penguins

```r
knitr::opts_chunk$set(fig.pos = "H", out.extra = "")

# to construct mapper graphs
if (!require(mappeR)) {
  install.packages("mappeR")
}
```

```
## Loading required package: mappeR
```

```r
# to visualize mapper graphs
if (!require(RCy3)) {
  install.packages("RCy3")
}
```

```
## Loading required package: RCy3
```

```r
library(mappeR)
library(RCy3)
```

In this vignette we will explore the `penguins` data set with `mappeR`. The data set contains measurements and characteristics of 344 penguins from three islands in the Palmer Archipelago, Antarctica. It is available natively in R.

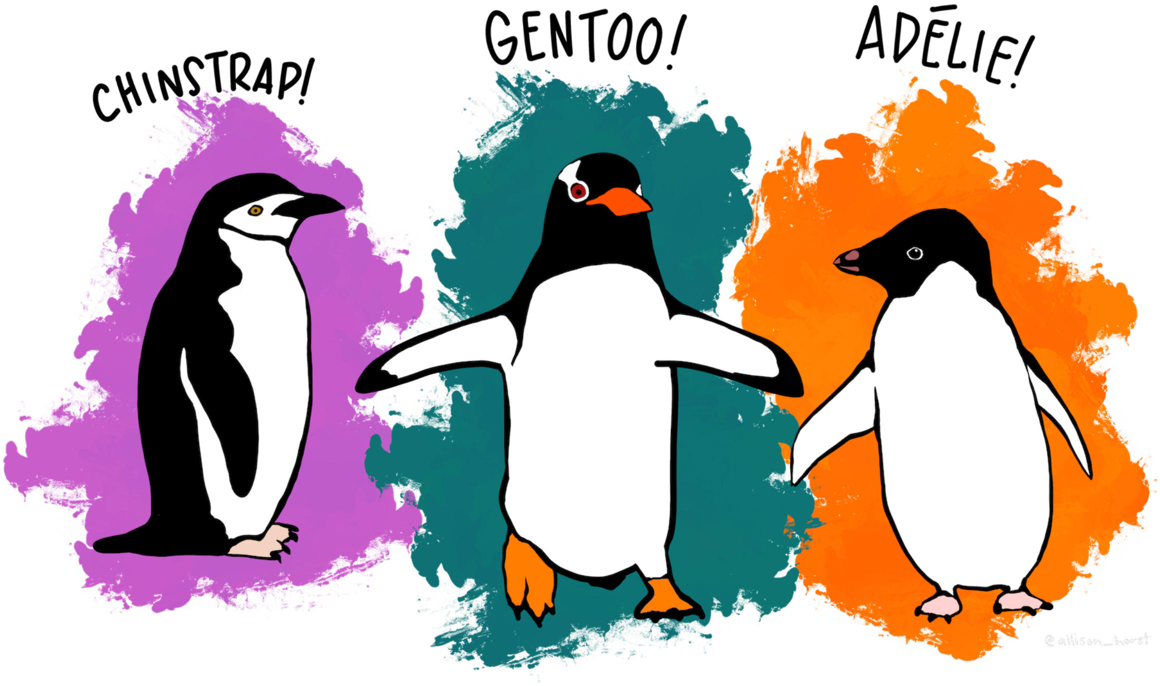


Figure 1: Palmer Penguins Art

```
head(penguins)
```

```
##   species    island bill_len bill_dep flipper_len body_mass    sex year
## 1  Adelie Torgersen     39.1     18.7         181      3750   male 2007
## 2  Adelie Torgersen     39.5     17.4         186      3800 female 2007
## 3  Adelie Torgersen     40.3     18.0         195      3250 female 2007
## 4  Adelie Torgersen       NA       NA          NA        NA   <NA> 2007
## 5  Adelie Torgersen     36.7     19.3         193      3450 female 2007
## 6  Adelie Torgersen     39.3     20.6         190      3650   male 2007
```

For simplicity, we will omit any rows with missing values from our data.

```
clean_penguins = na.omit(penguins)
head(clean_penguins)
```

```
##   species    island bill_len bill_dep flipper_len body_mass    sex year
## 1  Adelie Torgersen     39.1     18.7         181      3750   male 2007
## 2  Adelie Torgersen     39.5     17.4         186      3800 female 2007
## 3  Adelie Torgersen     40.3     18.0         195      3250 female 2007
## 5  Adelie Torgersen     36.7     19.3         193      3450 female 2007
## 6  Adelie Torgersen     39.3     20.6         190      3650   male 2007
## 7  Adelie Torgersen     38.9     17.8         181      3625 female 2007
```

One idea of the Mapper algorithm is to look at data points "in their own space." One way to do this with our penguins is to treat each penguin as a point in $\mathbb{R}^4$, with coordinates given by our features: bill length, bill depth, flipper length, and body mass. To avoid issues of scale caused by these measurements being in different units, we will first normalize each feature via z-scaling; that is, we subtract the mean and divide by the standard deviation so each set of measurements has a mean of 0 and a standard deviation of 1. (There are other methods of normalization, but for the sake of example we'll stick to z-normalizing here.) Once this is done, we'll define pairwise distances between penguins as the usual Euclidean distance metric between their $\mathbb{R}^4$ representations.

```
# get numerical columns only
penguin_numbers = subset(clean_penguins,
                         select = c(bill_len, bill_dep, flipper_len, body_mass))
head(penguin_numbers)
```

```
##   bill_len bill_dep flipper_len body_mass
## 1     39.1     18.7         181      3750
## 2     39.5     17.4         186      3800
## 3     40.3     18.0         195      3250
## 5     36.7     19.3         193      3450
## 6     39.3     20.6         190      3650
## 7     38.9     17.8         181      3625
```

```
# perform z normalization
normal_penguin_numbers = scale(penguin_numbers)
head(normal_penguin_numbers)
```

```
##     bill_len  bill_dep flipper_len  body_mass
## 1 -0.8946955 0.7795590  -1.4246077 -0.5676206
## 2 -0.8215515 0.1194043  -1.0678666 -0.5055254
## 3 -0.6752636 0.4240910  -0.4257325 -1.1885721
## 5 -1.3335592 1.0842457  -0.5684290 -0.9401915
## 6 -0.8581235 1.7444004  -0.7824736 -0.6918109
## 7 -0.9312674 0.3225288  -1.4246077 -0.7228585
```

```r
colMeans(normal_penguin_numbers) # means are zero (up to machine nearsightedness)
```

```
##      bill_len      bill_dep   flipper_len     body_mass
##  3.552797e-16  5.572578e-16  1.834379e-16 -9.274780e-17
```

```r
diag(var(normal_penguin_numbers)) # variances are one
```

```
##    bill_len    bill_dep flipper_len   body_mass
##           1           1           1           1
```

```r
# calculate euclidean distances
penguin_dists = dist(normal_penguin_numbers, method = "euclidean")
```

Now we'll use Ball Mapper to visualize penguin space. The algorithm first creates a cover of the space composed of "penguin ε-balls," which are sets of penguins ε distance away from a central penguin. What it means for these balls to *cover* penguin space is that every penguin lives in at least one ball. The balls may overlap if they are large enough — but the cover is constructed so that no penguin is the center of more than one ball to control its density. We then create a graph structure from the cover by constructing its 1-dimensional nerve, which is a graph whose vertices are penguin balls, and whose edges connect pairs of penguin balls with penguins in common. This gives us a coordinate-free "map" of penguin space we can investigate.

To aid in our exploration, we'll compute some statistics about the balls of penguins in the graph:

-Average bill length

-Average bill depth

-Average flipper length

-Average body mass

-Species count (Adélie, Chinstrap, or Gentoo)

-Island count (Biscoe, Dream, or Torgersen)

-Sex count (male or female)

```r
# generate ball mapper object with an epsilon value of 1 (chosen for example)
penguin_ball_mapper = create_ball_mapper_object(clean_penguins, penguin_dists, 1)

# match the data from each penguin ball to observations in original data set
penguin_balls = lapply(penguin_ball_mapper[[1]]$data, function(x)
  clean_penguins[unlist(strsplit(x, ", ")), ])

# calculate mean statistics for penguins in each vertex
penguin_ball_mapper[[1]]$average_bill_length = sapply(penguin_balls, function(penguin_ball)
  mean(clean_penguins[row.names(penguin_ball), "bill_len"]))
penguin_ball_mapper[[1]]$average_bill_depth = sapply(penguin_balls, function(penguin_ball)
  mean(clean_penguins[row.names(penguin_ball), "bill_dep"]))
penguin_ball_mapper[[1]]$average_flipper_length = sapply(penguin_balls, function(penguin_ball)
  mean(clean_penguins[row.names(penguin_ball), "flipper_len"]))
penguin_ball_mapper[[1]]$average_body_mass = sapply(penguin_balls, function(penguin_ball)
  mean(clean_penguins[row.names(penguin_ball), "body_mass"]))

# count islands in each vertex
penguin_ball_mapper[[1]]$biscoe_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "island"] == "Biscoe", na.rm = TRUE))
penguin_ball_mapper[[1]]$dream_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "island"] == "Dream", na.rm = TRUE))
```

```r
penguin_ball_mapper[[1]]$torgersen_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "island"] == "Torgersen", na.rm = TRUE))

# count species in each vertex
penguin_ball_mapper[[1]]$adelie_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "species"] == "Adelie", na.rm = TRUE))
penguin_ball_mapper[[1]]$chinstrap_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "species"] == "Chinstrap", na.rm = TRUE))
penguin_ball_mapper[[1]]$gentoo_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "species"] == "Gentoo", na.rm = TRUE))

# count sexes in each vertex
penguin_ball_mapper[[1]]$male_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "sex"] == "M", na.rm = TRUE))
penguin_ball_mapper[[1]]$female_count = sapply(penguin_balls, function(penguin_ball)
  sum(clean_penguins[row.names(penguin_ball), "sex"] == "F", na.rm = TRUE))
```

We'll use the RCy3 package to send the graph data over to Cytoscape, a network analysis program which allows us to manipulate style parameters with a nice GUI. It's also possible to interface with it here; see the RCy3 documentation for examples.

```r
# This line requires a running instance of Cytoscape to execute!
createNetworkFromDataFrames(penguin_ball_mapper[[1]], penguin_ball_mapper[[2]])
```

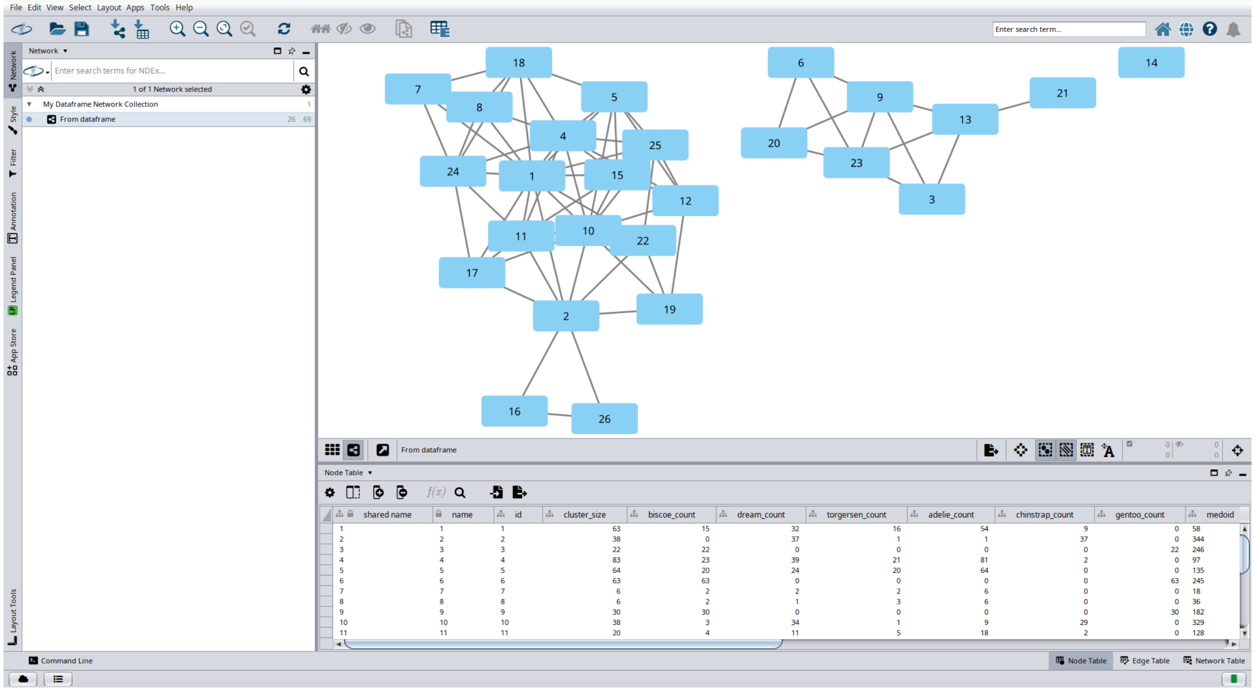
```
## Loading data...

## Applying default style...

## Applying preferred layout...

## networkSUID
##        2615
```

We see a graph with three connected components, presumably representing three different "families" of penguins. One of those is an isolated vertex, which we can see from the Node Table represents just a single penguin (#294, see below) — what makes that one so different from the other two families? Can we tell just from the "map" of penguin space?

```r
penguins[294, ]
```

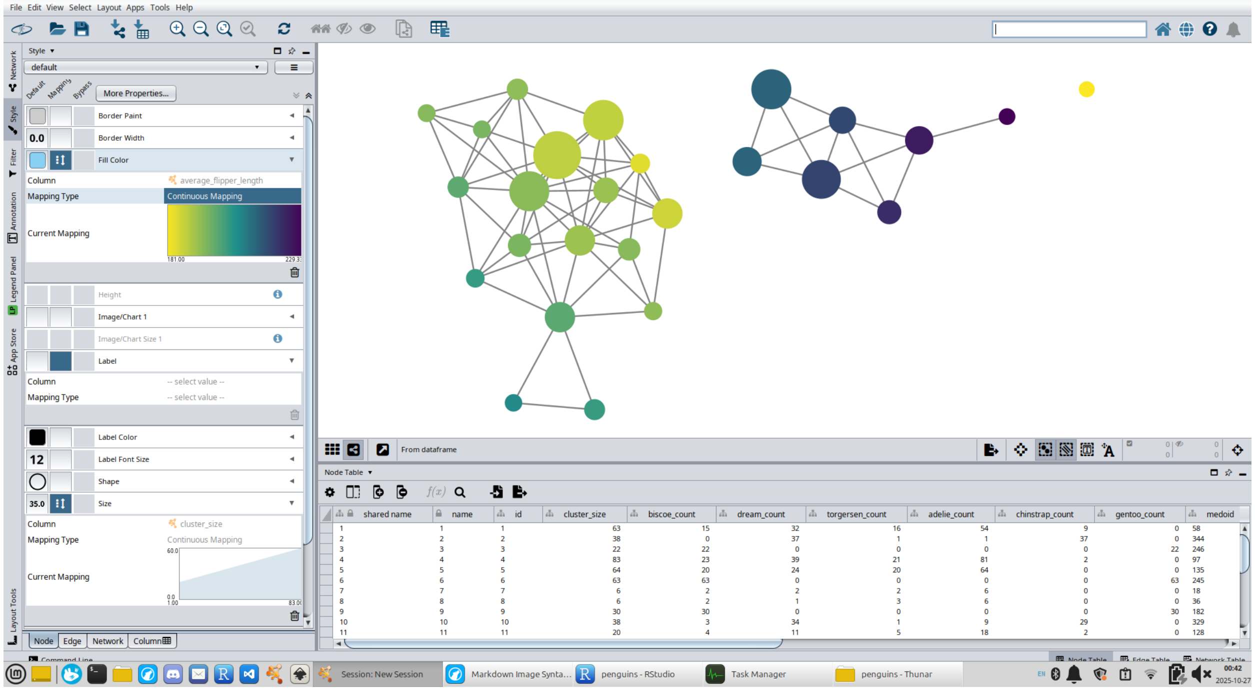
```
##       species island bill_len bill_dep flipper_len body_mass    sex year
## 294 Chinstrap  Dream       58     17.8         181      3700 female 2007
```

Over in the style tab, we've changed the node shape to ellipse and locked node width and height. Now we can start to ask questions about our graph; can we see any organization with respect to any of our features?



shared name	name	id	cluster_size	biscoe_count	dream_count	torgersen_count	adelie_count	chinstrap_count	gentoo_count	medoid
1	1	1	63	15	32	16	54	9	0	58
2	2	2	38	0	37	1	1	37	0	344
3	3	3	22	22	0	0	0	0	22	246
4	4	4	83	23	39	21	81	2	0	97
5	5	5	64	20	24	20	64	0	0	135
6	6	6	63	63	0	0	0	0	63	245
7	7	7	6	2	2	2	6	0	0	18
8	8	8	6	2	1	3	6	0	0	36
9	9	9	30	30	0	0	0	0	30	182
10	10	10	38	3	34	1	9	29	0	329
11	11	11	20	4	11	5	18	2	0	128

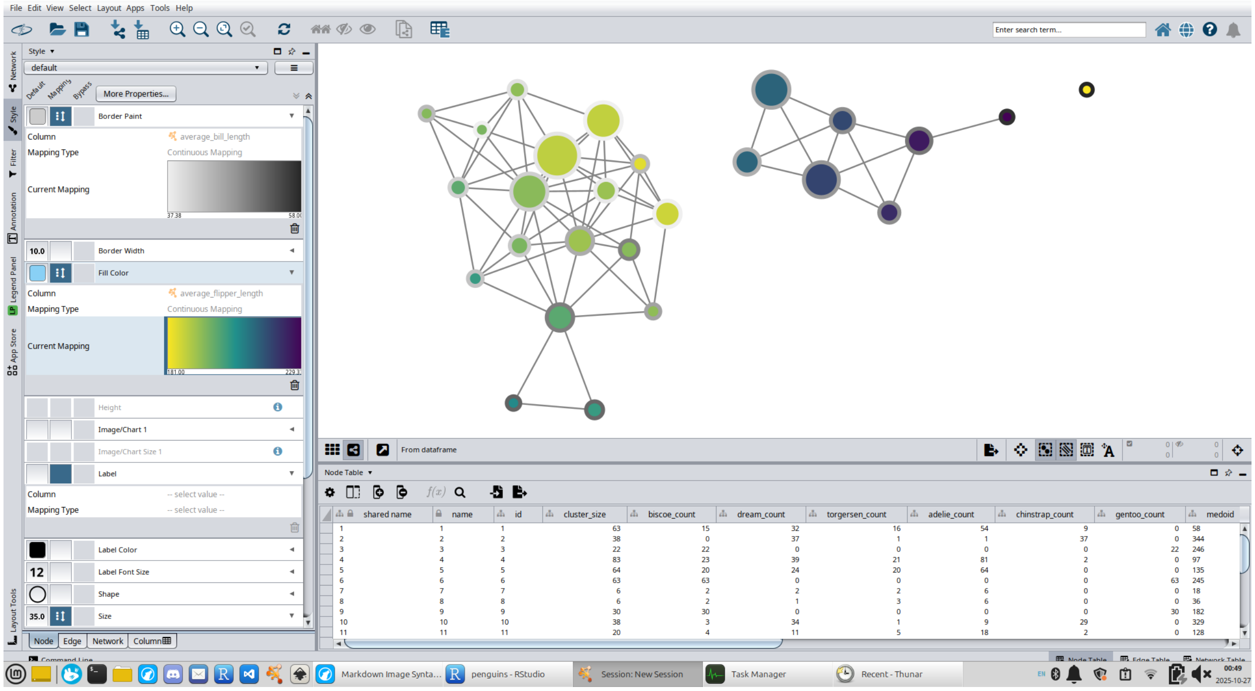
Figure 2: Initial Cytoscape Screen



Border Paint
0.0 Border Width
Fill Color
Column: average_flipper_length
Mapping Type: Continuous Mapping
Current Mapping: 181.00 – 229.33
Height
Image/Chart 1
Image/Chart Size 1
Label
Column: -- select value --
Mapping Type: -- select value --
Label Color
12 Label Font Size
○ Shape
35.0 Size
Column: cluster_size
Mapping Type: Continuous Mapping
Current Mapping: 60.0 / 0.0; 1.00 – 83.00

shared name	name	id	cluster_size	biscoe_count	dream_count	torgersen_count	adelie_count	chinstrap_count	gentoo_count	medoid
1	1	1	63	15	32	16	54	9	0	58
2	2	2	38	0	37	1	1	37	0	344
3	3	3	22	22	0	0	0	0	22	246
4	4	4	83	23	39	21	81	2	0	97
5	5	5	64	20	24	20	64	0	0	135
6	6	6	63	63	0	0	0	0	63	245
7	7	7	6	2	2	2	6	0	0	18
8	8	8	6	2	1	3	6	0	0	36
9	9	9	30	30	0	0	0	0	30	182
10	10	10	38	3	34	1	9	29	0	329
11	11	11	20	4	11	5	18	2	0	128

Figure 3: Adding Flipper Length Node Fill Coloring



Border Paint
Column: average_bill_length
Mapping Type: Continuous Mapping
Current Mapping: 37.38 – 58.00

10.0 Border Width

Fill Color
Column: average_flipper_length
Mapping Type: Continuous Mapping
Current Mapping: 181.00 – 229.33

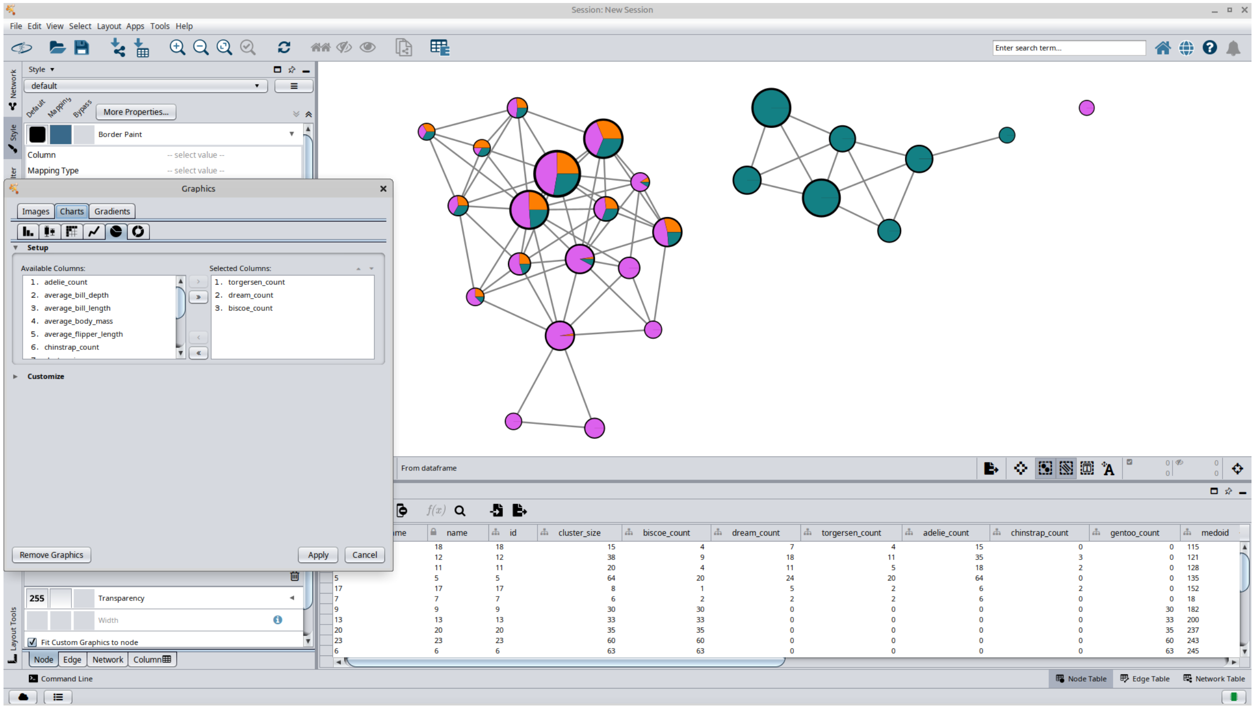
Height
Image/Chart 1
Image/Chart Size 1

Label
Column: -- select value --
Mapping Type: -- select value --

■ Label Color
12 Label Font Size
○ Shape
35.0 Size

shared name	name	id	cluster_size	biscoe_count	dream_count	torgersen_count	adelie_count	chinstrap_count	gentoo_count	medoid
1	1	1	63	15	32	16	54	9	0	58
2	2	2	38	0	37	1	1	37	0	344
3	3	3	22	22	0	0	0	0	22	246
4	4	4	83	23	39	21	81	2	0	97
5	5	5	64	20	24	20	64	0	0	135
6	6	6	63	63	0	0	0	0	63	245
7	7	7	6	2	2	2	6	0	0	18
8	8	8	6	2	1	3	6	0	0	36
9	9	9	30	30	0	0	0	0	30	182
10	10	10	38	3	34	1	9	29	0	329
11	11	11	20	4	11	5	18	2	0	128

Here is one way to compare two variables at once. We have kept the flipper length node fill coloring as before, but now we have augmented the graph with node border colors. Nodes with lighter borders contain penguins with shorter bill lengths, and darker nodes contain penguins with longer bill lengths. We can see that longer bill lengths belong to all connected components of the graph, with the singleton node representing a penguin with quite a large one!



■ Border Paint
Column: -- select value --
Mapping Type: -- select value --

Available Columns:
1. adelie_count
2. average_bill_depth
3. average_bill_length
4. average_body_mass
5. average_flipper_length
6. chinstrap_count

Selected Columns:
1. torgersen_count
2. dream_count
3. biscoe_count

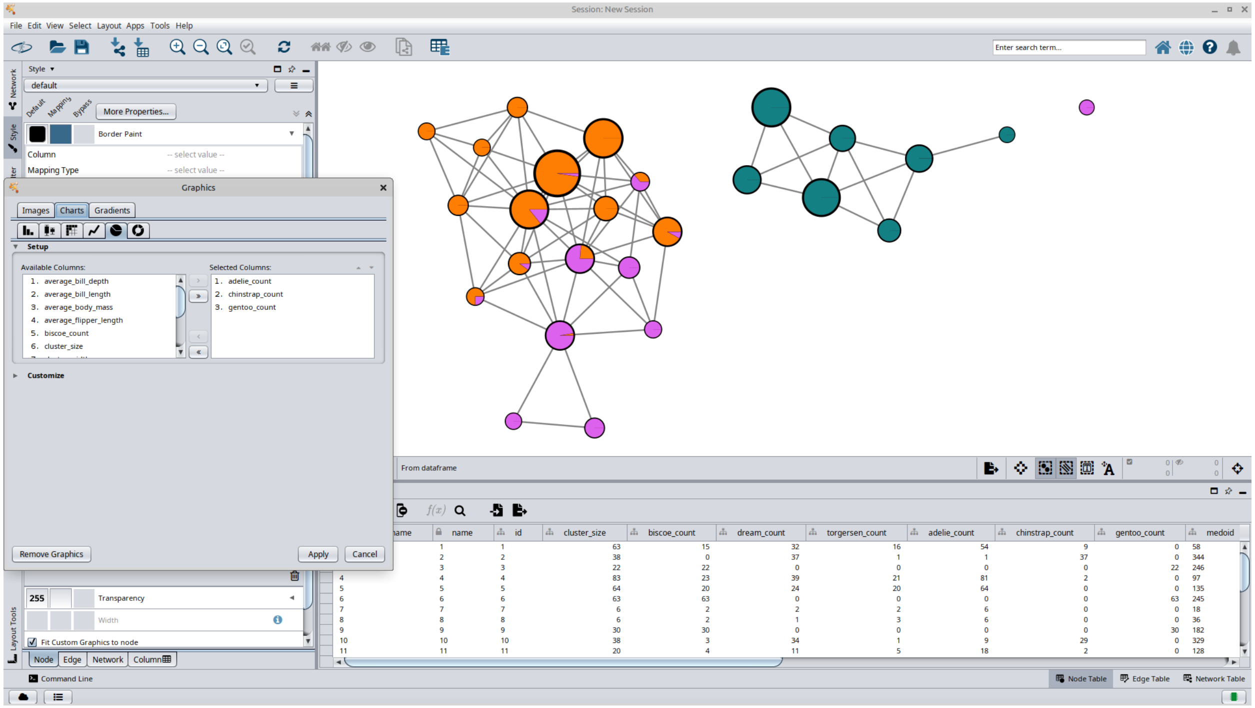
255 Transparency
Width
☑ Fit Custom Graphics to node

shared name	name	id	cluster_size	biscoe_count	dream_count	torgersen_count	adelie_count	chinstrap_count	gentoo_count	medoid
	18	18	15	4	7	4	15	0	0	115
	12	12	38	9	18	11	35	3	0	121
	11	11	20	4	11	5	18	2	0	128
5	5	5	64	20	24	20	64	0	0	135
17	17	17	8	1	5	2	6	2	0	152
7	7	7	6	2	2	2	6	0	0	18
9	9	9	30	30	0	0	0	0	30	182
13	13	13	33	33	0	0	0	0	33	200
20	20	20	35	35	0	0	0	0	35	237
23	23	23	60	60	0	0	0	0	60	243
6	6	6	63	63	0	0	0	0	63	245

Figure 4: Adding Island Membership Pie Charts to Nodes

Here we have used some custom graphics to make pie charts showing island membership per node. Torgersen

Island is orange, Dream Island is purple, and Biscoe Island is teal. We can see that Torgersen Island appears to be the most diverse; penguins living there are similar to ones living on the other two islands. The opposite is true for Biscoe Island, whose members are generally more similar to each other than those living on the other islands — an entire connected component is composed of only Biscoe natives!



Border Paint
Column -- select value --
Mapping Type -- select value --

Graphics

Images | Charts | Gradients

Setup

Available Columns:
1. average_bill_depth
2. average_bill_length
3. average_body_mass
4. average_flipper_length
5. biscoe_count
6. cluster_size

Selected Columns:
1. adelie_count
2. chinstrap_count
3. gentoo_count

Customize

Remove Graphics | Apply | Cancel

255 Transparency
Width
☑ Fit Custom Graphics to node

ame	name	id	cluster_size	biscoe_count	dream_count	torgersen_count	adelie_count	chinstrap_count	gentoo_count	medoid
	1	1	63	15	32	16	54	9	0	58
	2	2	38	0	37	1	1	37	0	344
	3	3	22	22	0	0	0	0	22	246
4	4	4	83	23	39	21	81	2	0	97
5	5	5	64	20	24	20	64	0	0	135
6	6	6	63	63	0	0	0	0	63	245
7	7	7	6	2	2	2	6	0	0	18
8	8	8	6	2	1	3	6	0	0	36
9	9	9	30	30	0	0	0	0	30	182
10	10	10	38	3	34	1	9	29	0	329
11	11	11	20	4	11	5	18	2	0	128

Figure 5: Adding Species Membership Pie Charts to Nodes

We can do a similar pie chart analysis of the species of our penguins in the Ball Mapper graph. Here, Adelie penguins are orange, Chinstrap penguins are purple, and Gentoo penguins are teal. Most of the nodes are composed of just one species of penguin, with one connected component — made up of penguins on Biscoe Island — entirely composed of Gentoo penguins. There *do* exist nodes in the other connect component with both Adelie and Chinstrap penguins, suggesting there are members of those species that are similar to each other. Representatives from these nodes can be found in the original data set through the `medoid` column, which contains the labels of the data points in each node with a minimum sum of distances to every other data point.