Principal component analysis (PCA)

Difference with Clustering:

Clustering assumes data points to be in one of the clusters that are mutually exclusive and collectively exhaustive (think: market segmentation), so cluster is partition.

PCA, we assume one observation isn't necessarily in one and only one cluster.

Analogy:

Reverse engineering recipes

What are the constituent parts of baked goods
e.g. flour, sugar, butter, stuff, raise, lemon juice…

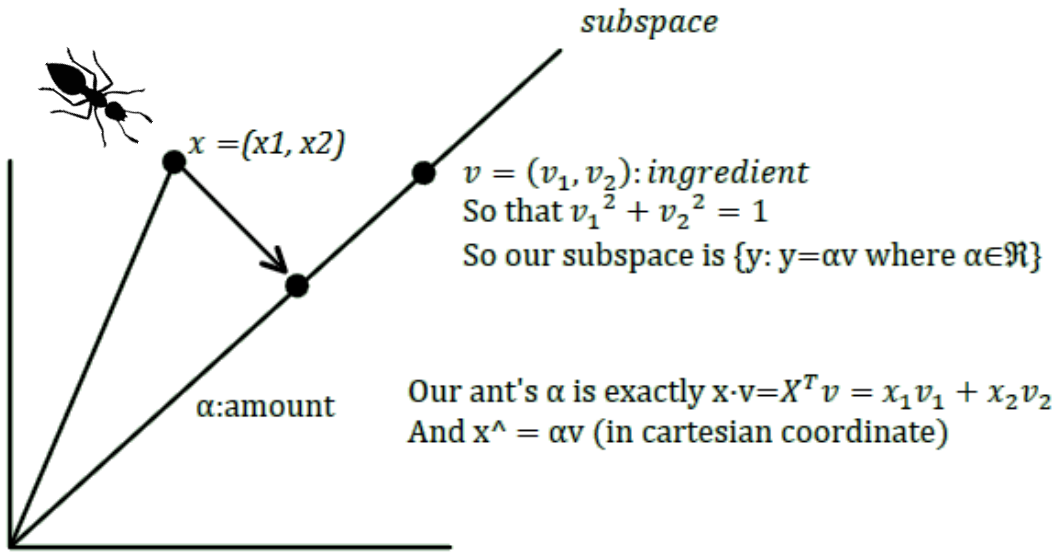
and their weight
e.g. for a pound cake, each ingredient gets a 'weight' of 1 pound

Ingredients = components("loadings")
Amounts of each ingredient = "scores"

15-minute math

Projection (perpendicular/orthogonal)

- Data point (read: an ant) projecting on a subspace (2D to 1D)



- For data matrix X
  - $X = \begin{pmatrix} x_{11} & \cdots & x_{12} \\ \vdots & \ddots & \vdots \\ x_1 v_1 & \cdots & x_x v_2 \end{pmatrix}$
  - v: candidate unit vector
  - $\alpha_i = x_i \cdot v$ where $x_i = (x_{i1}, x_{i2})$
  - idea: choose v so that $variance(\alpha_i) = variance(x_i \cdot v)$ is as large as possible
    - we want it because the minimal loss of information and maximum variance of the projections give the best.
  - constrained so that v has length 1 (Lagrange multiplier problem), so $v1^2 + v2^2 = 1$.

- Once we selected this first principle component, we then look for the next unit vector that explains the most of remaining variance in feature space. Obviously, this second principle component must be perpendicular to the first.
- Next we try to run an example PCA in R.

R examples – pca_2D.R

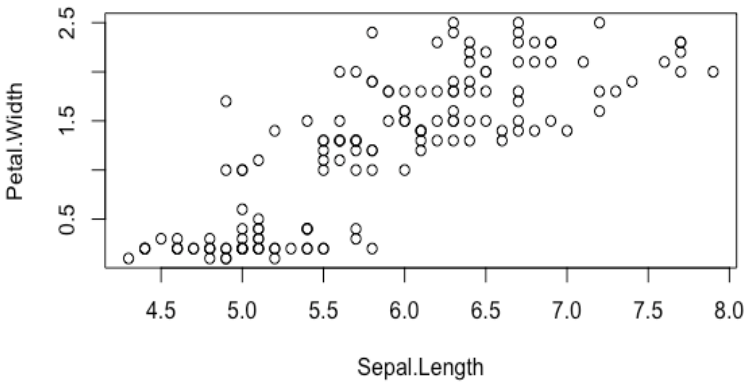
We first load and plot the iris data set:

```
# Load a toy data and peak at the numbers
data(iris)
# Pick out the first two columns
Z = iris[,c(1,4)]
# Clearly a lot of correlation structure in the measurements
plot(Z)
```

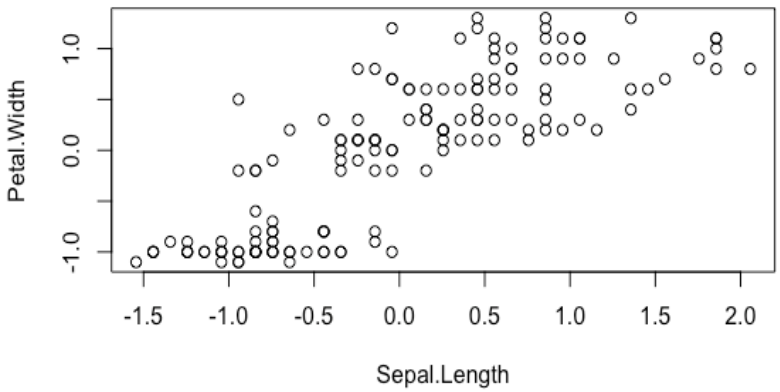
After centering the data, we get Z centered on (0,0).

```
Z_centered = scale(Z, center=TRUE, scale=FALSE)
plot(Z_centered)
```

plot of Z



plot of centered Z



We start to randomly choose a vector v_try.

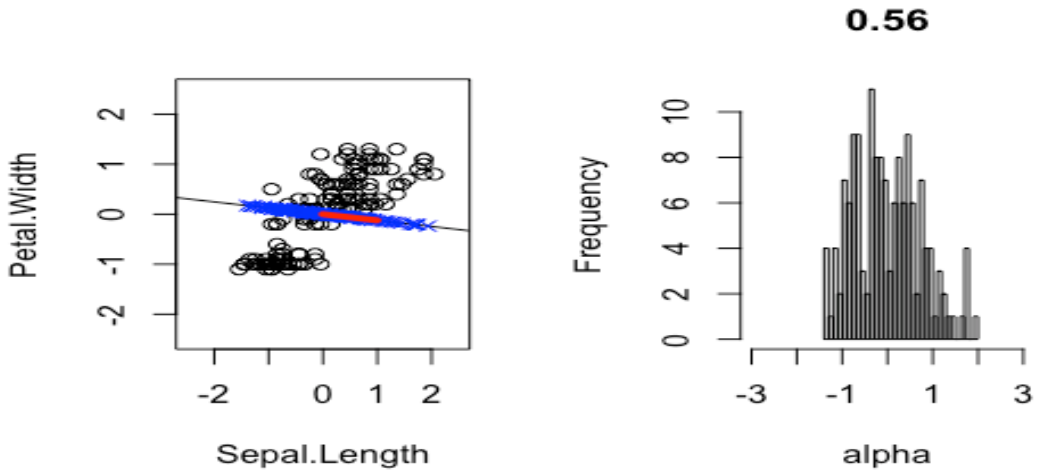
```
# Pick some random unit-norm vectors to define subspace
# and project the points onto that subspace
v_try = rnorm(2)
v_try = v_try/sqrt(sum(v_try^2))
slope = v_try[2]/v_try[1]  # intercept = 0, slope = rise/run

# Show the subspace
par(mfrow=c(1,2))
plot(Z_centered, xlim=c(-2.5,2.5), ylim=c(-2.5,2.5))
abline(0, slope)  # plot the subspace as a line

# Project the points onto that subspace
alpha = Z_centered %*% v_try  # inner product of each row with v_try
z_hat = alpha %*% v_try  # locations in R^2
points(z_hat, col='blue', pch=4)
segments(0, 0, v_try[1], v_try[2], col='red', lwd=4)

hist(alpha, 25, xlim=c(-3,3), main=round(var(alpha), 2))
```

From the plot, we can see that the randomly chosen vector v_try cannot explain the relationship in data so well. The blue line is the subspace of our random choice for the original data set and our unit-form vector v-try has been marked in red. 0.56 represents the standard deviation of the inner products of each row with v_try.



Instead of randomly picking a vector v, we then use PCA, with the syntax of prcomp(dataset).

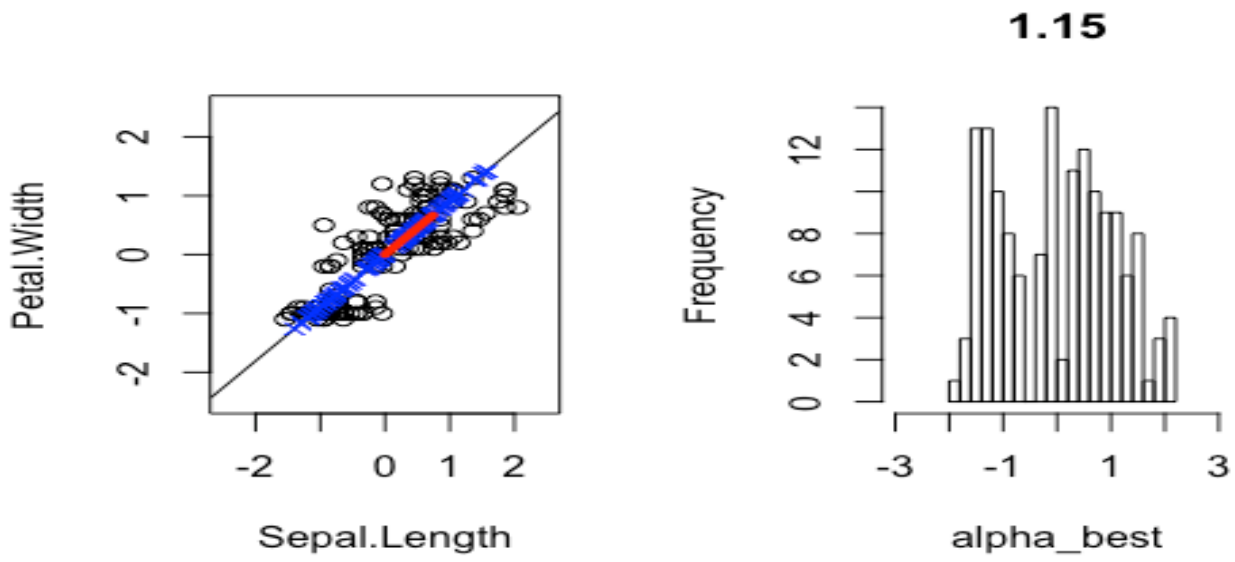
```
# Compare these random projections to the first PC
pc1 = prcomp(Z_centered)
v_best = pc1$rotation[,1]
v_best
slope_best = v_best[2]/v_best[1]  # intercept = 0, slope = rise/run

par(mfrow=c(1,2))
plot(Z_centered, xlim=c(-2.5,2.5), ylim=c(-2.5,2.5))
abline(0, slope_best)  # plot the subspace as a line

alpha_best = Z_centered %*% v_best  # inner product of each row with v_best
z_hat = alpha_best %*% v_best  # locations in R^2
points(z_hat, col='blue', pch=4)
segments(0, 0, v_best[1], v_best[2], col='red', lwd=4)

hist(alpha_best, 25, xlim=c(-3,3), main=round(var(alpha_best), 2))
```

We can see from the plots that using PCA to get a subspace is more accurate in explaining the data.

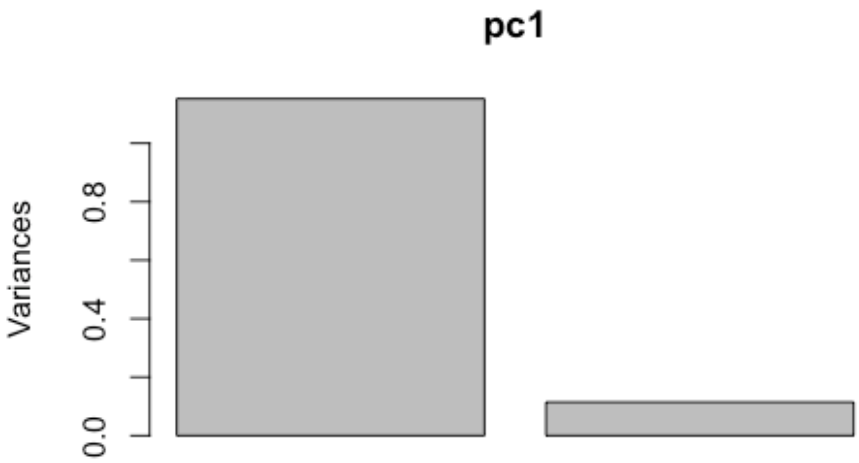


Finally, we want to know how much of the variation this first principal component predicts, by first getting the variance sum of each component. The sum is 1.2667.

```
var_bycomponent = apply(Z_centered, 2, var)
sum(var_bycomponent)
```

Next, we calculate the variance of points projected on our principal component.

```
# Compare with variance of the projected points
var(alpha_best)
var(alpha_best)/sum(var_bycomponent)  # as a ratio
# Compare with the answer from prcomp's plot method
par(mfrow=c(1,1))
plot(pc1)
pc1$sdev^2  # the standard deviation, rather than the variance
```



With the variance being 1.152267 and the ratio being 90.9%, our principal component shows the majority of information in the data set.