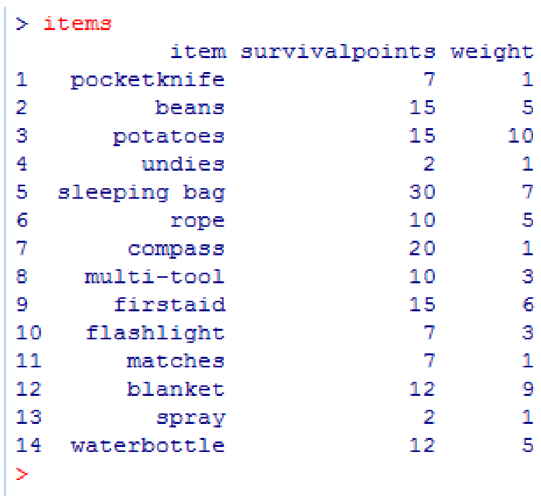
Multi-objective optimisation lab

The Problem – The Knap Sack Problem (Backpack problem in NZ)

NOTE: IF you didn't do the Genetic Algorithm Lab please go back and do this now!

As a reminder, the problem we will consider is defined as follows:

We have a list of items that we can put in our backpack, and each item has a weight and a survival value (there is also a "robust" column, but we will ignore this until the last step of the lab). What is the optimal combination of items that we can pack when we go on a tramp (i.e. walk) given that we want to **maximise our survival value**, but have some **weight limit** (since we are going to carry the backpack)? For example, the choice of items might be:



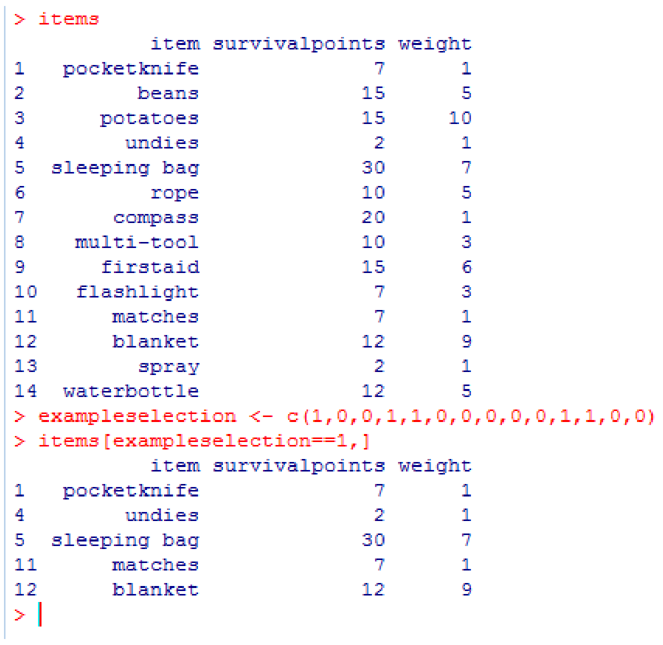
```
> items
            item survivalpoints weight
1    pocketknife              7      1
2          beans             15      5
3       potatoes             15     10
4         undies              2      1
5   sleeping bag             30      7
6           rope             10      5
7        compass             20      1
8     multi-tool             10      3
9       firstaid             15      6
10    flashlight              7      3
11       matches              7      1
12       blanket             12      9
13         spray              2      1
14   waterbottle             12      5
>
```

and we could be constrained to a **weight limit of 20**.

We have previously used a Genetic Algorithm (GA) using a binary string representation to search for a good solution that maximises survival points but keeps our total items under our weight limit. Given these are 2 constraints that are trade-offs, we could also treat the problem as a multi-objective optimisation. We will look at 2 representations – a binary string (like the GA example you have previously done), and using a vector of real values.

Original GA Representation

Since we know the total number of possible items to choose, and we have a binary selection (we either do or don't put an item in our backpack), it is appropriate to represent an individual as a binary string. Hence a "1" will represent that we are selecting a particular item, while a "0" indicates that we are not putting it into our backpack. For the item list above, with 14 items, the GA representation for each individual in the population is a binary string of 14 bits. For example, a possible solution might be **1 0 0 1 1 0 0 0 0 0 1 1 0 0**. This would correspond to selecting the pocketknife, undies, sleeping bag, matches and blanket, as shown below.



```
> items
           item survivalpoints weight
1   pocketknife              7      1
2         beans             15      5
3      potatoes             15     10
4        undies              2      1
5  sleeping bag             30      7
6          rope             10      5
7       compass             20      1
8    multi-tool             10      3
9      firstaid             15      6
10   flashlight              7      3
11      matches              7      1
12      blanket             12      9
13        spray              2      1
14  waterbottle             12      5
> exampleselection <- c(1,0,0,1,1,0,0,0,0,0,1,1,0,0)
> items[exampleselection==1,]
           item survivalpoints weight
1   pocketknife              7      1
4        undies              2      1
5  sleeping bag             30      7
11      matches              7      1
12      blanket             12      9
> 
```

STEP 1. **Run the mco.R script,** which will do a multi-objective optimisation for the bag problem.

Have a look at the output, and go through the script, making sure that you understand what it is doing.

WHAT DOES THE PLOT REPRESENT?

NOTE that this version uses the same representation for an individual as the original GA work that we used to solve this problem – i.e. a binary string. However, because the mco package assumes a vector of real values we had to write the function as.binary, which is a little complicated but just converts an integer value into the corresponding binary representation (given a total length of representation). You can always call this function on its own to check what it is doing. CHECK that you understand how this representation is defined.

We will look at a simpler representation in STEP 3.

STEP 2. **Edit the mco.R script and uncomment the section at the bottom. This adds an additional constraint. Run this script (or the uncommented section) to examine the change in behaviour.**

QUESTIONS:

1. What is the additional constraint that has been added?
2. What effect does this have on the plot of the pareto front and the solutions?

Try changing the weightConstraint function so that we only accept weights between 15 and 20 (currently it just accepts any weight <= 20), and check that the pareto front makes sense with this additional constraint.

HINT: You will need to edit the function weightConstraint:

```
weightConstraint <- function(x)

{

        return(weightlimit - packWeight(x))    # Constraints must be >= 0

}
```

CHANGE this constraint function so that weights can only be between 15 and the weightlimit (currently set at 20). Don't forget to source the new version to see the effect of this change.

STEP 3. Run mco2.R – this solves the same problem, but uses a representation that has a single value for each item. Open mco2.R and examine how the representation is used.

QUESTIONS

1. **How does the representation in mco2.r differ from mco.R?**
2. **How is an individual represented?**
3. **How is an individual decoded to work out the objectives?**

STEP 4. EXTENDING THE OBJECTIVES in MCO2.R

1. **Have another look at the items table. Note that it also has a column for indicating whether an item is "robust". Note that bigger is more robust.**
2. **Extend mco2.r to incorporate an additional OBJECTIVE: MAXIMISE ROBUST as part of our objective space.**

NOTE that this means we now have **3 objectives**, so plotting the **Pareto front requires the use of the 3d plotting package** (which we have already installed for you, but didn't say why!). Note, however, that it isn't as easy to see what is going on with 3 dimensions. How might you make this easier to understand? Check out **scatterplot options using the results from mco** – you could also consider how you might **plot 2 of the objectives at a time**.....