2. Reworking and Recoding Data

Often when working on a project you will have a data set that will contain additional information that you don't need for your analysis; or, have attributes which aren't specified as you require. This helpsheet explains how to remove and add additional attributes.

For example, let's say we have a data set as follows:

Name	Age	Place	School	Degree
John	20	Liverpool	Hillside High School	Geography BA (Hons)
Rachel	21	Norwich	Colman High School	Geography & Archaeology BA (Joint Hons)
...	...	...	...	...

And we are only interested in people's age for this exercise. As such, we don't need all of the other data.

Before we start, we need to setup the working directory and read in the data:

> *For more information on working directories, see the worksheet '1. R Basics'. Remember to create the folder* `R work` *if it doesn't exist already.*

```
# Set working directory
setwd("M:/R work")
# Read data from the web
data <- read.csv("http://data.alex-singleton.com/r-helpsheets/2/example.csv", header = TRUE)
```

We will now display this to check it has been read in correctly:

```
data
```

Which should give you this:

```
    Name Age     Place               School                                  Degree
1   John  20 Liverpool Hillside High School                     Geography BA (Hons)
2 Rachel  21   Norwich   Colman High School Geography & Archaeology BA (Joint Hons)
3  Helen  34 Liverpool Hillside High School                     Geography BA (Hons)
4    Mia  20 Liverpool  Central High School                     Geography BA (Hons)
5   Carl  26    Exeter  Central High School                    Geography BSc (Hons)
6 Kerryn  21    Exeter  Central High School                    Geography BSc (Hons)
```

The `subset` command can be used to extract just the specified columns (and/or rows) from the data set. For example:

```
subset(data, select = c("Name", "Age"))

subset(data, Place == "Liverpool", select = c("Name", "Age"))
```

We can also store this as a new object:

```
data.Liverpool <- subset(data, Place == "Liverpool", select = c("Name", "Age"))
```

Because the statement assigns the output of the subset function to the new object called `"data.Liverpool"`, nothing will be printed. As such, we can check by typing `data.Liverpool`:

```
   Name Age
1  John  20
3 Helen  34
4   Mia  20
```

Adding a column to a data frame is done using the $ symbol. We will initially store `NA` (i.e. no value) in the column.

```
data.Liverpool$diff100 <- NA
```

We also use the same principle to calculate the age difference from 100

```
data.Liverpool$diff100 <- 100 - data.Liverpool$Age
```

Perhaps we decide that we don't like the label of the first column "`Name`" and that it would be more appropriate to call it "`FirstName`". To make this change we create a variable with the column labels that we want:

```
new_column_names <- c("FirstName", "Age", "diff100")
```

When doing this it is always a good idea to check that the length of the object we have just created (it should be `3`) is the same as the number of columns in our data frame.

```
length(new_column_names)
```

```
ncol(data.Liverpool)
```

We can then add the new column names to the data frame:

```
colnames(data.Liverpool) <- new_column_names
```

Check the data frame now, and the names should be changed.

```
  FirstName Age diff100
1      John  20      80
3     Helen  34      66
4       Mia  20      80
```

Instead of recording people's age in years, perhaps we just need this in two categories - 21 and over, and under 21. We can *recode* the `Age` variable into a new variable as follows:

```
data.Liverpool$AgeCat[data.Liverpool$Age < 21] <- "Under 21"
data.Liverpool$AgeCat[data.Liverpool$Age >= 21] <- "21 or over"
```

This will create the new variable, `AgeCat`. To see what has happened to the object, print the `data.Liverpool` again:

```
  FirstName Age diff100     AgeCat
1      John  20      80   Under 21
3     Helen  34      66 21 or over
4       Mia  20      80   Under 21
```