Chapter 34: Set operations

Section 34.1: Set operators for pairs of vectors

Comparing sets

In R, a vector may contain duplicated elements:

```
v = "A"
w = c("A", "A")
```

However, a set contains only one copy of each element. R treats a vector like a set by taking only its distinct elements, so the two vectors above are regarded as the same:

```
setequal(v, w)
# TRUE
```

Combining sets

The key functions have natural names:

```
x = c(1, 2, 3)
y = c(2, 4)

union(x, y)
# 1 2 3 4

intersect(x, y)
# 2

setdiff(x, y)
# 1 3
```

These are all documented on the same page, `?union`.

Section 34.2: Cartesian or "cross" products of vectors

To find every vector of the form (x, y) where x is drawn from vector X and y from Y, we use `expand.grid`:

```
X = c(1, 1, 2)
Y = c(4, 5)

expand.grid(X, Y)

#   Var1 Var2
# 1    1    4
# 2    1    4
# 3    2    4
# 4    1    5
# 5    1    5
# 6    2    5
```

The result is a data.frame with one column for each vector passed to it. Often, we want to take the Cartesian product of sets rather than to expand a "grid" of vectors. We can use `unique`, `lapply` and `do.call`:

```
m = do.call(expand.grid, lapply(list(X, Y), unique))
```

```
#   Var1 Var2
# 1    1    4
# 2    2    4
# 3    1    5
# 4    2    5
```

Applying functions to combinations

If you then want to apply a function to each resulting combination `f(x,y)`, it can be added as another column:

```
m$p = with(m, Var1*Var2)
#   Var1 Var2  p
# 1    1    4  4
# 2    2    4  8
# 3    1    5  5
# 4    2    5 10
```

This approach works for as many vectors as we need, but in the special case of two, it is sometimes a better fit to have the result in a matrix, which can be achieved with `outer`:

```
uX = unique(X)
uY = unique(Y)

outer(setNames(uX, uX), setNames(uY, uY), `*`)

#   4  5
# 1 4  5
# 2 8 10
```

For related concepts and tools, see the combinatorics topic.

Section 34.3: Set membership for vectors

The `%in%` operator compares a vector with a set.

```
v = "A"
w = c("A", "A")

w %in% v
# TRUE TRUE

v %in% w
# TRUE
```

Each element on the left is treated individually and tested for membership in the set associated with the vector on the right (consisting of all its distinct elements).

Unlike equality tests, `%in%` always returns `TRUE` or `FALSE`:

```
c(1, NA) %in% c(1, 2, 3, 4)
# TRUE FALSE
```

The documentation is at `?`%in%``.

Section 34.4: Make unique / drop duplicates / select distinct

elements from a vector

`unique` drops duplicates so that each element in the result is unique (only appears once):

```
x = c(2, 1, 1, 2, 1)

unique(x)
# 2 1
```

Values are returned in the order they first appeared.

`duplicated` tags each duplicated element:

```
duplicated(x)
# FALSE FALSE TRUE TRUE TRUE
```

`anyDuplicated(x) > 0L` is a quick way of checking whether a vector contains any duplicates.

Section 34.5: Measuring set overlaps / Venn diagrams for vectors

To count how many elements of two sets overlap, one could write a custom function:

```
xtab_set <- function(A, B){
    both    <-  union(A, B)
    inA     <-  both %in% A
    inB     <-  both %in% B
    return(table(inA, inB))
}

A = 1:20
B = 10:30

xtab_set(A, B)

#         inB
# inA      FALSE TRUE
#   FALSE      0   10
#   TRUE       9   11
```

A Venn diagram, offered by various packages, can be used to visualize overlap counts across multiple sets.