Dimension Reduction and Clustering

Zack Treisman

Spring 2021

WESTERN
COLORADO UNIVERSITY
MATHEMATICS & COMPUTER
SCIENCE DEPARTMENT

```r
library(ggplot2)
library(dplyr)
library(tidyr)
library(vegan)
#library(devtools)
#install_github("ggbiplot")
#library(ggbiplot)

set.seed(23)
```

Philosophy

There are many situations where we would like to discover structure in data without explicitly specifying a predictor-response relationship between variables. Generally, this means creating new variables out of combinations of existing variables that provide more concise summaries.

- Data with correlated predictors or high dimension (many predictors) can be problematic.
  - Correlation leads to high variance in parameter estimates and significance levels.
  - The curse of dimsionality (eg. $1 - 0.9^{20} \approx 0.89$) means that high dimensional data are sparse, and statistical power is hard to come by.
- Reparameterizing may remove correlations and reduce dimension.

Dimension Reduction

Variable selection is a *supervised* technique for dimension reduction (see also partial least squares), meaning that a response variable guides the process.

Unsupervised techniques create a set of new variables to replace the existing ones. No response variable is specified in the algorithms. Instead, we search for pattern within the predictors themselves. These algorithms fall into two categories, according to the sort of variable created.

- **Ordination** refers to tools that create new numeric variables.
  - PCA, NMDS, etc.
- Tools that perform **clustering** create new categorical variables.
  - K-means, hierarchical clustering.

Scaling and centering

All of the methods that we'll discuss expect that all variables in the data are either recorded on the same measurement scale or have been rescaled so that the numerical ranges are comparable.

- Divide each variable by its standard deviation/(max-min)/IQR.
- Centering is also a helpful thing to do. Subtract the mean of each variable from each observation, so that all variables have mean 0.

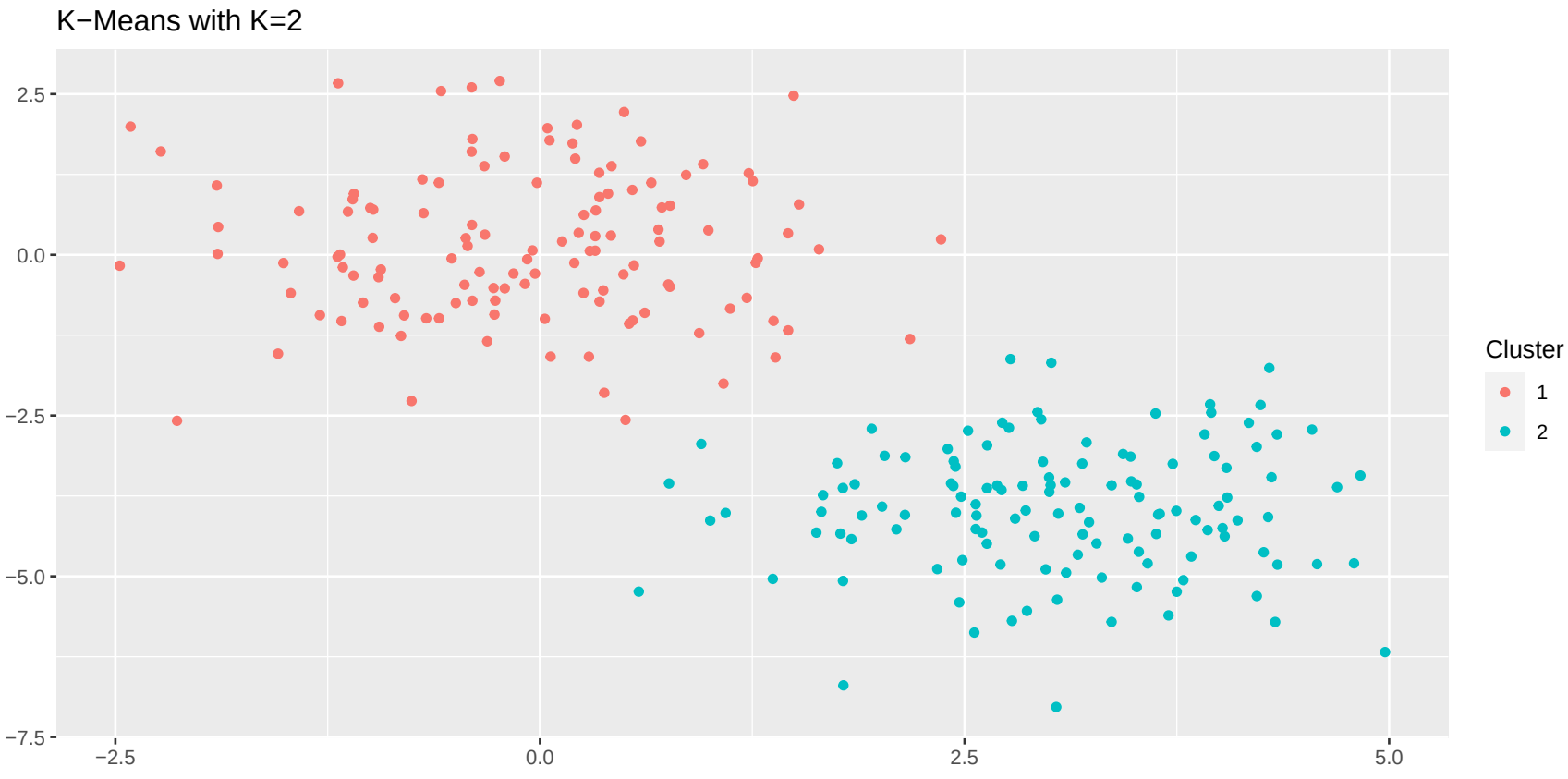
K-means clustering

Suppose you have a set of numeric variables $X = (X_1, \ldots, X_p)$, and a collection of observations $x_1, \ldots, x_n$, with $x_i = (x_{i1}, \ldots, x_{ip})$.

1. Choose a positive integer K. This is how many groups you plan to create in the data.
2. Randomly select points $C_1, \ldots, C_K$ with coordinates in the range determined by the data.
3. For each observation x_i, assign it to the group j for which the distance from x_i to the point C_j is the smallest.
4. For each group j, compute the centroid of the observations in that group and replace C_j with this point.
5. Repeat steps 3 and 4 until no points change groups in step 3 or some predefined stopping criterion is met.

K-means example

```
x=matrix(rnorm(250*2), ncol=2) # 250 random 2D points
x[1:125,1]=x[1:125,1]+3; x[1:125,2]=x[1:125,2]-4 # Shift half right and down
km.out=kmeans(x,2,nstart=20) # perform K-means with 2 clusters
ggplot(data.frame(x), aes(X1, X2, color=factor(km.out$cluster))) +
    geom_point()+labs(title="K-Means with K=2", x="", y="", color="Cluster")
```



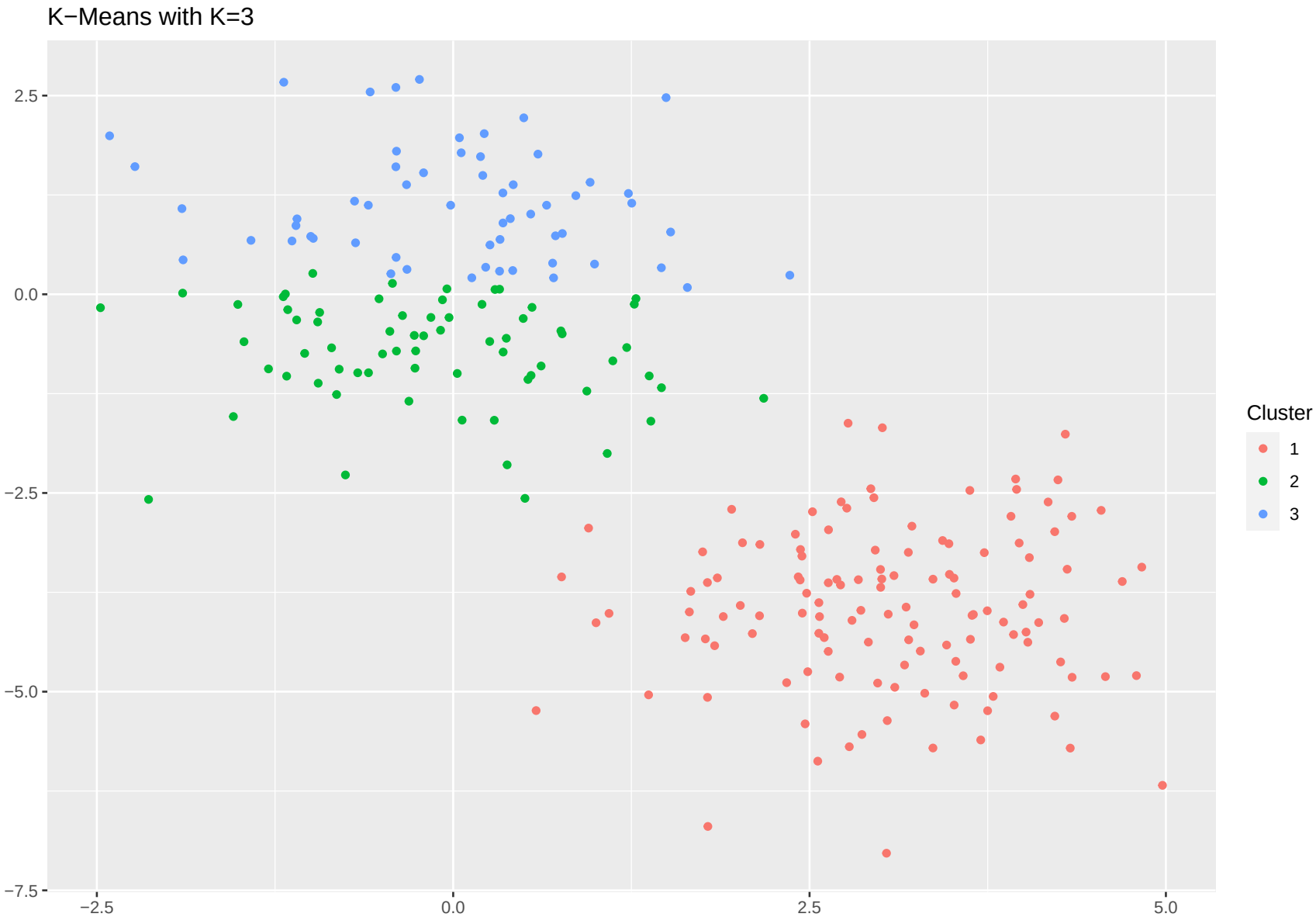
K-means example (cont.)

```
km.out
```

```
## K-means clustering with 2 clusters of sizes 125, 125
##
## Cluster means:
##          [,1]       [,2]
## 1 -0.08315394  0.1278647
## 2  3.05729531 -3.9631181
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 280.3319 232.1237
##  (between_SS / total_SS =  76.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

The number of clusters is a parameter to choose

```
km.out=kmeans(x,3,nstart=20)
```



Another parameter is how many times to restart the algorithm

A bad initial choice of $C_1 \ldots C_K$ can lead the algorithm to a local minimum that's not optimal.

```
set.seed(1)
km.out=kmeans(x,5,nstart=1)
km.out$tot.withinss
```

```
## [1] 317.9313
```

```
km.out=kmeans(x,5,nstart=20)
km.out$tot.withinss
```

```
## [1] 268.3861
```

Extensions and other considerations

- K-means-like algorithms with other notions of distance can be done with the `flexclust` package.
- Data that change multiplicatively (eg. percentages) should perhaps be log transformed first.
- When plotting clusters in high dimensional data, it may not be possible to see the separation in a plot of only two dimensions.

Hierarchical clustering

A totally different method of clustering is *hierarchical clustering*

1. Each observation starts in its own cluster.
2. Pairwise distances between clusters are computed.
3. The two closest clusters are merged.
4. Repeat steps 2 and 3 until all observations are in one cluster.
5. The resulting tree of merging clusters can be cut at various levels to give set numbers of clusters.

There are many ways to compute the distances in step 2.

West Fork Fire Complex Data

This is a project I'm working on with Jonathan Coop. We're looking at vegetation data from sites in the San Juans recovering from impacts of beetles and fire.

```
veg_data <- read.csv("data/West_Fork_Plants.csv")
set.seed(1)
sample_n(veg_data,10)[1:8]
```

```
##        Plot_ID     Type  Code Stratum Cover Height      Genus      Species
## 1    F_24_720   Burned ASAL7       5   0.5    0.1 Astragalus      alpinus
## 2   F_14_5080   Burned POGR9       5   0.1    0.1 Potentilla     gracilis
## 3   F_14_8494   Burned DAPA2       5   2.5    0.5  Danthonia       parryi
## 4   N_31_5843 Unburned POPU3       5   0.5    0.2 Polemonium   pulchellum
## 5   F_24_4594   Burned CARO5       5   2.5    0.2      Carex       rossii
## 6   N_14_9086 Unburned  PIEN       2  17.5    3.0      Picea  engelmannii
## 7   F_33_3456   Burned ELTR7       5   0.5    0.6     Elymus trachycaulus
## 8   N_41_5768 Unburned  ABLA       2   2.5    3.5      Abies   lasiocarpa
## 9  N_41_10975 Unburned  OSDE       5   2.5    0.5  Osmorhiza  depauperata
## 10 N_34_11403 Unburned POTR5       3   0.1    1.0    Populus  tremuloides
```

Pivot Wider

Create a data matrix where the rows are sites and the columns are species, with the entries percent cover for that species at that site.

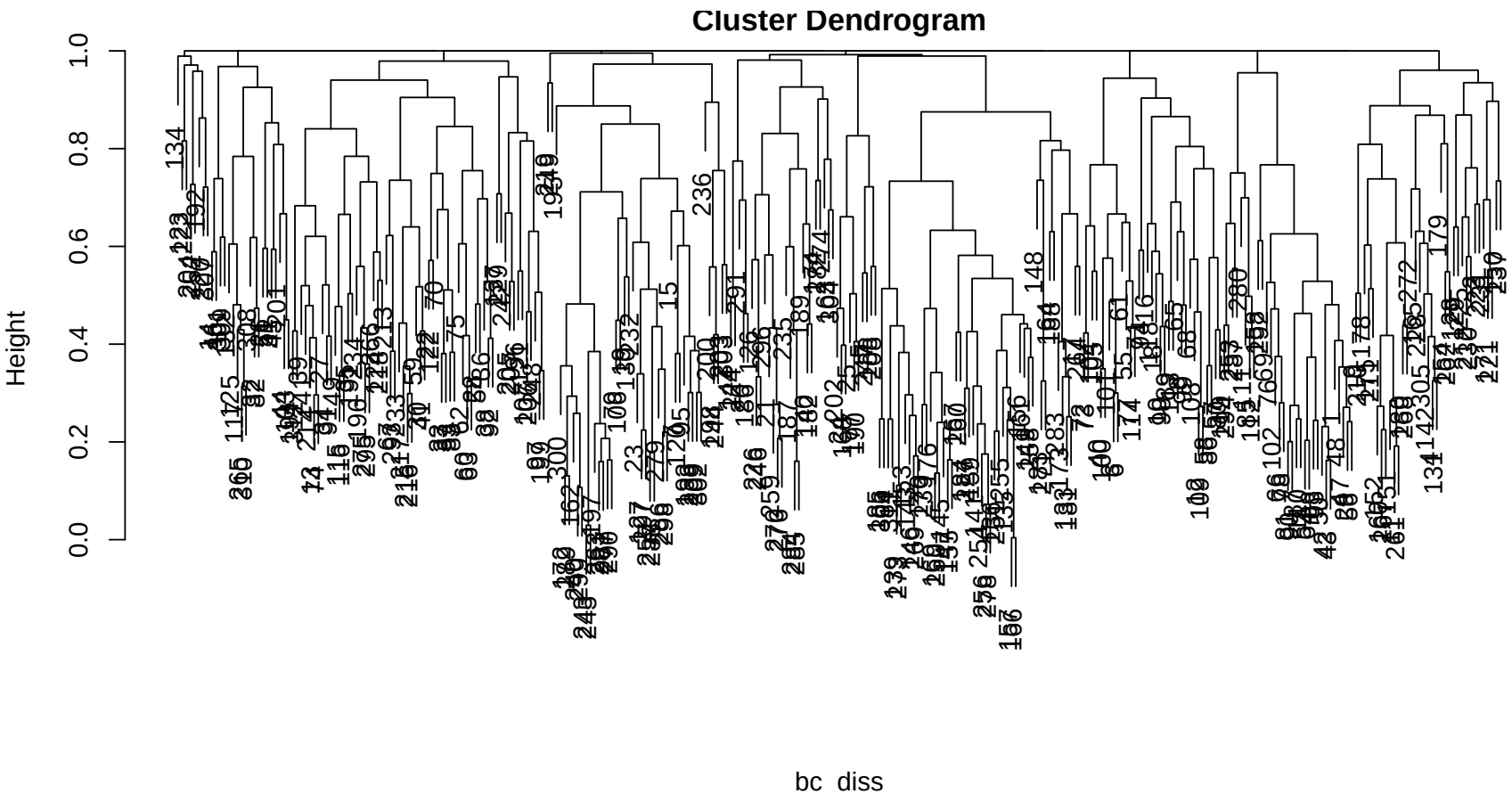
```
spp_cov<-as_tibble(veg_data[c("Plot_ID", "Code", "Cover")])
spp_cov<-with(spp_cov, spp_cov[order(spp_cov$Code), ])
spp_cov<-spp_cov[!(spp_cov$Code == "Unknown"),] # get rid of unknowns
spp_cov_matrix<-spp_cov %>% pivot_wider(names_from = Code,
                                        values_from = Cover,
                                        values_fn = sum)
spp_cov_matrix[is.na(spp_cov_matrix)] <- 0 # zeros instead of NAs
head(spp_cov_matrix[,1:10],10)
```

```
## # A tibble: 10 x 10
##    Plot_ID     ABCO  ABLA ACCO4  ACGL ACMI2 ACRU2  ADMO  ADPE AGAU2
##    <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 F_14_6810   17.5  49       0   0     0.1     0     0     0   0
##  2 F_32_9673    2.5   3       0   2.5   0       0     0     0   0
##  3 F_13_4275    0    39       0   0     0       0     0     0   0
##  4 F_13_6435    0     0.5     0   0     0.5     0     0     0   0
##  5 F_14_4273    0     0.1     0   0     0       0     0     0   0
##  6 F_14_4283    0    51.5     0   0     0       0     0     0   0
##  7 F_14_9251    0    12.5     0   0     0       0     0     0   0
##  8 F_22_10708   0    18.5     0   0     0       0     0     0   0.1
##  9 F_22_9346    0     7.5     0   0     0       0     0     0   0
## 10 F_23_3312    0    10       0   0     0       0     0     0   0
```

Compute the tree

Use Bray-Curtis dissimilarity to measure "distance" between clusters.

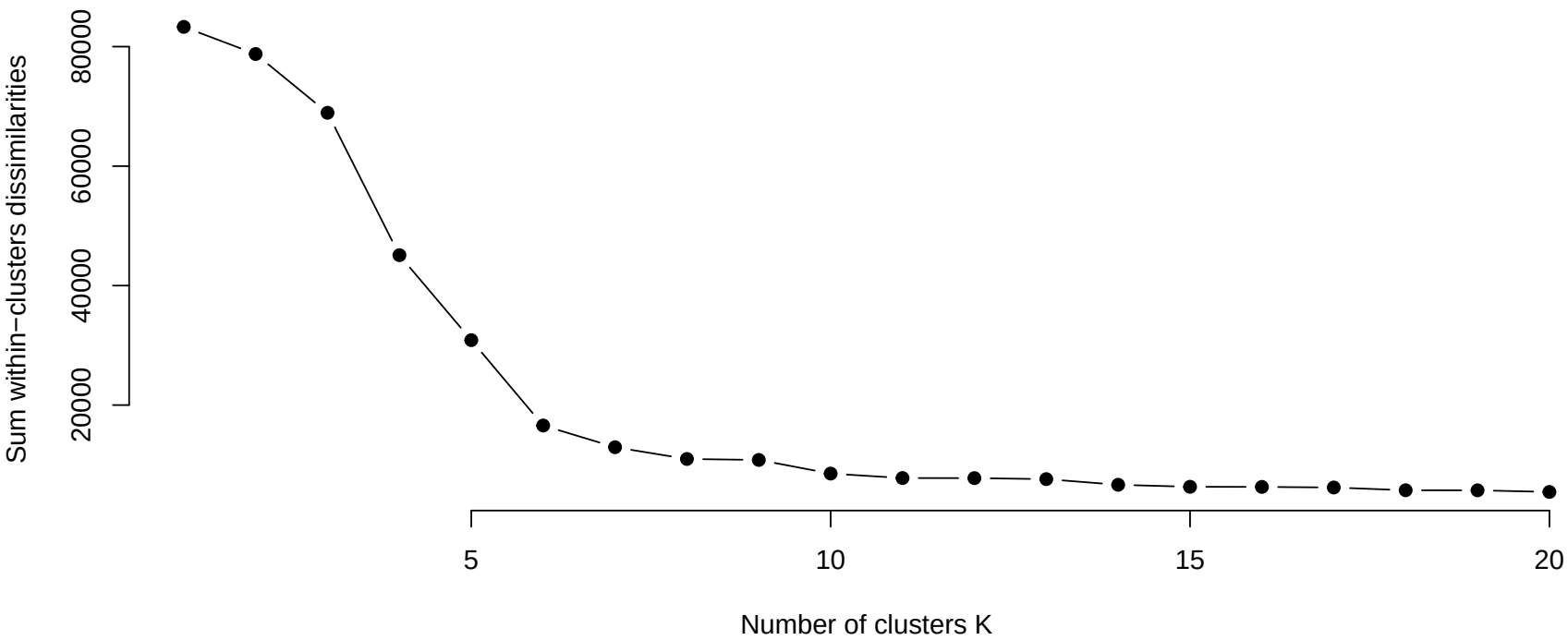
```
bc_diss <- vegdist(spp_cov_matrix[,-1],"bray")
diss_mat <- as.matrix(bc_diss)
bc_tree <- hclust(bc_diss)
plot(bc_tree)
```



Determine optimal number of clusters (scree plot)

Devise a metric of the information in the clustering and compute it as you vary K.

```
k.max<-20
dpc <- sapply(1:k.max, function(k){
    bc<-cutree(bc_tree, k)
    sum(sapply(1:k, function(x) sum(diss_mat[bc==x,bc==x]) ))
  } )
```

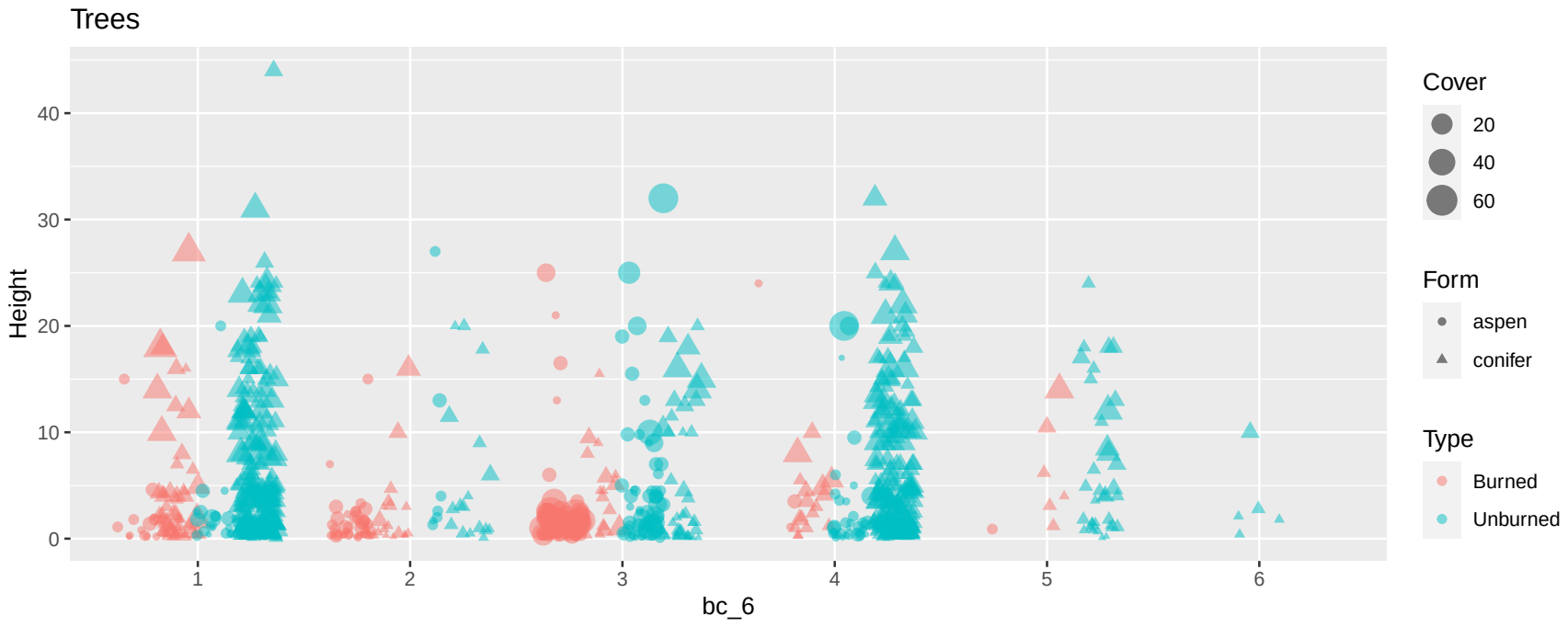


There's an elbow at 6 clusters, so that's K.

```
bc6<-cutree(bc_tree, k=6)
```

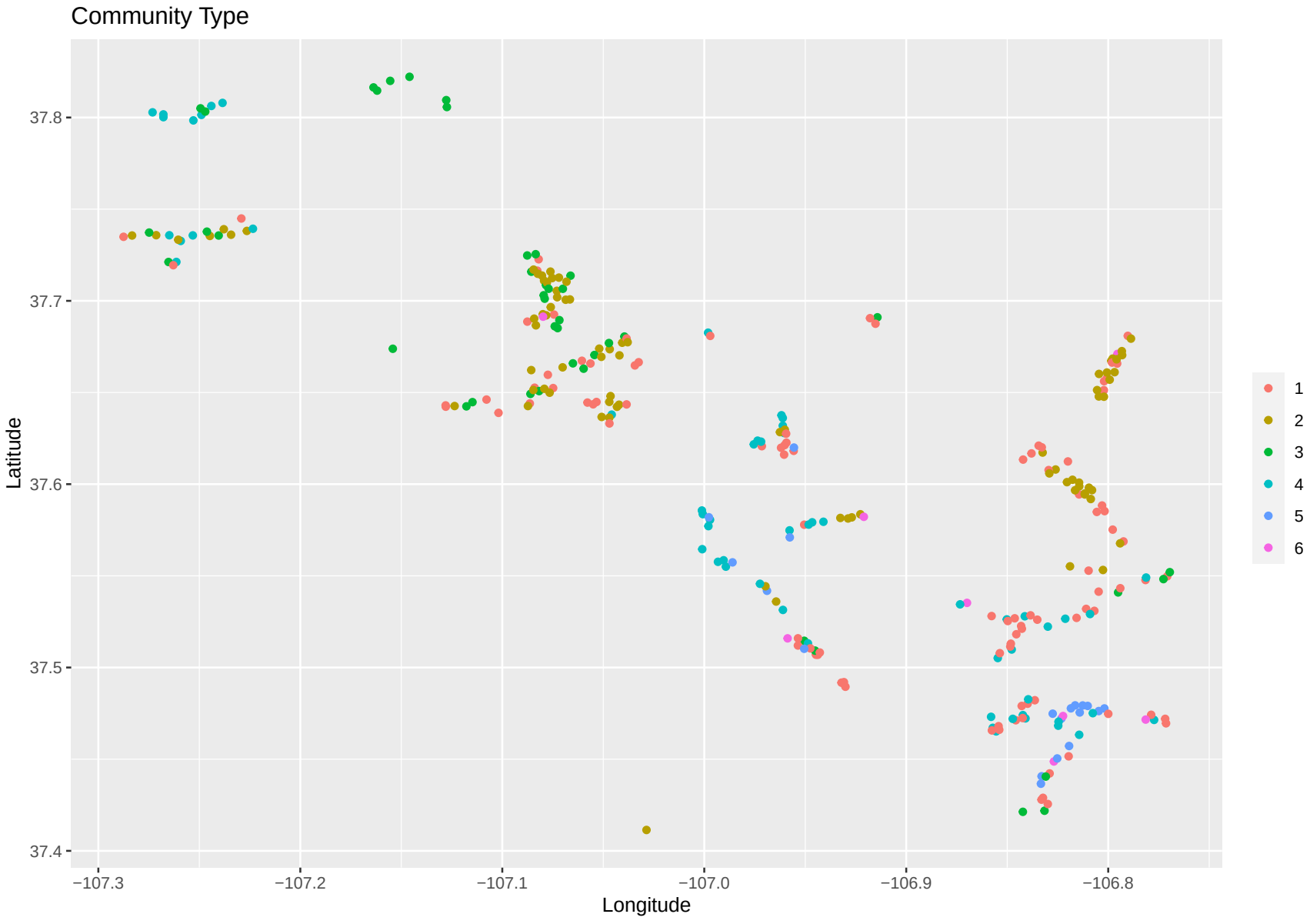
Interpreting the clusters

```
plot_groups <- tibble(Plot_ID = spp_cov_matrix$Plot_ID, bc_6 = bc6)
plot_groups$bc_6 <- factor(plot_groups$bc_6)
veg_data <- merge(veg_data, plot_groups)
west_fork_trees <- droplevels(veg_data[veg_data$LifeForm=="Tree", ])
west_fork_trees$Form <- factor(ifelse(grepl("POTR",west_fork_trees$Code),
                                      "aspen", "conifer"))
```



Having created this variable, it remains to determine what it means.

Geographic distribution of the groups



What plants make up the communities?

We might be interested in what plants are specific to each community.

```
get_codes <- function(x)unique(veg_data[veg_data$bc_6==x,]$Code)
only_here <-
    function(x) {
    onlyX <- get_codes(x)
    for(i in setdiff(1:6,x)) {onlyX <- setdiff(onlyX, get_codes(i))}
    onlyX}
only_here(1)
```

```
##  [1] "VACE"   "ABCO"   "CHDE2"  "EPHA"   "PECR5"  "ANMI3"  "ARMI4"  "SOSC2"
##  [9] "MIGU"   "JUEN"   "CAAU3"  "SIDR"   "BOGR2"  "VAAT2"  "LICO6"  "COMA25"
## [17] "POBI6"  "GEPA"   "PIPO"   "PONE2"  "HEQU2"  "FESO"   "DRRE"   "MARE"
## [25] "FRSP"   "CARU"
```

```
only_here(2)
```

```
##  [1] "ARCA13" "AGEX"   "JUME3"  "ROBL"   "HECO26" "CICL"   "DYGR"   "FEBR"
##  [9] "PIPU9"  "MELA2"  "PONEI2"
```

What plants are most common in each community?

```
group_summary_plants <- veg_data %>% dplyr::group_by(bc_6) %>%
  dplyr::summarise(num_sites = length(unique(Plot_ID)),
            num_species = length(unique(Code)),
            burn_prop = sum(Type=="Burned")/num_sites,
            most_com = tail(names(sort(table(Code))), 1),
            second_com = tail(names(sort(table(Code))), 2)[1],
            third_com = tail(names(sort(table(Code))), 3)[1])
group_summary_plants
```

```
## # A tibble: 6 x 7
##   bc_6  num_sites num_species burn_prop most_com second_com third_com
##   <fct>     <int>       <int>     <dbl> <chr>    <chr>      <chr>
## 1 1            99         229      9.24 ABLA     PIEN       CHAN9
## 2 2            83         175     16.6  TAOF     CASI12     CHAN9
## 3 3            43         152     11.1  POTR5    CHAN9      TAOF
## 4 4            60         210      2.15 PIEN     ABLA       FRVI
## 5 5            19         112      5.79 PIEN     ABLA       BRCI2
## 6 6             8          95     15.8  TRSP2    POPR       RIMO2
```

Ordination

With the data that are correlated rather than clustered, the goal is to find a new set of directions in which the data are uncorrelated.

- Ordination produces a low-dimensional representation of a dataset. It finds a sequence of combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, ordination also serves as a tool for data visualization. For example, this is useful after clustering for plotting and interpreting results.

Principal Components Analysis (PCA)

Here, I am closely following James et al. (2013).

- The *first principal component* of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p$$

that has the largest variance. By *normalized*, we mean that $\sum_{j=1}^{p} \phi_{j1} = 1$.

- We refer to the elements $\phi_{11}, \ldots, \phi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11} \phi_{21} \ldots \phi_{p1})^t$.

- The second principal component is the linear combination of $X_1, X_2, \ldots, X_p$ that has maximal variance among all linear combinations that are uncorrelated with Z_1.

- Further principal components are defined similarly.

Computation of principal components

To compute the principal components, continuing with the notation from above, the data are a matrix $M = [x_{ij}]$ where i denotes the observation and j the variable. Assume that the column means are zero (data are centered).

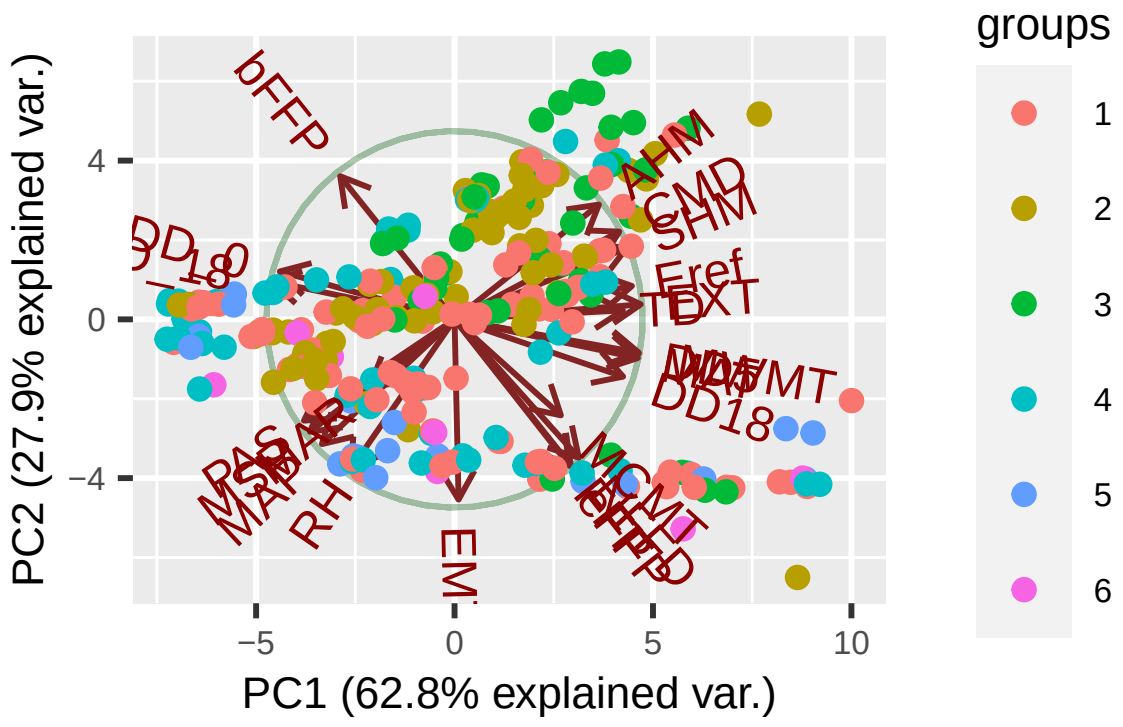
- The *singular value decomposition* factors the matrix into $M = U\Sigma V^t$ where U and V are orthogonal and Σ is diagonal.
- The columns of V are called the **principal directions**.
- The columns of $U\Sigma$ are the **principal components**

The details of this computation are probably not important, but in order to understand what is going on with ordination in general, it is useful to think in terms of linear algebra.

PCA example

The plot of the observations and the variables on the axes defined by the principal components is called a biplot.

```
pca.climate <- prcomp(climate71_00[,-c(1:5,29)], scale = TRUE)
library(ggbiplot) # library(devtools); install_github("ggbiplot")
ggbiplot(pca.climate, obs.scale = 1, var.scale = 1, circle = TRUE,
         groups = climate71_00$bc_6) + theme(text = element_text(size = 8))
```



Interpreting the components

When working with PCA, we are often most interested in the loadings (representations of the features in terms of the principal components)

```
climate_basis <- pca.climate$rotation
climate_basis[1:5,1:3]
```

```
##              PC1         PC2         PC3
## MAT   0.2562305 -0.08088240  0.04614361
## MWMT  0.2575102 -0.06975190 -0.08307110
## MCMT  0.1485913 -0.20112174  0.59966061
## TD    0.2353614  0.01881473 -0.41535238
## MAP  -0.1853677 -0.26127079 -0.15620109
```

and the scores (representations of the observations)

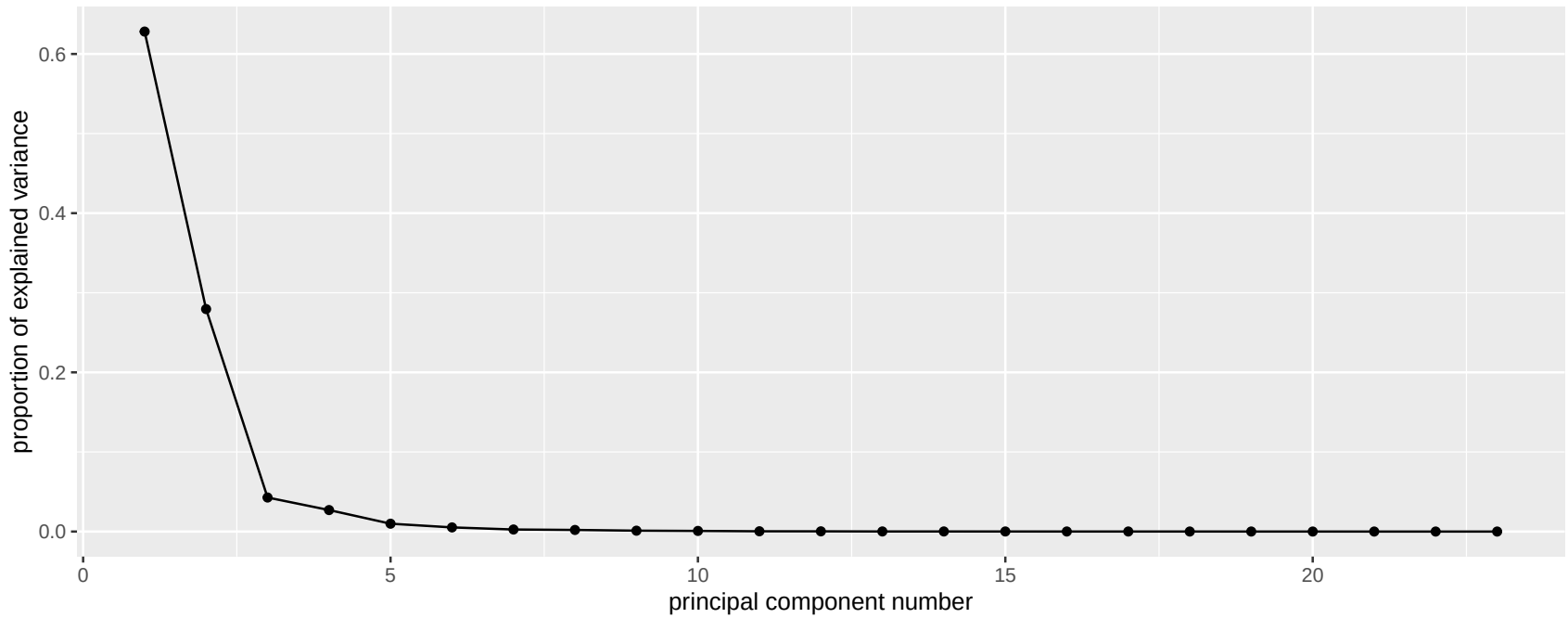
```
climate_obs <- as.data.frame(pca.climate$x)
climate_obs[1:5,1:3]
```

```
##         PC1        PC2         PC3
## 1 -2.618284 -2.0528583 -0.95501793
## 2  5.885337  4.8155813 -2.08786588
## 3 -2.187944  0.1882582  1.49270195
## 4 -4.158979 -1.2607389  0.53573323
## 5 -3.858269 -0.2627610 -0.07169377
```

How many components?

Similar to choosing the number of groups for clustering, choosing the number of principal components to retain can be done by looking for an elbow in a scree plot. 2 or 3 is probably good.

```
ggscreeplot(pca.climate)
```




```
summary(pca.climate)$importance[,1:5]
```

```
##                             PC1      PC2       PC3       PC4       PC5
## Standard deviation     3.800932 2.535391 0.9915583 0.7882199 0.4761904
## Proportion of Variance 0.628130 0.279490 0.0427500 0.0270100 0.0098600
## Cumulative Proportion  0.628130 0.907620 0.9503700 0.9773800 0.9872400
```

Extensions and other considerations

- PCA assumes that the structure in the data is that of a normally distributed ellipsoidal point cloud, just one that's not lined up with the axes.
- Other techniques allow for different structures in data. This is an area that is not far from the frontiers of statistics.
- One area that has not been much investigated by ecologists and may have some low-hanging fruit is *topological data analysis*.

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: With Applications in R*. Springer. https://faculty.marshall.usc.edu/gareth-james/ISL/.