# Parallel Rcpp Portfolio

Rachel Wood

2023-04-20

For this portfolio we consider the `electBook` dataset on Irish smart meters:

```
library(electBook)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
data("Irish")
summary(Irish)
```

```
##         Length Class      Mode
## indCons 2672   data.frame list
## survey    12   data.frame list
## extra      7   data.frame list
```

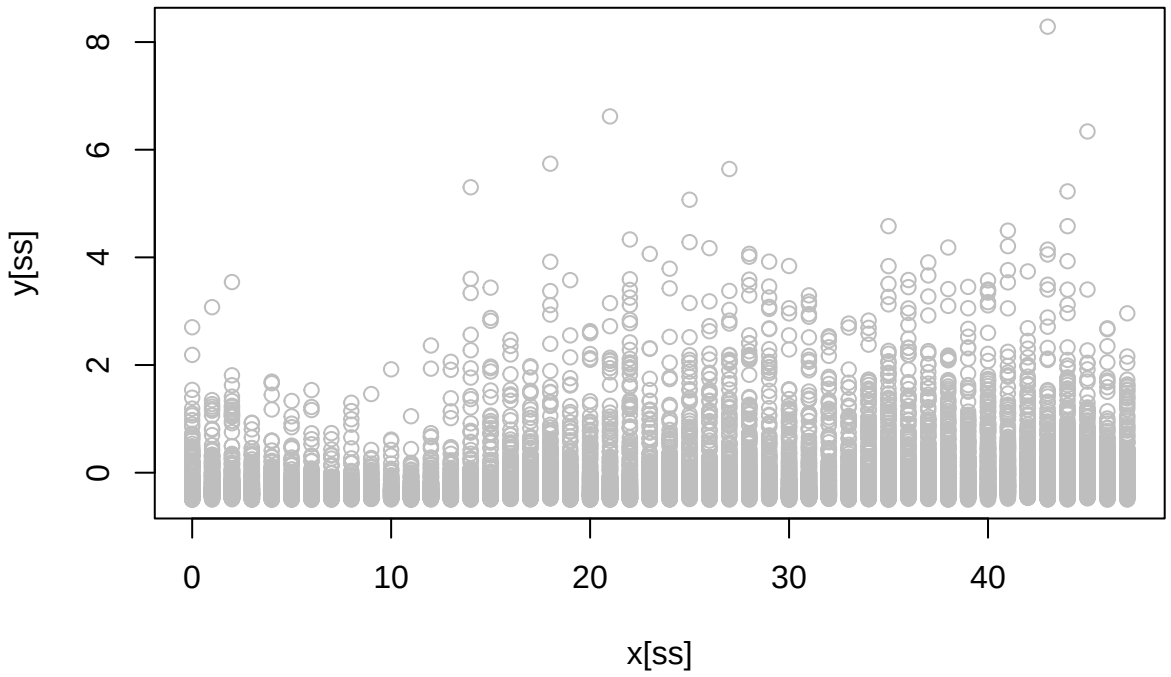
We concatenate the household demand into a vector - as there are a huge number of observations, we only plot a sub sample of the data:

```
y <- do.call("c", Irish$indCons)
y <- y - mean(y)
```

```
ncust <- ncol(Irish$indCons)

x <- rep(Irish$extra$tod, ncust)

n <- length(x)
ss <- sample(1:n, 1e4)
plot(x[ss], y[ss], col = "grey")
```




##

Modelling Demand We consider modelling the demand as a function of the time of day. Our first model is a simple linear regression $\mathbb{E}(y|x) = \beta x$, which we fit using least squares:

```
reg1D <- function(y, x){

  b <- t(x) %*% y / (t(x) %*% x)

  return(as.vector(b))

}
```

We compare this to the built in `lm` function:

```
system.time( lm(y ~ -1 + x)$coeff )[3]
```

```
## elapsed
##  14.835
```

```
system.time( reg1D(y, x) )[3]
```

```
## elapsed
##   0.473
```

We can see our function has a much faster implementation, however we can still attempt to improve this by parallelising the code in Rcpp using OpenMP

```
#include <Rcpp.h>
#include <omp.h>
using namespace Rcpp;

// [[Rcpp::plugins(openmp)]]
// [[Rcpp::export]]
double reg1DParallel(NumericVector y, NumericVector x) {
  int n = y.size();
  double numerator = 0.0;
  double denominator = 0.0;
```

```
  #pragma omp parallel for reduction(+:numerator, denominator)
  for (int i = 0; i < n; i++) {
    numerator += x[i] * y[i];
    denominator += x[i] * x[i];
  }

  return numerator / denominator;
}
```

We first check the results are the same:

```
all.equal(reg1D(y,x), reg1DParallel(y,x))
```

```
## [1] TRUE
```

and now compare computational time:

```
system.time( reg1D(y, x) )[3]
```

```
## elapsed
##   0.442
```

```
system.time( reg1DParallel(y, x) )[3]
```

```
## elapsed
##   0.023
```

We can see this has improved by a factor of 10.