Introduction to Data

Lessons from DataCamp

Contents

Introduction **2**
required packages for this session 2

Chapter 1: Language of data **3**
Loading data into R 3
Identify variable types 3
Categorical data in R: factors 4
Filtering based on a factor 4
Complete filtering based on a factor 5
Discretize variables 6
Discretize a different variable 6
Combining levels of a different factor 6
Visualizing numerical and categorical data 7

Chapter 2: Study types and cautionary tales **9**
Identify study type 9
Identify the type of study 9
Generalizability & causal inference: random sampling vs random assignment 10
Random sampling or random assignment? 10
Identify the scope of inference of study 10
Understanding Simspsons Paradox 11
Number of males and females admitted 11
Proportion of males admitted overall 12
Proportion of males admitted for each department 12
Contingency table results by group 13

Chapter 3: Sampling strategies and experimental design **14**
Sampling strategies, determine which 14
Sampling strategies, choose worst 14
Sampling in R 14
Simple random sample (SRS) in R 14
Stratified sample in R 15
Compare SRS vs. stratified sample 16
Principles ofexperimental design 16
Identifying components of a study 16
Experimental design terminology 16
Connect blocking and stratifying 16

Chapter 4: Case study **18**
Beauty in the classroom 18
Inspect the data 18
Identify the type of study 18
Sampling / experimental attributes 19
Identify variable types 19
Recode a variable 20
Create a scatterplot 20
Create a scatterplot, with an added layer 21

Introduction

The following documentoutlines the written portion of the lessons from DataCamp's "Introduction to Data" course. This is a beginner course that requires you to have a basic understanding of R-programming.

As a note: All text is completely copied and pasted from the course. There are insances where the document refers to the "editor on the right", please note, that in this notebook document all of the instances are noted in the "r-chunks" (areas containing working r-code), which occurs below the text, rather than to the right. Furthermore, This lesson contained instructional videos at the beginign of new concepts that are not detailed in this document. However, even without these videos, the instructions are quite clear in indicating what the code is accomplishing.

If you have this document open on "R-Notebook", simply click "run" -> "Run all" (Or just press 'ctrl + alt + r'), let the "r-chunks" run (This might take a bit of time) then click "Preview". All necssary data is embedded within the code, no need to set a working directory or open an R-project.

This document was created by Neil Yetz on 10/01/2017. Please send any questions or concerns in this document to Neil at ndyetz@gmail.com

required packages for this session

```
#NOTE FROM NEIL: You will need to install and load these packages for this r-notebook to work.

#install.packages("openintro")
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("gapminder")
#install.packages("tidyr")

library(openintro)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
library(ggplot2)
library(gapminder)
```

```
## Warning: package 'gapminder' was built under R version 3.4.2
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.4.2
```

Chapter 1: Language of data

This chapter introduces terminology of datasets and data frames in R.

Loading data into R

In the video, you saw how to load the `hsb2` dataset into R using the `data()` function and how to preview its contents with `str()`.

In this exercise, you'll practice on another dataset, `email50`, which contains a subset of incoming emails for the first three months of 2012 for a single email account. You will examine the structure of this dataset and determine the number of rows (observations) and columns (variables).

Instructions

Load the `email50` dataset with the `data()` function.

View the structure of this dataset with the `str()` function. *How many observations and variables are there?*

```r
# Load data
data(email50)

# View its structure
str(email50)
```

```
## 'data.frame':    50 obs. of  21 variables:
##  $ spam        : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ to_multiple : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ from        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ cc          : int  0 0 4 0 0 0 0 0 1 0 ...
##  $ sent_email  : num  1 0 0 0 0 0 0 1 1 0 ...
##  $ time        : POSIXct, format: "2012-01-04 06:19:16" "2012-02-16 13:10:06" ...
##  $ image       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ attach      : num  0 0 2 0 0 0 0 0 0 0 ...
##  $ dollar      : num  0 0 0 0 9 0 0 0 0 23 ...
##  $ winner      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ inherit     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ viagra      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ password    : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ num_char    : num  21.705 7.011 0.631 2.454 41.623 ...
##  $ line_breaks : int  551 183 28 61 1088 5 17 88 242 578 ...
##  $ format      : num  1 1 0 0 1 0 0 1 1 1 ...
##  $ re_subj     : num  1 0 0 0 0 0 0 1 1 0 ...
##  $ exclaim_subj: num  0 0 0 0 0 0 0 0 1 0 ...
##  $ urgent_subj : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ exclaim_mess: num  8 1 2 1 43 0 0 2 22 3 ...
##  $ number      : Factor w/ 3 levels "none","small",..: 2 3 1 2 2 2 2 2 2 2 ...
```

Identify variable types

Recall from the video that the `glimpse()` function from `dplyr` provides a handy alternative to `str()` for previewing a dataset. In addition to telling you the number of observations and variables, it shows the name and type of each column, along with a neatly printed preview of its values.

Let's have another look at the `email50` data, so you can practice identifying variable types.

Instructions

Use the `glimpse()` function to view the variables in the `email50` dataset. *Identify each variable as either numerical or categorical, and further as discrete or continuous (if numerical) or ordinal or not ordinal (if categorical).*

```
# Glimpse email50
glimpse(email50)
```

```
## Observations: 50
## Variables: 21
## $ spam         <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0...
## $ to_multiple  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0...
## $ from         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ cc           <int> 0, 0, 4, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0...
## $ sent_email   <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1...
## $ time         <dttm> 2012-01-04 06:19:16, 2012-02-16 13:10:06, 2012-0...
## $ image        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ attach       <dbl> 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0...
## $ dollar       <dbl> 0, 0, 0, 0, 9, 0, 0, 0, 0, 23, 4, 0, 3, 2, 0, 0, ...
## $ winner       <fctr> no, no, no, no, no, no, no, no, no, no, no, no, ...
## $ inherit      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ viagra       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ password     <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0...
## $ num_char     <dbl> 21.705, 7.011, 0.631, 2.454, 41.623, 0.057, 0.809...
## $ line_breaks  <int> 551, 183, 28, 61, 1088, 5, 17, 88, 242, 578, 1167...
## $ format       <dbl> 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1...
## $ re_subj      <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1...
## $ exclaim_subj <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0...
## $ urgent_subj  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ exclaim_mess <dbl> 8, 1, 2, 1, 43, 0, 0, 2, 22, 3, 13, 1, 2, 2, 21, ...
## $ number       <fctr> small, big, none, small, small, small, small, sm...
```

Categorical data in R: factors

Filtering based on a factor

Categorical data are often stored as factors in R. In this exercise, you'll get some practice working with a factor variable, `number`, from the `email50` dataset. This variable tells you what type of number (none, small, or big) an email contains.

Recall from the video that the `filter()` function from `dplyr` allows you to filter a dataset to create a subset containing only certain levels of a variable. For example, the following code filters the `mtcars` dataset for cars containing 6 cylinders:

```
mtcars %>%
  filter(cyl == 6)
```

Instructions

Create a new dataset called `email50_big` that is a subset of the original `email50` dataset containing only emails with `"big"` numbers. This information is stored in the `number` variable.

Report the dimensions of `email50_big` using the `glimpse()` function again. How many emails contain big numbers?

```r
# Subset of emails with big numbers: email50_big
email50_big <- email50 %>%
  filter(number == "big")

# Glimpse the subset
glimpse(email50_big)
```

```
## Observations: 7
## Variables: 21
## $ spam         <dbl> 0, 0, 1, 0, 0, 0, 0
## $ to_multiple  <dbl> 0, 0, 0, 0, 0, 0, 0
## $ from         <dbl> 1, 1, 1, 1, 1, 1, 1
## $ cc           <int> 0, 0, 0, 0, 0, 0, 0
## $ sent_email   <dbl> 0, 0, 0, 0, 0, 1, 0
## $ time         <dttm> 2012-02-16 13:10:06, 2012-02-04 16:26:09, 2012-0...
## $ image        <dbl> 0, 0, 0, 0, 0, 0, 0
## $ attach       <dbl> 0, 0, 0, 0, 0, 0, 0
## $ dollar       <dbl> 0, 0, 3, 2, 0, 0, 0
## $ winner       <fctr> no, no, yes, no, no, no, no
## $ inherit      <dbl> 0, 0, 0, 0, 0, 0, 0
## $ viagra       <dbl> 0, 0, 0, 0, 0, 0, 0
## $ password     <dbl> 0, 2, 0, 0, 0, 0, 8
## $ num_char     <dbl> 7.011, 10.368, 42.793, 26.520, 6.563, 11.223, 10.613
## $ line_breaks  <int> 183, 198, 712, 692, 140, 512, 225
## $ format       <dbl> 1, 1, 1, 1, 1, 1, 1
## $ re_subj      <dbl> 0, 0, 0, 0, 0, 0, 0
## $ exclaim_subj <dbl> 0, 0, 0, 1, 0, 0, 0
## $ urgent_subj  <dbl> 0, 0, 0, 0, 0, 0, 0
## $ exclaim_mess <dbl> 1, 1, 2, 7, 2, 9, 9
## $ number       <fctr> big, big, big, big, big, big, big
```

Complete filtering based on a factor

The `droplevels()` function removes unused levels of factor variables from your dataset. As you saw in the video, it's often useful to determine which levels are unused (i.e. contain zero values) with the `table()` function.

In this exercise, you'll see which levels of the `number` variable are dropped after applying the `droplevels()` function.

Instructions

Make a `table()` of the `number` variable in the `email50_big` dataset. *Which two levels are unused?*

Apply the `droplevels()` function to the `number` variable. Assign the result back to `email50_big$number`.

Remake the `table()` of the `number` variable in the `email50_big` dataset. *How is this output different from the first?*

```r
# Table of number variable
table(email50_big$number)
```

```
## 
##  none small   big 
##     0     0     7
```

```r
# Drop levels
email50_big$number <- droplevels(email50_big$number)

# Another table of number variable
table(email50_big$number)
```

```
##
## big
##   7
```

Discretize variables

Discretize a different variable

In this exercise, you will create a categorical version of the `num_char` variable in the `email50` dataset, which tells you the number of characters in an email, in thousands. This new variable will have two levels—`"below median"` and `"at or above median"`—depending on whether an email has less than the median number of characters or equal to or more than that value.

The median marks the 50th percentile, or midpoint, of a distribution, so half of the emails should fall in one category and the other half in the other. You will learn more about the median and other measures of center in the next course in this series.

Instructions

The `email50 dataset` is available in your workspace.

Use the `num_char` variable to find the median number of characters in the emails and store the result as `med_num_char`.

Create a new variable called `num_char_cat`, which discretizes the `num_char` variable into `"below median"` or`"at or above median"`. Use the `mutate()` function from `dplyr` to accomplish this.

Apply `table()` on this new variable `num_char_cat` to determine how many emails are in each category and evaluate whether these counts match the expected numbers.

```r
# Calculate median number of characters: med_num_char
med_num_char <- median(email50$num_char)

# Create num_char_cat variable in email50
email50 <- email50 %>%
  mutate(num_char_cat = ifelse(num_char < med_num_char, "below median", "at or above median"))

# Count emails in each category
table(email50$num_char_cat)
```

```
##
## at or above median       below median
##                 25                 25
```

Combining levels of a different factor

Another common way of creating a new variable based on an existing one is by combining levels of a categorical variable. For example, the `email50` dataset has a categorical variable called number with levels `"none"`, `"small"`, and `"big"`, but suppose you're only interested in whether an email contains a number. In this exercise, you will create a variable containing this information and also visualize it.

For now, do your best to understand the code we've provided to generate the plot. We will go through it in detail in the next video.

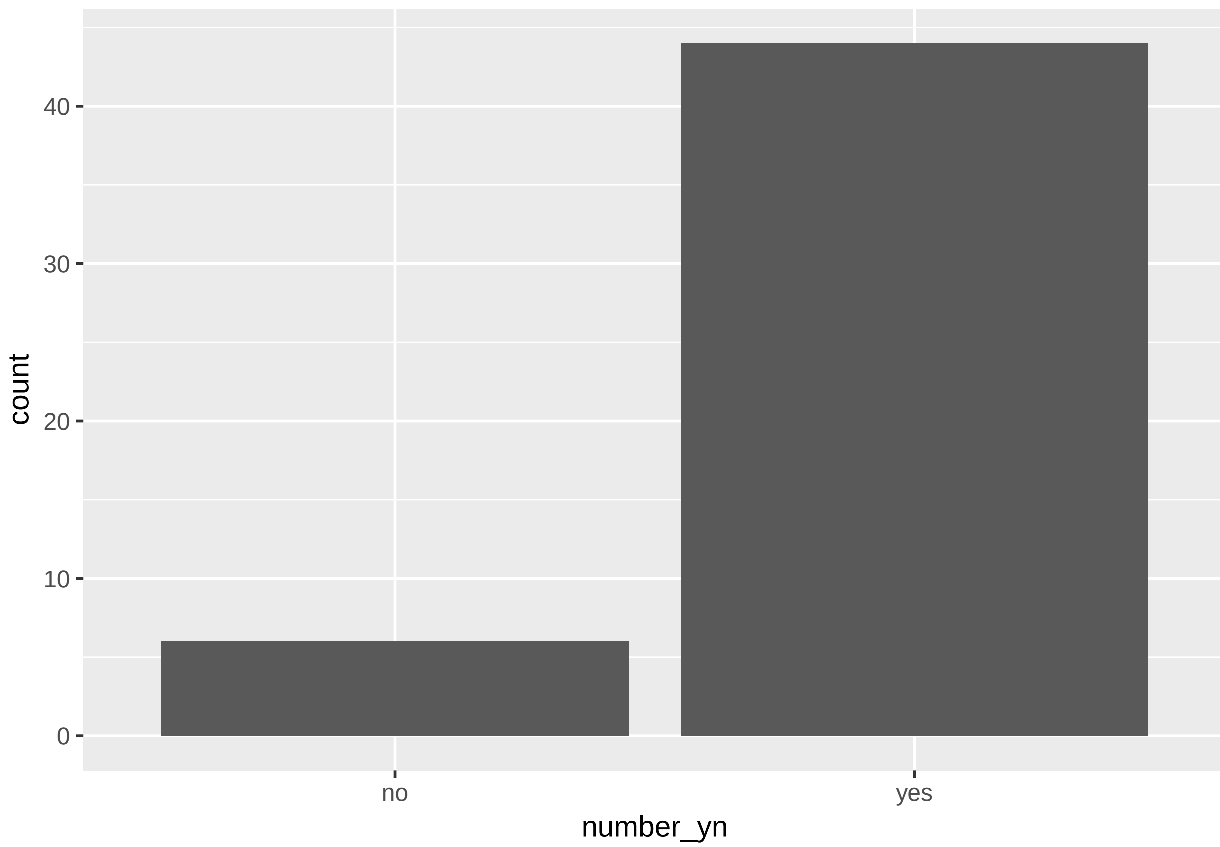
Instructions

Create a new variable in `email50` called `number_yn` that is `"no"` if there is no number in the email and `"yes"` if there is a small or a big number. The `ifelse()` function may prove useful here.

Run the code provided to visualize the distribution of the `number_yn` variable.

```
# Create number_yn column in email50
email50 <- email50 %>%
  mutate(number_yn = ifelse(number == "none", "no", "yes"))

# Visualize number_yn
ggplot(email50, aes(x = number_yn)) +
  geom_bar()
```



Visualizing numerical and categorical data

In this exercise, you will visualize the relationship between two numerical variables from the `email50` dataset, conditioned on whether or not the email was spam. This means that we will use some aspect of the plot (like color or shape) to separate the groups in the `spam` column so that we can compare plotted values between them.

Recall that in the `ggplot()` function, the first argument gives the dataset, then the aesthetics map the variables to certain features of the plot, and finally the `geom_*()` layer informs the type of plot you want to make. In this exercise, you will make a scatterplot by adding the `geom_point()` layer to your `ggplot()` call.

Instructions

Create a scatterplot of number of exclamation points in the email message (`exclaim_mess`) vs. number of characters (`num_char`).

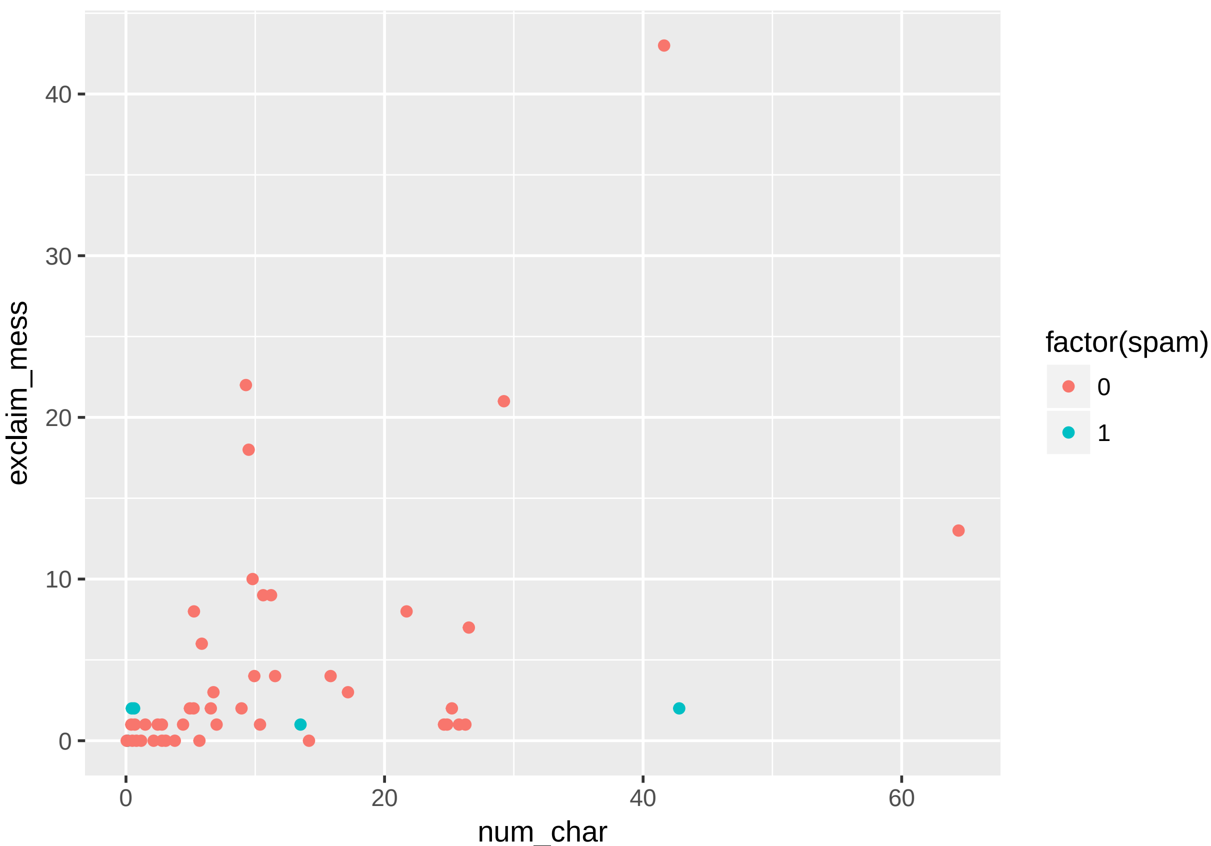
Color points by whether or not the email is `spam`.

Note that the spam variable is stored as numerical (0/1) but you want to use it as a categorical variable in this plot. To do this, you need to force R to think of it as such with the `factor()` function.

Based on the plot, what's the relationship between these variables?

```
# Load ggplot2
library(ggplot2)

# Scatterplot of exclaim_mess vs. num_char
ggplot(email50, aes(x = num_char, y = exclaim_mess, color = factor(spam))) +
  geom_point()
```



Chapter 2: Study types and cautionary tales

In this chapter, you will learn about observational studies and experiments, scope of inference, and Simpson's paradox.

Identify study type

A study is designed to evaluate whether people read text faster in Arial or Helvetica font. A group of volunteers who agreed to be a part of the study are randomly assigned to two groups: one where they read some text in Arial, and another where they read the same text in Helvetica. At the end, average reading speeds from the two groups are compared.

What type of study is this?

Possible Answers (Correct answer is **Bolded**)

Observational study
Experiment
Neither, since the sample consists of volunteers

Identify the type of study

Next, let's take a look at data from a different study on country characteristics. You'll load the data first and view it, then you'll be asked to identify the type of study. Remember, an experiment requires random assignment.

Instructions

Load the `gapminder` data. This dataset comes from the `gapminder` R package, which has already been loaded for you.

View the variables in the dataset with `glimpse()`.

If these data come from an observational study, assign `"observational"` to the `type_of_study` variable. If experimental, assign `"experimental"`.

```r
# Load data
data(gapminder)

# Glimpse data
glimpse(gapminder)
```

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

```r
# Identify type of study
type_of_study <- "observational"
```

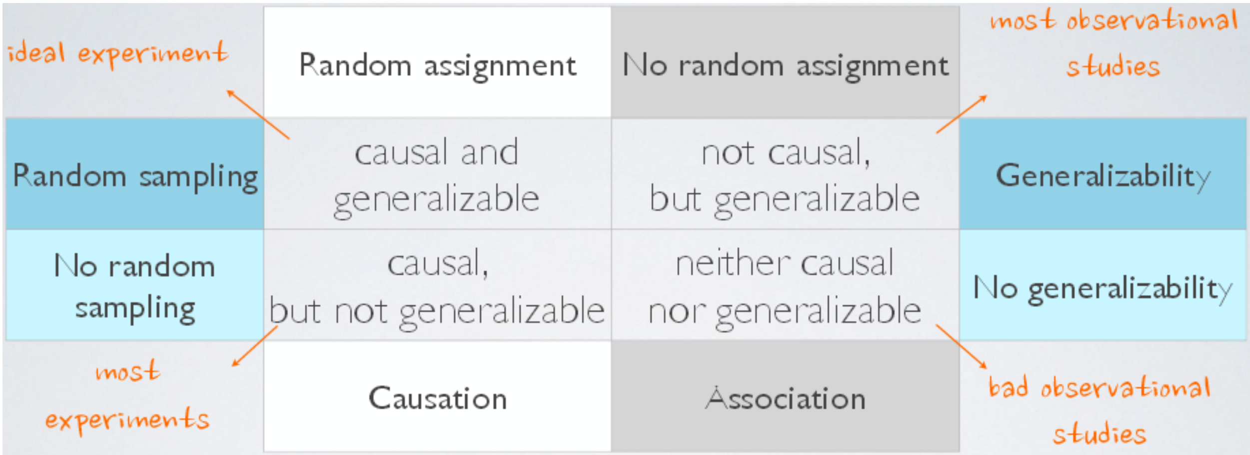


Figure 1:

Generalizability & causal inference: random sampling vs random assignment

Random sampling or random assignment?

One of the early studies linking smoking and lung cancer compared patients who are already hospitalized with lung cancer to similar patients without lung cancer (hospitalized for other reasons), and recorded whether each patient smoked. Then, proportions of smokers for patients with and without lung cancer were compared.

Does this study employ random sampling and/or random assignment?

Possible Answers (Correct answer is **Bolded**) Random sampling, but not random assignment
Random assignment, but not random sampling
Neither random sampling nor random assignment
Both random sampling and random assignment

Identify the scope of inference of study

Volunteers were recruited to participate in a study where they were asked to type 40 bits of trivia—for example, "an ostrich's eye is bigger than its brain"—into a computer. A randomly selected half of these subjects were told the information would be saved in the computer; the other half were told the items they typed would be erased.

Then, the subjects were asked to remember these bits of trivia, and the number of bits of trivia each subject could correctly recall were recorded. It was found that the subjects were significantly more likely to remember information if they thought they would not be able to find it later.

The results of the study "_____" be generalized to all people and a causal link between believing information is stored and memory "_____" be inferred based on these results.

Possible Answers (correct answer is **Bolded**)

cannot, cannot
cannot, can
can, cannot
can, can

Understanding Simspsons Paradox

Simpson's paradox, or the Yule–Simpson effect, **is a phenomenon in probability and statistics, in which a trend appears in different groups of data but disappears or reverses when these groups are combined** *(i.e. with the presence of confounders within the statistical model)*. It is sometimes given the descriptive title reversal paradox or amalgamation paradox.

Number of males and females admitted

In order to calculate the number of males and females admitted, we will introduce two new functions: `count()` from the `dplyr` package and `spread()` from the `tidyr` package.

In one step, `count()` allows you to group the data by certain variables (in this case, admission status and gender) and then counts the number of observations in each category. These counts are available under a new variable called `n`.

`spread()` simply reorganizes the output across columns based on a key-value pair, where a pair contains a key that explains what the information describes and a value that contains the actual information. `spread()` takes the name of the dataset as its first argument, the name of the `key` column as its second argument, and the name of the `value` column as its third argument, all specified without quotation marks.

Instructions

Use the `ucb_admit` dataset (which is already pre-loaded) and the `count()` function to determine the total number of males and females admitted. Assign the result to `ucb_counts`.

Print `ucb_counts` to the console.

Then, use the `spread()` function to spread the output across columns based on admission status (key) and `n` (value).

```r
# Load packages
library(dplyr)
library(tidyr)

# Count number of male and female applicants admitted
ucb_counts <- ucb_admit %>%
  count(Admit,Gender)

# View result
ucb_counts
```

```
## # A tibble: 4 x 3
##      Admit Gender     n
##     <fctr> <fctr> <int>
## 1 Admitted   Male  1198
## 2 Admitted Female   557
## 3 Rejected   Male  1493
## 4 Rejected Female  1278
```

```r
# Spread the output across columns
ucb_counts %>%
  spread(Admit, n)
```

```
## # A tibble: 2 x 3
##   Gender Admitted Rejected
## * <fctr>    <int>    <int>
## 1   Male     1198     1493
```

```
## 2 Female      557      1278
```

Proportion of males admitted overall

You can now calculate the percentage of males admitted. To do so, you will create a new variable with `mutate()` from the `dplyr` package.

Instructions

`dplyr` and `tidyr` have been loaded for you.

Use the code from the previous exercise to construct a table of counts of admission status and gender.

Then, use the `mutate()` function create a new variable, `Perc_Admit`, which is the ratio of those admitted, `Admitted`, to all applicants of that gender, (`Admitted + Rejected`).

Which gender has a higher admission rate, male or female?

```
ucb_admit %>%
# Table of counts of admission status and gender
  count(Admit, Gender) %>%
# Spread output across columns based on admission status
  spread(Admit, n) %>%
# Create new variable
  mutate(Perc_Admit = Admitted / (Admitted + Rejected))
```

```
## # A tibble: 2 x 4
##   Gender Admitted Rejected Perc_Admit
##   <fctr>    <int>    <int>      <dbl>
## 1   Male     1198     1493  0.4451877
## 2 Female      557     1278  0.3035422
```

Proportion of males admitted for each department

Next you'll make a table similar to the one you constructed earlier, except you will first group the data by department. Then, you'll use this table to calculate the proportion of males admitted in each department.

Instructions

`dplyr` and `tidyr` have been loaded for you.

Use the code from earlier to create a table of counts of admission status and gender, but this time group first by `Dept.` Assign this result to `admit_by_dept`.

Print `admit_by_dept` to the console.

Calculate the percentage of those admitted in each department, `Perc_Admit`, by applying the `mutate()` function to `admit_by_dept`.

```
# Table of counts of admission status and gender for each department
admit_by_dept <- ucb_admit %>%
  count(Dept, Gender, Admit) %>%
  spread(Admit, n)

# View result
admit_by_dept
```

```
## # A tibble: 12 x 4
##     Dept Gender Admitted Rejected
```

```
##    * <chr> <fctr>    <int>    <int>
##   1     A   Male      512      313
##   2     A Female       89       19
##   3     B   Male      353      207
##   4     B Female       17        8
##   5     C   Male      120      205
##   6     C Female      202      391
##   7     D   Male      138      279
##   8     D Female      131      244
##   9     E   Male       53      138
##  10     E Female       94      299
##  11     F   Male       22      351
##  12     F Female       24      317
```

```
# Percentage of those admitted to each department
admit_by_dept %>%
  mutate(Perc_Admit = Admitted / (Admitted + Rejected))
```

```
## # A tibble: 12 x 5
##      Dept Gender Admitted Rejected Perc_Admit
##     <chr> <fctr>    <int>    <int>      <dbl>
##   1     A   Male      512      313 0.62060606
##   2     A Female       89       19 0.82407407
##   3     B   Male      353      207 0.63035714
##   4     B Female       17        8 0.68000000
##   5     C   Male      120      205 0.36923077
##   6     C Female      202      391 0.34064081
##   7     D   Male      138      279 0.33093525
##   8     D Female      131      244 0.34933333
##   9     E   Male       53      138 0.27748691
##  10     E Female       94      299 0.23918575
##  11     F   Male       22      351 0.05898123
##  12     F Female       24      317 0.07038123
```

Contingency table results by group

The final result from the previous exercise is available in your workspace as `perc_admit_by_dept`. Which of the following best describes the relationship between admission status and gender?

Possible Answers (COrrect answer is **Bolded**)

Within most departments, female applicants are more likely to be admitted.

Within most departments, male applicants are more likely to be admitted.

Within most departments, male and female applicants are equally likely to be admitted.

Chapter 3: Sampling strategies and experimental design

This chapter defines various sampling strategies and their benefits/drawbacks as well as principles of experimental design.

Sampling strategies, determine which

A consulting company is planning a pilot study on marketing in Boston. They identify the zip codes that make up the greater Boston area, then sample 50 randomly selected addresses from each zip code and mail a coupon to these addresses. They then track whether the coupon was used in the following month.

What sampling strategy has this company used?

Possible Answers (Correct answer is **Bolded**)

Simple random sample
Stratified sample
Cluster sample
Multistage sample

Sampling strategies, choose worst

A school district has requested a survey be conducted on the socioeconomic status of their students. Their budget only allows them to conduct the survey in some of the schools, hence they need to first sample a few schools.

Students living in this district generally attend a school in their neighborhood. The district is broken into many distinct and unique neighborhoods, some including large single-family homes and others with only low-income housing.

Which approach would likely be the **least effective** for selecting the schools where the survey will be conducted?

Possible Answers (Correct answer is **Bolded**)

Simple random sampling
Stratified sampling, where each stratum is a neighborhood
Cluster sampling, where each cluster is a neighborhood

Sampling in R

Simple random sample (SRS) in R

Suppose you want to collect some data from a sample of eight states. A list of all states and the region they belong to (Northeast, Midwest, South, West) are given in the `us_regions` data frame.

Instructions

The `dplyr` package and `us_regions` data frame have been loaded for you.

Use simple random sampling to select eight states from `us_regions`. Save this sample in a data frame called states_srs.

Count the number of states from each region in your sample.

```r
#load the US_regions dataframe
load("us_regions.Rdata")

# Simple random sample: states_srs
#Note from Neil, you will get different results everytime for your "sample_n()" function. It is taking
states_srs <- us_regions %>%
  sample_n(8)

# Count states by region
states_srs %>%
  group_by(region) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   region [4]
##      region     n
##      <fctr> <int>
## 1   Midwest     4
## 2 Northeast     1
## 3     South     1
## 4      West     2
```

Stratified sample in R

In the last exercise, you took a simple random sample of eight states. However, as you may have noticed when you counted the number of states selected from each region, this strategy is unlikely to select an equal number of states from each region. The goal of stratified sampling is to select an equal number of states from each region.

Instructions

The `dplyr` package has been loaded for you and `us_regions` is still available in your workspace.

Use stratified sampling to select a total of eight states, where each stratum is a region. Save this sample in a data frame called `states_str`.

Count the number of states from each region in your sample to confirm that each region is represented equally in your sample.

```r
# Stratified sample
states_str <- us_regions %>%
  group_by(region) %>%
  sample_n(2)

# Count states by region
states_str %>%
  group_by(region) %>%
  count()
```

```
## # A tibble: 4 x 2
## # Groups:   region [4]
##      region     n
##      <fctr> <int>
## 1   Midwest     2
## 2 Northeast     2
## 3     South     2
```

```
## 4      West     2
```

Compare SRS vs. stratified sample

Which method, simple random sampling or stratified sampling, ensures an equal number of states from each region?

Possible Answers (Correct answer is **Bolded**)

Simple random sampling
Stratified sampling

Principles ofexperimental design

Identifying components of a study

A researcher designs a study to test the effect of light and noise levels on exam performance of students. The researcher also believes that light and noise levels might have different effects on males and females, so she wants to make sure both genders are represented equally under different conditions.

Which of the below is correct?

Possible Answers (Correct answer is **Bolded**)

There are 3 explanatory variables (light, noise, gender) and 1 response variable (exam performance).

There is 1 explanatory variable (gender) and 3 response variables (light, noise, exam performance).

There are 2 blocking variables (light and noise), 1 explanatory variable (gender), and 1 response variable (exam performance).

There are 2 explanatory variables (light and noise), 1 blocking variable (gender), and 1 response variable (exam performance).

Experimental design terminology

"_____" variables are conditions you can impose on the experimental units, while "_____" variables are characteristics that the experimental units come with that you would like to control for.

Possible Answers (Correct answer is **BOLDED**)

Blocking, explanatory
Explanatory, blocking
Control, treatment
Treatment, control

Connect blocking and stratifying

In random sampling, you use "_____" to control for a variable. In random assignment, you use "_____" to achieve the same goal.

Possible Answers (Correct answer is **BOLDED**)

stratifying, blocking

blocking, stratifying

confounding, stratifying

confounding, blocking

Chapter 4: Case study

Apply terminology, principles, and R code learned in the first three chapters of this course to a case study looking at how the physical appearance of instructors impacts their students' course evaluations.

```
load("evals.RData")
```

Beauty in the classroom

For this chapter, you will be working with a dataset [dataset = `evals`] of student course evaluations, including information about their opinions on the professor. We are interested in seeing if the physical attractiveness of the professor has an impact on evaluation scores.

Inspect the data

The purpose of this chapter is to give you an opportunity to apply and practice what you've learned on a real world dataset. For this reason, we'll provide a little less guidance than usual.

The data from the study described in the video are available in your workspace as `evals`. Let's take a look!

Instructions

Inspect the `evals` data frame using techniques you learned in previous chapters. Use an approach that shows you how many observations and variables are included in the dataset.

```
# Inspect evals
str(evals)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    463 obs. of  21 variables:
##  $ score        : num  4.7 4.1 3.9 4.8 4.6 4.3 2.8 4.1 3.4 4.5 ...
##  $ rank         : Factor w/ 3 levels "teaching","tenure track",..: 2 2 2 2 3 3 3 3 3 3 ...
##  $ ethnicity    : Factor w/ 2 levels "minority","not minority": 1 1 1 1 2 2 2 2 2 2 ...
##  $ gender       : Factor w/ 2 levels "female","male": 1 1 1 1 2 2 2 2 2 1 ...
##  $ language     : Factor w/ 2 levels "english","non-english": 1 1 1 1 1 1 1 1 1 1 ...
##  $ age          : int  36 36 36 36 59 59 59 51 51 40 ...
##  $ cls_perc_eval: num  55.8 68.8 60.8 62.6 85 ...
##  $ cls_did_eval : int  24 86 76 77 17 35 39 55 111 40 ...
##  $ cls_students : int  43 125 125 123 20 40 44 55 195 46 ...
##  $ cls_level    : Factor w/ 2 levels "lower","upper": 2 2 2 2 2 2 2 2 2 2 ...
##  $ cls_profs    : Factor w/ 2 levels "multiple","single": 2 2 2 2 1 1 1 2 2 2 ...
##  $ cls_credits  : Factor w/ 2 levels "multi credit",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ bty_f1lower  : int  5 5 5 5 4 4 4 5 5 2 ...
##  $ bty_f1upper  : int  7 7 7 7 4 4 4 2 2 5 ...
##  $ bty_f2upper  : int  6 6 6 6 2 2 2 5 5 4 ...
##  $ bty_m1lower  : int  2 2 2 2 2 2 2 2 2 3 ...
##  $ bty_m1upper  : int  4 4 4 4 3 3 3 3 3 3 ...
##  $ bty_m2upper  : int  6 6 6 6 3 3 3 3 3 2 ...
##  $ bty_avg      : num  5 5 5 5 3 ...
##  $ pic_outfit   : Factor w/ 2 levels "formal","not formal": 2 2 2 2 2 2 2 2 2 2 ...
##  $ pic_color    : Factor w/ 2 levels "black&white",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Identify the type of study

What type of study is this?

Possible Answers (Correct answer is **Bolded**)

Observational study
Experiment

Sampling / experimental attributes

The data from this study were gathered by _____.

Possible Answers (Correct answer is **Bolded**)

randomly sampling classes
randomly sampling students
randomly assigning students to classes
randomly assigning professors to students

Identify variable types

It's always useful to start your exploration of a dataset by identifying variable types. The results from this exercise will help you design appropriate visualizations and calculate useful summary statistics later in your analysis.

Instructions

Explore the evals data once again with the following goals in mind:

- *Identify each variable as numerical or categorical.*
- *If numerical, determine if discrete or continuous.*
- *If categorical, determine if ordinal or not.*

We've created a vector of variable names in the editor called `cat_vars`. To test your understanding of the data, remove the names of any variables that are not categorical.

```r
# Inspect variable types
str(evals)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    463 obs. of  21 variables:
##  $ score        : num  4.7 4.1 3.9 4.8 4.6 4.3 2.8 4.1 3.4 4.5 ...
##  $ rank         : Factor w/ 3 levels "teaching","tenure track",..: 2 2 2 2 3 3 3 3 3 3 ...
##  $ ethnicity    : Factor w/ 2 levels "minority","not minority": 1 1 1 1 2 2 2 2 2 2 ...
##  $ gender       : Factor w/ 2 levels "female","male": 1 1 1 1 2 2 2 2 2 1 ...
##  $ language     : Factor w/ 2 levels "english","non-english": 1 1 1 1 1 1 1 1 1 1 ...
##  $ age          : int  36 36 36 36 59 59 59 51 51 40 ...
##  $ cls_perc_eval: num  55.8 68.8 60.8 62.6 85 ...
##  $ cls_did_eval : int  24 86 76 77 17 35 39 55 111 40 ...
##  $ cls_students : int  43 125 125 123 20 40 44 55 195 46 ...
##  $ cls_level    : Factor w/ 2 levels "lower","upper": 2 2 2 2 2 2 2 2 2 2 ...
##  $ cls_profs    : Factor w/ 2 levels "multiple","single": 2 2 2 2 1 1 1 2 2 2 ...
##  $ cls_credits  : Factor w/ 2 levels "multi credit",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ bty_f1lower  : int  5 5 5 5 4 4 4 5 5 2 ...
##  $ bty_f1upper  : int  7 7 7 7 4 4 4 2 2 5 ...
##  $ bty_f2upper  : int  6 6 6 6 2 2 2 5 5 4 ...
##  $ bty_m1lower  : int  2 2 2 2 2 2 2 2 2 3 ...
##  $ bty_m1upper  : int  4 4 4 4 3 3 3 3 3 3 ...
##  $ bty_m2upper  : int  6 6 6 6 3 3 3 3 3 2 ...
```

```
##  $ bty_avg      : num  5 5 5 5 3 ...
##  $ pic_outfit   : Factor w/ 2 levels "formal","not formal": 2 2 2 2 2 2 2 2 2 2 ...
##  $ pic_color    : Factor w/ 2 levels "black&white",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Remove non-factor variables from this vector
#cat_vars <- c("score", "rank", "ethnicity", "gender", "language", "age",
#              "cls_students", "cls_level", "cls_profs", "cls_credits",
#              "bty_avg", "pic_outfit", "pic_color")

#Correct vector with nan-factor variables removed:
cat_vars <- c("rank", "ethnicity", "gender", "language",
              "cls_level", "cls_profs", "cls_credits",
             "pic_outfit", "pic_color")
```

Recode a variable

The `cls_students` variable in `evals` tells you the number of students in the class. Suppose instead of the exact number of students, you're interested in whether the class is

- `"small"` (18 students or fewer),
- `"midsize"` (19 - 59 students), or
- `"large"` (60 students or more).

Since you'd like to have three distinct levels (instead of just two), you will need a nested call to `ifelse()`, which means that you'll call `ifelse()` a second time from within your first call to `ifelse()`. We've provided some scaffolding for you in the editor—see if you can figure it out!

Instructions

Recode the `cls_students` variable into a new variable, `cls_type`, having the three levels described above. Save the resulting data frame (with the new variable) as `evals`.

What type of variable is `cls_type`?

```
# Recode cls_students as cls_type: evals
evals <- evals %>%
  # Create new variable
  mutate(cls_type = ifelse(cls_students <= 18, "small",
                      ifelse(cls_students >18 & cls_students <=59, "midsize", "large")))
```

Create a scatterplot

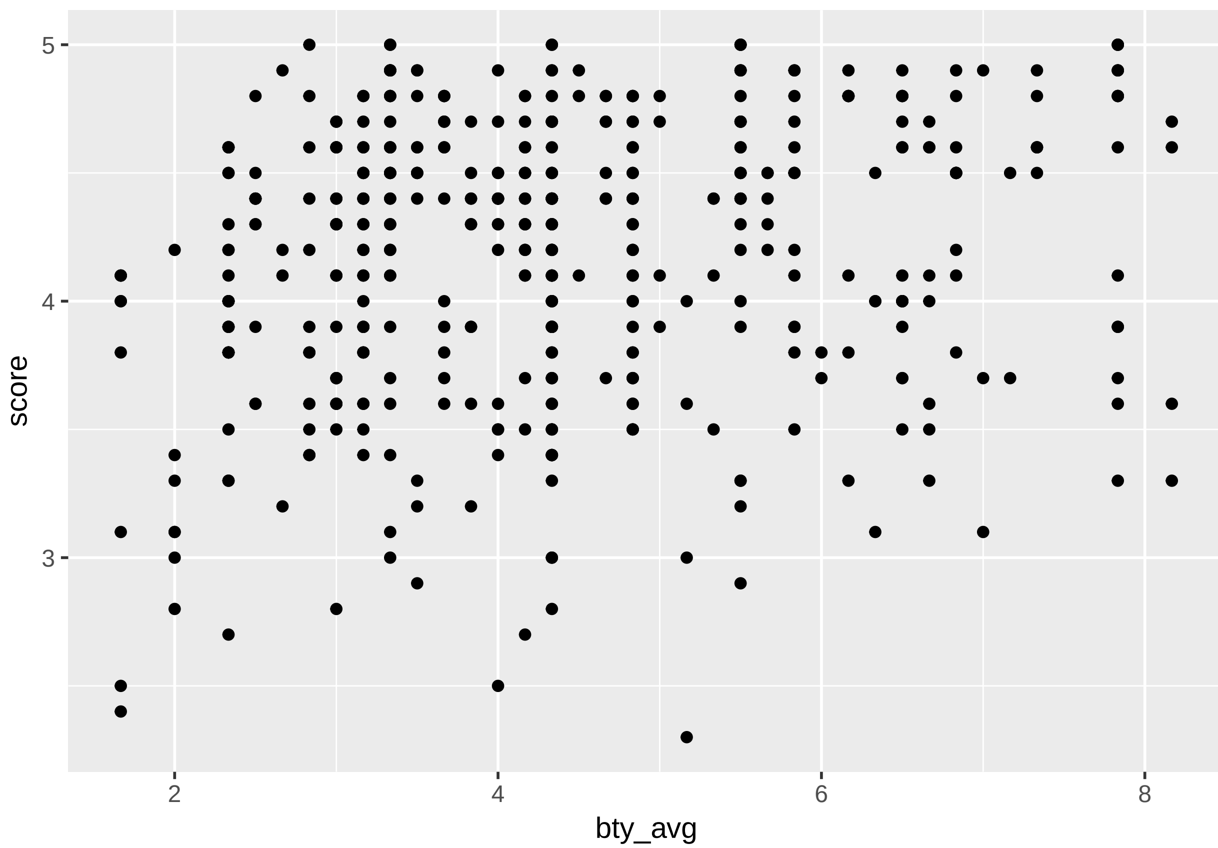
The `bty_avg` variable shows the average beauty rating of the professor by the six students who were asked to rate the attractiveness of these faculty. The `score` variable shows the average professor evaluation score, with 1 being *very unsatisfactory* and 5 being *excellent*.

Instructions

Use `ggplot()` to create a scatterplot displaying the relationship between these two variables.

How would you describe the relationship apparent in this visualization?

```
# Scatterplot of score vs. bty_avg
ggplot(evals, aes(x = bty_avg, y = score)) +
  geom_point()
```



Create a scatterplot, with an added layer

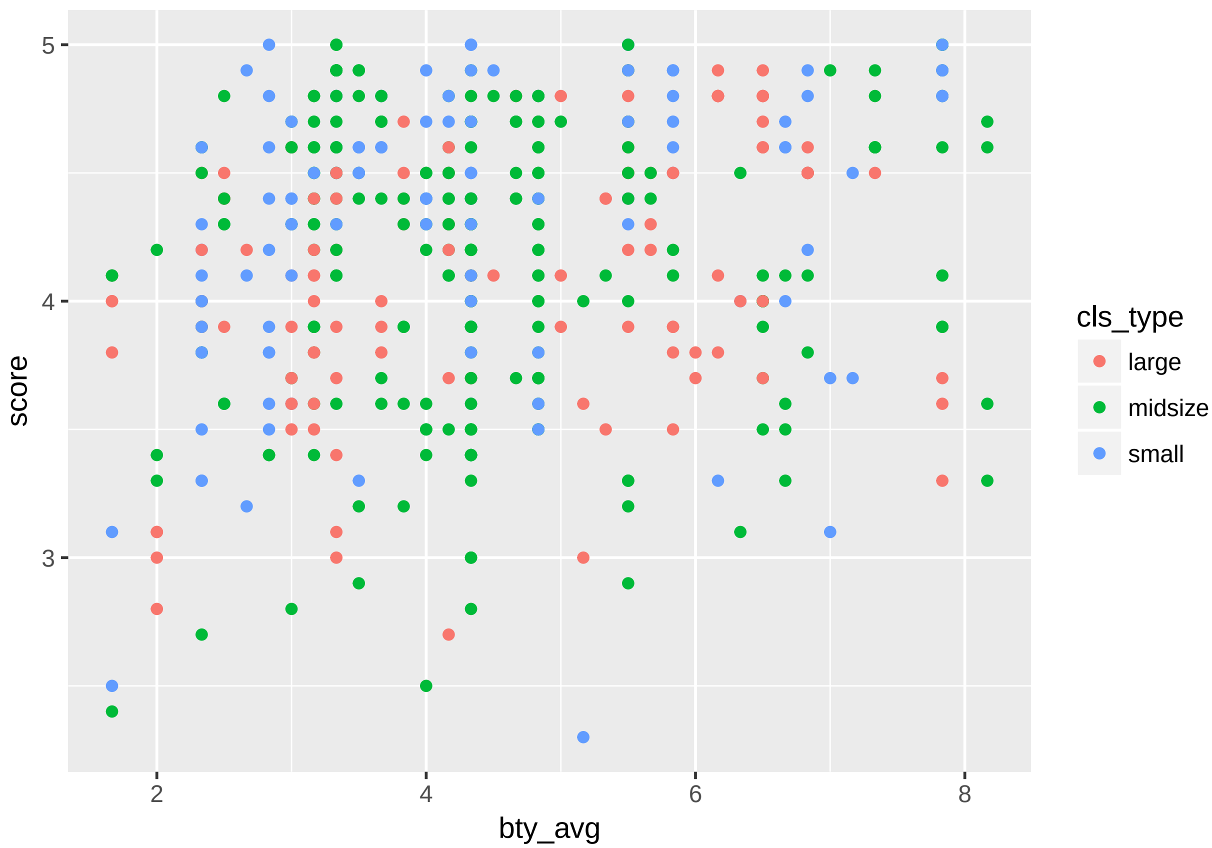
Suppose you are interested in evaluating how the relationship between a professor's attractiveness and their evaluation score varies across different class types (small, midsize, and large).

Instructions

Recreate your visualization from the previous exercise, but this time coloring the points by class type.

How would you describe the relationship apparent in this visualization?

```
# Scatterplot of score vs. bty_avg colored by cls_type
ggplot(evals, aes(x = bty_avg, y = score, color = cls_type)) +
  geom_point()
```


5
4
3
score
2
4
6
8
bty_avg
cls_type
large
midsize
small