Code optimization using commuting example

Commuting rate formula:

The expected commuting rate σ_{ji} from a source node j and a destination node i is:

$$\langle \sigma_{ji} \rangle = \sigma_j \frac{n_j n_i}{(n_j + s_{ji})(n_j + n_i + s_{ji})} \tag{1}$$

where n_j and n_i are the populations for nodes j and i, and s_{ji} is the total population (excluding j and i) in a circle centered on j with a radius equal to the distance between j and i. The total commuting rate of individuals in j is $\sigma_j = N_c/N = 11\%$, where N_c is the total number of commuters and N is the total population in the country.

External libraries used in this document:

`here` and `igraph`

Sourcing code:

```
invisible(lapply(here::here(list.files("R", full = TRUE)), source))
```

We're specifically working with code in the `R/commuting.R` file.

Reading in example data:

```
# read in data and make a subset of it
g = readRDS(here::here("inst/sampleData/flu-g.RDS"))
g = igraph::induced.subgraph(g, c("890", sample(1:1000, 250)))

# network edges (distances between nodes)
head(igraph::as_data_frame(g, "edges"))
```

```
##   from to Total_Length
## 1    9 12     59.21775
## 2    9 23     38.87634
## 3    9 26     12.45560
## 4    9 29     39.76065
## 5    9 34     28.00177
## 6    9 38     59.26846
```

```
# network node information
head(igraph::as_data_frame(g, "vertices"))
```

```
##    name       pop       lat      lon  id
## 9     9 19.037111 -2.731854 29.85823  n9
## 12   12 65.992627 -2.713930 29.54045 n12
## 23   23 18.746073 -2.688227 29.71347 n23
## 26   26  4.163806 -2.673082 29.89298 n26
```

```
## 29   29 70.058099 -2.660633 29.68806 n29
## 34   34 23.153439 -2.653949 29.75348 n34
```

Calculating commuting proportions:

```
# calculate commuting rates over it
t1 = system.time(disnet_commuting(g))
t2 = system.time(disnet_commuting2(g))
```

Time taken by the two methods:

```
# Method 1
t1
```

```
##    user  system elapsed
##   6.238   0.271   6.522
```

```
# Method 2
t2
```

```
##    user  system elapsed
##   4.304   0.092   4.430
```

There's some improvement with the second method but definitely scope for better code optimization!