Fast forward a few years, and now you're finishing up university. As your time in university comes to an end, your goal is now to build a career. As job seekers can attest, finding employment can be a grueling, unorganized process. In this mission, we'll learn how to use **lists** to organize the job-seeking process.

In the first four missions, we've learned about three different data structures:

**Vector**: A *one* dimensional data structure that can hold *one* type of data.

**Matrix**: A *two* dimensional data structure that can hold *one* type of data.

**Dataframe** A two dimensional data structure that can hold *multiple* types of data.

And the fourth most common data structure are **lists**:

A **list** is a one-dimensional data structure that can hold *multiple* types of R objects. A list behaves similar to a dataframe, in that it can hold multiple types of data. However, a list is not limited to single data types. A list can also hold vectors, matrices, dataframes, all in one structure. It is often good practice to store dataframes within lists. It is *not* good practice to store objects in dataframes.

In many scenarios, we've created multiple objects that we might want to access later. A list gives us a good method of storing all the objects we've created. In addition, lists give us more flexibility than a vector, since lists can hold multiple data types.

To create a list, we'll use the list() function. Similar to creating a vector, we'll pass any value we'd like in our list, through the list() function. Then, we'll assign the list a name:



companies <- list("axis\_tech","bio\_fuel.co","moon\_land.co")

Creating a list is similar to creating a vector. Since we're looking to organize the job-seeking process, let's store a few companies we're applying to in a list!

instructions

* Store the following company names in a list called companies:
  + "boom\_tech"
  + "car\_mechanics"
  + "Pineapple.co"
  + "slurp\_snacks"

We've created a list with a few company names of character type. However, we've only stored one data type, which we could use a vector to store. Let's utilize the added benefits of a list. When interviewing for a specific position, you might be doing a variety of research. You'd look up the job description, position title, multiple salary amounts etc.

After gathering all this information, we can store this all in a list, despite the information being of different data types. Note, data in this mission is fictitious:



salary <- c(75000,100000,110000)

​

job\_description <- c("Requires a college degree. 3 years of experience in data.")

​

job\_title <- c("data scientist")

If we tried to store this all in a vector:



vector <-c(salary, job\_description, job\_title)

If we tried to store these 3 objects in a single vector, the R interpreter will convert all the values into the character data type. Here's what vector would look like if we printed all the values:



75000' '1e+05' '110000' 'Requires a college degree. 3 years of experience in data.' 'data scientist

Instead, let's use a list:



list(salary, job\_description, job\_title)

This would display:



1. 75000 1e+05 110000

2. 'Requires a college degree. 3 years of experience in data.'

3. 'data scientist'

Storing all this information in a list, allows them all to maintain their original data types. In addition, notice, that the R interpreter automatically assigns a number to each element in the list. We'll dive deeper into this in a later screen.

Now that you've gathered information about a few jobs, let's store these values in a list!

instructions

* Store the following objects in a list called research:
  + "Data Analyst"
  + c(74000,60000,80000)
  + "Must have skills with R programming"

In the previous screen, we stored our job research inside a list. While all our information is conveniently in one location, imagine researching & storing 100+ positions. Storing our information as it currently is might be a bit messy. Let's organize our list, by *naming* the values in our list.

Naming a list is the same as naming a vector. We'll be using the names() function to access the names of the list. Then, we'll store the names of our values within names(list):



job\_1 <- list(c(75000,100000,110000), 'Requires a college degree. 3 years of experience in data.', 'data scientist')

​

names(job\_1) <- c("salary","job\_description","job\_title")

Now, let's print our job\_1 list to display our list:



$salary

[1] 75000 100000 110000

​

$job\_description

[1] "Requires a college degree. 3 years of experience in data."

​

$job\_title

[1] "data scientist"

We've named each object in our list a specific name. Well go over what the [1] means in a later screen.

By default, the names of a list behave similar to the vectors we created in earlier missions. If you call the names() function on our list, without naming it, it will return null:



job\_1 <- list(c(75000,100000,110000), 'Requires a college degree. 3 years of experience in data.', 'data scientist')

​

names(job\_1)

This returns:



NULL

Let's use what we know about names() to organize our information in our job research!

instructions

* Let's name our research list with the proper names:
  + "job\_title"
  + "salaries"
  + "job\_requirements"
* Store the names of research in the variable named\_research.

Now that we've named the values in our research, let's say we wanted to acquire more information on a specific company. During our research, we found that the company we're applying for also offers a four week vacation. To add this new information, we'll learn an alternative way to naming the values in our list. Then, we'll add the four week vacation data to our current list.

In the previous screen, we used names() to name the values in our list. Instead of using the names() function, we can actually set our name when creating our list. In the previous screen, we did the following:



job\_1 <- list(c(75000,100000,110000), 'Requires a college degree. 3 years of experience in data.', 'data scientist')

​

names(job\_1) <- c("salary","job\_requirements","job\_title")

Instead, we can use our original list and set the names using =:



job\_1 <- list(salary = c(75000,100000,110000),

job\_requirements = 'Requires a college degree. 3 years of experience in data.',

job\_title = 'data scientist')

This would display the same value if we called names() on job\_2:



names(job\_1)

This would display:

'salary' 'job\_requirements' 'job\_title'

Let's name our previous list of job information by using this alternative way!

instructions

* Name research list using the following names using the new method. This should not require any additional lines of code, just editing the current research:
  + "job\_title"
  + "salaries"
  + "job\_requirements"

Now that we've learned a new way of naming our research list, let's return to our conundrum of adding four weeks of vacation to our list. Adding a new value to a list, is similar to adding a new value to a vector. To add a new value, we'll use the c() function to store the original list and then add the new value at the end. Let's add "Vacation: 28" to the current list:



job\_1 <- list(salary = c(75000,100000,110000),

job\_requirements = 'Requires a college degree. 3 years of experience in data.',

job\_title = 'data scientist')

​

job\_1 <- c(job\_1, vacation = 28)

If we were to display the results of job\_1 using print():



$salary

[1] 75000 100000 110000

​

$job\_requirements

[1] "Requires a college degree. 3 years of experience in data."

​

$job\_title

[1] "data scientist"

​

$vacation

[1] 28

We *could* use the list() function instead of the c() function to add a new value. However, we're not looking to create a list of lists right now.

Now, let's add a vacation total to our current job research!

instructions

* Add the following value to the research list:

vacation = 21

So far, we've added more values to our jobs research and created a list holding all values. Since our list contains multiple values, there may be instances where we'd only want to look at the salaries, or vacations. To return a single value, we'll *index* into our list. Indexing into a list is similar to indexing into a vector. However, there are a few differences. Let's take our current job\_1 list:



job\_1 <- list(salary = c(75000,100000,110000),

job\_requirements = 'Requires a college degree. 3 years of experience in data.',

job\_title = 'data scientist', vacation = 28)

When we learned vectors, we used brackets [] to index into our vector. However, there are multiple ways to index into a list:

Notice how job\_1[1] *does not* return the same output as job\_1[[1]]. This is because, using a single bracket in job\_1[1] returns the *entire list object containing the value*. Using a double bracket job\_1[[1]] will return the single *component* of the list. job\_1$salary will also return a single component. To better understand the difference between a double bracket index [[]] and a single bracket index [], let's look at the output visually:

Let's index into our research list and return our salary!

instructions

* Index into the research list and return *only* the salaries. Do not include the name as well. Store this in salaries.

Now that we've returned a single value from our research list, we'll use indexing to change the values in our list. Let's say, that we noticed an error in our research. The salaries listed on the website, are much higher than the salaries the company can offer for that position. As a result, we'll have to update our research.

To update our research, we'll use the indexing we learned in the previous screen. Then we'll store a new value for that index. Let's take our job\_1 list and store a new set of salaries. First, we'll index into job\_1 where the salary is located. Then we'll re-store our new values:



job\_1[[1]] <- c(50000,60000,70000)

If we were to display our entire list, it would look like:



$salary

[1] 50000 60000 70000

​

$job\_requirements

[1] "Requires a college degree. 3 years of experience in data."

​

$job\_title

[1] "data scientist"

Using our research list, you also discover that the company has decided to rename this position as Jr. Data Scientist rather than Data Analyst. Make the necessary changes to the research list.

instructions

* Using the research list, change the job\_title from "Data Analyst" to "Jr. Data Scientist".

Throughout this mission, we've been using the vector job\_1 and research. To make our lives easier, let's merge these lists in one list, so we have one place where we can access the values. Merging lists into one list, is similar to adding a new value to a list. Instead of one value, we'll add an entire list.

Let's look at an example using job\_1 and additional\_data:



job\_1 <- list(salary = c(50000,60000,70000), job\_requirements = "Requires a college degree. 3 years of experience in data.", job\_title = "data scientist")

​

more\_data <- list(vacation = 28, retirement = "pension", bonus = 5000)

To combine these two lists, we'd perform the following:



combined <- c(job\_1,more\_data)

This would display:



$salary

[1] 50000 60000 70000

​

$job\_requirements

[1] "Requires a college degree. 3 years of experience in data."

​

$job\_title

[1] "data scientist"

​

$vacation

[1] 28

​

$retirement

[1] "pension"

​

$bonus

[1] 5000

Let's introduce another list of research information and combine this with our research list!

instructions

* Combine the list research with more\_research. Store this in combined.

Congratulations! You've completed the R Fundamentals Basics course. Throughout this course, we learned the most common data structures: vectors, matrices, dataframes & lists. In addition, we not only learned how to create these structures, we used this structures to figure out which college to attend, what to major in and to find a job! We used a number of R tools to guide you through your journey, from high school to employment.