Lab 9 - Network Science

Setup

- Create a project in RStudio for this lab.
- Download the **edges_2012.csv** and **vertices_2012.csv** data files from Canvas and move them to your project folder.
- Open a new R script for this lab and insert comments with your name and lab number.
- Load the `tidyverse`, `tidygraph`, and `ggraph` packages.

This lab contains example code that you will need to type into your R script and run. Additionally, there are exercises in green text that you should complete. Be sure to use appropriate comments in your R script so that it is easy to identify where your solutions to these exercises are located.

Terminology

The roots of network science are in the mathematical discipline of *graph theory*. There a few basic definitions that we need before we can proceed.

- A *graph* is simply a set of *vertices* (or nodes) and a set of *edges* (or links) between those nodes. It may be more convenient to think about a graph as being a *network*.
- In general, graphs do not have coordinates. Thus, there is no right way to draw a graph. Visualizing a graph is more art than science, but there are several popular graph layout algorithms.
- Edges in a graph can be *directed* or *undirected*. The difference is whether the relationship is mutual (i.e., undirected) or one-sided (i.e., directed).
- Edges may be *weighted*. The value of the weight represents some quantitative measure.
- A *path* is a non-self-intersecting sequence of edges that connect two vertices. There may be many paths, or no paths, between two vertices in a graph, but if there are many paths, then there is at least one shortest path. The notion of a shortest path is dependent upon a distance measure in the graph (usually, just the number of edges, or the sum of the edge weights). A graph is connected if there is a path between all pairs of vertices.
- The *diameter* of a graph is the length of the longest shortest path between any pairs of vertices.The *eccentricity* of a vertex is the length of the longest shortest path starting at that vertex. Thus, a vertex with a low eccentricity is more central to the graph.
- It is frequently of interest to determine which nodes are the most "central" to the network, but there are many different notions of *centrality*.
  - The *degree* of a vertex within a graph is the number of edges connected to it. Thus, the degree of a node is a simple measure of centrality in which more highly connected nodes rank higher.

- *Betweenness centrality* is based on the notion that if a vertex is more central to a graph, the more shortest paths between vertices would pass through it. Specifically, the betweenness centrality for a vertex *v* considers the proportion of shortest paths between any two vertices that pass through *v*, then sums over all possible pairs of vertices in the graph.

Example: Six Degrees of Kristen Stewart

In this example, we will explore a fun application of network science to Hollywood movies. The notion of *Six Degrees of Separation* was conjectured by a Hungarian network theorist in 1929, and later popularized by a play (and movie starring Will Smith). That is, we are all part of a social network with a relatively small diameter (as small as 6).

One popular incarnation of these ideas is the Kevin Bacon game. The idea is that, in principle, every actor in Hollywood can be connected to Kevin Bacon in at most six movie hops. We'll explore this idea using the Internet Movie Database (IMDb.com).

Collecting Hollywood Data

We will populate a Hollywood network using actors in the IMDb. In this network, each actor is a node, and two actors share an edge if they have ever appeared in a movie together. Our goal will be to determine the centrality of Kevin Bacon.

First, we want to determine the edges, since we can look up the node information based on the edges that are present. One caveat is that these networs can grow very rapidly. For this example, we will be conservative by incuding popular (at least 150,000 ratings) feature films (i.e., `kind_id` equal to 1) in 2012, and we will consider only the top-20 credited roles in each film.

To retrieve the list of edges, we need to consider all possible cast assignment types. To get this list, we will start by forming all total pairs using the `CROSS JOIN` operation in SQL, which has no direct `dplyr` equivalent. Thus, it will be necessary to actually write the SQL code in this case.

The following code was used to create the **edges_2012.csv** and **vertices_2012.csv** data files. Note that this code will only work for you if you have an established SQL connection and is included here for information purposes only. If you would like to learn more about connecting to SQL, see Appendix F in the Modern Data Science with R textbook.

```
library(mdsr)
library(DBI)

db <- dbConnect_scidb("imdb")

E <- dbGetQuery(db,
"SELECT a.person_id AS src, b.person_id AS dest,
  a.movie_id,
  a.nr_order * b.nr_order AS weight,
  t.title, idx.info AS ratings
FROM imdb.cast_info AS a
  CROSS JOIN imdb.cast_info AS b USING (movie_id)
  LEFT JOIN imdb.title AS t ON a.movie_id = t.id
```

```
  LEFT JOIN imdb.movie_info_idx AS idx ON idx.movie_id = a.movie_id
WHERE t.production_year = 2012 AND t.kind_id = 1
  AND info_type_id = 100 AND idx.info > 150000
  AND a.nr_order <= 20 AND b.nr_order <= 20
  AND a.role_id IN (1,2) AND b.role_id IN (1,2)
  AND a.person_id < b.person_id
GROUP BY src, dest, movie_id")

E <- E %>%
  mutate(ratings = parse_number(ratings))

actor_ids <- unique(c(E$src, E$dest))

V <- db %>%
  tbl("name") %>%
  filter(id %in% actor_ids) %>%
  select(actor_id = id, actor_name = name) %>%
  collect() %>%
  arrange(actor_id) %>%
  mutate(id = row_number())

edges <- E %>%
  left_join(select(V, from = id, actor_id), by = c("src" = "actor_id")) %>%
  left_join(select(V, to = id, actor_id), by = c("dest" = "actor_id"))

write_csv(edges, "edges_2012.csv")
write_csv(V, "vertices_2012.csv")
```

Note that a `weight` variable was created that we can use to weight the edges in the resulting graph. In this case, the weight is based on the order in which each actor appears in the credits (`nr_order`), where a rank of 1 means the actor had top billing. As `weight` is the product of the ranks for two connected actors, the smallest possible value of `weight` is 2. These weights will be useful because a higher order in the credit usually means more screen time.

Now, you can simply read in the **edges_2012.csv** and **vertices_2012.csv** data files.

```
edges <- read.csv("edges_2012.csv")
V <- read.csv("vertices_2012.csv")

edges %>% summarize(num_rows = n(),
                    num_titles = n_distinct(title))

##   num_rows num_titles
## 1    10223         55
```

The IMDb query resulted in 10,223 connections between 55 films. We can see that *Batman: The Dark Knight Rises* received the most user ratings on IMDb.

```
movies <- edges %>%
  group_by(movie_id) %>%
  summarize(title = max(title), N = n(), numRatings = max(ratings)) %>%
  arrange(desc(numRatings))
```

```
movies
```

```
## # A tibble: 55 × 4
##    movie_id title                                N numRatings
##       <int> <chr>                              <int>      <int>
##  1  4339115 The Dark Knight Rises                190    1258255
##  2  3519403 Django Unchained                     190    1075891
##  3  4316706 The Avengers                         190    1067306
##  4  4368646 The Hunger Games                     190     750674
##  5  4366574 The Hobbit: An Unexpected Journey    190     681957
##  6  4224391 Silver Linings Playbook              190     577500
##  7  4231376 Skyfall                              190     557652
##  8  4116220 Prometheus                           190     504980
##  9  4300124 Ted                                  190     504893
## 10  3298411 Argo                                 190     493001
## # … with 45 more rows
```

Building the Hollywood Network

To build a graph, we specify the edges, whether we want them to be directed, and add information about the vertices.

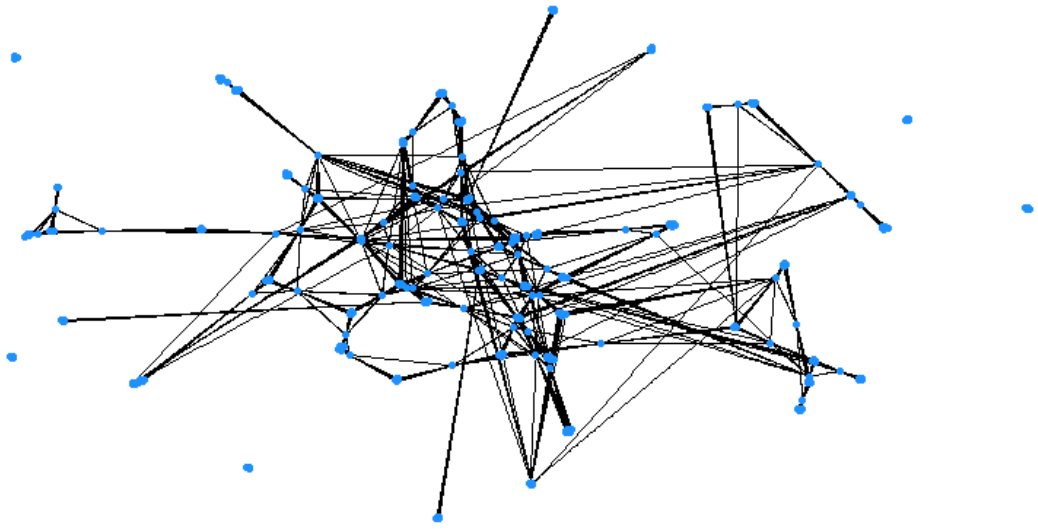
```
g <- tbl_graph(nodes = V, directed = FALSE, edges = edges)
summary(g)
```

```
## IGRAPH 4a3da65 U-W- 1010 10223 --
## + attr: actor_id (v/n), actor_name (v/c), id (v/n), src (e/n), dest
## | (e/n), movie_id (e/n), weight (e/n), title (e/c), ratings (e/n)
```

From the `summary` command above, we can see that we have 1,010 actors and 10,223 edges between them. Note that we have associated metadata with each edge: namely, information about the movie that gave rise to the edge, and the aforementioned `weight` metric based on the order in the credits were each actor appeared (the idea is that top-billed stars are likely to appear on screen longer, and thus have more meaningful interactions with more of the cast).

With our network intact, we can visualize it. For plotting network graphs directly with `ggplot2`, you can use the `ggraph` package, whcih provides `geom_edge_*` and `geom_node_*` functions (alternative plotting packages include `ggnetwork`, `geomnet`, and `GGally`). There are many graphical parameters that you may wish to set, and the default choices are not always good. In this case we have 1,010 vertices, so we'll make them small, and omit labels.

```
ggraph(g, 'drl') +
  geom_edge_fan(width = 0.1) +
  geom_node_point(color = "dodgerblue") +
  theme_void()
```



It is easy to see the clusters based on movies, but you can also see a few actors who have appeared in multiple movies, and how they tend to be more "central" to the network. If an actor has appeared in multiple movies, then it stands to reason that they will have more connections to other actors. This is captured by *degree centrality*.

```
g <- g %>%
  mutate(degree = centrality_degree())

g %>%
  as_tibble() %>%
  arrange(desc(degree)) %>%
  head()
```

```
## # A tibble: 6 × 4
##   actor_id actor_name               id degree
##      <int> <chr>                 <int>  <dbl>
## 1   502126 Cranston, Bryan         113     57
## 2   891094 Gordon-Levitt, Joseph   228     57
## 3   975636 Hardy, Tom              257     57
## 4  1012171 Hemsworth, Chris        272     57
## 5  1713855 Neeson, Liam            466     57
## 6  1114312 Ivanek, Zeljko          304     56
```

There are a number of big name actors on this list who appeared in multiple movies in 2012. Why does Bryan Cranston (actor_id = 502126) have so many connections? The following function will retrieve the list of movies for a particular actor.

```
show_movies <- function(g, id) {
  g %>%
    activate(edges) %>%
    as_tibble() %>%
    filter(src == id | dest == id) %>%
    group_by(movie_id) %>%
    summarize(title = first(title), num_connections = n())
}
```

```
show_movies(g, 502126)
```

```
## # A tibble: 3 × 3
##   movie_id title        num_connections
##      <int> <chr>                  <int>
## 1  3298411 Argo                      19
## 2  3780482 John Carter               19
## 3  4472483 Total Recall              19
```

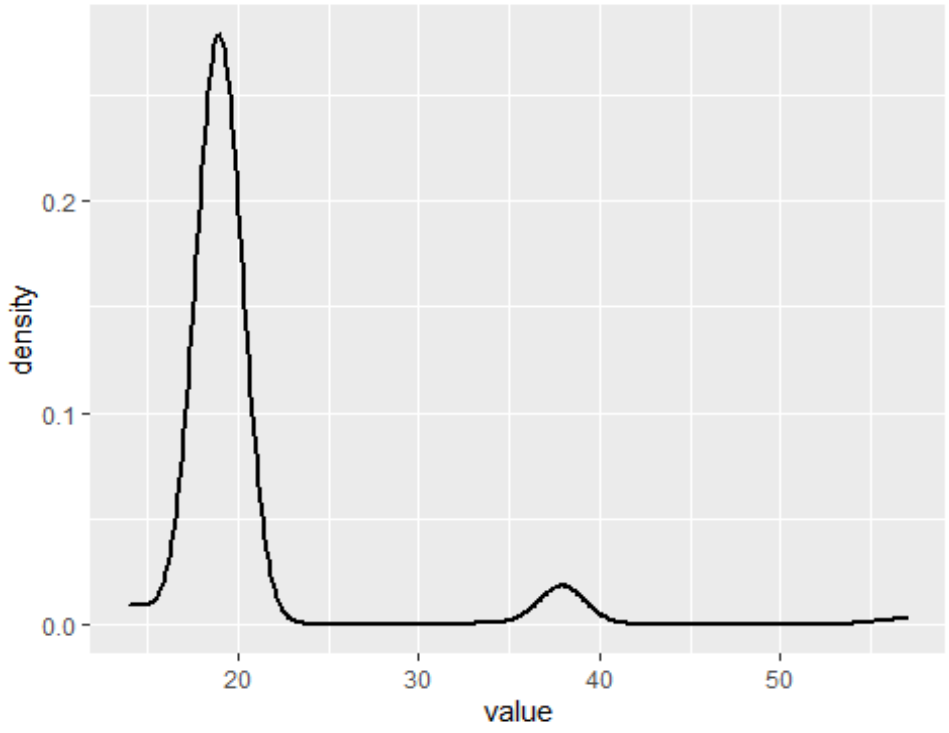
Cranston appeared in all three of these movies.

Exercise 1

- Use the `show_movies` function to determine the 2012 movies that Kristen Stewart (actor_id = 3945132) appeared in. Then use the select and filter functions on as_tibble(g) to determine Kristen Stewart's degree centrality.
- View the list of actors in the `V` data set. Select two that you recognize and make note of their actor_id value. Then determine the 2012 movies that they appeared in and their degree centrality.

Note that the distribution of degrees is not terribly smooth (see figure below). That is, the number of connections that each actor has appears to be limited to a few discrete possibilities. Can you think of why that might be?

```
ggplot(data = enframe(igraph::degree(g)), aes(x = value)) +
  geom_density(size = 1)
```

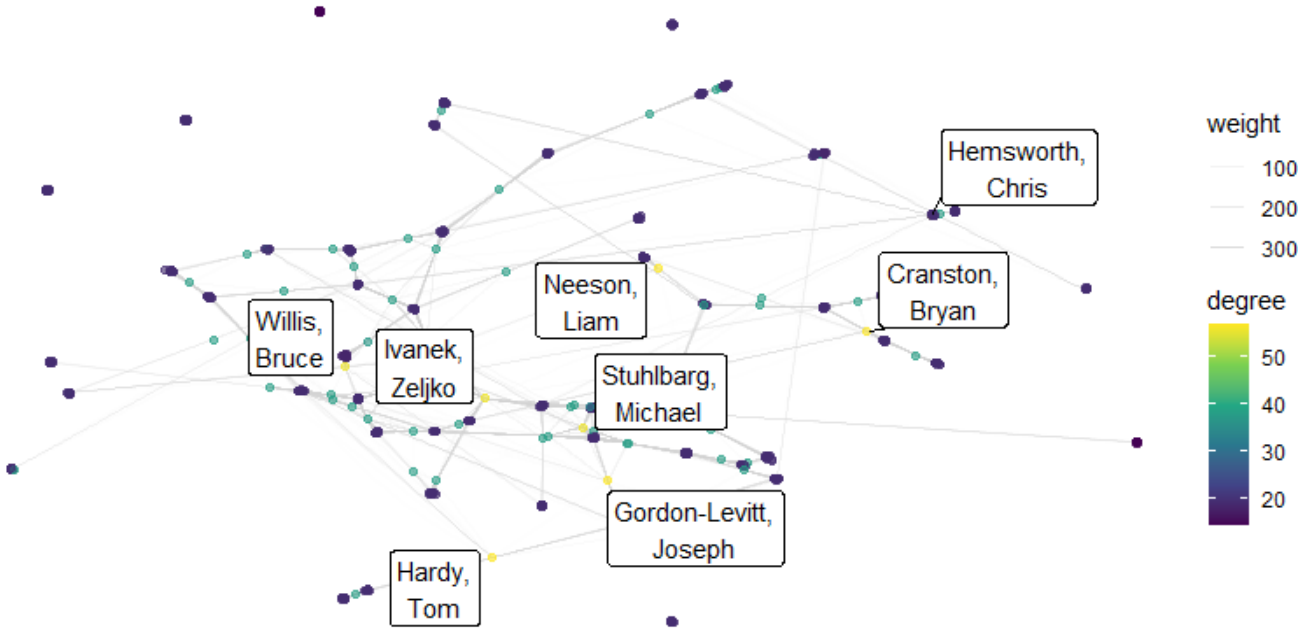


Returning to the network graph, the nodes can be shaded according to their degree centrality and the transparency of the edges can be scaled relative to their `weight` measure.

```
hollywood <- ggraph(g, layout = 'drl') +
  geom_edge_fan(aes(alpha = weight), color = "lightgray") +
  geom_node_point(aes(color = degree), alpha = 0.6) +
  scale_edge_alpha_continuous(range = c(0, 1)) +
  scale_color_viridis_c() +
  theme_void()
```

We don't want to show vertex labels for everyone, because that would result in an unreadable mess. However, it would be nice to see the highly central actors.

```
hollywood +
  geom_node_label(aes(filter = degree > 40, label = str_replace_all(actor_name, ",
", ",\n")), repel = TRUE)
```



The Hollywood network for popular 2012 movies. Color is mapped to degree centrality.

Buidling a Kristen Stewart Oracle

Degree centrality does not take into account the weights on the edges. If we want to emphasize the pathways through leading actors, we could consider betweenness centrality.

```
g <- g %>%
  mutate(btw = centrality_betweenness(weights = weight, normalized = TRUE))

## Warning in betweenness(graph = graph, v = V(graph), directed = directed, :
## 'nobigint' is deprecated since igraph 1.3 and will be removed in igraph 1.4
```

```
g %>%
  as_tibble() %>%
  arrange(desc(btw)) %>%
  head(10)
```

```
## # A tibble: 10 × 5
##    actor_id actor_name                id degree   btw
##       <int> <chr>                  <int>  <dbl> <dbl>
##  1  3945132 Stewart, Kristen         964     38 0.236
##  2   891094 Gordon-Levitt, Joseph    228     57 0.217
##  3  3346548 Kendrick, Anna           857     38 0.195
##  4   135422 Bale, Christian           27     19 0.179
##  5    76481 Ansari, Aziz              15     19 0.176
##  6   558059 Day-Lewis, Daniel        135     19 0.176
##  7  1318021 LaBeouf, Shia            363     19 0.156
##  8  2987679 Dean, Ester              787     38 0.152
##  9  2589137 Willis, Bruce            694     56 0.141
## 10   975636 Hardy, Tom               257     57 0.134
```

Notice that Kristen Stewart has the highest betweenness centrality, while Joseph Gordon-Levitt and Tom Hardy (and others) have the highest degree centrality. Moreover, Christian Bale has the third-highest betweenness centrality despite appearing in only one movie. This is because he played the lead in *The Dark Knight Rises*, the movie responsible for the most edges. Most shortest paths through *The Dark Knight Rises* pass through Christian Bale.

If Kristen Steward is very central to this network, then perhaps instead of a Bacon number, we could consider a Stewart number. Charlize Theron's Stewart number is obviously 1, since they appeared in *Snow White and the Huntsman* together.

```
ks <- V %>%
  filter(actor_name == "Stewart, Kristen")

ct <- V %>%
  filter(actor_name == "Theron, Charlize")

g %>%
  convert(to_shortest_path, from = ks$id, to = ct$id)
```

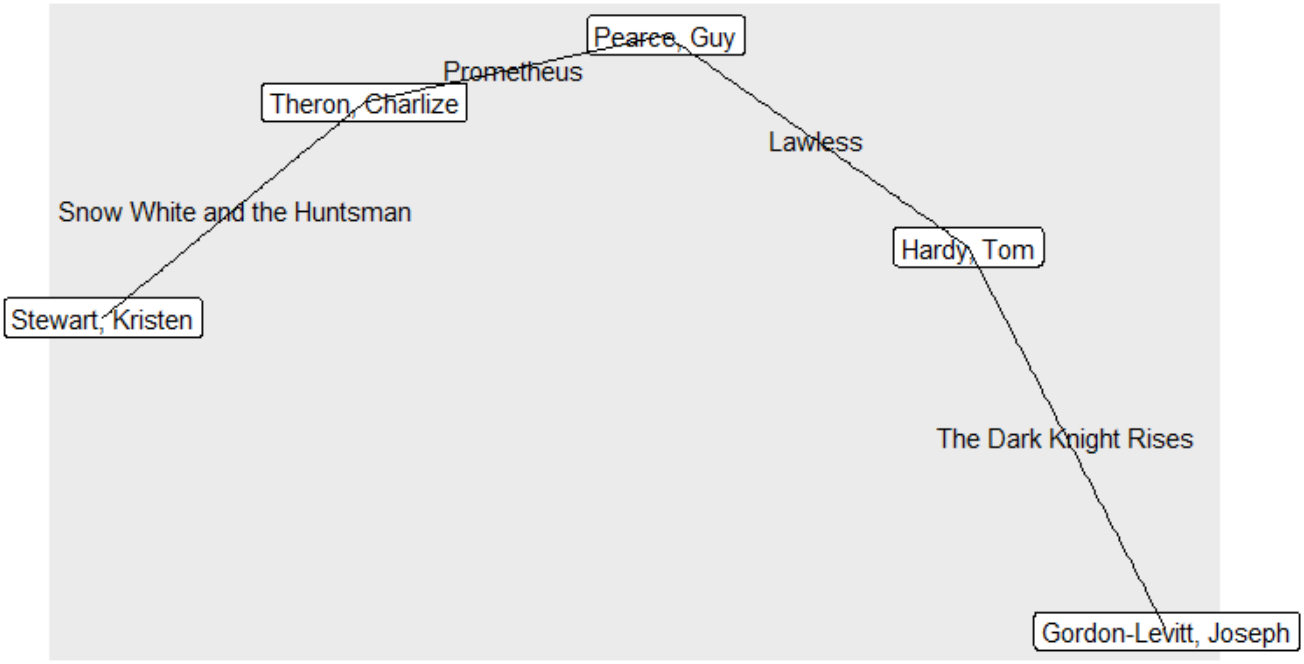
```
## # A tbl_graph: 2 nodes and 1 edges
## #
## # An unrooted tree
## #
## # Node Data: 2 × 6 (active)
##   actor_id actor_name          id degree    btw .tidygraph_node_index
##      <int> <chr>            <int>  <dbl>  <dbl>                 <int>
## 1  3945132 Stewart, Kristen   964     38 0.236                   964
## 2  3990819 Theron, Charlize   974     38 0.0940                  974
## #
## # Edge Data: 1 × 9
##    from    to     src    dest movie_id weight title         ratings .tidygraph_…
##   <int> <int>   <int>   <int>    <int>  <int> <chr>           <int>        <int>
## 1     1     2 3945132 3990819  4237818      3 Snow White a…  243824        10198
```

On the other hand, her distance from Joseph Gordon-Levitt is 4. The interpretation here is that Joseph Gordon-Levitt was in The Dark Night Rises with Tom Hardy, who was in Lawless with Guy Pearce, who was in Prometheus with Charlize Theron, who was in Snow White and the Huntsman with Kristen Stewart.

```
jgl <- V %>%
  filter(actor_name == "Gordon-Levitt, Joseph")

set.seed(47)
h <- g %>%
  convert(to_shortest_path, from = jgl$id, to = ks$id, weights = NA)

h %>%
  ggraph('gem') +
  geom_node_point() +
  geom_node_label(aes(label = actor_name)) +
  geom_edge_fan2(aes(label = title)) +
  coord_cartesian(clip = "off") +
  theme(plot.margin = margin(6, 36, 6, 36))
```



Note, however, that these shortest paths are not unique. In fact, there are 9 shortest paths between Kristen Stewart and Joseph Gordon-Levitt, each having a length of 4.

```
igraph::all_shortest_paths(g, from = ks$id, to = jgl$id, weights = NA) %>%
  pluck("res") %>%
  length()

## [1] 9
```

Exercise 2

For the two actors that you identified in Exercise 1, determine their Stewart number (if it exists). If so, create a visualization of a path from them to Kristen Stewart.

As we saw in the network graph, our Hollywood network is not connected, and thus its diameter is infinite. However, the diameter of the largest connected component can be computed. Thus number (in this case, 10) indicates how many hops separate the two most distant actors in the network.

```
igraph::diameter(g, weights = NA)
```

```
## [1] 10
```

```
g %>%
  mutate(eccentricity = node_eccentricity()) %>%
  filter(actor_name == "Stewart, Kristen")
```

```
## # A tbl_graph: 1 nodes and 0 edges
## #
## # An unrooted tree
## #
## # Node Data: 1 × 6 (active)
##   actor_id actor_name          id degree   btw eccentricity
##      <int> <chr>            <int>  <dbl> <dbl>        <dbl>
## 1  3945132 Stewart, Kristen   964     38 0.236            6
## #
## # Edge Data: 0 × 8
## # … with 8 variables: from <int>, to <int>, src <int>, dest <int>,
## #   movie_id <int>, weight <int>, title <chr>, ratings <int>
```

On the other hand, we note that Kristen Stewart's eccentricity is 6. This means that there is no actor in the connected part of the network who is more than 6 hops away from Kristen Stewart. Six degrees of separation indeed!

References

Modern Data Science with R (2nd ed) - Chapter 20