**Loading data into R**

0xp

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).

# Load data

data(email50)

# View its structure

str(email50)

# Identify variable types

100xp

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)

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?

# Subset of emails with big numbers: email50\_big

email50\_big <- email50 %>%

filter(number == "big")

# Glimpse the subset

glimpse(email50\_big)

# Complete filtering based on a factor

100xp

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?

# Table of number variable

table(email50\_big$number)

# Drop levels

email50\_big$number <- droplevels(email50\_big$number)

# Another table of number variable

table(email50\_big$number)

# Table of number variable

table(email50\_big$number)

# Drop levels

email50\_big$number <- droplevels(email50\_big$number)

# Another table of number variable

table(email50\_big$number)

# Discretize a different variable

100xp

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.

# 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)

# Combining levels of a different factor

100xp

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

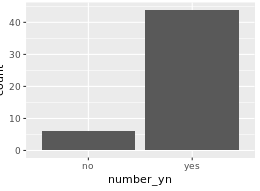
email50 <- email50 %>%

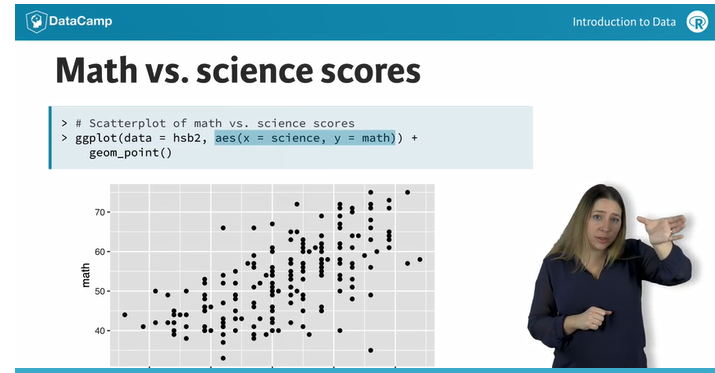
mutate(number\_yn = ifelse(number == "none", "no", "yes"))

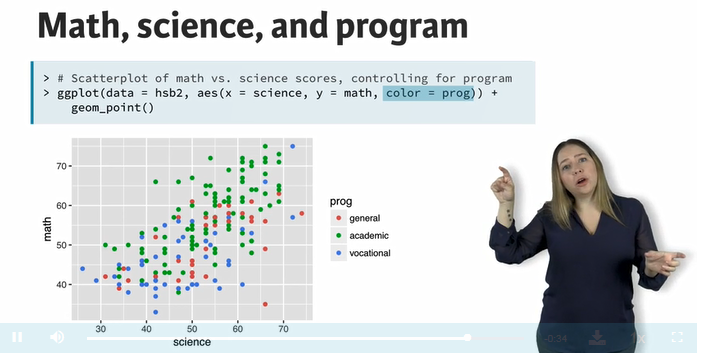
# Visualize number\_yn

ggplot(email50, aes(x = number\_yn)) +

geom\_bar()







# Visualizing numerical and categorical data

100xp

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?

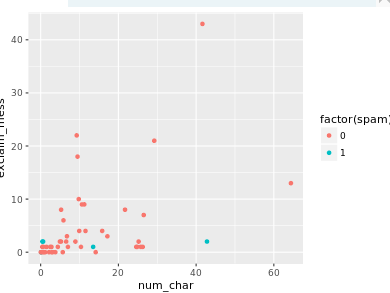
# Load ggplot2

library(ggplot2)

# Scatterplot of exclaim\_mess vs. num\_char

ggplot(email50, aes(x = num\_char , y = exclaim\_mess, color = factor(spam))) +

geom\_point()



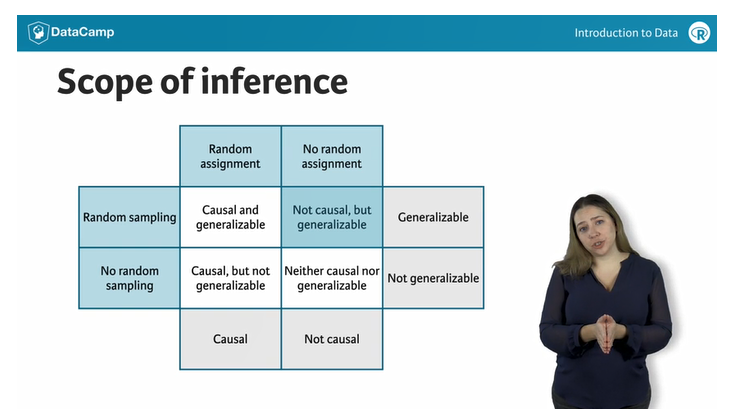
# Identify the type of study

0xp

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".



# Identify the scope of inference of study

50xp

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.

# Load data

data(gapminder)

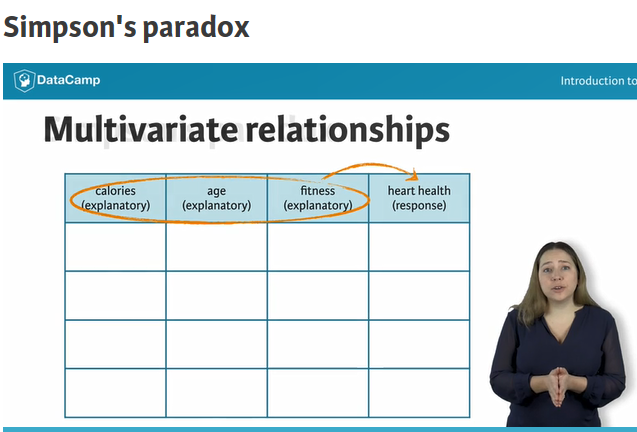
# Glimpse data

glimpse(gapminder)

# Identify type of study

type\_of\_study <- "observational"

Simpson paradox



# Number of males and females admitted

0xp

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()](https://www.rdocumentation.org/packages/dplyr/topics/tally) 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()](https://www.rdocumentation.org/packages/tidyr/topics/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).

# Load packages

library(dplyr)

library(tidyr)

# Count number of male and female applicants admitted

ucb\_counts <- ucb\_admit %>%

count(Gender, Admit)

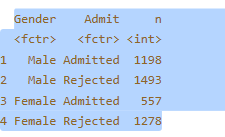
# View result

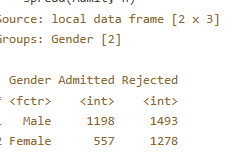
ucb\_counts

# Spread the output across columns

ucb\_counts %>%

spread(Admit, n)





# Proportion of males admitted overall

100xp

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

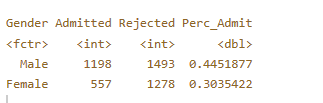
count(Gender, Admit) %>%

# Spread output across columns based on admission status

spread(Admit, n) %>%

# Create new variable

mutate(Perc\_Admit = Admitted / (Admitted + Rejected))



# Proportion of males admitted for each department

0xp

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)

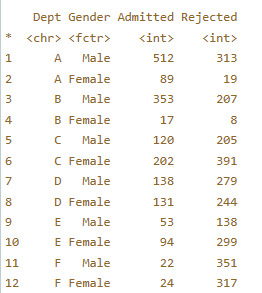
# View result

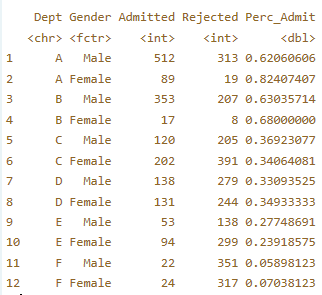
admit\_by\_dept

# Percentage of those admitted to each department

admit\_by\_dept %>%

mutate(Perc\_Admit = Admitted / (Admitted + Rejected))





# Simple random sample in R

0xp

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

# Simple random sample

states\_srs <- us\_regions %>%

sample\_n(8)

# Count states by region

states\_srs %>%

group\_by(region) %>%

count()

# Stratified sample in R

100xp

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

# Stratified sample

states\_str <- us\_regions %>%

group\_by(region) %>%

sample\_n(8)

# Count states by region

states\_str %>%

group\_by(region) %>%

count()

# Stratified sample

states\_str <- us\_regions %>%

group\_by(region) %>%

sample\_n(2)

# Count states by region

states\_str %>%

group\_by(region) %>%

count()

**Identifying components of a study**

50xp

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?

