



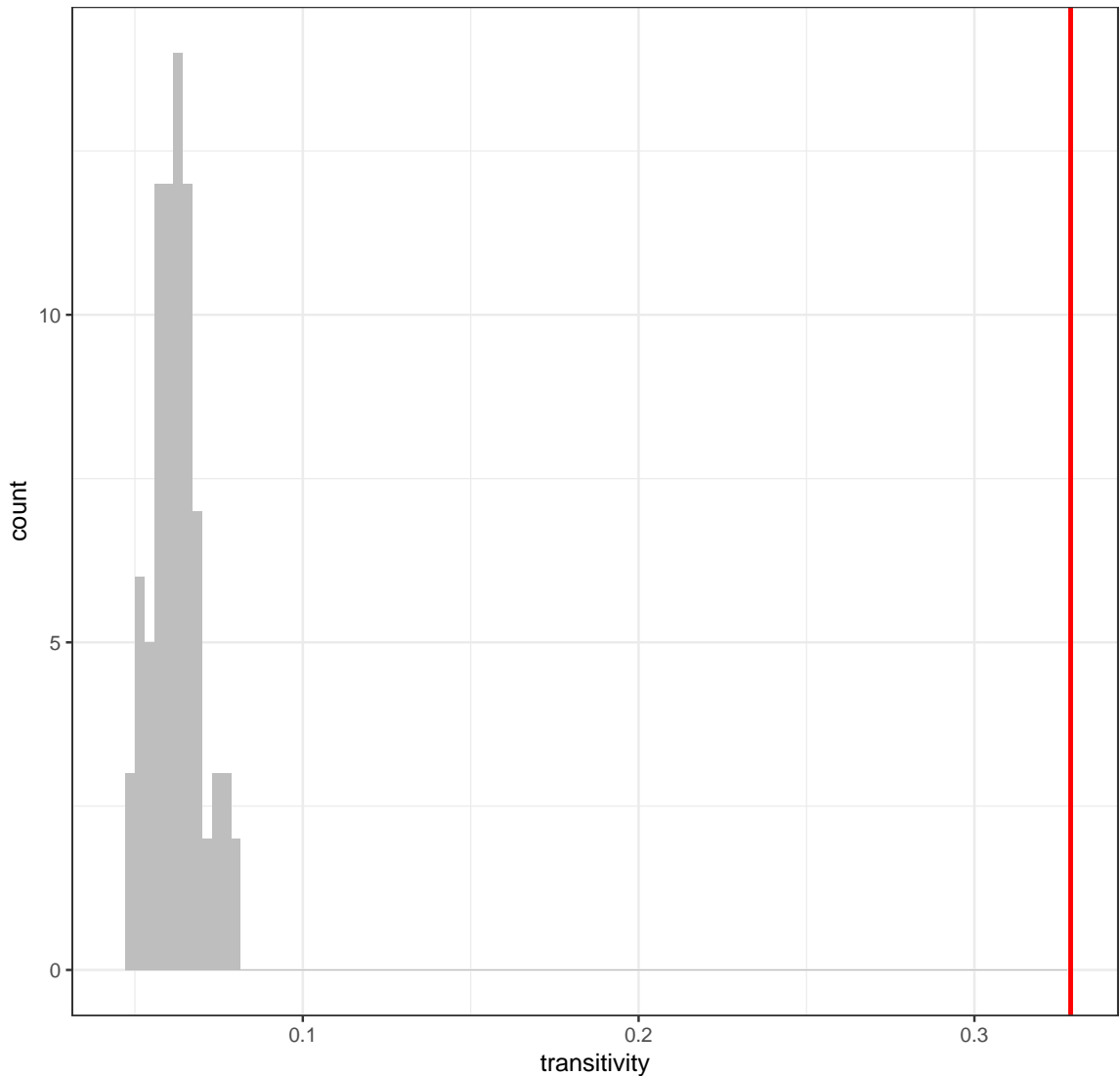
```
}  
}  
colnames(global_char) <- c("density", "transitivity", "diameter")  
global_char <- na.omit(global_char)
```

What does the 'for' loop returns for a row that comes from a non connected graph? How is this problem handled?  
Using this script, answer to the following questions:

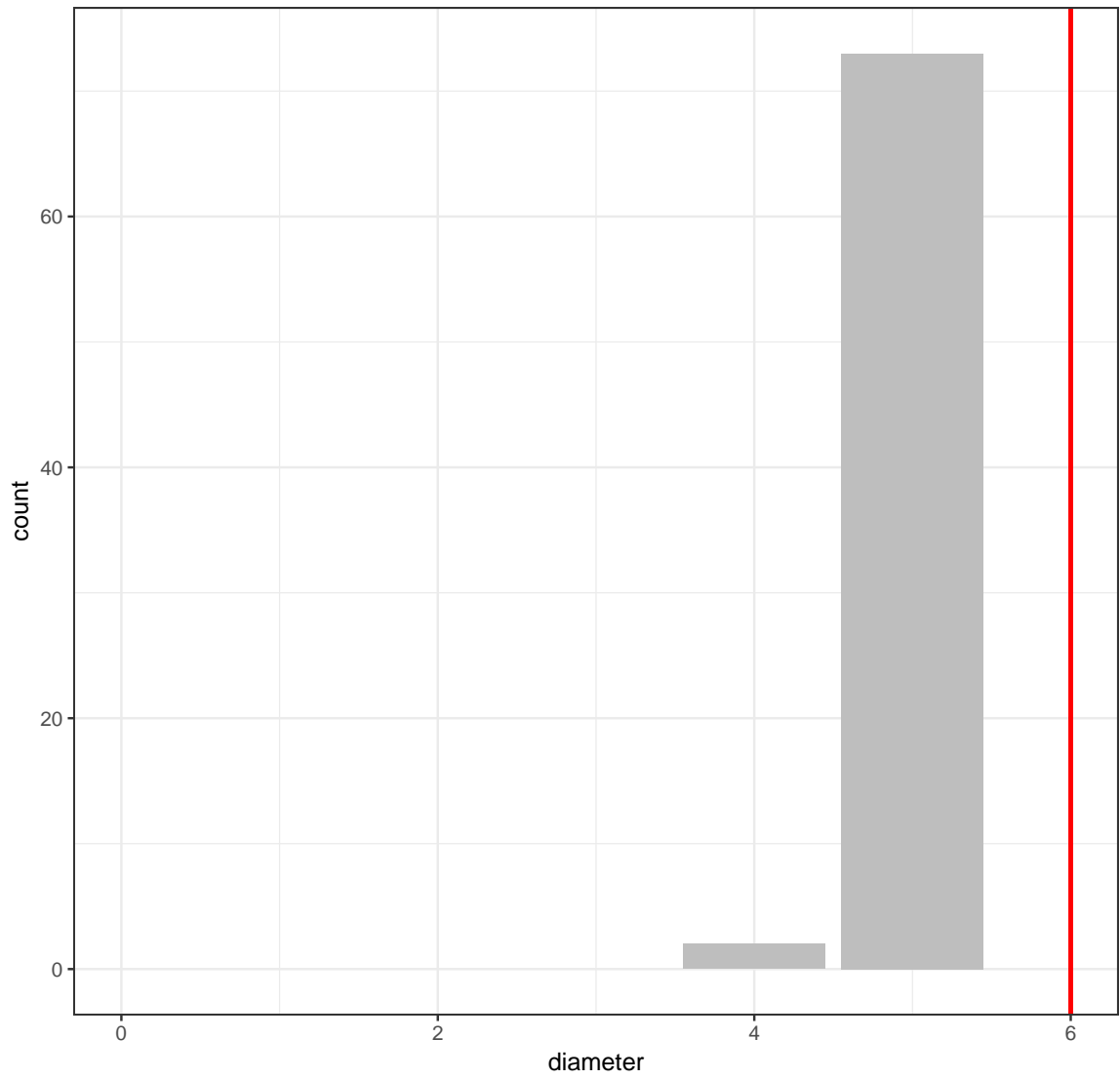
- (a) How many generated graphs were connected?

```
## [1] 82
```

- (b) How do the transitivity and the diameter of these graphs compare to the transitivity of the real graph `got_net`?  
transitivity of ER graphs similar to GOT

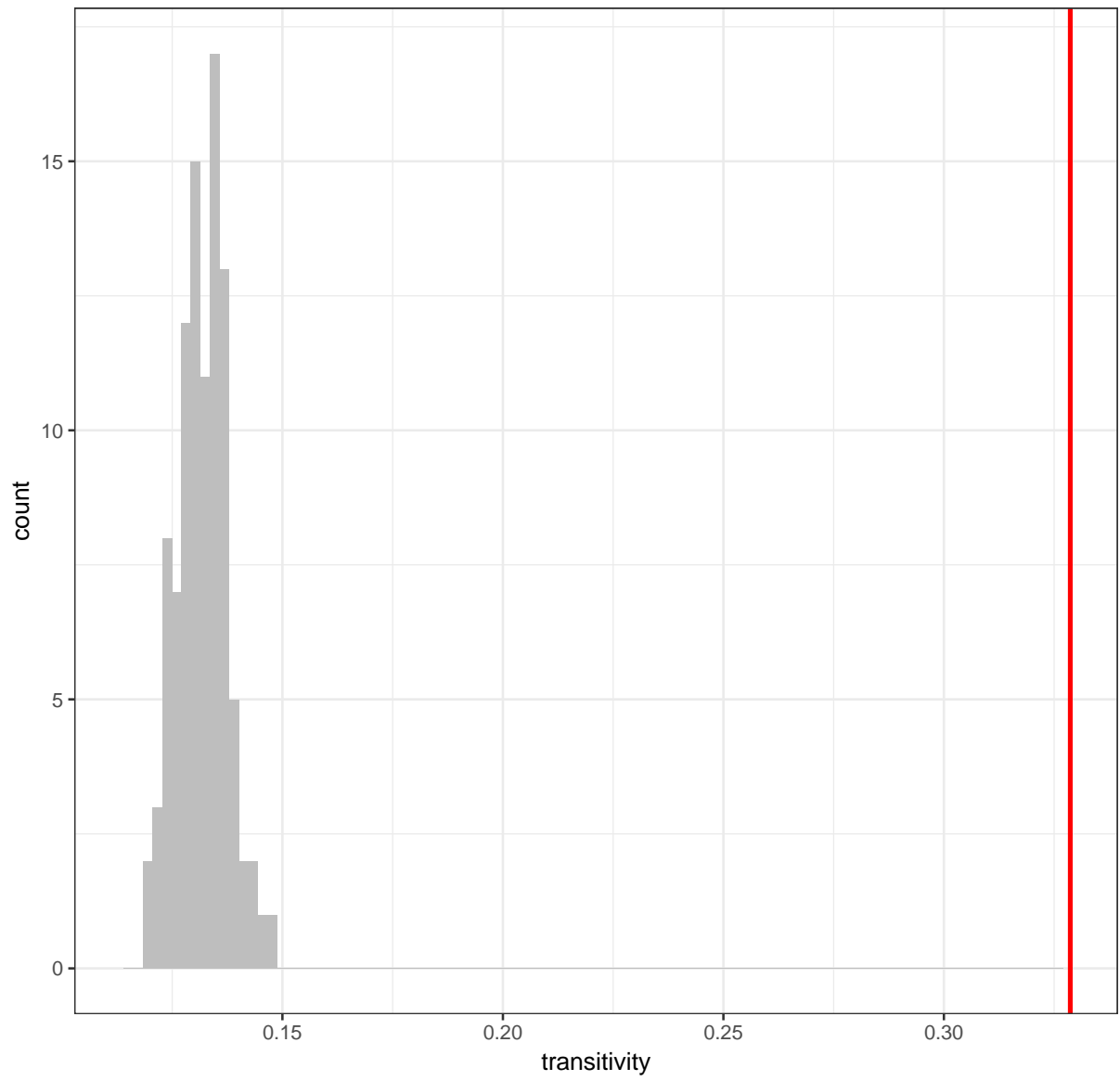


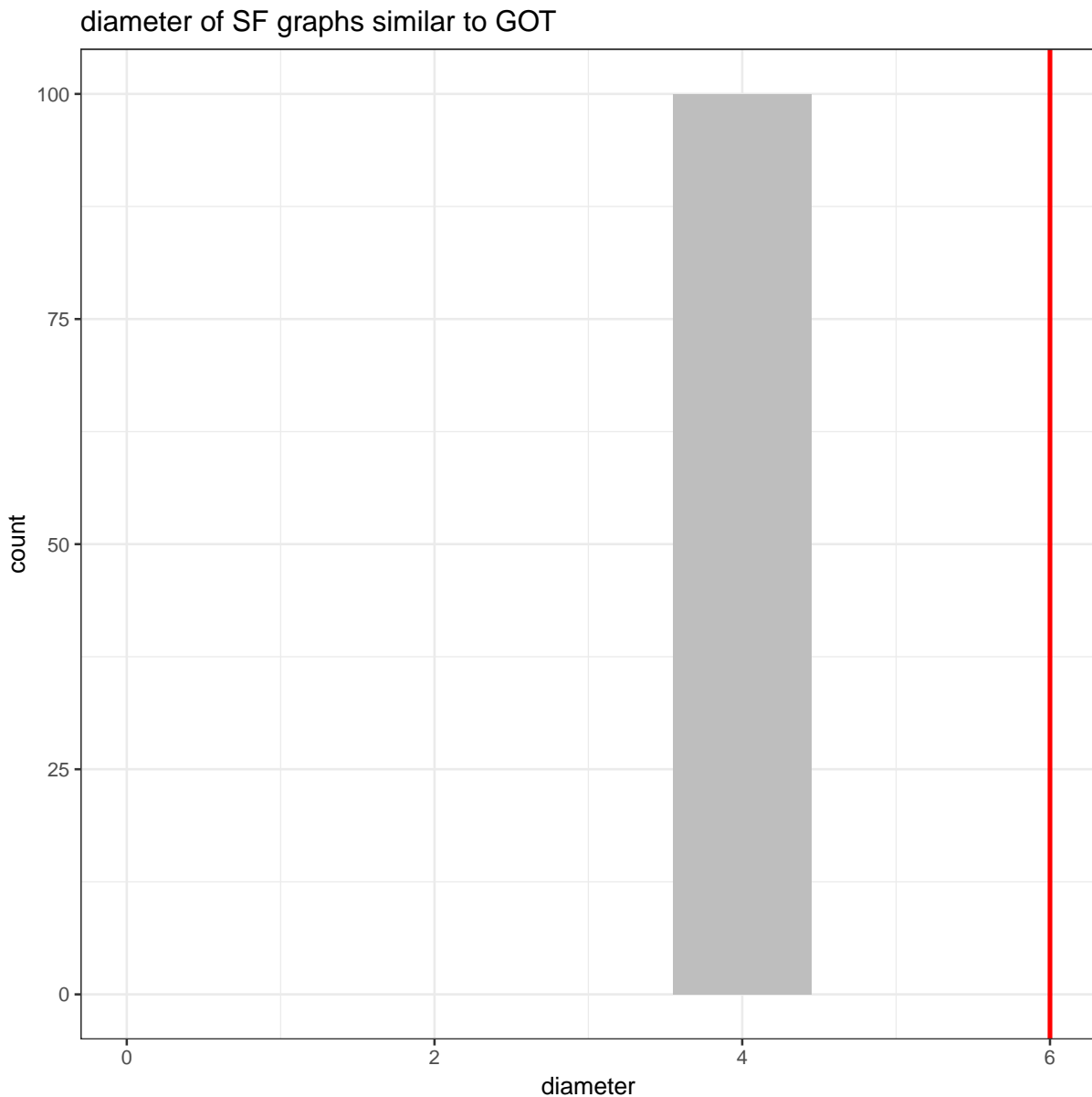
diameter of ER graphs similar to GOT



2. Use the same approach and answer the same question with the function `sample_pa` that generates scale free graphs according to Barabasi-Albert model. Use `m = 4` to run this function (how to tune this number is out of the scope of this lesson).

transitivity of SF graphs similar to GOT





## Exercise 2 Permutation tests

- The function `rewire` is used to generate random graphs by randomly permuting two edge endpoints. The second argument `with` of this function specifies a function call to one of the rewiring method, `keeping_degseq` indicating to keep the degree distribution. What does the following code perform and which value to use for `Q`?  
*Be careful, when using this script, that it uses a parallel backend. For Windows, the proper parallel backend is handled with the functions of the R package `doParallel`.*

```
set.seed(22011600)
iter <- 100
B <- 100
all_seeds <- sample(1:22011600, B, replace = FALSE)
registerDoMC(cores = 7)
global_char <- foreach (ind=1:B, .combine = rbind) %dopar% {
  set.seed(all_seeds[ind])
  rg <- rewire(got_net, keeping_degseq(n = iter * Q))
  if (is.connected(rg) & is.simple(rg)) {
    res <- c("transitivity" = transitivity(rg, weights = NA),
            "diameter" = diameter(rg, weights = NA))
  }
}
```

```

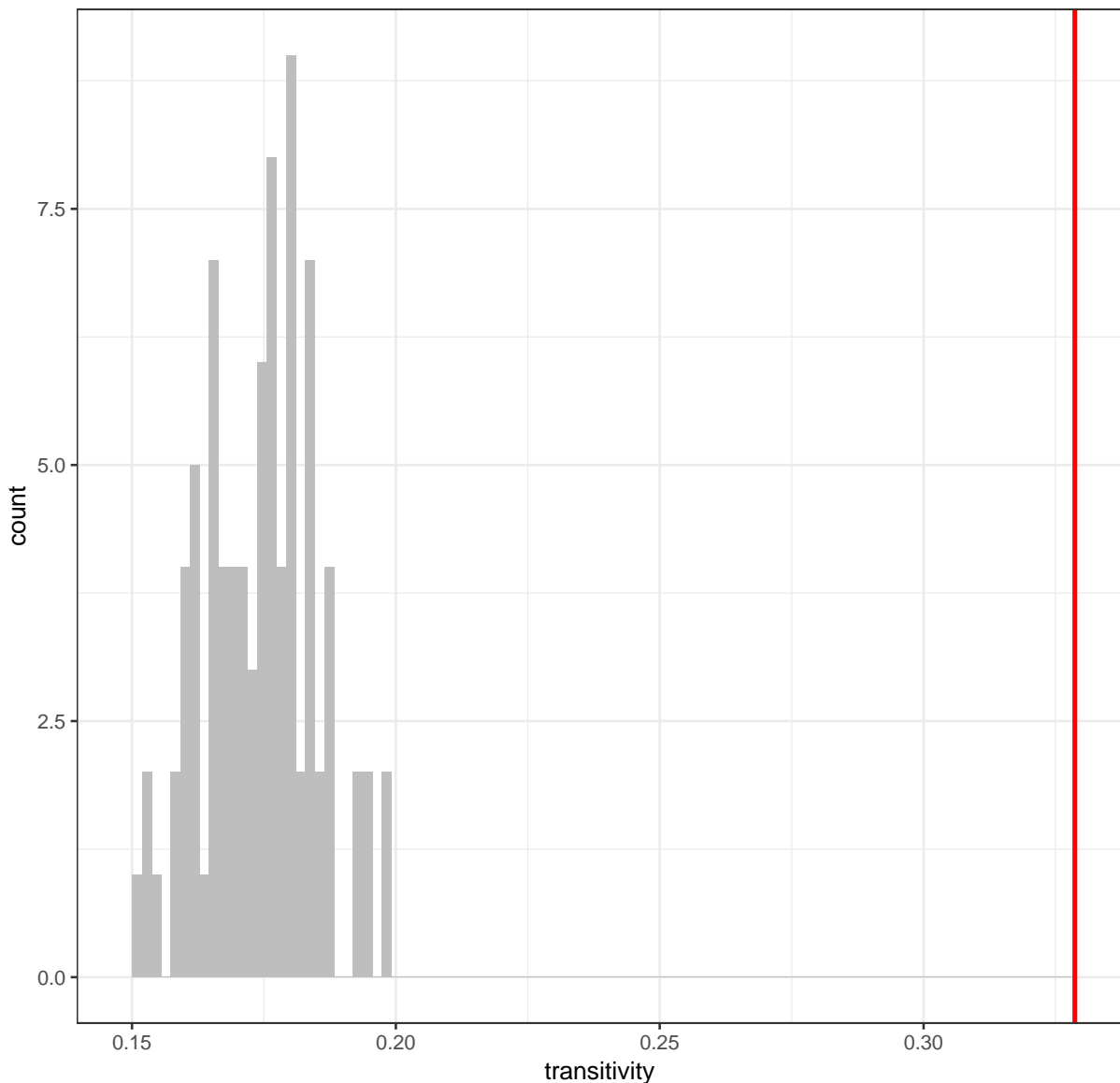
} else res <- rep(NA, 2)
  return(res)
}
global_char <- na.omit(global_char)

```

How many of these networks were connected?

```
## [1] 87
```

- Use the previous result to compare the transitivity of the observed graph with the transitivity of random graphs with the same degree distribution. transitivity of graphs with the same degree distribution than GOT



- Use the same type of script to generate a list of `iter` betweenness distributions for random graphs with the same degree distribution than GOT. How many of these networks were connected?

```

set.seed(22011706)
iter <- 100
B <- 100
all_seeds <- sample(1:22011706, B, replace = FALSE)
registerDoMC(cores = 7)

```

```

global_char <- foreach (ind=1:B, .combine = rbind) %dopar% {
  set.seed(all_seeds[ind])
  rg <- rewire(got_net, keeping_degseq(n = iter * Q))
  if (is.connected(rg) & is.simple(rg)) {
    res <- betweenness(rg, weights = NA)
  } else res <- rep(NA, vcount(got_net))
  return(res)
}
global_char <- na.omit(global_char)
nrow(global_char)

## [1] 94

```

4. What does the following code compute?

```

# obtain estimated p-values
bet_got <- betweenness(got_net, weights = NA)
valid_exp <- nrow(global_char)
c_betweenness <- rbind(bet_got, global_char)
p_high <- apply(c_betweenness, 2, function(acol)
  sum(acol[1] > acol[-1]) / valid_exp)
p_low <- apply(c_betweenness, 2, function(acol)
  sum(acol[1] < acol[-1]) / valid_exp)

```

5. Use the obtained `p_high` and `p_low` to obtain the following plot: blue nodes are those that have a betweenness lower than expected by random chance at risk 5%, red nodes are those with a betweenness larger than expected. Size of the nodes are proportionnal to the log-transformed betweenness and only blue and red nodes have their names displayed.

