Chapter 5 - Working with large datasets

Edson Raul Cepeda Marquez

The concepto of *Big Data* to very large datasets. Dealing with Big Data is also part of data science. The science of what you can do with data in a given amount of time, or a given amount of space, is called *complexity theory*.

Subsample your data before you analize the full dataset.

You very rarely need to analyze a complete dataset to get a least an idea of how the data behaves. You should explore a smaller sample of your data. Here it is important that you pick a random sample. If you have a large dataset, and your analysis is being slowed down because of ti, don't be afraid to pick a random subset and analyze that.

You can use the `dplyr` functions `sample_n()` and `sample_frac()` to sample from a data frame. Use `sample_n()` to get fixed numbers of rows and `sample_frac()` to get a fraction of the data:

```
library(dplyr)
library(magrittr)

iris %>% sample_n(size = 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          4.4         3.2          1.3         0.2
## 2          4.9         3.6          1.4         0.1
## 3          6.7         3.1          4.7         1.5
## 4          6.3         2.8          5.1         1.5
## 5          7.7         2.8          6.7         2.0
##      Species
## 1     setosa
## 2     setosa
## 3 versicolor
## 4  virginica
## 5  virginica
```

```
iris %>% sample_frac(size = 0.02)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          6.6         3.0          4.4         1.4
## 2          5.8         4.0          1.2         0.2
## 3          6.3         2.9          5.6         1.8
##      Species
## 1 versicolor
## 2     setosa
## 3  virginica
```

You need your data in a form `dplyr` can manipulate, and if the data is too large even to load into R, then you cannot have it in a data frame to sample from.

Luckily, `dplyr` has support for using data stored on disk rather than in RAM, in various backend formats. It

is, possible to connect a database to `dplyr` and sample from a large dataset this way.

Running Out of Memory During Analysis

You can examine memory usage and memory changes using the `pryr` package:

```
library(pryr)
mem_change(x <- rnorm(10000))
```

```
## -10.9 kB
```

Modifying this vector:

```
mem_change(x[1] <- 0)
```

```
## 528 B
```

If we assing vector to another variable, we do not use twice as memory:

```
mem_change(y <- x)
```

```
## -79.5 kB
```

But then if we modify one vector, we will have to make a copy so the other vector remains the same:

```
mem_change(x[1] <- 0)
```

```
## 80.6 kB
```

This is one reason for using pipelines rather than assigning to many variables during analysis.

You can remove stored data using `rm()` function to free up memory.