PCA (rda function) output

- prin_comp$rda(object)
- summary(prin_comp)
- prin_comp$CA
- prin_comp$Xbar
- prin_comp$Tot.chi
- prin_comp$v.eig
- prin_comp$u.eig

Within the object returned by the function rda(), results are within CA object (?)ca.object):
- prin_comp$CA contains
  - eig: eigenvalues
  - u: site scores
  - v: variable scores (eigenvectors)
  - u: site scores scaled by eigenvalues
  - v: variable scores scaled by eigenvalues
  - Tot.chi: total inertia
  - Xbar: centered and scaled data matrix

Repeat the same analysis, this time with species 1 randomized (sampled without replacement)

PCA (rda function) output

- v: variable scores or loadings, listed as "species scores" by rda function
- Represent the relationship between each variable and the axes.

- Look for where each variable has its highest (greatest magnitude) loading. Remember that early axes are more important (explain more variation).
How do you know what “good” loadings are?

- Axis score-variable correlations are difficult to interpret
- Convention is some arbitrary eigenvalue cut off values
- Permutation approach (see script and paper) to assess “significance”
  - Randomizes matrix (sample without replacement), calculates loadings, builds distribution

GIVING MEANINGFUL INTERPRETATION TO ORDINATION AXES: ASSESSING LOADING SIGNIFICANCE IN PRINCIPAL COMPONENT ANALYSIS

Pierko R. Peres-Neto, Donald A. Jackson, and Keith M. Somers
Department of Zoology, University of Toronto, Toronto, Ontario, Canada M5S 3G7

Biplot function
- Function `biplot()` (stats package, should load automatically) will plot sample points and loadings/scores as vectors.
  
```r
prin_comp$rda(community, scaled=T)
biplot(prin_comp$CA$u, prin_comp$CA$v)
```

How well does the analysis find and summarize structure?

- Sample scores are calculated from centered and scaled data multiplied by the eigenvector matrix.
  - In an `rda` object, these are returned as `CA$u`, the centered and scaled data is returned as `CA$Xbar`.

```r
PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9    PC10
Inertia Proportion
0.4567 0.7798 0.8822 0.6541 0.4628 0.1468
Cumulative Proportion
0.2280 0.3935 0.5418 0.3119 0.2264 0.2029
```

- Because they are centered and scaled, PCA points are centered at the origin.

### PCA

```
> prin_comp$rda(community, scaled=T)
```

```
> biplot(prin_comp$CA$u, prin_comp$CA$v)
```

---

3/7/2018
Scores function
- Different software/functions will scale points and variable scores differently.
- The `scores` function (vegan) returns scores for samples and variables.
- `scores(prin_comp)` – returns an object with sample and site scores.

Projecting Scores
- This also means that the PCA can serve as a model. New data can be "projected" into PCA space without changing the underlying model (re-running the PCA).
- Function `predict` will project new data (must have the same variables with the same names)
  ```r
  predict(prin_comp, new_data)
  ```

Table 9.1 from Legendre and Legendre (1998)

<table>
<thead>
<tr>
<th>Method</th>
<th>Distance</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>Euclidean</td>
<td>Quantitative, linear relationships assumed, beware of double-zeros</td>
</tr>
<tr>
<td>PCoA</td>
<td>Any</td>
<td>Quantitative, qualitative or mixed</td>
</tr>
<tr>
<td>NMDS</td>
<td>Any</td>
<td>Quantitative, qualitative or mixed</td>
</tr>
<tr>
<td>CA</td>
<td>$\chi^2$</td>
<td>Non-negative, quantitative or binary</td>
</tr>
<tr>
<td>FA</td>
<td>Euclidean</td>
<td>Quantitative, linear relationships assumed, beware of double-zeros</td>
</tr>
</tbody>
</table>

Axis Rotation
- PCA constructs the new variables (axes) to maximize variance explained. Loadings are used to relate variables to axes.
- What if you want the relationship between variables and axes to be clearer?
- You can have the analysis "rotate" axes to maximize loadings with variables.
- The tradeoff is that less overall variance will be explained
Factor Analysis

- Function `factanal()`
  - Data matrix
  - Number of factors
  - Method of calculating scores
  - Type of rotation (none = PCA)
- First, try factor analysis with no rotation. Results look very similar to a PCA

```r
fac <- factanal(community, 6, scores="regression", rotation="none")
```

<table>
<thead>
<tr>
<th>Factor1</th>
<th>Factor2</th>
<th>Factor3</th>
<th>Factor4</th>
<th>Factor5</th>
<th>Factor6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS loadings</td>
<td>4.523</td>
<td>3.215</td>
<td>1.033</td>
<td>0.939</td>
<td>0.096</td>
</tr>
<tr>
<td>Proportion Var</td>
<td>0.452</td>
<td>0.322</td>
<td>0.103</td>
<td>0.094</td>
<td>0.010</td>
</tr>
<tr>
<td>Cumulative Var</td>
<td>0.452</td>
<td>0.774</td>
<td>0.877</td>
<td>0.971</td>
<td>0.981</td>
</tr>
</tbody>
</table>

Sample Grouping

- Function `dataEllipse` (car package) will add a centroid and confidence ellipse to a group of points
- 75% confidence ellipses for groups (creek_factor):

```r
dataEllipse(prin_coord$points[,1:2], add=T, levels=0.75, group=creek_factor)
```

Factor Analysis

```r
fac <- factanal(community, 6, scores="regression", rotation="varimax")
```

<table>
<thead>
<tr>
<th>Factor1</th>
<th>Factor2</th>
<th>Factor3</th>
<th>Factor4</th>
<th>Factor5</th>
<th>Factor6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS loadings</td>
<td>3.144</td>
<td>2.820</td>
<td>2.731</td>
<td>0.932</td>
<td>0.174</td>
</tr>
<tr>
<td>Proportion Var</td>
<td>0.314</td>
<td>0.282</td>
<td>0.273</td>
<td>0.093</td>
<td>0.017</td>
</tr>
<tr>
<td>Cumulative Var</td>
<td>0.314</td>
<td>0.596</td>
<td>0.869</td>
<td>0.963</td>
<td>0.980</td>
</tr>
</tbody>
</table>

Loadings:

<table>
<thead>
<tr>
<th>Factor1</th>
<th>Factor2</th>
<th>Factor3</th>
<th>Factor4</th>
<th>Factor5</th>
<th>Factor6</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp1</td>
<td>-0.419</td>
<td>0.153</td>
<td>0.849</td>
<td>0.121</td>
<td>0.244</td>
</tr>
<tr>
<td>sp2</td>
<td>0.178</td>
<td>0.507</td>
<td>0.683</td>
<td>0.441</td>
<td>0.130</td>
</tr>
<tr>
<td>sp3</td>
<td>0.210</td>
<td>0.762</td>
<td>0.394</td>
<td>0.460</td>
<td></td>
</tr>
<tr>
<td>sp4</td>
<td>0.647</td>
<td>0.687</td>
<td>0.147</td>
<td>0.211</td>
<td>-0.101</td>
</tr>
<tr>
<td>sp5</td>
<td>0.950</td>
<td>0.276</td>
<td>-0.111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sp6</td>
<td>0.950</td>
<td>-0.276</td>
<td>0.111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sp7</td>
<td>0.647</td>
<td>-0.687</td>
<td>-0.147</td>
<td>0.211</td>
<td></td>
</tr>
<tr>
<td>sp8</td>
<td>0.210</td>
<td>-0.762</td>
<td>-0.394</td>
<td>0.460</td>
<td></td>
</tr>
<tr>
<td>sp9</td>
<td>-0.178</td>
<td>-0.507</td>
<td>-0.683</td>
<td>0.441</td>
<td>0.130</td>
</tr>
<tr>
<td>sp10</td>
<td>-0.419</td>
<td>-0.153</td>
<td>-0.849</td>
<td>0.121</td>
<td>0.244</td>
</tr>
</tbody>
</table>

Sample Grouping

- Another approach is to plot minimum convex polygons for groups of samples.
- Function `chull` (grDevices package) returns the points that form a polygon for a groups of samples.

```r
chull(prin_coord$points[,1:2])
```

| [1] 2 1 3 4 5 7 10 11 12 13 14 15 |

- Function `polygon` (graphics package) will add a polygon to your plot.

```r
gmphc-prin_coord$points[1:15,1:2]
polygon(grmphc[chull(prin_coord$points[,1:2]),],col=rgb(1,0,0,0.3))
```