# Self incompatibility in plants — Golberg et al. 2010

Golberg et al. (2010) studied self-incompatibility in Solenaceae. Self-incompatible plants can recognize and reject their own pollen. The file `data/Golberg2010_data.csv` contains a list of 356 species, along with a flag determining the self-incompatibility status: `0` stands for self-incompatible; `1` for self-compatible; `2-5` for more complex scenarios.

1. Write a program that counts how many species are in each category of `Status`. The output should be a `data.frame`, like

```
  Status count
1      0   116
2      1   196
3      2    17
4      3     1
5      4    25
6      5     1
```

This can be accomplished in a number of ways. First of all, we need to read the data using the function `read.csv`:

```r
# Read the table (use stringsAsFactors == FALSE, otherwise species names
# will become factors!)
g2010 <- read.csv("../data/Goldberg2010_data.csv", stringsAsFactors = FALSE)
# Check the dimensions
dim(g2010)
```

```
## [1] 356   2
```

```r
# Print the first few lines
head(g2010)
```

```
##                      Species Status
## 1      Acnistus_arborescens      1
## 2       Anisodus_tanguticus      1
## 3          Atropa_belladonna      1
## 4 Brachistus_stramonifolius      1
## 5           Brugmansia_aurea      0
## 6      Brugmansia_sanguinea      0
```

Now we want to count how many species are associated with each `Status`, and store the results in a `data.frame`.

A possible strategy is to extract all the unique occurrences of `Status`, and then write a `for` loop iterating over them. For each `Status`, we can append to the `data.frame`:

```r
# Get unique values in Status
possible_status <- sort(unique(g2010$Status))
# Create empty data.frame
results <- data.frame()
# Cycle through the possible values:
```

```
for (status in possible_status){
  # Add to the data.frame
  results <- rbind(results,
                   data.frame(Status = status, count = sum(g2010$Status == status)))
}
# And here are the results
results
```

```
##   Status count
## 1      0   116
## 2      1   196
## 3      2    17
## 4      3     1
## 5      4    25
## 6      5     1
```

Alternatively, we can manage without writing a loop, by exploiting the function `table`, which automatically builds a vector of counts (see `?table` for the manual):

```
# Example usage
table(g2010$Status)
```

```
##
##   0   1   2   3   4   5
## 116 196  17   1  25   1
```

```
# Store it in a temporary variable
tmp <- table(g2010$Status)
# Build the data.frame
results <- data.frame(Status = as.numeric(names(tmp)),
                      count = as.numeric(tmp))
# And here are the results
results
```

```
##   Status count
## 1      0   116
## 2      1   196
## 3      2    17
## 4      3     1
## 5      4    25
## 6      5     1
```

2. Write a program that builds a `data.frame` specifying how many species are in each `Status` for each genus (note that each species name starts with the genus, followed by an underscore).

To complete this task, we need to extract the genus name for each species. The functions `strsplit` can break a string into pieces according to one (or more) characters. It returns a list for each string. For example:

```
# Take the first species
sp1 <- g2010$Species[1]
# Here's the species name
sp1
```

```
## [1] "Acnistus_arborescens"
```

```
# Split using "_" as a separator
strsplit(sp1, "_")
```

```
## [[1]]
## [1] "Acnistus"    "arborescens"
```

```
# To access the genus
# [[1]] -> First element of the list;
# [1] -> First element of the vector
strsplit(sp1, "_")[[1]][1]
```

```
## [1] "Acnistus"
```

With this function at hand, we can for example add a new `Genus` column to `g2010`:

```
# Create empty vector of strings
genera <- character(0)
# For each species, extract genus
for (sp in g2010$Species){
  genus <- strsplit(sp, "_")[[1]][1]
  # update genus list
  genera <- c(genera, genus)
}
# Finally, append to g2010
g2010$Genus <- genera
# See the first few
head(g2010)
```

```
##                      Species Status      Genus
## 1      Acnistus_arborescens      1   Acnistus
## 2       Anisodus_tanguticus      1   Anisodus
## 3         Atropa_belladonna      1     Atropa
## 4 Brachistus_stramonifolius      1 Brachistus
## 5          Brugmansia_aurea      0 Brugmansia
## 6      Brugmansia_sanguinea      0 Brugmansia
```

There are other, clever ways to obtain the same results. However, they are more advanced, so we stick to the for loop.

Once we have extracted the `Genus`, we can for example build a table using two nested for loops, or alternatively calling

```
table(g2010$Genus, g2010$Status)
```

```
##
##                  0   1   2   3   4   5
##   Acnistus       0   1   0   0   0   0
##   Anisodus       0   1   0   0   0   0
##   Atropa         0   1   0   0   0   0
##   Brachistus     0   1   0   0   0   0
##   Brugmansia     2   0   0   0   0   0
##   Capsicum       0  14   2   0   0   0
##   Cuatresia      2   0   0   0   0   0
##   Datura         0   4   0   0   0   0
##   Dunalia        1   0   0   0   1   0
##   Dyssochroma    1   0   0   0   0   0
##   Eriolarynx     1   0   0   0   0   0
##   Grabowskia     1   0   0   0   0   0
##   Hyoscyamus     0   3   0   0   0   0
##   Iochroma       4   0   0   0   0   0
##   Jaborosa       1   0   0   0   0   0
##   Jaltomata      0   6   0   0   0   0
##   Lycianthes     3   0   0   0   0   0
##   Lycium         9   1   0   1   8   0
##   Mandragora     1   0   0   0   0   0
##   Nicotiana      7  46   2   0   0   0
##   Nolana         8   1   2   0   0   0
##   Physalis       5   5   0   0   0   0
##   Salpichroa     1   0   0   0   0   0
##   Solandra       1   0   0   0   0   0
##   Solanum       63 110  10   0  13   1
##   Symonanthus    0   0   0   0   2   0
##   Vassobia       0   1   0   0   0   0
##   Withania       0   1   0   0   1   0
##   Witheringia    5   0   1   0   0   0
```