packages('readxl')
audit_budget_data <- readxl::read_excel('audit.xlsm', sheet = "Raw Data")
```

```
dim(audit_budget_data)
```

```
## [1] 215 766
```

```
head(audit_budget_data$OUTCOMES_KPIs, 2)
```

```
## [1] "Stakeholder feedback; % of certified employees; Staff Utilization; % of Advisory Services; % of r
## [2] "Quantity of management requests that were completed and time to complete them."
```

Generate a Data Report

When you have a lot of variables and you're not familiar with the data, it can be useful to get a full report on the dataset you are working with.

```
# install.packages('dlookr')
library(dlookr)
```

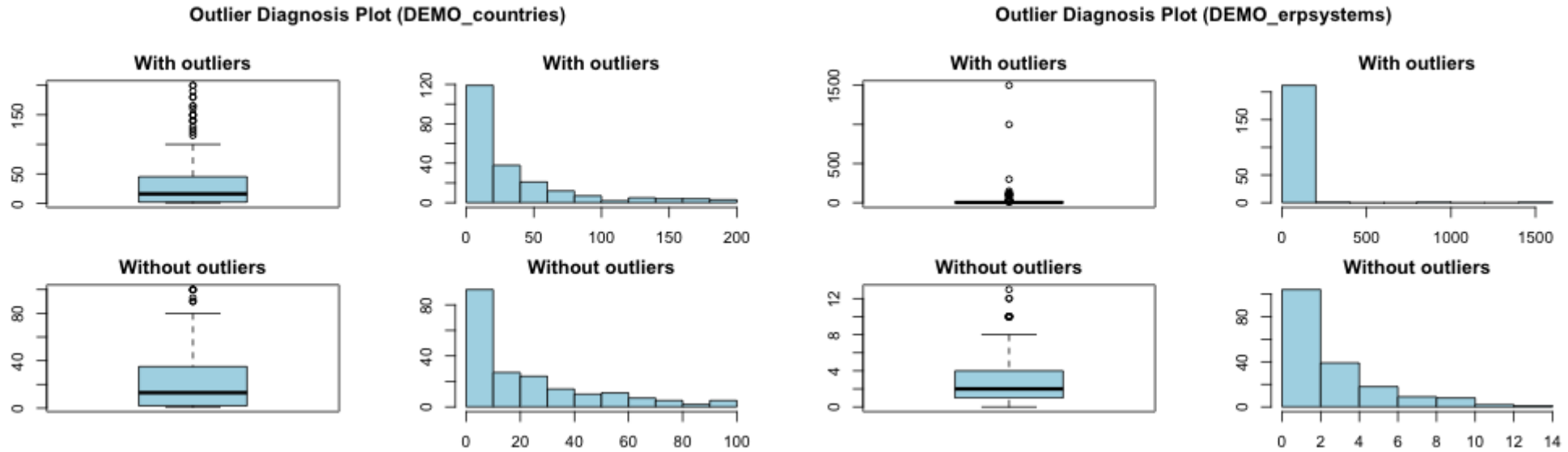
```
##
## Attaching package: 'dlookr'

## The following object is masked from 'package:base':
##
##     transform
```

```
diagnose(audit_budget_data[,1:5])
```

```
## # A tibble: 5 x 6
##   variables   types missing_count missing_percent unique_count unique_rate
##   <chr>       <chr>         <int>           <dbl>        <int>       <dbl>
## 1 RECORDID    char…             0               0          215      1
## 2 DEMO_reven… char…             0               0          196      0.912
## 3 DEMO_count… nume…             0               0           64      0.298
```

A Graphical Look at Outliers

```
plot_outlier(audit_budget_data[,1:4])
```



Putting it All Into a Single Report

This single line of code produces a full PDF report om the entire dataset. Very useful for getting familiar with large datasets! 

Reports like this can help you to think differently and creatively about your analysis.

```
#diagnose_report(audit_budget_data)
```

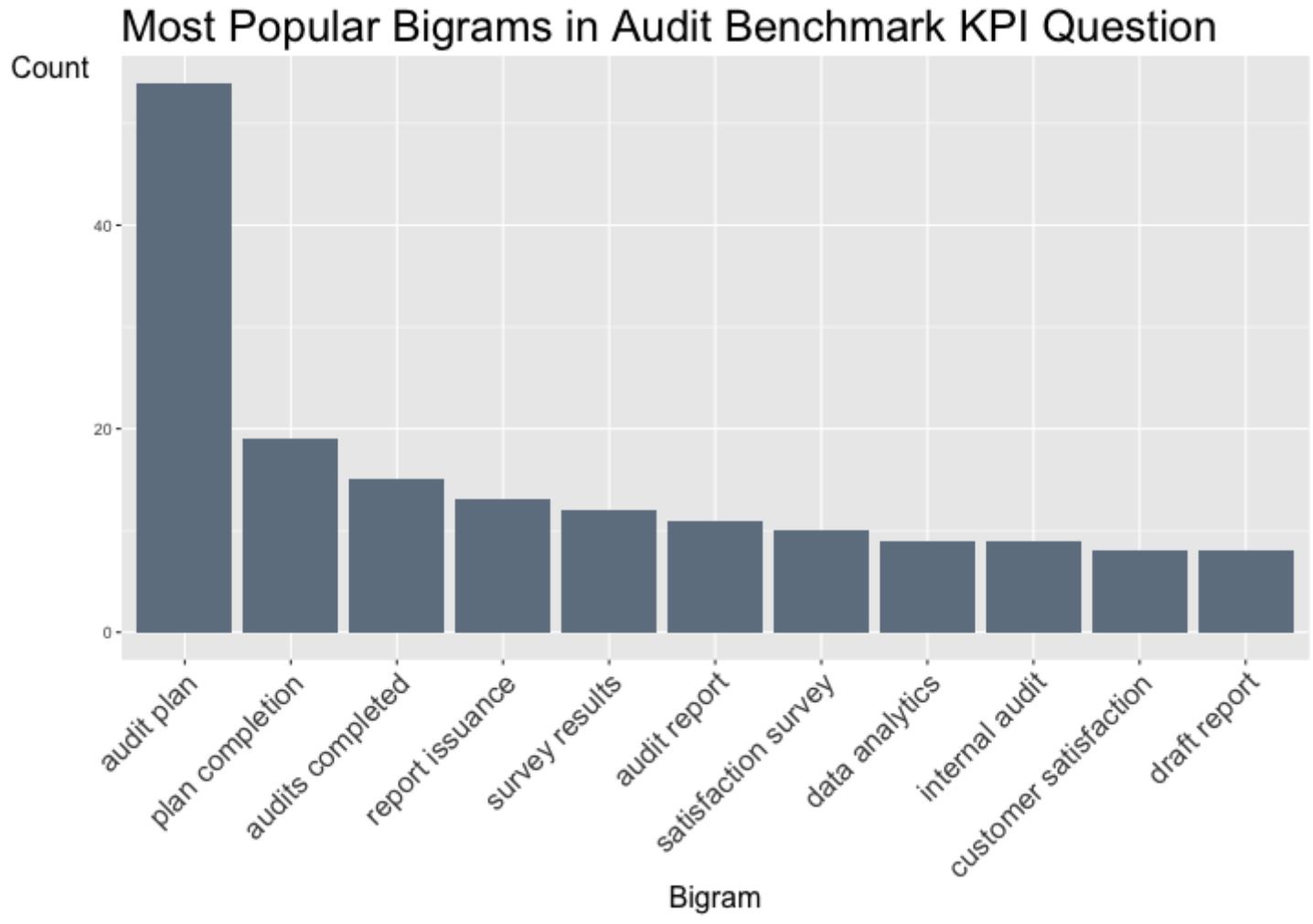
For more information on `dlookr` check out the full vignettes here.

What Else Can We Do?

Because our data is pretty close to tidy, less the terrible variable naming scheme, we can do anything we can imagine with the data.

```
library(tidytext)

audit_budget_data %>% # data
  select(OUTCOMES_KPIs) %>% # select a text variable
  tidytext::unnest_tokens(bigram, OUTCOMES_KPIs, token = "ngrams", n = 2) %>% # gen bigrams
  separate(bigram, c("word1", "word2"), sep = " ") %>% # separate bigrams...
  filter(!word1 %in% stop_words$word) %>% # filter out the stop words
  filter(!word2 %in% stop_words$word) %>% # ditto
  unite(bigram, word1, word2, sep = " ") %>% # reunite bigrams
  count(bigram, sort = TRUE) %>% #count them and sort descending
  top_n(10) %>% # top 10
  ggplot(aes(reorder(bigram, -n), n))+ # plot results...
  geom_bar(stat = "identity", fill = "slategray") + #in a bar chart
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.title.y = element_text(angle = 0, hjust = 1)) +
  labs(title = "Most Popular Bigrams in Audit Benchmark KPI Question",
       x = "Bigram",
       y = "Count")
```

The Output


Most Popular Bigrams in Audit Benchmark KPI Question
Count
40
20
0
audit plan
plan completion
audits completed
report issuance
survey results
audit report
satisfaction survey
data analytics
internal audit
customer satisfaction
draft report
Bigram