PCA for the uninitiated

Ben Mabey

PCA for the uninitiated

Intuitive motivation via maximum variance interpretation

Ben Mabey

For online viewers...

- Pressing 'p' toggles speaker notes (when available)
- Pressing 'f' toggles fullscreen viewing
- Pressing 'w' toggles widescreen
- Pressing 'o' toggles overview mode

N.B.: The deck isn't completely standalone since I don't explain every step made as I did when actually presenting it. That said I think the deck should be useful for anyone who wants to get a quick idea of what PCA is and the math behind it (I only take into account conventional PCA, not probabilistic interpretations). I am inconsistent with some of my equations to make some of the algebra easier (all legal though!) which I explained during the actual presentation. For people who want to go deeper and follow the math more closely I highly recommend the tutorial by Jonathan Shlens which is where I got most of my derivations.

See the last slide of the deck for additional resources.

— &twocol

The ubiquitous & versatile PCA

*** left * Dimensionality Reduction * Data Visualization * Learn faster * Lossy Data Compression * Noise Reduction * Exploration * Feature Extraction * Regression (Orthogonal)

*** right

- Unsupervised Learning Algorithm

- Anomaly Detection (not the best)
- Matching/Distance (e.g. Eigenfaces, LSI)
- K-Means
- Computer Graphics (e.g. Bounded Volumes)
- and many more across various domains…

Majority of PCA tutorials…

1. Organize dataset as matrix.
2. Subtract off the mean for each measurement.
3. Calculate the covariance matrix and perform eigendecomposition.
4. Profit!

Majority of PCA tutorials…

1. Organize dataset as matrix.
2. Subtract off the mean for each measurement.
3. Calculate the ~~covariance~~ correlation matrix and perform eigendecomposition.
4. Profit!

Majority of PCA tutorials…

1. Organize dataset as matrix.
2. Subtract off the mean for each measurement.
3. ~~Calculate the covariance correlation matrix and perform eigendecomposition.~~
4. Perform SVD.
5. Profit!

— &vcenter 

The intuitive ~~Magic~~ Math behind PCA

- **Maximize the variance.**
- Minimize the projection error.

— &vcenter

\(P_{m\times m}X_{m\times n} = Y_{m\times n}\)

*** pnotes - N.B.: The X here is the tranpose of a typical design matrix.. - See the Shlens paper for more info. - Its goal is to extract the important information from the data and to express this information as a set of new orthogonal variables called principal components. - A *linear* transformation! This is a big assumption. - *Is there another basis, which is a linear combination of the original basis, that best re-expresses our data set?* - This transformation will become the *principal components* of X. - What does the transformation boil down to? - Rotation and scale.. so how does that help us? - What should our P be doing? - What do we want our Y do look like?

— &full_image local:signal_noise.png source:http://www.squidoo.com/noise-sources-signal-noise-ratio-snr-and-a-look-at-them-in-the-frequency-domain text_class:white

*** pnotes

- Every dataset has noise and signal… How can we bring out the signal?

— &vcenter

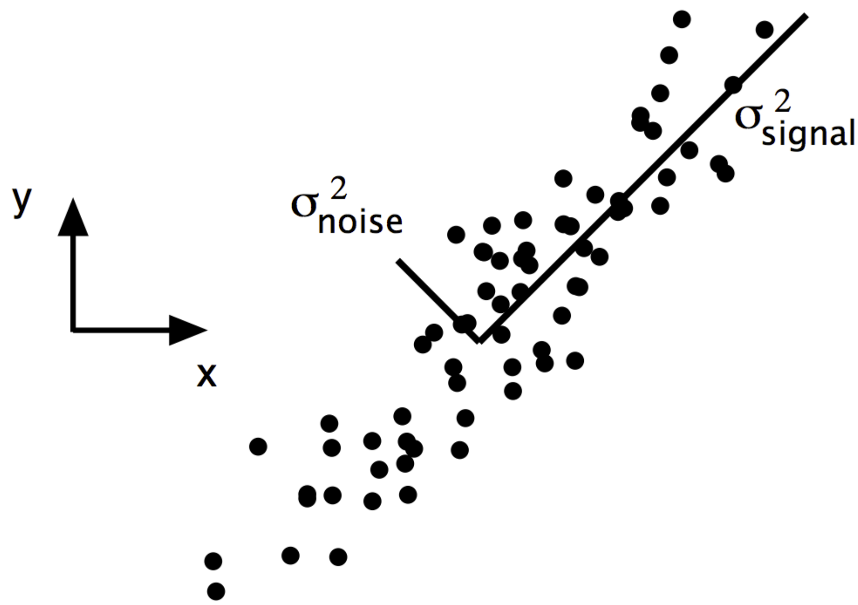
\(SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}\)

— &full_image local:svn.png

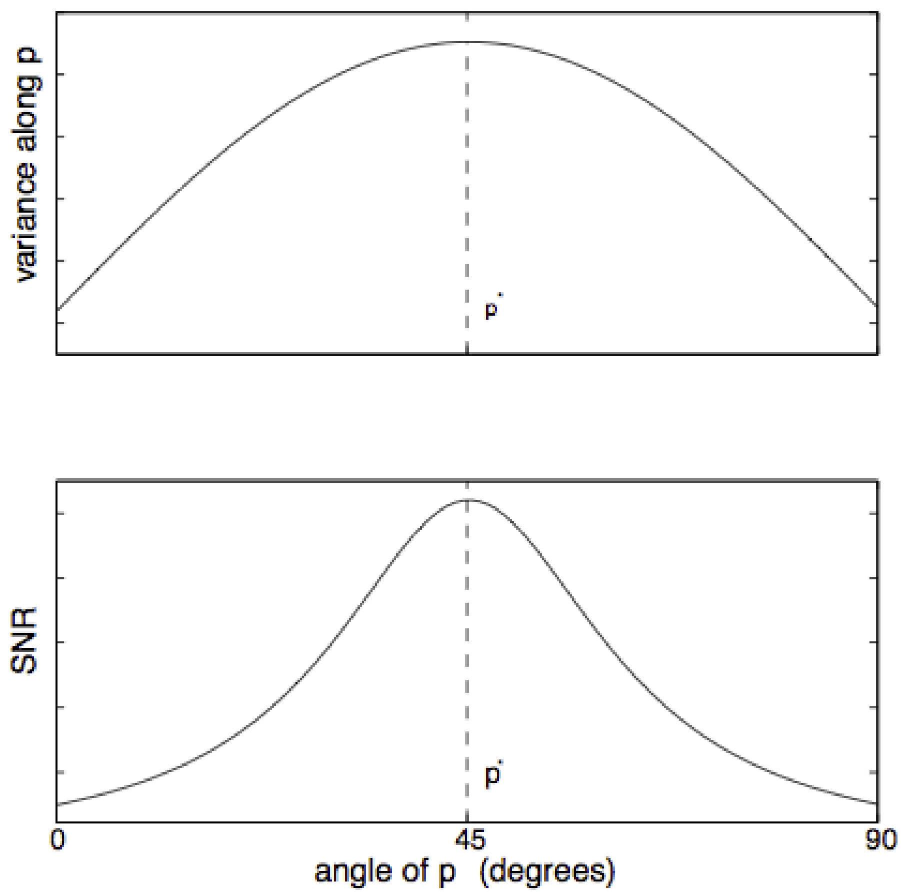
— &twocol

Rotate to maximize variance

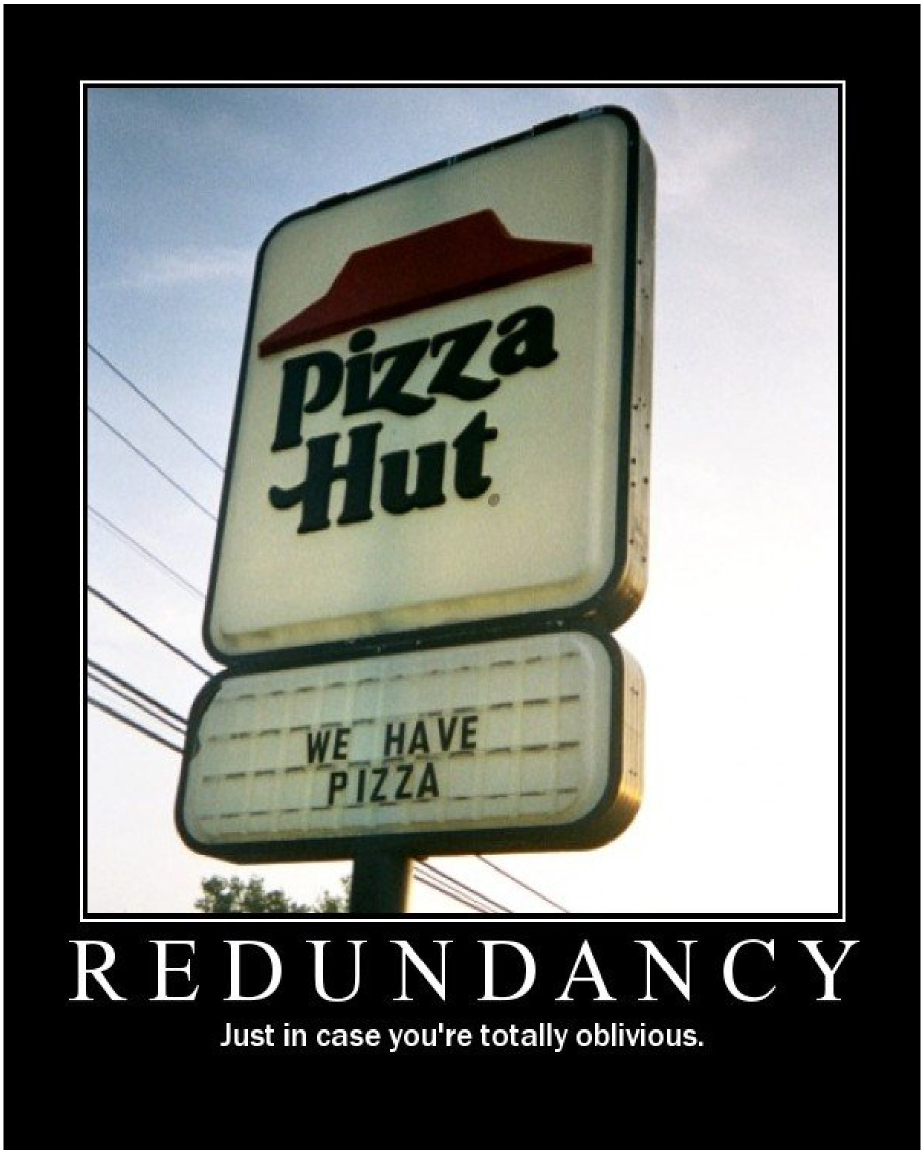
*** left



*** right

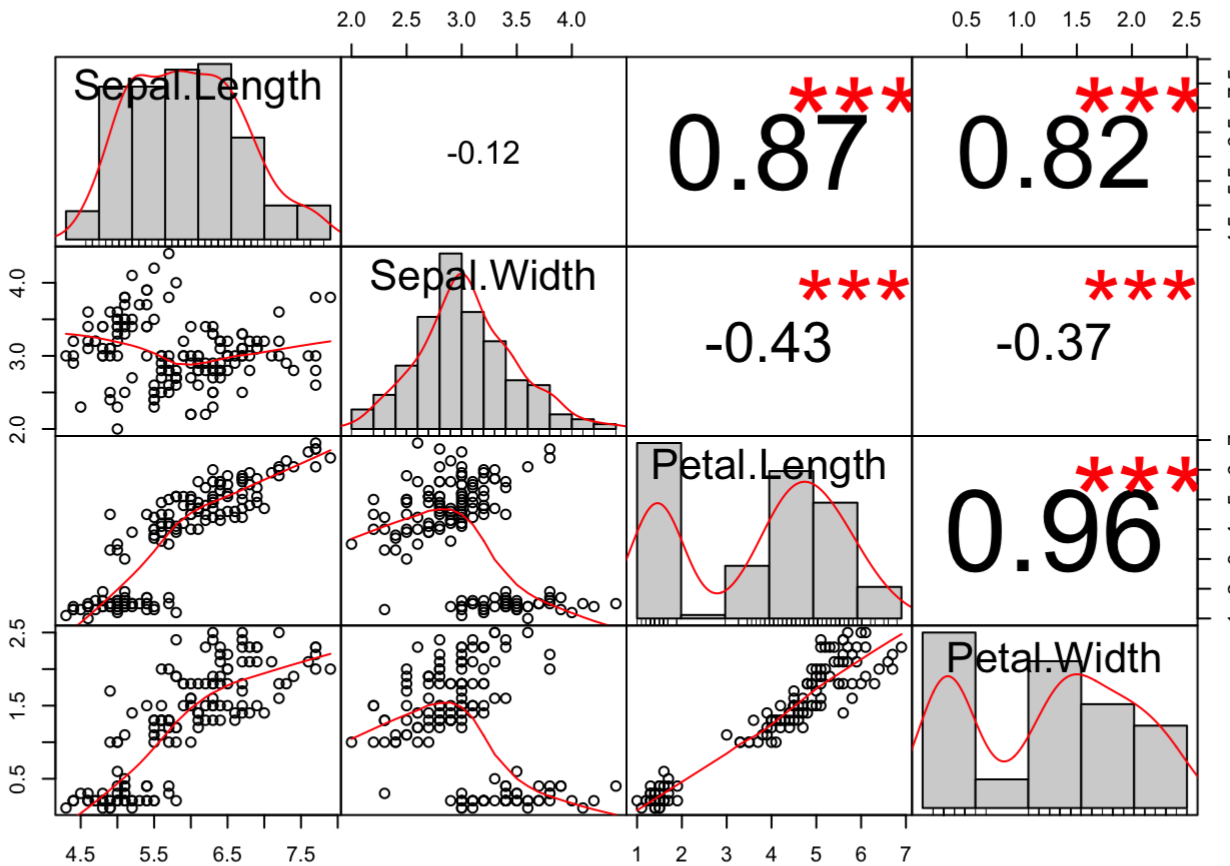


— &vcenter

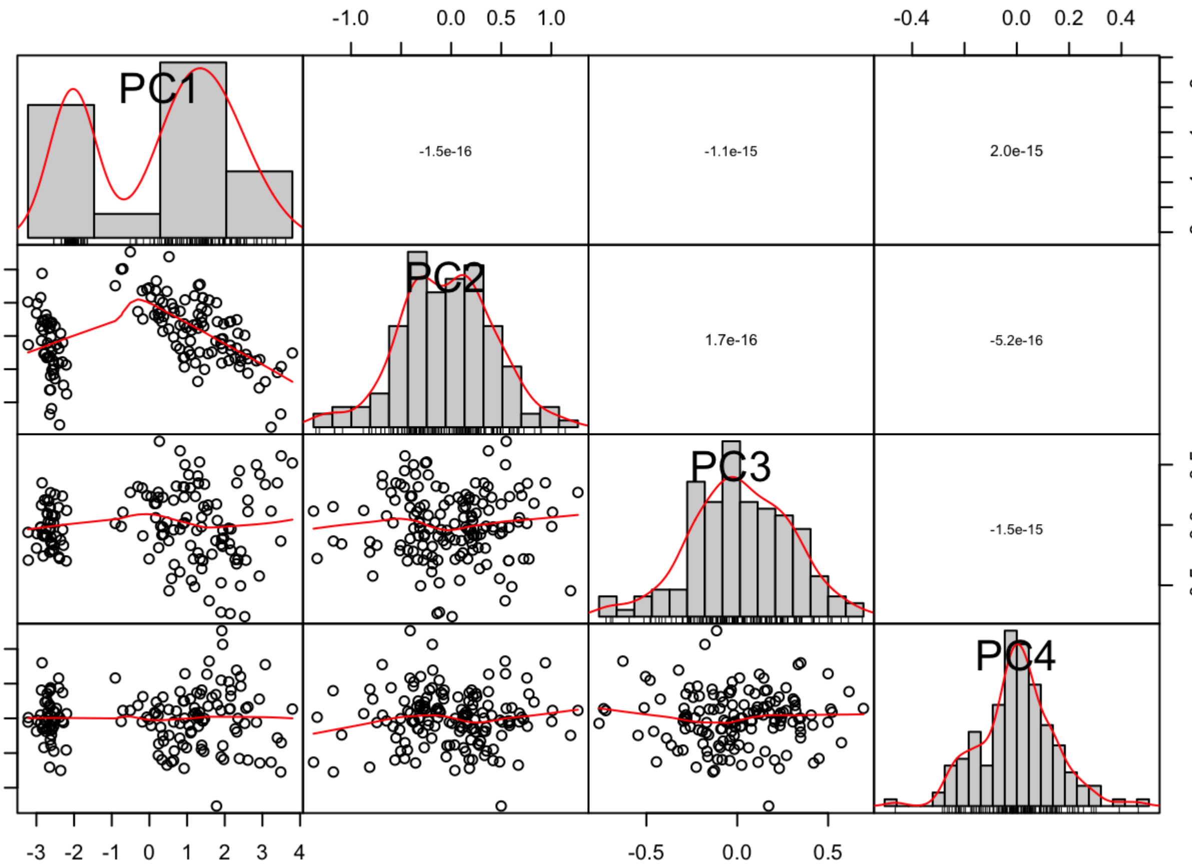


*** pnotes - The logo dimension and the text dimension are redunant and we can reduce it down to a single dimension. :) - “In the real world” you are given datasets with redundanat data all the time… - Different kinds of measurements of same event (i.e. different types of brain scans) - Even duplicated features with transform + noise. - We want a set of features, principal components, that are not redundant. That way we can select the most “principal” ones and throw away the rest. - Another name for redundancy is correlation. - We want to decorrelate the variables.

```
library(PerformanceAnalytics)
chart.Correlation(iris[-5], bg=iris$Species, pch=21)
```




```
chart.Correlation(decorrelated.iris, bg=iris$Species, pch=21)
```



*** pnotes

- Of course PCA doesn't work by looking at scatter plot pairs.
- it needs to optimize against the quantified version of this… Covariance/Correlation.

— &vcenter ## Variance and Covariance \(\DeclareMathOperator{\stddev}{stddev} \DeclareMathOperator{\var}{var} \DeclareMathOperator{\cov}{cov} \DeclareMathOperator{\corr}{corr}\)

Mathematically Useful

Intuitive

Dispersion

\[\begin{eqnarray*} \sigma^2_A = \var(A) &=& E[(A - \mu_A)^2] \\ &=& \frac{1}{n} \sum_{i=1}^n (a_i - \mu_A)^2 \end{eqnarray*} \]

\[\sigma_A = \stddev(A) = \sqrt{\var(A)}\]

Relationship

\[\begin{eqnarray*} \sigma_{AB} = \cov(A,B) &=& E[(A - \mu_A)(B - \mu_B)] \\ &=& \frac{1}{n} \sum_{i=1}^n (a_i - \mu_A)(b_i - \mu_B) \end{eqnarray*} \]

\[\rho_{AB} = \frac{\sigma_{AB}}{\sigma_A\ \sigma_B} = \frac{\cov(AB)}{\stddev(A) \stddev(B)}\] unitless measure \((-1.0..1.0)\)

\[\cov(A,A) = \var(A)\]

\(\sigma_{AB}\) or \(\rho_{AB}\) is \(0\) if and only if \(A\) and \(B\) are uncorrelated.

— $vcenter

Covariance Matrix

\[\Sigma = \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,n} \\ \\ \vdots & \vdots & \ddots & \vdots \\ \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_{n,n} \\ \\ \end{bmatrix} \]

Preprocess \(X\) so that it has zero mean. Now \(\sigma_{AB} = \frac 1n \sum_{i=1}^n a_i b_i\)

\[\Sigma_X = \frac{1}{n}X^TX\]

```r
center <- function(x) x - mean(x)
iris.centered <- apply(as.matrix(iris[-5]), 2, center)
(t(iris.centered) %*% iris.centered) / (nrow(iris) - 1)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```r
center <- function(x) x - mean(x)
m.centered <- apply(as.matrix(iris[-5]), 2, center)
(t(m.centered) %*% m.centered) / (nrow(iris) - 1)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
```

```
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```
cov(iris[-5])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

— $vcenter

What would our ideal \(\Sigma_Y\) look like?

\[PX = Y\]

\[\Sigma_Y = \begin{bmatrix} \sigma^2_1\\ & \sigma^2_2 & \Huge 0\\ & & \ddots\\ & \Huge 0 & & \sigma^2_n\\ \end{bmatrix} \] i.e. \(Y\) is decorrelated.

— &vcenter

Our goal…

Find some orthonormal matrix \(P\) in \(PX = Y\) such that \(\Sigma_Y = YY^T\) is a diagonal matrix. The rows \(Y_n\) of \(P\) are the **principal components** of \(X\).

Note, that I transposed the design matrix (the data) so that covariance calculation is also reversed. This will make our life easier…

Turn the Algebra crank…

\[\begin{eqnarray*} \Sigma_Y &=& P \Sigma_X P^T \\ &=& P(Q\Lambda Q^T) P^T \\ &=& P(P^T\Lambda P) P^T \\ &=& (PP^T)\Lambda (PP^T) \\ &=& I\Lambda I \\ \Sigma_Y &=& \Lambda_{\Sigma_X} \\ \end{eqnarray*} \]

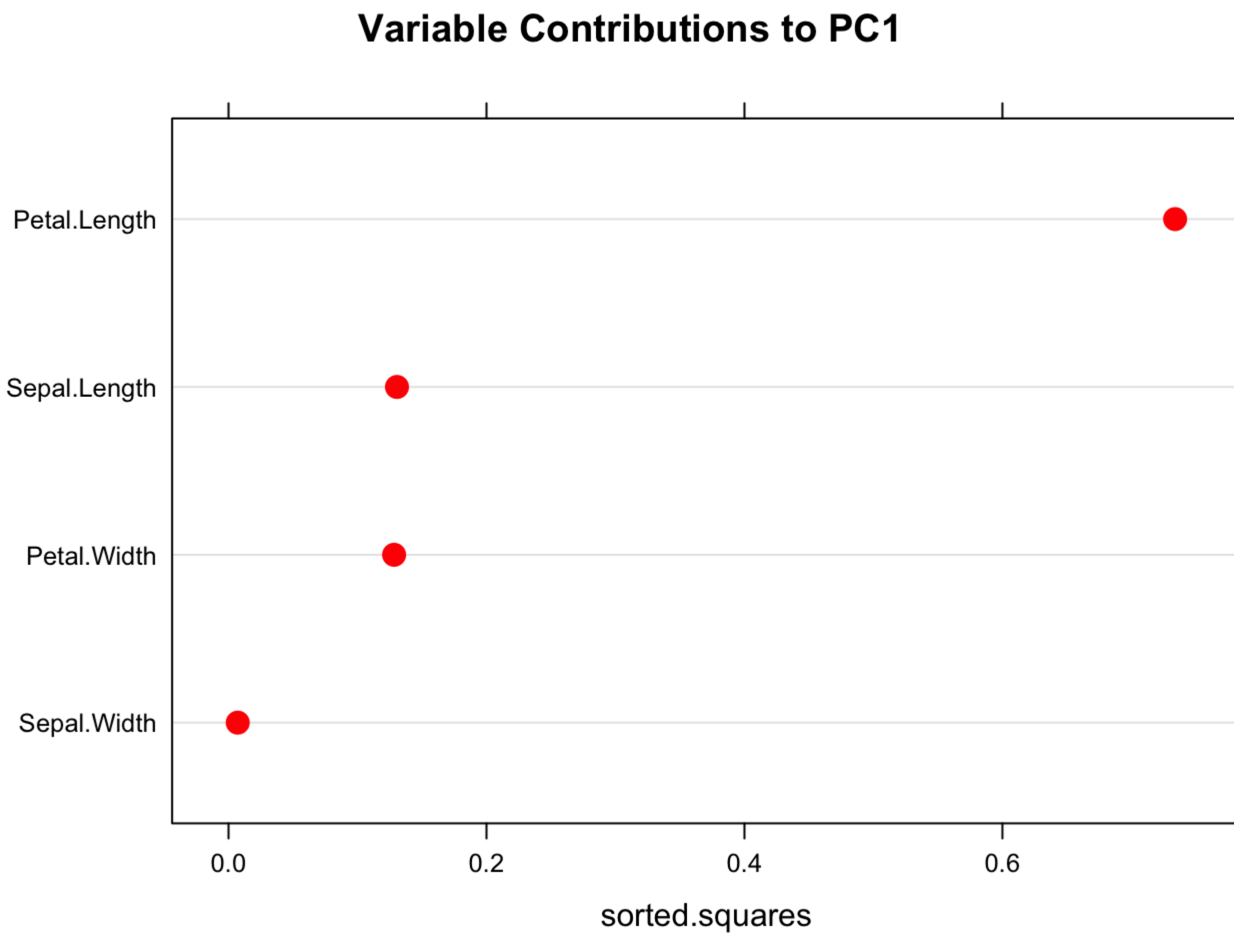
- The principal components are linear combinations of original features of \(X\).
- The principal components of \(X\) are the eigenvectors of \(\Sigma_X\).
- The corresponding eigenvaules lie in \(\Sigma_Y\) and represent the variance.

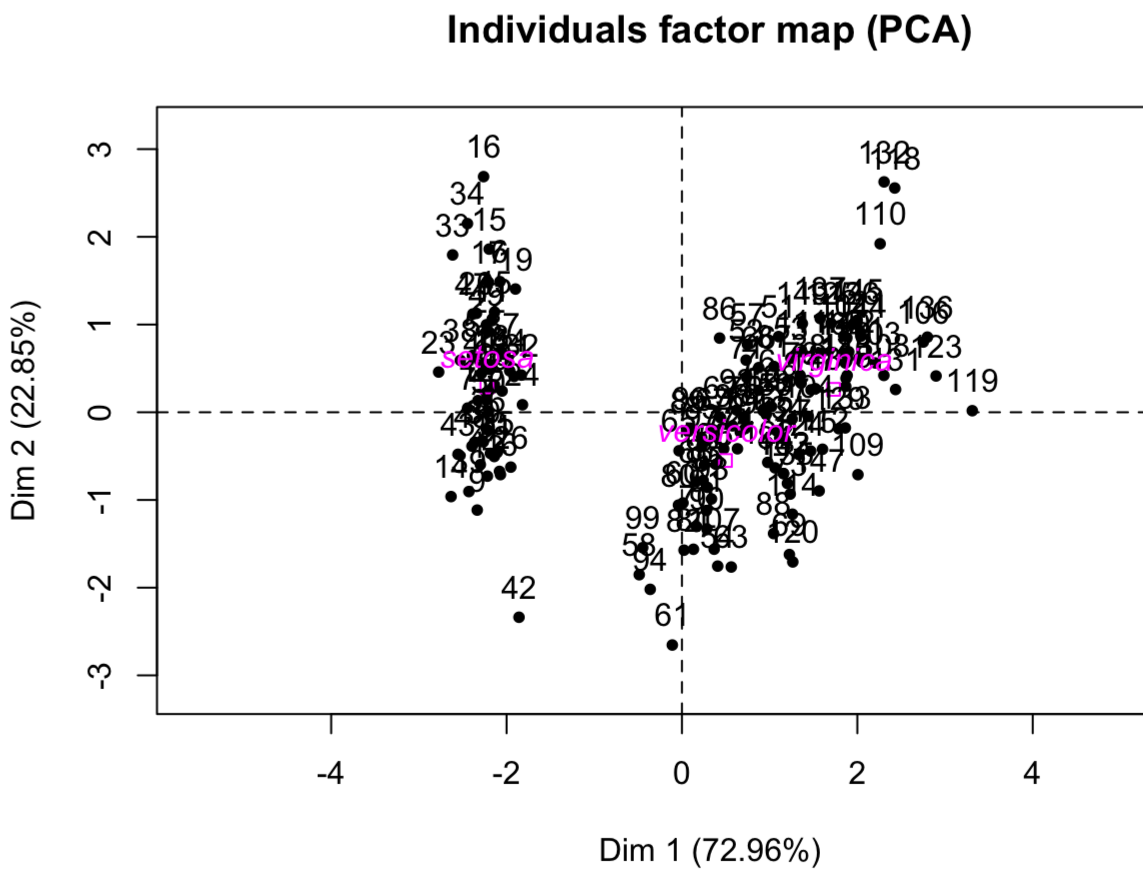
Make the contributions intuitive…

```
iris.eigen$vectors^2
```

```
##                      PC1         PC2         PC3        PC4
## Sepal.Length 0.130600269 0.431108815 0.338758748 0.09953217
## Sepal.Width  0.007144055 0.533135721 0.357497361 0.10222286
## Petal.Length 0.733884527 0.030058080 0.005811939 0.23024545
## Petal.Width  0.128371149 0.005697384 0.297931952 0.56799951
```

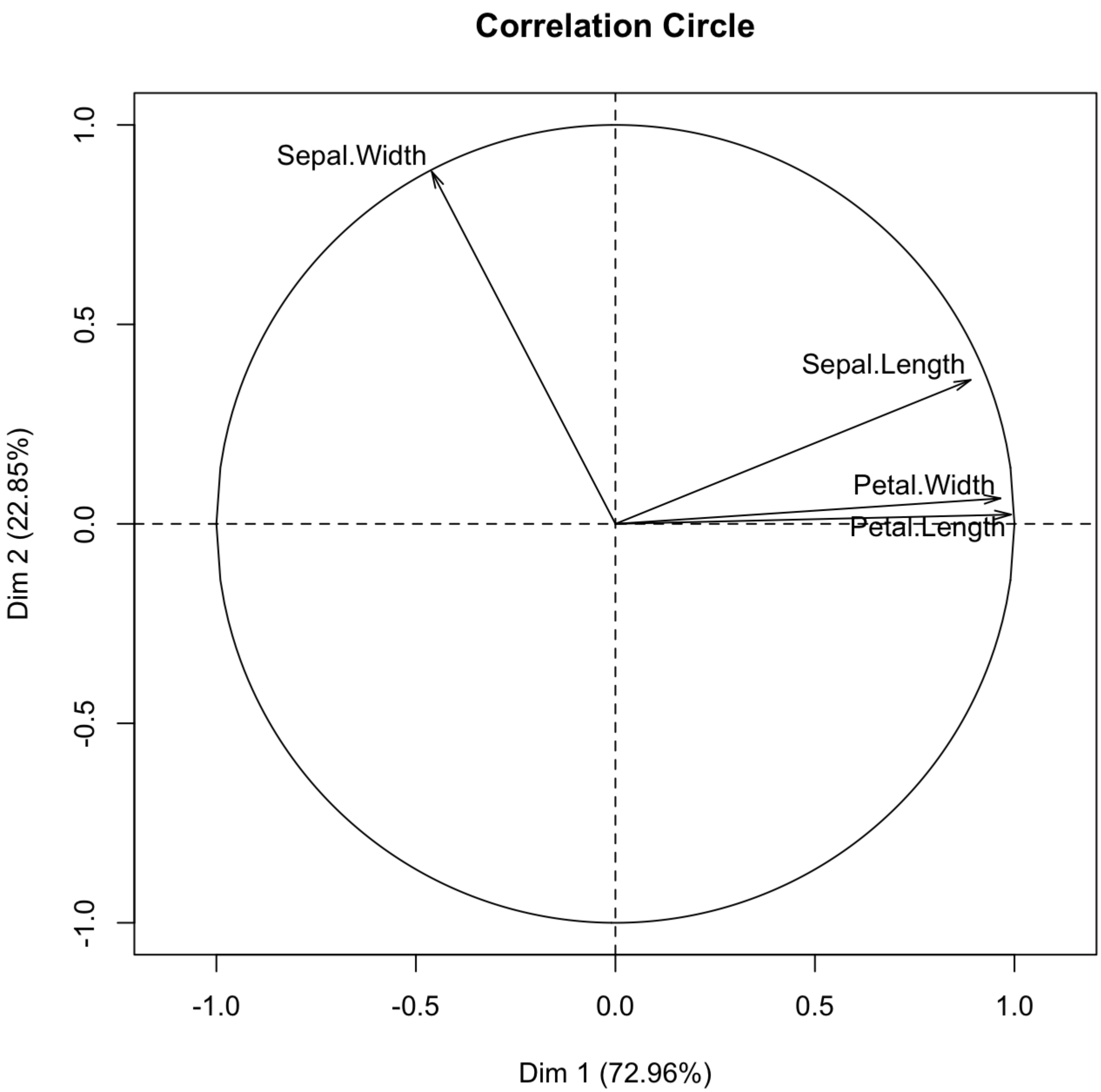
```
squared <- iris.eigen$vectors^2
sorted.squares <- squared[order(squared[,1]),1]
dotplot(sorted.squares,main="Variable Contributions to PC1",cex=1.5,col="red")
```

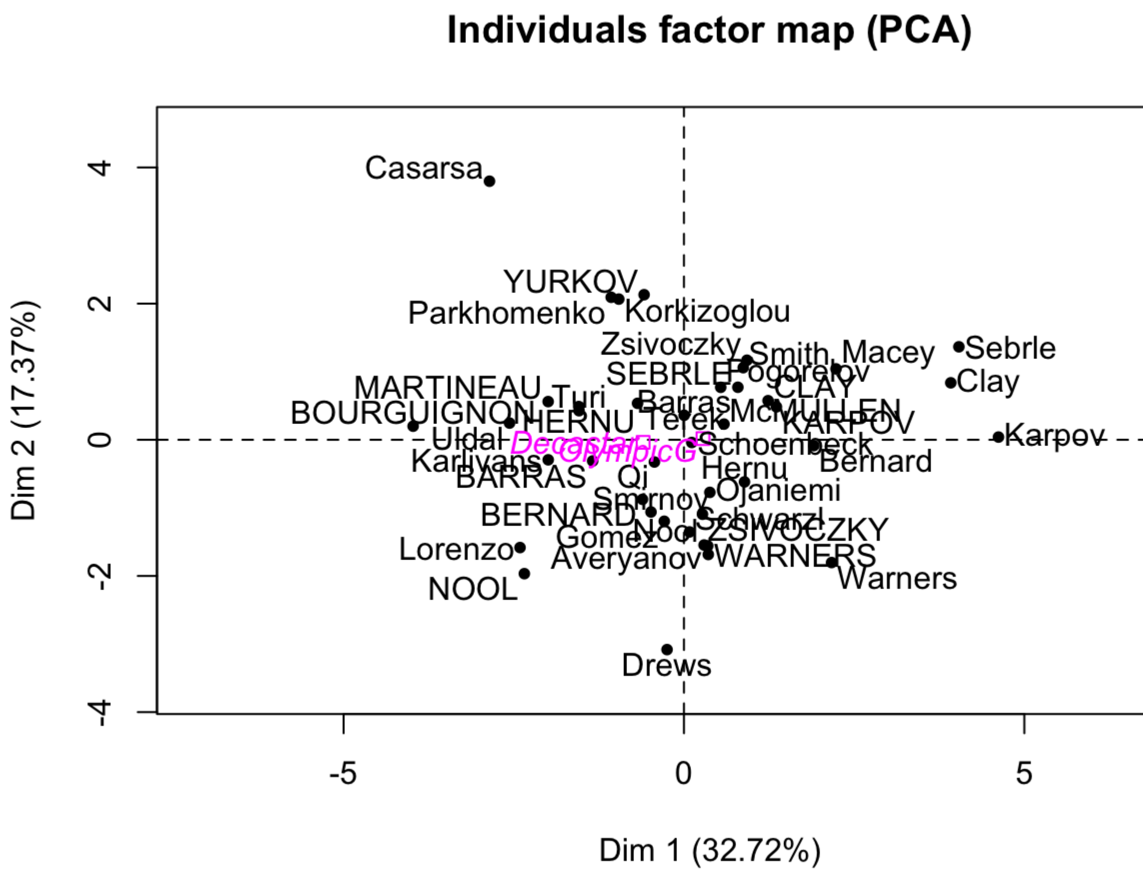




— &vcenter

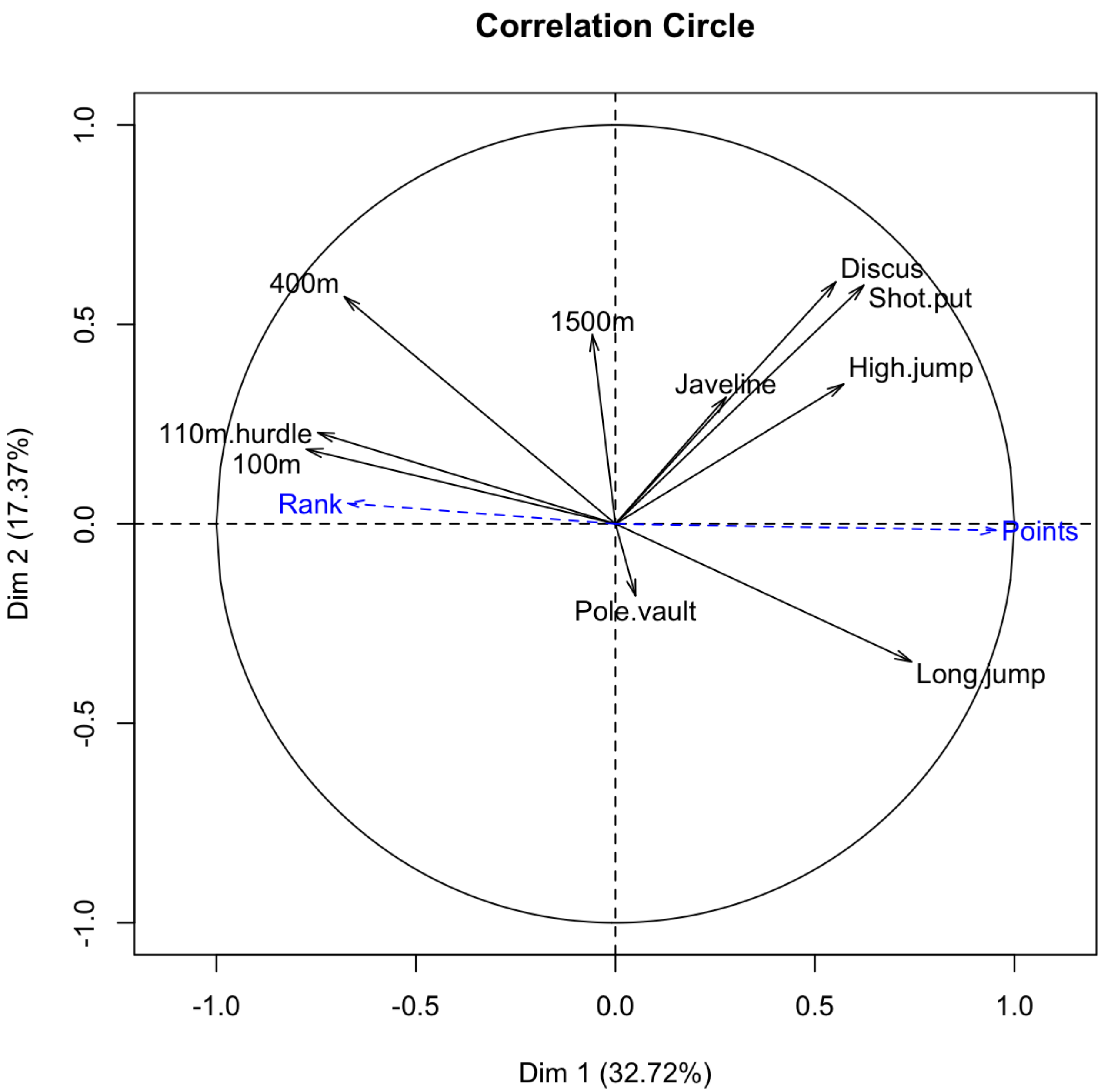
```
# library(FactoMineR); iris.pca <- PCA(iris, quali.sup=5)
plot(iris.pca, choix = "var", title="Correlation Circle")
```


Correlation Circle
Sepal.Width
Sepal.Length
Petal.Width
Petal.Length
1.0
0.5
0.0
-0.5
-1.0
Dim 2 (22.85%)
-1.0
-0.5
0.0
0.5
1.0
Dim 1 (72.96%)



— &vcenter

```
# res.pca <- PCA(decathlon, quanti.sup=11:12, quali.sup = 13)
plot(res.pca, choix = "var", title="Correlation Circle")
```



Correlation Circle
Discus
Shot.put
400m
1500m
High.jump
Javeline
110m.hurdle
100m
Rank
Points
Pole.vault
Long.jump
Dim 2 (17.37%)
Dim 1 (32.72%)
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0

How many components should you keep?

Ratio of variance retained (e.g. 99% is common):

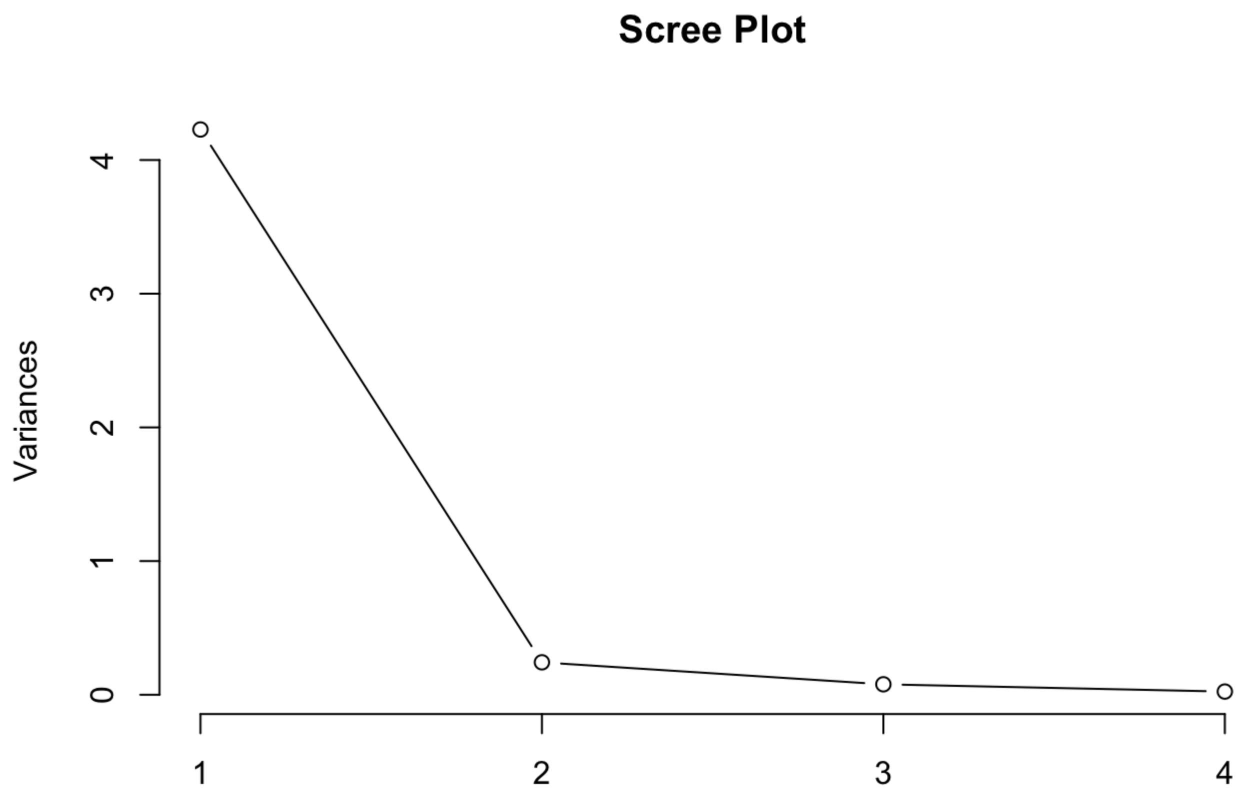
\[\frac{\sum_{i=1}^k \sigma_i}{\sum_{i=1}^n \sigma_i}\]

```
cumsum(iris.eigen$values / sum(iris.eigen$values))
```

```
## [1] 0.9246187 0.9776852 0.9947878 1.0000000
```

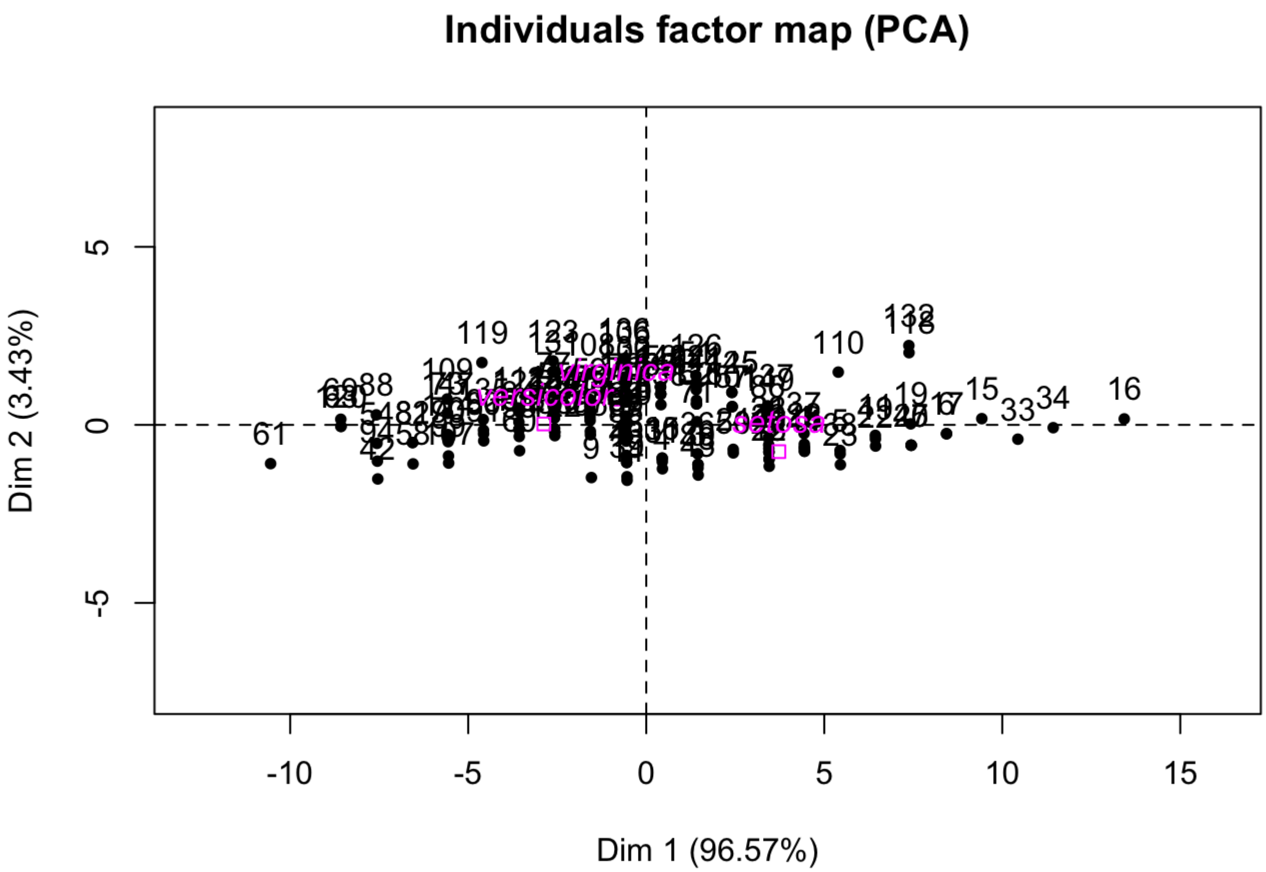
— .scree_plot ## The Elbow Test

```
iris.prcomp <- prcomp(iris[-5], center=TRUE, scale = FALSE)
screeplot(iris.prcomp,type="line",main="Scree Plot")
```

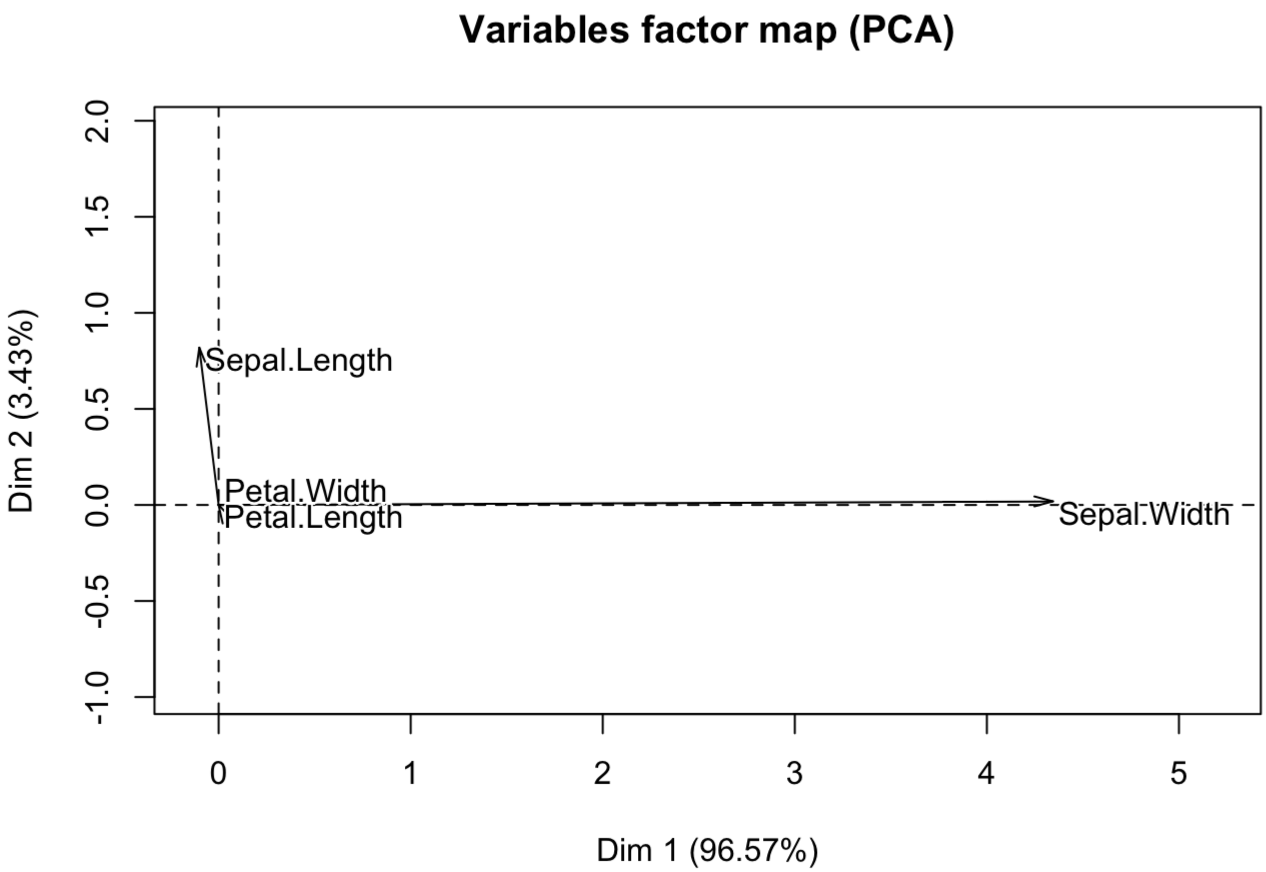


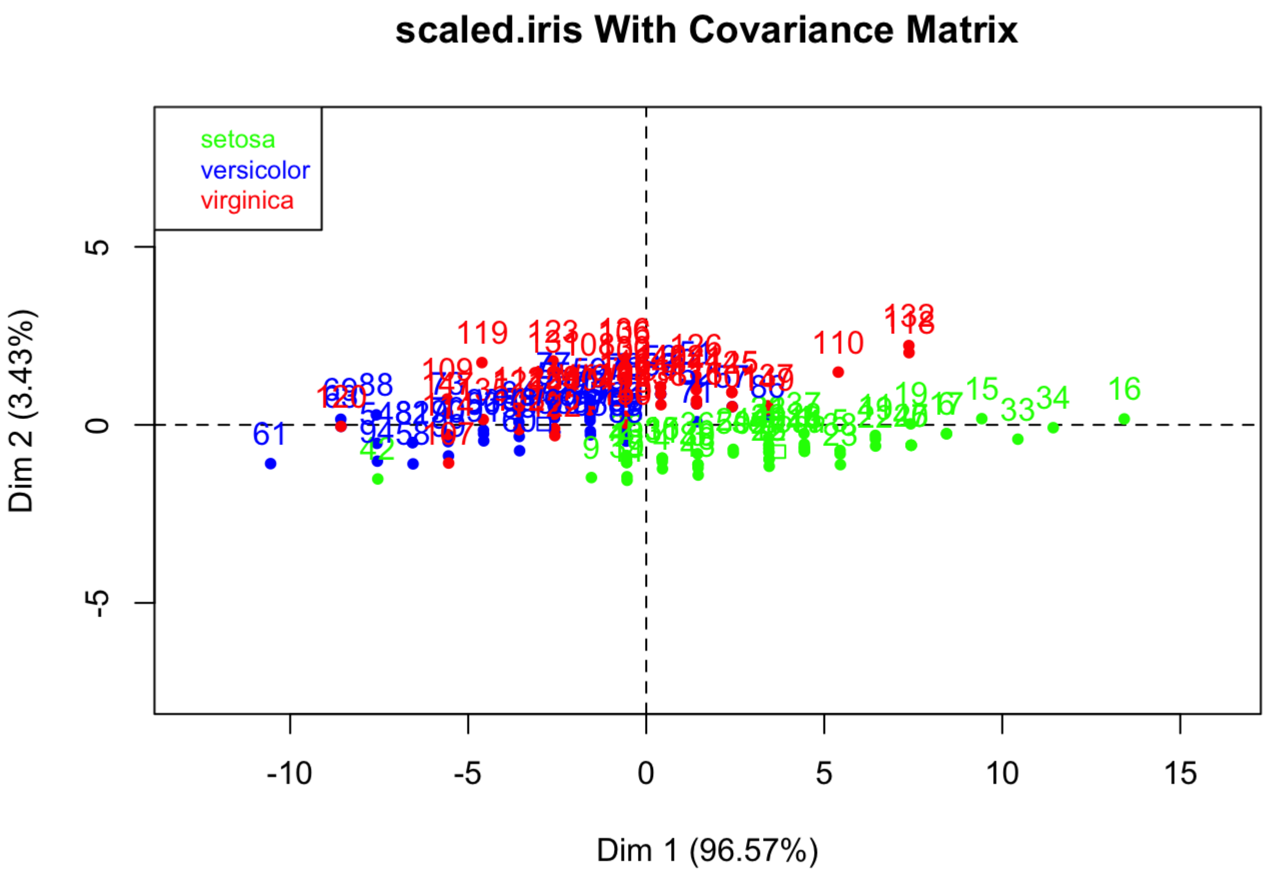
How will PCA perform?

```
scaled.iris <- iris
scaled.iris$Petal.Length <- iris$Petal.Length / 1000
scaled.iris$Petal.Width <- iris$Petal.Width / 1000
scaled.iris$Sepal.Width <- iris$Sepal.Width * 10
```



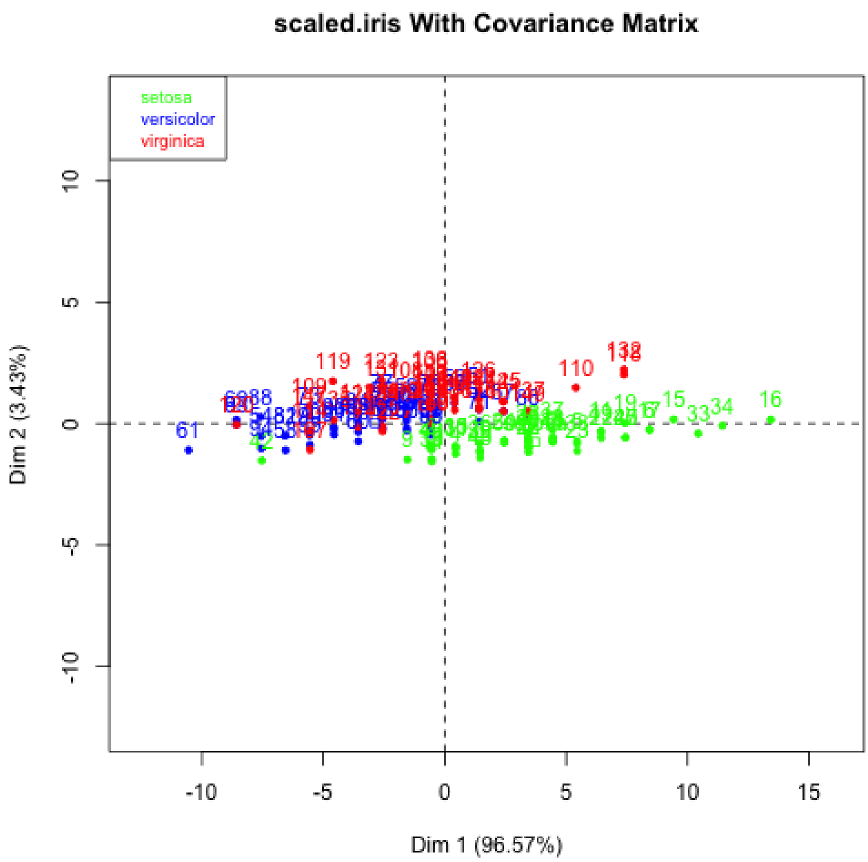

```
## Warning in arrows(0, 0, coord.var[v, 1], coord.var[v, 2], length = 0.1, :
## zero-length arrow is of indeterminate angle and so skipped
```


Variables factor map (PCA)
Sepal.Length
Petal.Width
Petal.Length
Sepal.Width
Dim 2 (3.43%)
Dim 1 (96.57%)
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
0
1
2
3
4
5

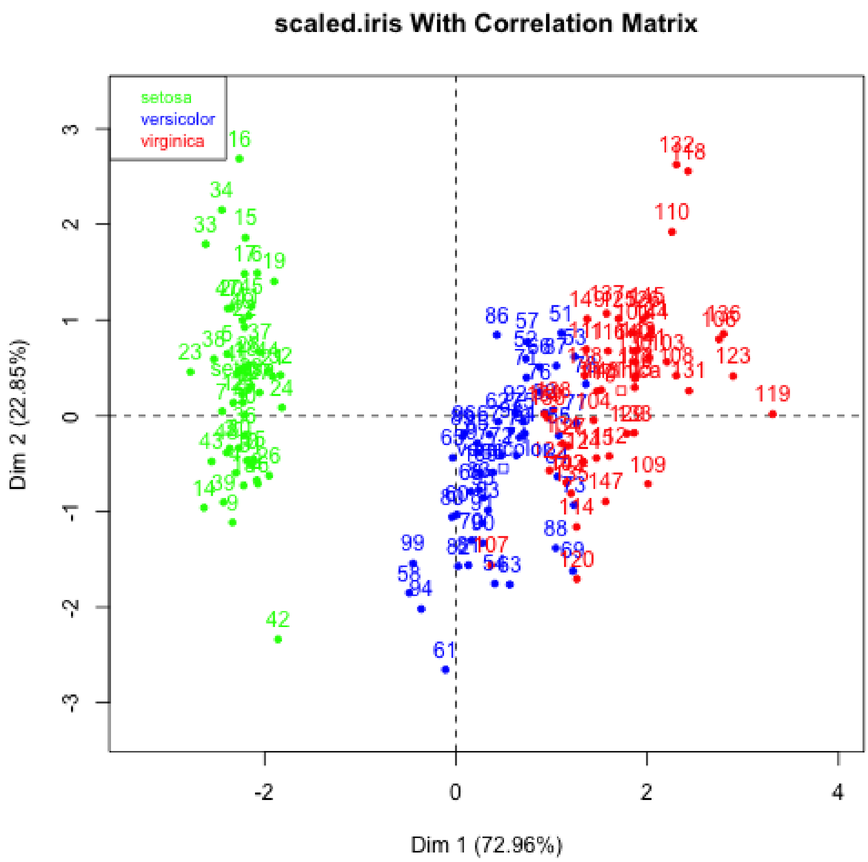


— &twocol ## Scale Matters

*** left


scaled.iris With Covariance Matrix
setosa
versicolor
virginica
Dim 2 (3.43%)
Dim 1 (96.57%)
-10
-5
0
5
10
15

*** right



Correlation Matrix - Standardize the data

```
# (In practice just use the built-in cor function)
standardize <- function(x) {centered <- x - mean(x); centered / sd(centered)}
scaled.iris.standardized <- apply(as.matrix(scaled.iris[-5]), 2, standardize)
(t(scaled.iris.standardized) %*% scaled.iris.standardized) / (nrow(iris) - 1)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

*** pnotes - Mention Kernel PCA.. you can apply non-linear transforms to the data before as well.

Hey, \(AA^T\) and \(A^TA\) look familar…

\[\begin{eqnarray*} A &=& U DV^T \\ AA^T &=& UDV^T(UDV^T)^T \\ &=& UDV^TVD^T U^T \\ &=& UDD^TU^T \ \ (V^TV = I\ \mbox{since V, and U, are orthonormal}) \\ AA^T &=& U D^2 U^T \ \ (\mbox{since D is a diagnol matrix}) \\ \end{eqnarray*} \] Recall that eigendecomposition for an orthonormal matrix is \(A = Q \Lambda Q^T\).

Therefore \(U\) are the eigenvectors of \(AA^T\) and \(D^2\) are the eigenvalues.

Likewise \(V\) are the eigenvectors of \(A^TA\) and \(D^2\) are the eigenvalues.

Tada!

```
y <- iris.centered / sqrt(nrow(iris) - 1)
y.svd <- svd(y)
pcs <- y.svd$v
rownames(pcs)=colnames(iris.centered)
colnames(pcs)=c("PC1","PC2","PC3","PC4")
pcs
```

```
##                      PC1         PC2         PC3        PC4
## Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

```
y.svd$d^2 # variances
```

```
## [1] 4.22824171 0.24267075 0.07820950 0.02383509
```

— &full_image local:Gandalf_the_White_returns.png #references ## References and Resources 1. Jon Shlens (versions 2.0 and 3.1), Tutorial on Principal Component Analysis 1. H Abdi and L J Williams (2010), Principal component analysis 1. Andrew Ng (2009), cs229 Lecture Notes 10 1. Andrew Ng (2009), cs229 Lectures 14 & 15 1. Christopher Bishop (2006), Pattern Recognition and Machine Learning, section 12.1 1. Steve Pittard (2012), Principal Components Analysis Using R 1. Quick-R, Principal Components and Factor Analysis (good pointers to additional R packages) 1. C Ding, X He (2004), K-means Clustering via Principal Component Analysis