Week 3 Module 5: Named Vectors

CSCI E-5a: Programming in R

Let's clear the global computing environment:

```
rm( list = ls() )
```

Let's load in the R objects for this module:

```
load( "Module 5 R Objects.Rdata" )

ls()
```

```
[1] "incorrect.values.cereal.sales.vector"
```

Module Overview and Learning Outcomes

Hello! And welcome to Module 5: Lookup Vectors.

In this module, we'll focus on the concept of lookup vectors.

- In Section 1, we'll learn what named vectors are, and how to construct them.
- In Section 2, we'll see how to use character string values to index a named vector.
- In Section 3, we'll work through a case study to see how we can summarize the results of a survey by using the concepts of named vectors and character string indexing.
- In Section 4, we'll use this method to repair invalid values in data.

When you've completed this module, you should be able to:

- Explain what a named vector is
- Create a named vector
- Use character string values to index a vector
- Use named vectors and character string indexing to summarize data from a survey
- Detect and repair invalid values in data

There are 2 new built-in R functions in this module:

- `str()`
- `unique()`

All right! Let's get started by defining the concept of a named vector.

Section 1: Named Vectors

Main Idea: *We can define the concept of a named vector*

In this section, we'll learn what named vectors are, and how to construct them.

When I first learned about named vectors and character value indexing, I didn't see how it was useful or practical.

Very few authors of R books devote much attention to named vectors, and there aren't a lot of great examples available.

Now I see them very useful in the right situation.

The key idea here is that the named vector allows us to convert a vector of character values into another type of vector.

In particular, we can use a named vector to convert a vector of character values into a vector of numeric values.

In this case, we're "looking up" a numeric value, given a character value.

This is a common task, and data structures that enable you to lookup values like this are often called "lookup tables" in the programming world.

However, the word "table" has a special meaning in R, so I don't want to use that term.

Instead, I will use the term "lookup vector", because we are using a vector to lookup a value.

So what is a "named vector"?

The fundamental concept of a *named vector* is that we can *name* the elements or components of a vector by using character string values as identifiers for the elements of that vector.

We can use the `c()` function to create a named vector by explicitly indicating the name for each value:

```
named.vector <-
    c(
        first = 3,
        second = 1,
        third = 5
    )
```

Let's directly display this vector:

```
named.vector
```

```
 first second  third 
     3      1      5 
```

Notice that this displays a little differently from the numeric vectors we've seen previously.

The `str()` function reports the structure of an R object.

Let's call the `str()` function using `named.vector` as the input argument:

```
str( named.vector )
```

```
 Named num [1:3] 3 1 5
 - attr(*, "names")= chr [1:3] "first" "second" "third"
```

You can see that the names are an "attribute" of the vector.

The names of a vector are stored as a character vector.

We can obtain a character vector consisting of the names of `named.vector` by using the `names()` function:

```
names( named.vector )
```

```
[1] "first"  "second" "third"
```

In fact, another way to create a named vector is to first create the vector itself, and then assign a character vector containing the names by using the `names()` function:

```
another.named.vector <-
    c( 4, -7, 6 )

names( another.named.vector ) <-
    c( "Red", "Green", "Blue" )
```

Let's directly display this vector:

```
another.named.vector
```

```
  Red Green  Blue 
    4    -7     6 
```

When we look at the structure of this named vector, we can see that it has exactly the same form as before:

```
str( another.named.vector )
```

```
 Named num [1:3] 4 -7 6
 - attr(*, "names")= chr [1:3] "Red" "Green" "Blue"
```

Sometimes the names can be annoying and it can be desirable to eliminate them.

To get rid of the names, assign the value `NULL`.

```
names( named.vector ) <- NULL
```

Notice that when we directly display this vector, it now looks different:

```
named.vector
```

```
[1] 3 1 5
```

Also, when we examine the structure of this vector we no longer see a `names` attribute:

```
str( named.vector )
```

```
 num [1:3] 3 1 5
```

Finally, note that we can actually use any character string we want for the name, even character strings that violate the rules for naming variables:

```
c(
    "The First One" = 3,
    "7482" = 1,
    "*^#)" = 7
)
```

```
The First One          7482          *^#)
            3             1             7
```

So that's how to define the concept of a named vector.

Now let's explore how to use character strings to index a named vector.

Exercise 5.1: Creating a named vector

A grocery store sells three brands of breakfast cereal:

- Sugar Bomz (SBZ), which sells for $2.99 per box.
- Krispee Yummm (KYM), which sells for $3.49 per box.
- Healthy Kale and Tofu (HKT), which sells for $7.99 per box.

Let's make a table of these brands and their price:

Brand	Price
SBZ	2.99
KYM	3.49
HKT	7.99

Create a named vector that associates brand names with prices.

Solution

```
# Type your answer here
```

Section 2: Indexing with Character Values

Main Idea: *We can use character values to index a named vector*

In this section, we'll see how to use character values to index a named vector.

The power of named vectors arises from the fact that we can use character values to index named vectors.

Let's remake our `named.vector`:

```
named.vector <-
    c(
        first = 3,
        second = 1,
        third = 5
    )

named.vector
```

```
 first second  third
     3      1      5
```

We can index this vector by using a character string:

```
named.vector[ "first" ]
```

```
first
    3
```

Even though this value has a name, it still behaves just like a regular numeric value:

```
4 * named.vector[ "first" ] + 7
```

```
first
   19
```

We can also use a vector of character string values to index `named.vector`:

```
character.indexing.vector <-
    c( "first", "third", "first",
       "second", "third" )

named.vector[ character.indexing.vector ]
```

```
 first  third  first second  third
     3      5      3      1      5
```

Notice what we've done here: we've taken a vector of character strings (`character.indexing.vector`) and converted it into a numeric vector.

This is what I meant at the beginning of Section 1 when I said:

> In particular, we can use a named vector to convert a vector of character values into a vector of numeric values.

So that's how to use character values to index a named vector.

Now let's see how to use named vectors and character string indexing to summarize data from a survey.

Exercise 5.2: Character Indexing

Suppose the grocery store sells this sequence of breakfast cereals:

- One box of Sugar Bomz (SBZ).
- One box of Krispee Yummm (KYM).
- One box of Sugar Bomz (SBZ).
- One box of Healthy Kale and Tofu (HKT).
- One box of Sugar Bomz (SBZ).

Create a vector of character strings to represent this sequence of sales, and then convert it to a numeric vector of the price per box of the brand by constructing a lookup vector.

Solution

Section 3: Case Study: Summarizing Survey Data

Main Idea: *We can use named vectors and character string indexing to summarize data from a survey*

In this section, we'll work through a case study to see how we can summarize the results of a survey by using the concepts of named vectors and character string indexing.

So far, we've seen some simple examples of how to convert data in character string form to numeric values.

As we've seen, the resulting vector is a numeric vector, and behaves exactly like other numeric vectors when we perform arithmetic operations on it.

Then we can use summary functions such as `sum()` or `mean()` to summarize the data.

Let's consider a practical example of this technique.

Suppose a restaurant asks their customers, "How would you rate your meal here today?", and they use a multiple-choice format where the answers are "Very Good", "Good", "OK", "Bad", and "Very Bad".

The responses are then converted into a numeric score:

Response	Numeric Score
Very Good	4
Good	3
OK	2
Bad	1
Very Bad	0

The restaurant then wants to calculate the average customer satisfaction rating.

Now suppose the restaurant gets these responses:

> Good, Very Good, Good, OK, Very Good, Bad, Very Good

How can we convert these character values into a numeric vector consisting of the numeric scores in the table?

We can create a named lookup vector:

```r
response.lookup.vector <-
    c(
        "Very Good" = 4,
        "Good" = 3,
        "OK" = 2,
        "Bad" = 1,
        "Very Bad" = 0
    )
```

Let's directly display this lookup vector:

```r
response.lookup.vector
```

```
Very Good      Good        OK       Bad  Very Bad
        4         3         2         1         0
```

Now we can represent the responses using a character vector:

```
response.vector <-
    c( "Good", "Very Good", "Good", "OK",
       "Very Good", "Bad", "Very Good" )
```

Then we can use the response vector to index the lookup vector:

```
response.lookup.vector[
    response.vector
]
```

```
     Good Very Good      Good        OK Very Good       Bad Very Good 
        3         4         3         2         4         1         4 
```

Now we can summarize these values using the sample mean:

```
response.score.vector <-
    response.lookup.vector[
        response.vector
    ]

mean.response.score <-
    mean( response.score.vector )

cat(
    "Mean response score:",
    formatC(
        mean.response.score,
        format = "f",
        digits = 2
    )
)
```

```
Mean response score: 3.00
```

So that's how we can summarize the results of a survey by using the concepts of named vectors and character string indexing.

Now let's see how to use named vectors and character string indexing to repair invalid values in data.

Exercise 5.3: Grocery Store

Let's return to our grocery store example.

Recall that these were the prices per box for each brand of breakfast cereal:

Brand	Price
SBZ	2.99
KYM	3.49
HKT	7.99

Also, recall that we had this sequence of sales:

- One box of Sugar Bomz (SBZ).
- One box of Krispee Yummm (KYM).
- One box of Sugar Bomz (SBZ).
- One box of Healthy Kale and Tofu (HKT).
- One box of Sugar Bomz (SBZ).

Calculate the total sales amount for the day by looking up the price for each box and then adding up the values in the vector using the `sum()` function.

Solution

Section 4: Detecting and Repairing Invalid Values

Main Idea: *We can use named vectors and character string indexing to repair invalid values in data*

In this section, we'll use named vectors and character string indexing to repair invalid labels in data.

So far, we've focused on lookup vectors that convert character string values to numeric values.

We can also use lookup vectors to convert character string values to other character string values.

This can be very useful when we want to repair invalid values.

Let's return to our example of the customer satisfaction survey.

Suppose we received the data in this form:

```
incorrect.values.response.vector <-
    c( "Good", "very good", "Good", "OK",
       "very good", "Bad", "very good" )
```

Notice that the values that represent a "Very Good" response are encoded with lower-case letters in the form "very good".

When we attempt to lookup the numeric values for this data, no name will exactly match with "very good", and we'll get a `NA` response:

```
response.lookup.vector[
    incorrect.values.response.vector
]
```

```
Good <NA> Good   OK <NA>  Bad <NA>
   3   NA    3    2   NA    1   NA
```

We'll explore this `NA` value later, in Week 5, but for right now I hope you can agree that this is undesirable.

The problem here is that we are expecting to see "Very Good" and instead the data has "very good".

We can use the `unique()` function to create a new vector consisting of all the different values in a given vector, but with each value displayed only once:

```
unique( incorrect.values.response.vector )
```

```
[1] "Good"      "very good" "OK"        "Bad"
```

If we have a large amount of data, the unique function can help us to quickly determine which values need to be repaired.

Let's construct a lookup vector to convert "very good" to "Very Good":

```
repair.bad.value.lookup.vector <-
  c(
    "Good" = "Good",
    "very good" = "Very Good",    # Here's where we fix the bad value
    "OK" = "OK",
    "Bad" = "Bad",
    "Very Bad" = "Very Bad"
  )
```

Now we can repair the bad values with this lookup vector:

```
repaired.response.vector <-
  repair.bad.value.lookup.vector[
    incorrect.values.response.vector
  ]

names( repaired.response.vector ) <- NULL

repaired.response.vector
```

```
[1] "Good"      "Very Good" "Good"      "OK"        "Very Good" "Bad"
[7] "Very Good"
```

Now we can convert to numeric values without any problems:

```
response.score.vector <-
    response.lookup.vector[
        repaired.response.vector
    ]

response.score.vector
```

```
     Good Very Good      Good        OK Very Good       Bad Very Good
        3         4         3         2         4         1         4
```

The vector is now ready for conversion to numeric values and averaging:

```
response.score.vector <-
    response.lookup.vector[
        repaired.response.vector
    ]

mean.response.score <-
    mean( response.score.vector )
```

```r
cat(
    "Mean response score:",
    formatC(
        mean.response.score,
        format = "f",
        digits = 2
    )
)
```

```
Mean response score: 3.00
```

So that's how to use named vectors and character string indexing to repair invalid labels in data.

Now let's review what we've learned in this module.

Exercise 5.4: Grocery Store

The R object `incorrect.values.cereal.sales.vector` contains data on cereal sales.

As with the previous exercise, your goal is to calculate the total sales amount for this data.

First, determine the number of entries in this vector, and convince yourself that it would be a lot of work to examine each entry individually.

Then use the `unique()` function to see all the different labels in this data.

Construct a lookup vector to repair any incorrect entries, and then use the vector of repaired values to calculate the total sales amount.

Solution

Module Review

In this module, we focused on the concept of lookup vectors.

- In Section 1, we learned what named vectors are, and how to construct them.
- In Section 2, we saw how to use character string values to index a named vector.
- In Section 3, we worked through a case study to see how we can summarize the results of a survey by using the concepts of named vectors and character string indexing.
- In Section 4, we used this method to repair invalid values in data.

Now that you've completed this module, you should be able to:

- Explain what a named vector is
- Create a named vector
- Use character string values to index a vector
- Use named vectors and character string indexing to summarize data from a survey
- Detect and repair invalid values in data

There were 2 new built-in R functions in this module:

- `str()`
- `unique()`

All right! That's it for Module 5: Lookup Vectors.

Now let's move on to Module 6: Stripcharts.

Solutions to the Exercises

Exercise 5.1: Creating a named vector

A grocery store sells three brands of breakfast cereal:

- Sugar Bomz (SBZ), which sells for $2.99 per box.
- Krispee Yummm (KYM), which sells for $3.49 per box.
- Healthy Kale and Tofu (HKT), which sells for $7.99 per box.

Let's make a table of these brands and their price:

Brand	Price
SBZ	2.99
KYM	3.49
HKT	7.99

Create a named vector that associates brand names with prices.

Solution

```
breakfast.cereal.price.lookup.vector <-
    c(
        "SBZ" = 2.99,
        "KYM" = 3.49,
        "HKT" = 7.99
    )
```

Now let's display this vector:

```
breakfast.cereal.price.lookup.vector
```

Exercise 5.2: Character Indexing

Suppose the grocery store sells this sequence of breakfast cereals:

- One box of Sugar Bomz (SBZ).
- One box of Krispee Yummm (KYM).

- One box of Sugar Bomz (SBZ).
- One box of Healthy Kale and Tofu (HKT).
- One box of Sugar Bomz (SBZ).

Create a vector of character strings to represent this sequence of sales, and then convert to a numeric vector of the price per box of the brand.

Solution

Let's start by creating the vector representing the sales, using a character string format:

```
breakfast.cereal.sales.vector <-
    c(
        "SBZ", "KYM", "SBZ", "HKT", "SBZ"
    )
```

Now let's convert that to a vector of prices per box:

```
breakfast.cereal.prices.vector <-
    breakfast.cereal.price.lookup.vector[
        breakfast.cereal.sales.vector
    ]

breakfast.cereal.prices.vector
```

Exercise 5.3: Grocery Store

Let's return to our grocery store example.

Recall that these were the prices per box for each brand of breakfast cereal:

Brand	Price
SBZ	2.99
KYM	3.49
HKT	7.99

Also, recall that we had this sequence of sales:

- One box of Sugar Bomz (SBZ).
- One box of Krispee Yummm (KYM).
- One box of Sugar Bomz (SBZ).
- One box of Healthy Kale and Tofu (HKT).
- One box of Sugar Bomz (SBZ).

Calculate the total sales amount for the day by looking up the price for each box and then adding up the values in the vector using the `sum()` function.

Solution

First, we'll lookup the prices (we did this in the previous problem) and then we'll add these using the `sum()` function to obtain the total sales amount:

```r
breakfast.cereal.prices.vector <-
    breakfast.cereal.price.lookup.vector[
        breakfast.cereal.sales.vector
    ]

total.breakfast.cereal.sales <-
    sum( breakfast.cereal.prices.vector )

cat(
    "Total sales:",
    formatC(
        total.breakfast.cereal.sales,
        format = "f",
        digits = 2
    )
)
```

Exercise 5.4: Grocery Store

The R object `incorrect.values.cereal.sales.vector` contains data on cereal sales, but it has some incorrect values.

As with the previous exercise, your goal is to calculate the total sales amount for this data.

First, determine the number of entries in this vector, and convince yourself that it would be a lot of work to examine each entry individually.

Then use the `unique()` function to see all the different labels in this data.

Construct a lookup vector to repair any incorrect entries, and then use the vector of repaired values to calculate the total sales amount.

Solution

Let's see how many items are in this vector:

```r
length( incorrect.values.cereal.sales.vector )
```

How many unique values are there in this vector?

```r
unique( incorrect.values.cereal.sales.vector )
```

Now we'll create the lookup vector to repair the incorrect values:

```r
repair.cereal.sales.lookup.vector <-
  c(
    "SBZ" = "SBZ",
    "KYM" = "KYM",
    "HKT" = "HKT",
    "sbz" = "SBZ",
    "kym" = "KYM",
    "hkt" = "HKT"
  )
```

Now we can repair the values in this vector:

```r
repaired.cereal.sales.vector <-
  repair.cereal.sales.lookup.vector[
    incorrect.values.cereal.sales.vector
  ]
```

We can convert the repaired values to numeric values:

```r
cereal.sales.numeric.vector <-
  breakfast.cereal.price.lookup.vector[
    repaired.cereal.sales.vector
  ]
```

Finally, we can add everything up to obtain the total sales amount:

```r
total.cereal.sales.amount <-
  sum( cereal.sales.numeric.vector )

cat(
  "Total cereal sales:",
  formatC(
    total.cereal.sales.amount,
    format = "f",
    digits = 2,
    big.mark = ","
  )
)
```