Statistical Analysis Using R

1. **Factor**

Categorical data are often stored as factors in R.

Eg : -

Data Set - 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, 1, ...
$ to_multiple  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ...
$ from         <dbl> 1, 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, 0, ...
$ sent_email   <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, ...
$ time         <dttm> 2012-01-04 13:19:16, 2012-02-16 20:10:06, 2012-01-04 ...
$ image        <dbl> 0, 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, 1, ...
$ dollar       <dbl> 0, 0, 0, 0, 9, 0, 0, 0, 0, 23, 4, 0, 3, 2, 0, 0, 0, 0,...
$ winner       <fctr> no, no, no, no, no, no, no, no, no, no, no, no, yes, ...
$ inherit      <dbl> 0, 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, 0, ...
$ password     <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, ...
$ num_char     <dbl> 21.705, 7.011, 0.631, 2.454, 41.623, 0.057, 0.809, 5.2...
$ line_breaks  <int> 551, 183, 28, 61, 1088, 5, 17, 88, 242, 578, 1167, 198...
$ format       <dbl> 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, ...
$ re_subj      <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, ...
$ exclaim_subj <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 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, 0, ...
$ exclaim_mess <dbl> 8, 1, 2, 1, 43, 0, 0, 2, 22, 3, 13, 1, 2, 2, 21, 10, 0...
$ number       <fctr> small, big, none, small, small, small, small, small, ...
> email50_big <- email50 %>% filter(number == 'big')
```

```
> email50_big <- email50 %>% filter(number == 'big')
> 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 20:10:06, 2012-02-04 23:26:09, 2012-01-24 ...
$ 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
```

In this observation Seven emails contains big numbers.

** The filter function and pipeline (%>%) operator is used from dplyr package.

** The `droplevels()` function removes unused levels of factor variables from your dataset.

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

 none small   big 
    0     0     7 
>
> # Drop levels
> email50_big$number <- droplevels(email50_big$number)
>
> # Another table of number variable
> table(email50_big$number)

big 
  7 
```

** Dropping the levels of the number variable gets rid of the levels with counts of zero.

Discretize a variable

Discretizing is a common way of creating a new variable from an existing variable. This means converting a numerical variable to a categorical variable based on certain criteria.

Created 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.

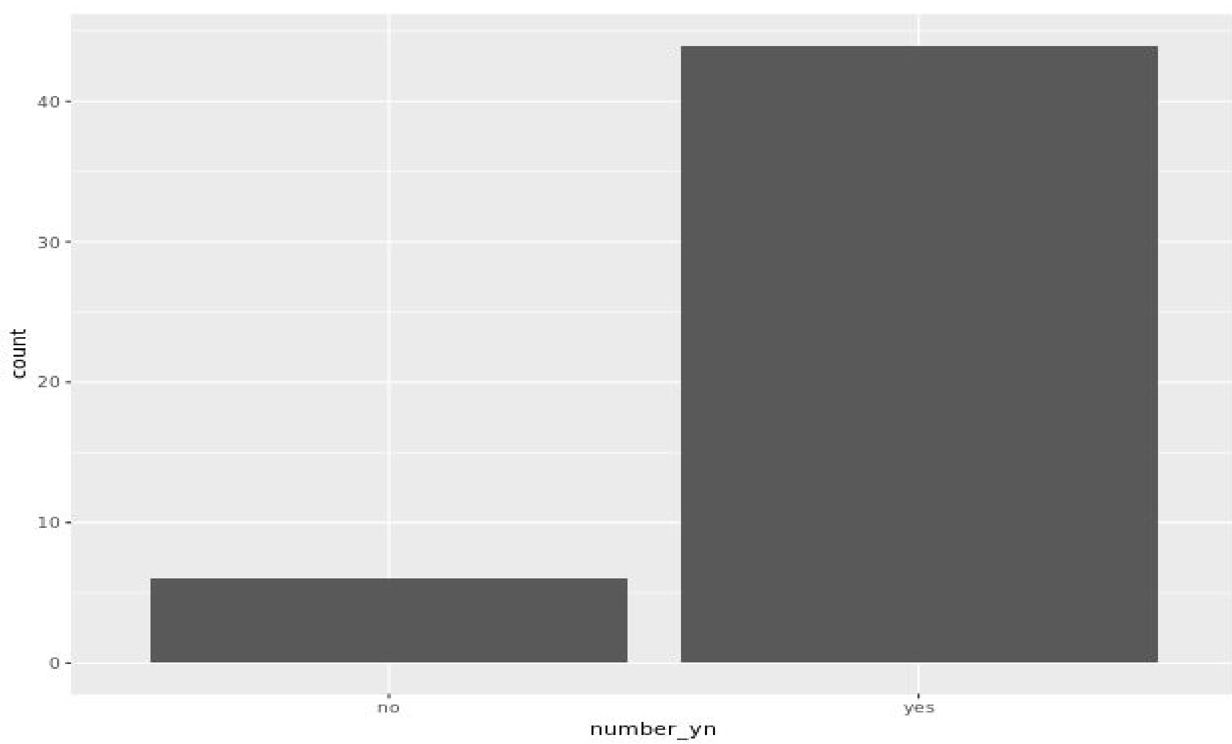
The median marks the 50th percentile, or midpoint, of a distribution, so half of the emails should fall in one category and the other half in the other.

```
> # 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.

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



Visualizing numerical data

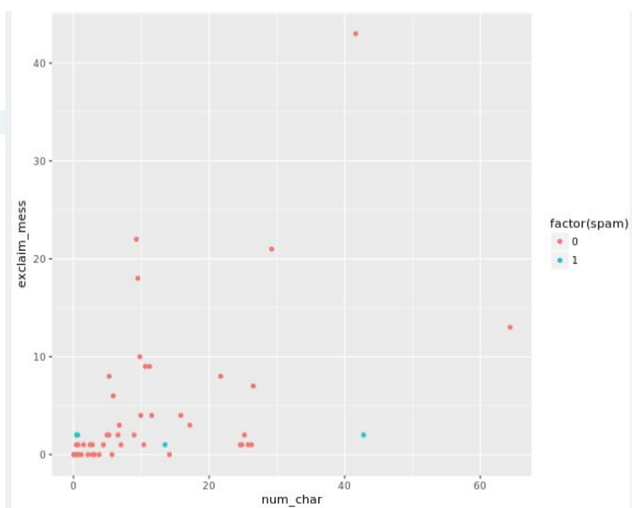
Visualize the relationship between two numerical variables from the `email50` dataset, conditioned on whether or not the email was spam. This means that we will use some aspect of the plot (like color or shape) to separate the groups in the `spam` column so that we can compare plotted values between 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 scatter plot by adding the `geom_point()` layer to your `ggplot()` call.

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?*

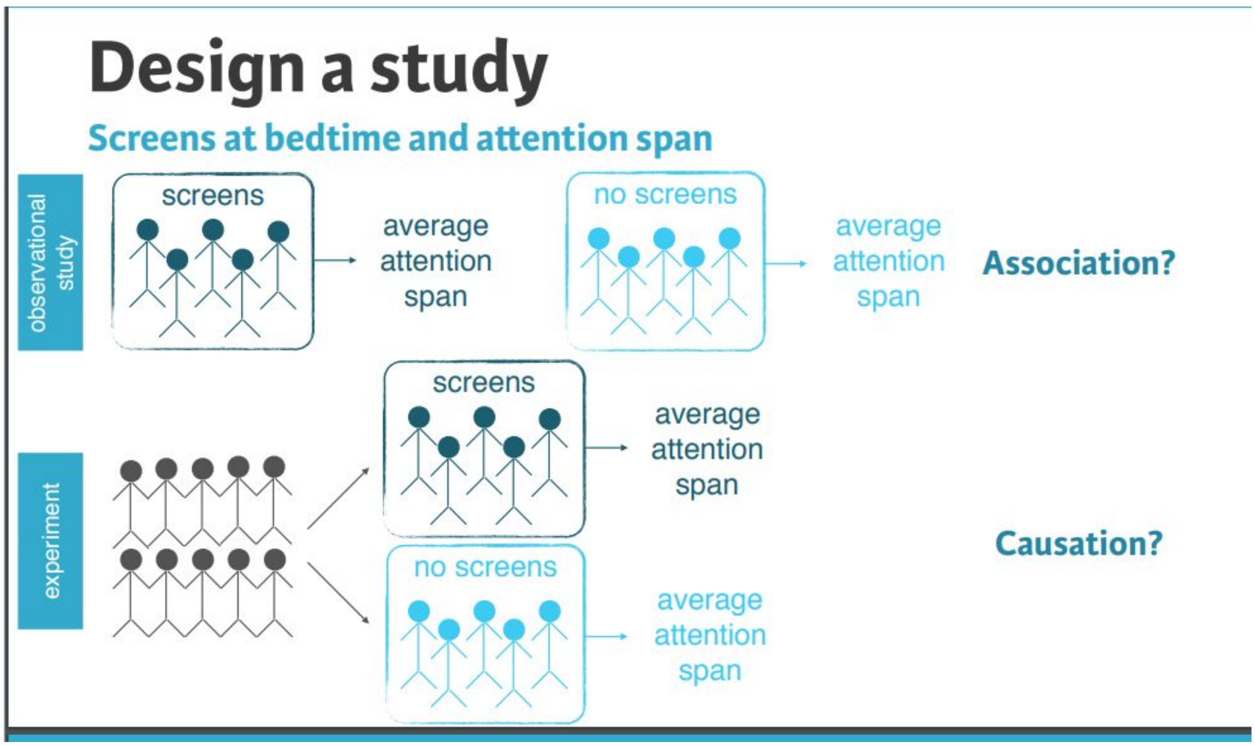
```
# Load ggplot2
library(ggplot2)

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



Observational studies and experiments

Types of Studies: -

1. **Observational Study**: - Collect data in a way that does not directly interfere with how the data arise. Only **correlation** can be inferred through this study.
2. **Experiment Study: -** Randomly assigns subject to various treatments. **Causation** can be inferred through this study.



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.

```
> # Load data
> data(gapminder)
>
> # Glimpse data
> glimpse(gapminder)
Observations: 1,704
Variables: 6
$ country   <fctr> Afghanistan, Afghanistan, Afghanistan, Afghanistan, Afgh...
$ continent <fctr> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, As...
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 199...
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 4...
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372,...
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.113...
>
```

Since there is no way to randomly assign countries to attributes, this is an observation study.

Random sampling and random assignment

Random Sampling:

Random Selection of subjects from population. Random Selection helps generalizability of results.

Random Assignment:

Assignment of subjects to various treatments. This helps to infer causation from results.

Scope of inference

	Random assignment	No random assignment	
Random sampling	Causal and generalizable	Not causal, but generalizable	Generalizable
No random sampling	Causal, but not generalizable	Neither causal nor generalizable	Not generalizable
	Causal	Not causal	

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?

Explanation - Random assignment is not employed because the conditions are not imposed on the patients by the people conducting the study; random sampling is not employed because the study records the patients who are already hospitalized, so it wouldn't be appropriate to apply the findings back to the population as a whole.

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 cannot be generalized to all people and a causal link between believing information is stored and memory can be inferred based on these results.

There is no random sampling since the subjects of the study were volunteers, so the results cannot be generalized to all people. However, due to random assignment, we are able to infer a causal link between the belief information is stored and the ability to recall that same information.ble to infer a causal link between the belief information is stored and the ability to recall that same information.cannot be generalized to all people. However, due to random assignment, we are able to infer a causal link between the belief information is stored and the ability to recall that same information.

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.

```
> library(tidyr)
>
> # Count number of male and female applicants admitted
> ucb_counts <- ucb_admit %>%
    count(Admit, Gender)
>
> # View result
> ucb_counts
Source: local data frame [4 x 3]
Groups: Admit [?]

      Admit Gender     n
     <fctr> <fctr> <int>
1 Admitted    Male  1198
2 Admitted  Female   557
3 Rejected    Male  1493
4 Rejected  Female  1278
>
> # Spread the output across columns
> ucb_counts %>%
    spread(Admit, n)
# A tibble: 2 × 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.

```
> ucb_admit %>% count(Admit,Gender) %>% spread(Admit,n) %>% 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
```

Fantastic! It looks like 44% of males were admitted versus only 30% of females.

Proportion of males admitted for each department

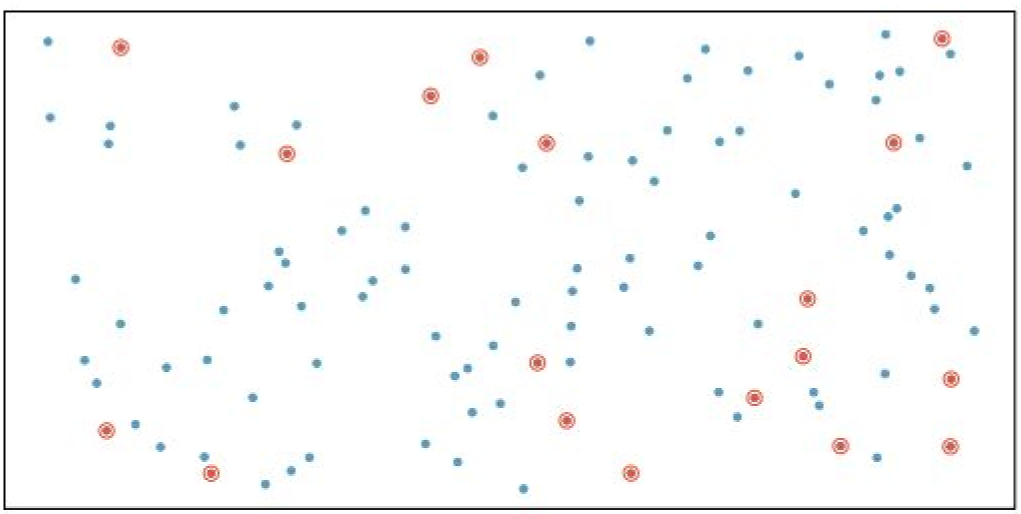
Next you'll make a table similar to the one you constructed earlier, except you will first group the data by department. Then, you'll use this table to calculate the proportion of males admitted in each department.

```
> admit_by_dept <- ucb_admit %>%
+   count(Dept, Admit, Gender) %>%
+   spread(Admit, n)
> admit_by_dept
Source: local data frame [12 x 4]
Groups: Dept [6]

    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
```

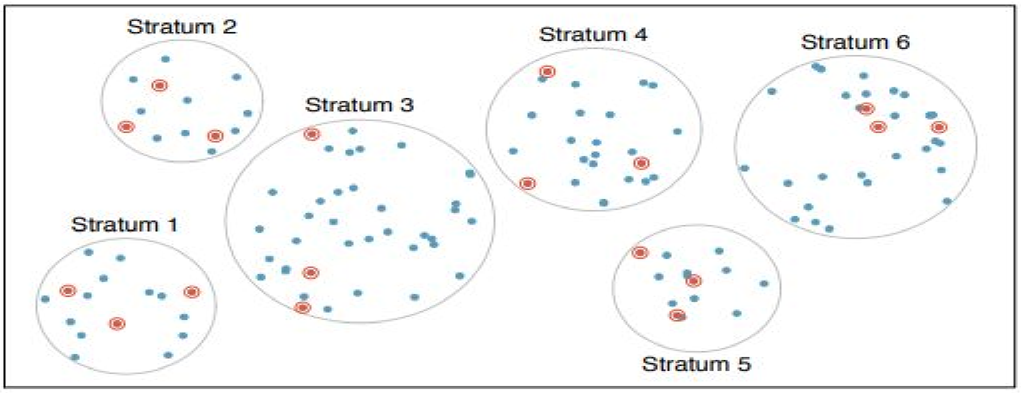
```
> admit_by_dept %>% mutate(Perc_Admit = Admitted / (Admitted + Rejected))
Source: local data frame [12 x 5]
Groups: Dept [6]

    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
```

Sampling strategies

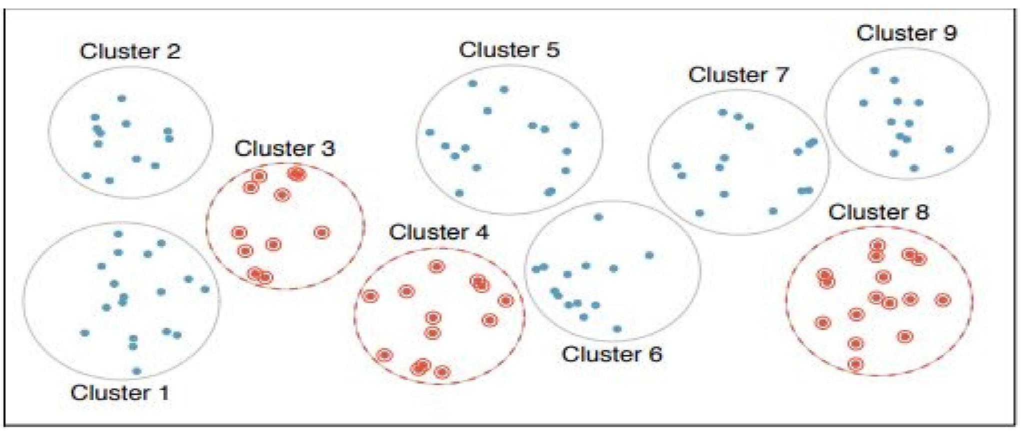
1. Simple random sample



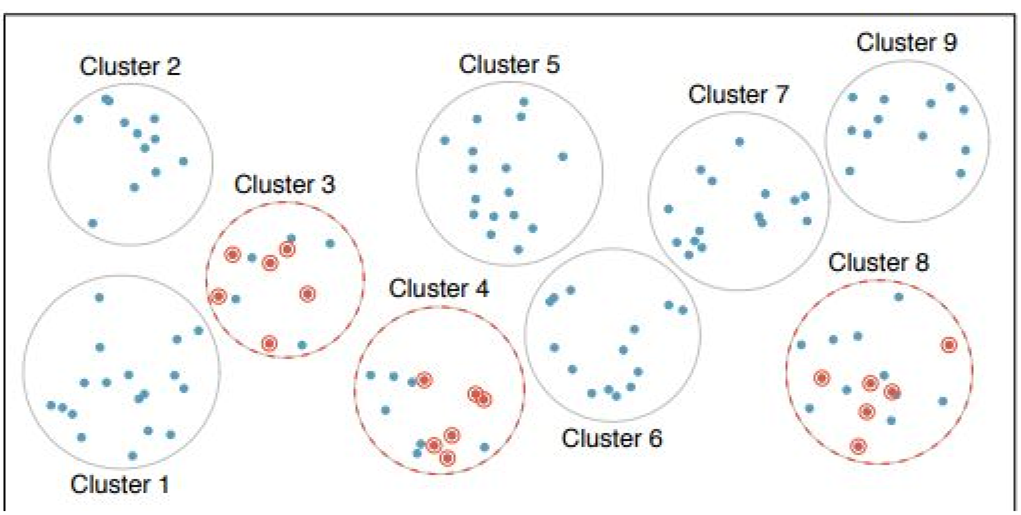
2. Stratified sample



3. Cluster sample



4. Multistage sample



Sampling in R

Simple random sample 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.

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

```
> # Simple random sample: states_srs
> states_srs <- us_regions %>%
    sample_n(size=8)
>
> # Count states by region
> states_srs %>%
    group_by(region) %>%
    count()
# A tibble: 3 × 2
     region     n
     <fctr> <int>
1   Midwest     2
2 Northeast     1
3     South     5
```

Notice that this strategy selects an unequal number of states from each region. In the next exercise, you'll implement stratified sampling to select an equal number of states from each region.

Stratified sample in R

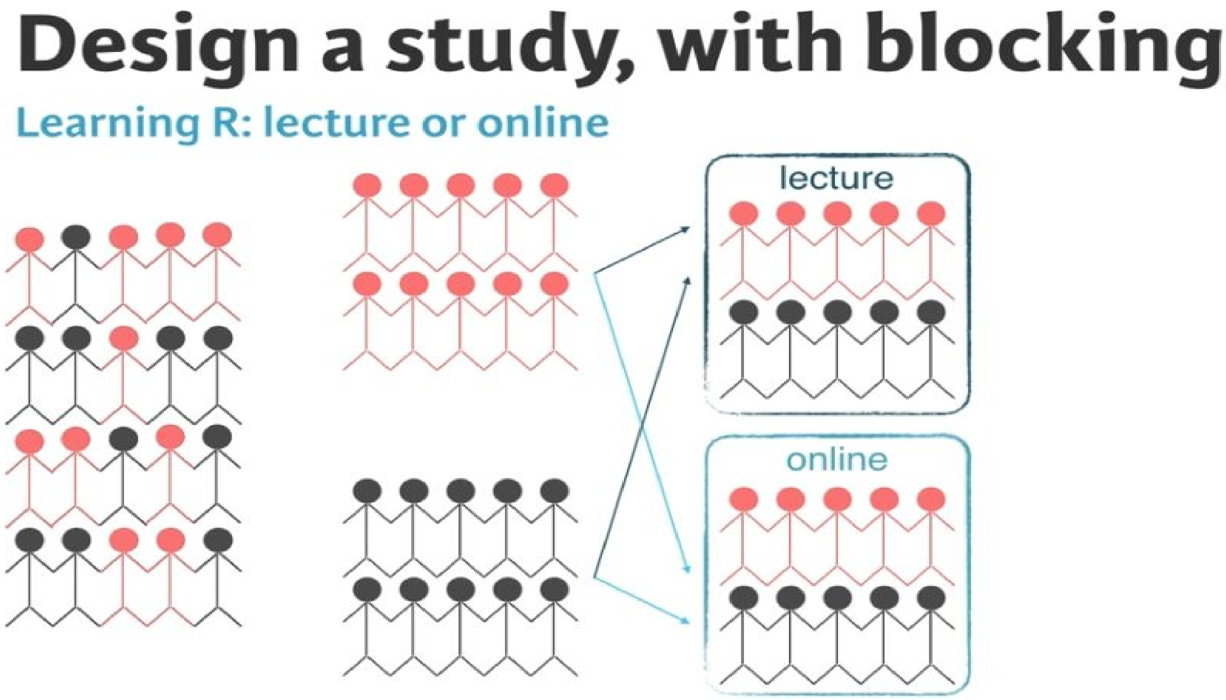
In the last exercise, you took a simple random sample of eight states. However, as you may have noticed when you counted the number of states selected from each region, this strategy is unlikely to select an equal number of states from each region. The goal of stratified sampling is to select an equal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

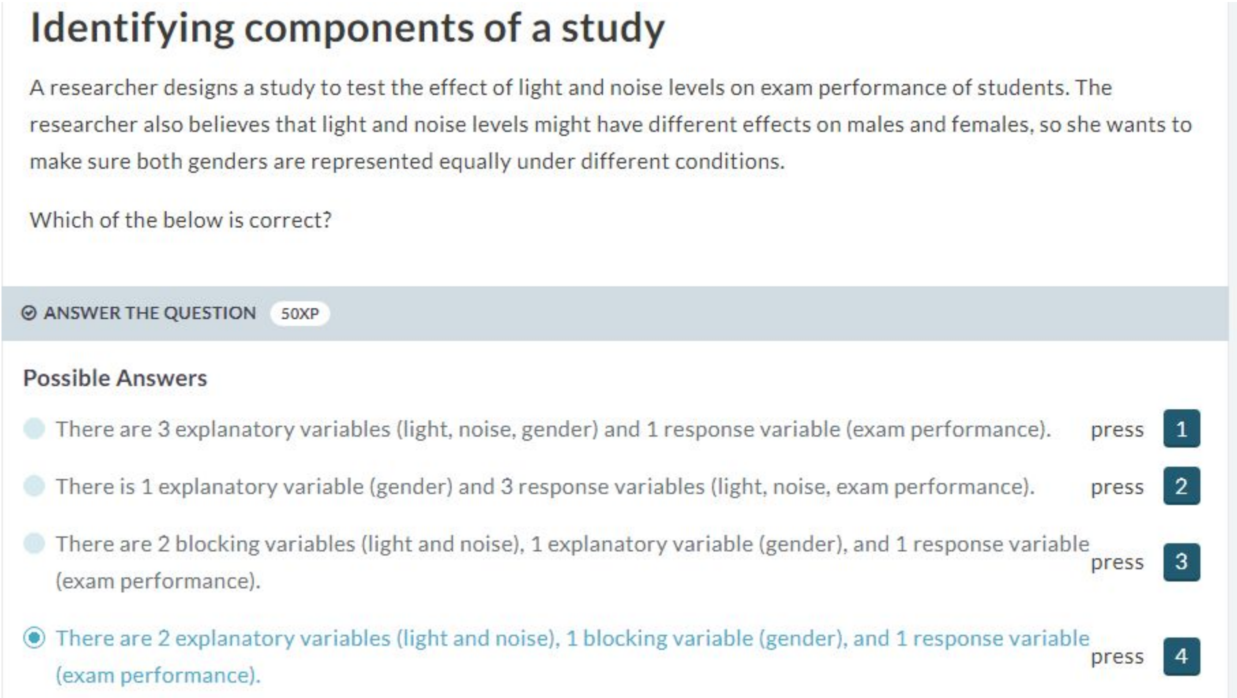
```
> # Stratified sample
> states_str <- us_regions %>%
    group_by(region) %>%
    sample_n(size = 2)
>
> # Count states by region
> states_str %>%
    group_by(region) %>%
    count()
# A tibble: 4 × 2
      region     n
      <fctr> <int>
1    Midwest     2
2  Northeast     2
3      South     2
4       West     2
```

In a stratified sample, each stratum (i.e. Region) is represented equally.

Principles of experimental design

1. **Control** - Compares treatment of interest to a control group.
2. **Randomize** - randomly assign subjects to treatments.
3. **Replicate** - Collect a sufficiently large sample within a study or replicate the entire study.
4. **Block** - Accounts for potential effect of confounding variables.
   Blocking can be achieved by
   First grouping subjects into blocks based on these variables.
   Randomize within each block to treatment groups.





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?

ANSWER THE QUESTION 50XP

Possible Answers

- There are 3 explanatory variables (light, noise, gender) and 1 response variable (exam performance). press 1
- There is 1 explanatory variable (gender) and 3 response variables (light, noise, exam performance). press 2
- There are 2 blocking variables (light and noise), 1 explanatory variable (gender), and 1 response variable (exam performance). press 3
- There are 2 explanatory variables (light and noise), 1 blocking variable (gender), and 1 response variable (exam performance). press 4

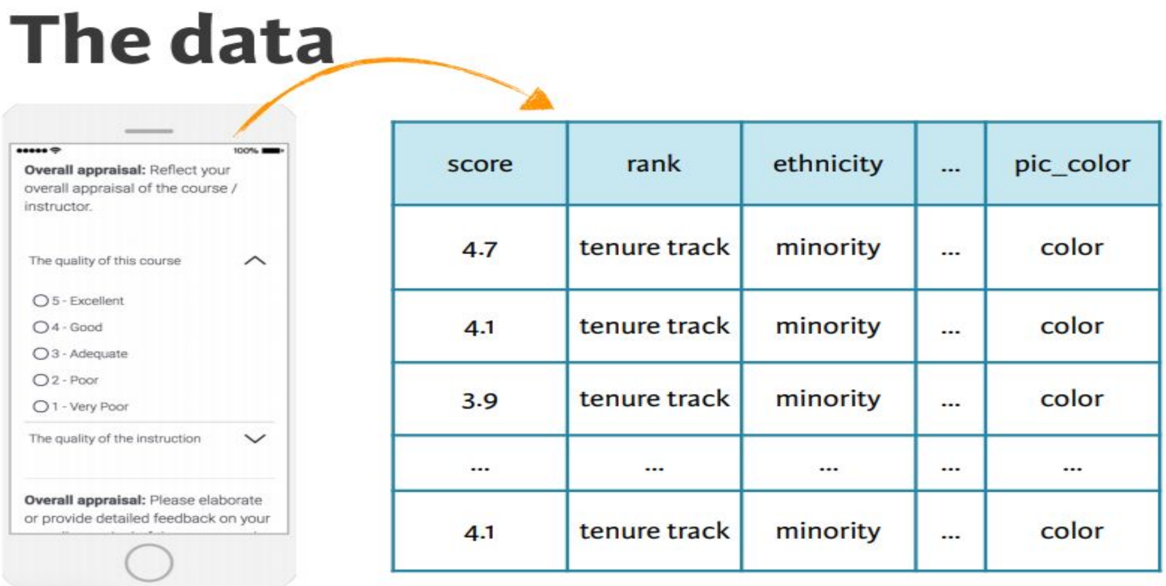
Explanatory variables are conditions you can impose on the experimental units, while **blocking** variables are characteristics that the experimental units come with that you would like to control for.

Connect blocking and stratifying

In random sampling, you use **stratifying** to control for a variable. In random assignment, you use **blocking** to achieve the same goal.

CASE STUDY

How the physical appearance of instructors impacts their students' course evaluations ?



Source: Hamermesh, Daniel S., and Amy Parker. "Beauty in the classroom: Instructors' pulchritude and putative pedagogical productivity." Economics of Education Review 24.4 (2005): 369-376.

```
> # Inspect evals
> glimpse(evals)
Observations: 463
Variables: 21
$ score         <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8...
$ rank          <fctr> tenure track, tenure track, tenure track, tenure tra...
$ ethnicity     <fctr> minority, minority, minority, minority, not minority...
$ gender        <fctr> female, female, female, female, male, male, male, ma...
$ language      <fctr> english, english, english, english, english, english...
$ age           <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 4...
$ cls_perc_eval <dbl> 55.81395, 68.80000, 60.80000, 62.60163, 85.00000, 87....
$ cls_did_eval  <int> 24, 86, 76, 77, 17, 35, 39, 55, 111, 40, 24, 24, 17, ...
$ cls_students  <int> 43, 125, 125, 123, 20, 40, 44, 55, 195, 46, 27, 25, 2...
$ cls_level     <fctr> upper, upper, upper, upper, upper, upper, upper, upp...
$ cls_profs     <fctr> single, single, single, single, multiple, multiple, ...
$ cls_credits   <fctr> multi credit, multi credit, multi credit, multi cred...
$ bty_f1lower   <int> 5, 5, 5, 5, 4, 4, 4, 5, 5, 2, 2, 2, 2, 2, 2, 2, 2, 7,...
$ bty_f1upper   <int> 7, 7, 7, 7, 4, 4, 4, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 9,...
$ bty_f2upper   <int> 6, 6, 6, 6, 2, 2, 2, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 9,...
$ bty_m1lower   <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 7,...
$ bty_m1upper   <int> 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 6,...
$ bty_m2upper   <int> 6, 6, 6, 6, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 6,...
$ bty_avg       <dbl> 5.000, 5.000, 5.000, 5.000, 3.000, 3.000, 3.000, 3.33...
$ pic_outfit    <fctr> not formal, not formal, not formal, not formal, not ...
$ pic_color     <fctr> color, color, color, color, color, color, color, col...
>
```

The Data Set consists of 463 observations and 21 variables.

What type of study this is ? - Observational Study

The Data for this study is collected by randomly sampling classes.

Identify variable types

It's always useful to start your exploration of a dataset by identifying variable types.

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()`.

```
# 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")))
```

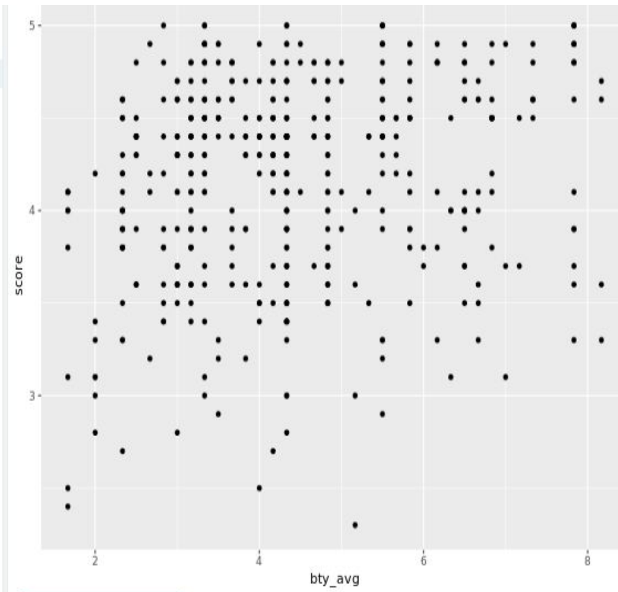
```
> str(evals)
Classes 'tbl_df', 'tbl' and 'data.frame':   463 obs. of  22 variables:
 $ score        : num  4.7 4.1 3.9 4.8 4.6 4.3 2.8 4.1 3.4 4.5 ...
 $ rank         : Factor w/ 3 levels "teaching","tenure track",..: 2 2 2 2 3 3 3 3 3 3 ...
 $ ethnicity    : Factor w/ 2 levels "minority","not minority": 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 ...
 $ cls_type     : chr  "midsize" "large" "large" "large" ...
```

Excellent! The cls_type variable is a categorical variable, stored as a character vector. You could have made it a factor variable by wrapping the nested ifelse() statements inside factor().

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*.

```
# 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).