Explain the methods provided by R for aggregation and reshaping of data. Give examples.

R provides a number of powerful methods for aggregating and reshaping data. When we aggregate data, we replace groups of observations with summary statistics based on those observations.

When reshaping data, we alter the structure (rows and columns) determining how the data is organized. We'll use the mtcars data frame. This dataset, describes the design and performance characteristics (number of cylinders, displacement, horsepower, mpg, and so on) of automobiles.

Transpose

The transpose (reversing rows and columns) is perhaps the simplest method of reshaping a dataset. Use the t() function to transpose a matrix or a data frame. An example is below.

```
> cars <- mtcars[1:5,1:4]
> cars
                   mpg cyl disp  hp
Mazda RX4         21.0   6  160 110
Mazda RX4 Wag     21.0   6  160 110
Datsun 710        22.8   4  108  93
Hornet 4 Drive    21.4   6  258 110
Hornet Sportabout 18.7   8  360 175
> t(cars)
     Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout
mpg         21             21       22.8           21.4              18.7
cyl          6              6        4.0            6.0               8.0
disp       160            160      108.0          258.0             360.0
hp         110            110       93.0          110.0             175.0
```

Listing above uses a subset of the mtcars dataset in order to conserve space on the page.

Aggregating data

It's relatively easy to collapse data in R using one or more by variables and a defined function. The format is aggregate(x, by, FUN) where x is the data object to be collapsed, by is a list of variables that will be crossed to form the new observations, and FUN is the scalar function used to calculate summary statistics that will make up the new observation values.

As an example, we'll aggregate the mtcars data by number of cylinders and gears, returning means on each of the numeric variables.

Listing 5.10 Aggregating data

```
> options(digits=3)
> attach(mtcars)
> aggdata <-aggregate(mtcars, by=list(cyl,gear), FUN=mean, na.rm=TRUE)
> aggdata
  Group.1 Group.2  mpg cyl disp  hp drat   wt qsec  vs   am gear carb
1       4       3 21.5   4  120  97 3.70 2.46 20.0 1.0 0.00    3 1.00
2       6       3 19.8   6  242 108 2.92 3.34 19.8 1.0 0.00    3 1.00
3       8       3 15.1   8  358 194 3.12 4.10 17.1 0.0 0.00    3 3.08
4       4       4 26.9   4  103  76 4.11 2.38 19.6 1.0 0.75    4 1.50
5       6       4 19.8   6  164 116 3.91 3.09 17.7 0.5 0.50    4 4.00
6       4       5 28.2   4  108 102 4.10 1.83 16.8 0.5 1.00    5 2.00
7       6       5 19.7   6  145 175 3.62 2.77 15.5 0.0 1.00    5 6.00
8       8       5 15.4   8  326 300 3.88 3.37 14.6 0.0 1.00    5 6.00
```

In these results, Group.1 represents the number of cylinders (4, 6, or 8) and Group.2 represents the number of gears (3, 4, or 5). For example, cars with 4 cylinders and 3 gears have a mean of 21.5 miles per gallon (mpg).

When we're using the aggregate() function , the by variables must be in a list (even if there's only one). You can declare a custom name for the groups from within the list, for instance, using by=list(Group.cyl=cyl, Group.gears=gear) . The function specified can be any built-in or user-provided function. This gives the aggregate command a great deal of power. But when it comes to power, nothing beats the reshape package.

The Reshape package

The reshape package can be used for both restructuring and aggregating datasets. Reshape can be installed using **install.packages("reshape").**

Basically, we'll "melt" data so that each row is a unique ID-variable combination. Then we'll "cast" the melted data into any shape of desire. During the cast, one can aggregate the data with any function of our wish.

The dataset we'll be working with is shown in table 5. In this dataset, the measurements are the values in the last two columns (5, 6, 3, 5, 6, 1, 2, and 4). Each measurement is uniquely identified by a combination of ID variables (in this case ID, Time, and whether the measurement is on X1 or X2). For example, the measured value 5 in the first row is uniquely identified by knowing that it's from observation (ID) 1, at Time 1, and on variable X1.

Table 5.8 The original dataset (mydata)

ID	Time	X1	X2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

MELTING

Melting a dataset means to restructure it into a format where each measured variable is in its own row, along with the ID variables needed to uniquely identify it.

If we melt the data from table 5.8, using the following code

library(reshape)

md <- melt(mydata, id=(c("id", "time")))

we end up with the structure shown in table 5.9.

Table 5.9 The melted dataset

ID	Time	Variable	Value
1	1	X1	5
1	2	X1	3
2	1	X1	6
2	2	X1	2
1	1	X2	6
1	2	X2	5
2	1	X2	1
2	2	X2	4

We must specify the variables needed to uniquely identify each measurement (ID and Time) and that the variable indicating the measurement variable names (X1 or X2) is created automatically. Now that we haves data in a melted form, we can recast it into any shape, using the cast() function.

CASTING

The cast() function starts with melted data and reshapes it using a formula that we provide and an (optional) function used to aggregate the data. The format is

newdata <- cast(md, formula, FUN)

where md is the melted data, formula describes the desired end result, and FUN is the (optional) aggregating function.