How to find Gamma in a Power Law Distribution

William John Holden

26 June 2021

```
library(neo4r)
library(tidyverse)
library(knitr)
```

A graph $G = (V, E)$ forms a **scale-free network** when the degree of its vertices, $\texttt{deg}(v)$, fits a **power law** distribution. In a power law distribution, the probability density function (PDF) $P(k)$ for vertices of degree k is approximately $k^{-\gamma}$.

$$P(k) \sim k^{-\gamma}$$

An alternative phrasing of the PDF definition is, "the probability of finding a random vertex v with degree $\texttt{deg}(v) = k$ can be modeled by $k^{-\gamma}$ where γ is a constant."

For example, the connectedness of *Game of Thrones* characters is said to follow a power law distribution. We will use the `neo4r` package.

First, we connect to the Neo4j database over HTTP. The `neo4r` package does not support the BOLT protocol.

```
con <- neo4j_api$new(
  url = "http://localhost:7474",
  user = "neo4j",
  password = "powerlaw"
)
```

Now, we can run a Cypher query to tally the number of connections in or out of each vertex in the *Game of Thrones* database.

```
tmp = "MATCH (u)-[r]-()
RETURN u.name as Name, COUNT(r) as Degree
ORDER BY Degree DESC" %>% call_neo4j(con)
```

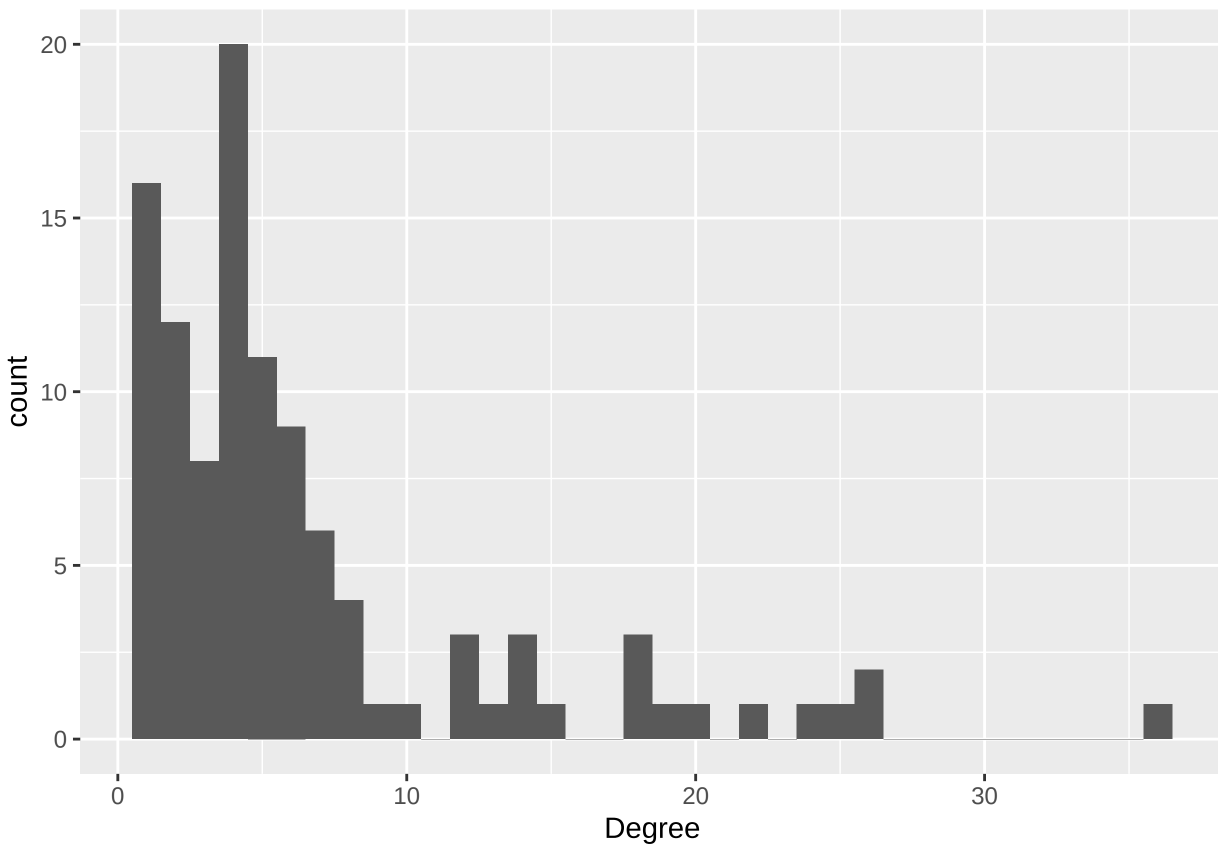
`neo4r` confusingly returns two "tibble" objects from this query. Additionally, the values in each object are named `value`.

```
df = tibble(Name = tmp$Name$value, Degree = tmp$Degree$value)
kable(head(df))
```

Name	Degree
Tyrion	36
Jon	26
Sansa	26
Robb	25
Jaime	24
Tywin	22

Now, we can analyze the power law distribution in R. We first plot the data on a histogram using the `ggplot2` library.

```
df %>% ggplot(aes(x = Degree)) + geom_histogram(binwidth=1)
```



γ is the exponent by which `count` decreases as `Degree` increases. We will compute a linear model to find γ. The `tabulate` function can count the number of occurrences for each degree in our data frame.

```
degree_dist = tibble(Degree = 1:max(df$Degree), Count = tabulate(df$Degree))
kable(head(degree_dist))
```

Degree	Count
1	16
2	12
3	8
4	20
5	11
6	9

We add a column to `degree_dist` by computing density from each row. We will also drop rows where density is zero.

```
degree_dist = degree_dist %>% mutate(Density = Count / sum(Count)) %>% filter(Density > 0)
kable(head(degree_dist))
```

Degree	Count	Density
1	16	0.1495327
2	12	0.1121495
3	8	0.0747664
4	20	0.1869159
5	11	0.1028037
6	9	0.0841121

We can find γ using a linear model. The command is simple:

```
model = lm(log(Density) ~ log(Degree), data = degree_dist)
summary(model)
```

```
##
## Call:
## lm(formula = log(Density) ~ log(Degree), data = degree_dist)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1098 -0.3505 -0.1083  0.4308  1.0759
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.3681     0.3568  -3.835  0.00103 **
## log(Degree)  -0.9989     0.1445  -6.915 1.03e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6045 on 20 degrees of freedom
## Multiple R-squared:  0.7051, Adjusted R-squared:  0.6903
## F-statistic: 47.82 on 1 and 20 DF,  p-value: 1.025e-06
```

This technique is called change of variables. The idea is that if $y = ax^b$, then we let $Y = \log y$ and $X = \log x$. By substitution,

$$y = e^Y = ax^b = ae^{X^b} = ae^{bX}.$$

Take the logarithm of both sides to find

$$\log e^Y = Y = \log ae^{bX} = \log a + \log e^{bX} = \log a + bX.$$

The slope of a linear model `lm(log(y) ~ log(x))` model is therefore equal to b. For the *Game of Thrones* power law distribution, the slope reveals that $\gamma = -0.9989486$.