Ve406 Lecture 20

Jing Liu

UM-SJTU Joint Institute

July 27, 2018

- For datasets with a large number of independent variables, k, we often need to "reduce" this number k before selecting variables for our model.
- For the extreme case, if the number of observations is too small, that is,

$$n < k$$

then it is simply impossible to include all k of them in the model.

- In general, consider some centred data in a matrix of $n \times k$,

$$\mathbf{X}_{n\times k} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{n1} & \cdots & \cdots & x_{nk} \end{bmatrix}$$

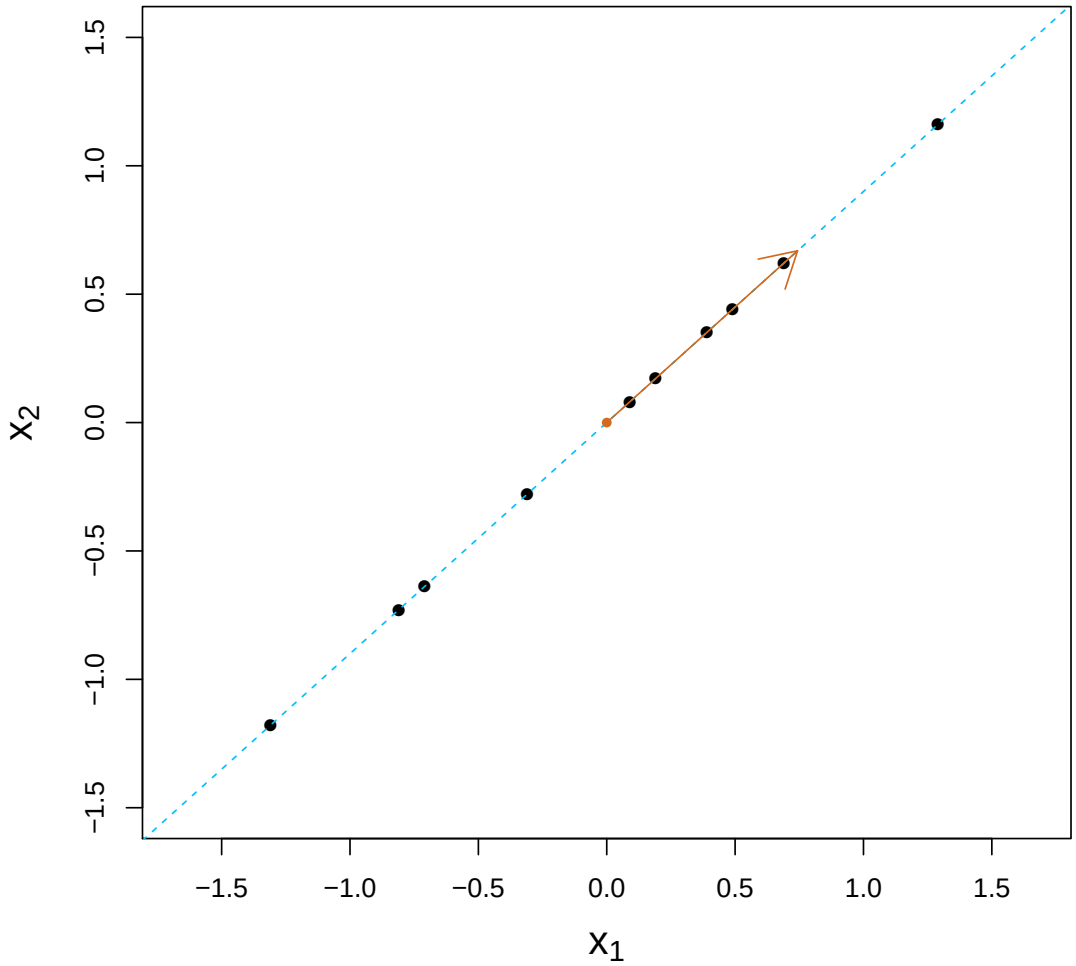
that is, the column means are all zero.

Q: How can we reduce the number k without losing too much information?

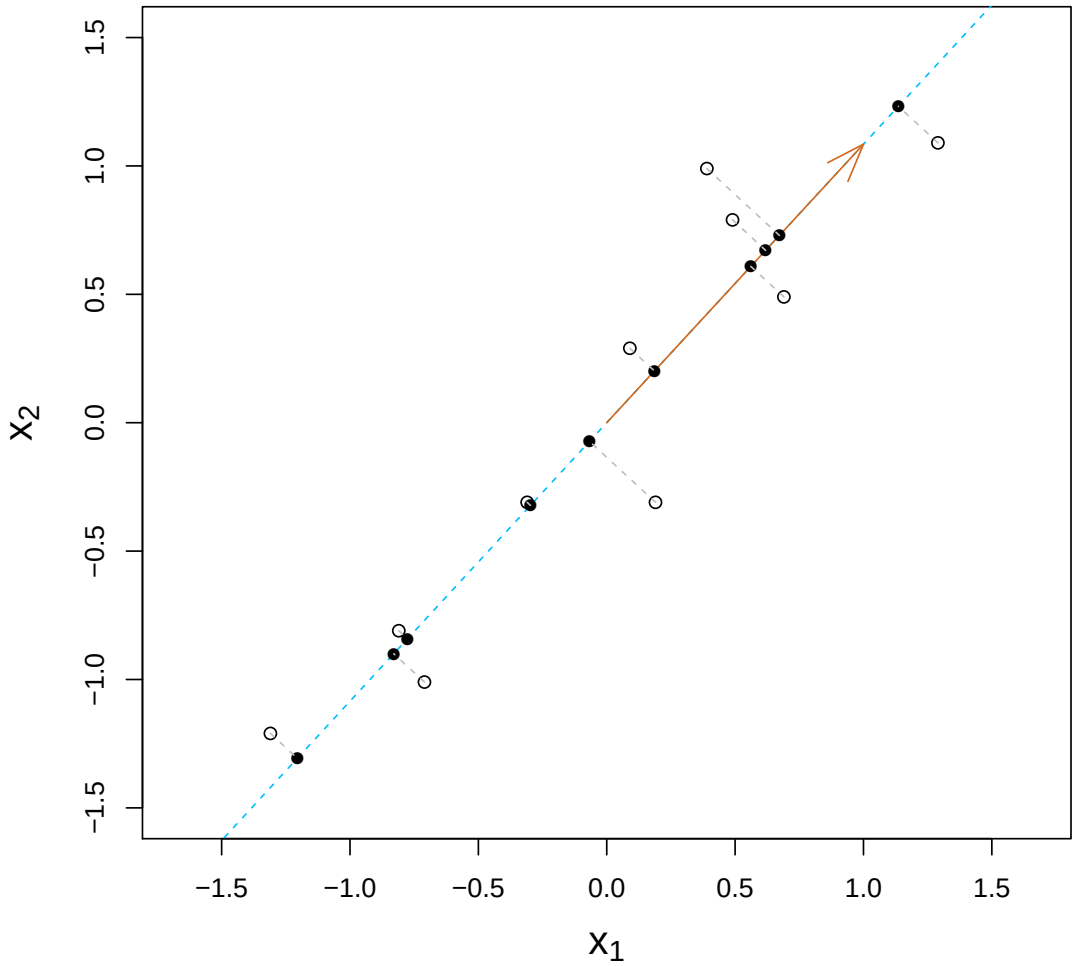
- If there is a perfect multicollinearity, discarding columns is appropriate

$$\alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k x_{ik} = 0 \qquad \text{for all } i.$$

where α_j are not all zeros. When $k = 2$, another choice seems reasonable.



Q: What if they were not perfect multicollinear? e.g.



- In general, imagine the extreme case of reducing k to just 1, i.e. replacing

$$\mathbf{X}_{n\times k}$$

by a single vector in $\mathbf{z} \in \mathbb{R}^n$, which can be thought as a new variable.

- The idea of PCA originated from identifying the principal component

$$\mathbf{w}$$

which is a unit vector in $\mathbb{R}^k$ that minimises

$$\sum_{i=1}^{n} \|\mathbf{r}_i - \left(\mathbf{r}_i^{\mathrm{T}}\mathbf{w}\right)\mathbf{w}\|^2, \qquad \text{where } \mathbf{r}_i^{\mathrm{T}} \text{ denote rows of } \mathbf{X}.$$

and use the following weighted average as the "reduced" data

$$\mathbf{z} = \mathbf{X}\mathbf{w}$$

- It can be show that the minimisation problem is equivalent to

$$\max_{\{\mathbf{w}:\ \|\mathbf{w}\|=1\}} f(\mathbf{w})$$

where the objective function is given by

$$f(\mathbf{w}) = \|\mathbf{X}\mathbf{w}\|^2$$

- Using Lagrange multiplier, we have a unconstrained optimisation problem

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \lambda\left(\mathbf{w}^{\mathrm{T}}\mathbf{w} - 1\right)$$

- Setting the first derivatives to zero, we have the following equations

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} = \lambda\mathbf{w}$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{w} = 1$$

Q: Have you seen the first equation before?

- Hence the principal component(s) is a unit eigenvector of

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}$$

- Since the objective function at local optimiser $\mathbf{w}^*$ can be written as

$$f(\mathbf{w}^*) = \lambda$$

the eigenvector associated to the largest eigenvalue λ is global maximiser.

- Since $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is symmetric, there are k orthonormal eigenvectors for $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ which in turn means all eigenvalues are non-negative.
- This gives us a way to "reduce" $\mathbf{X}$ to two columns, or three columns, etc.

$$\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \ldots$$

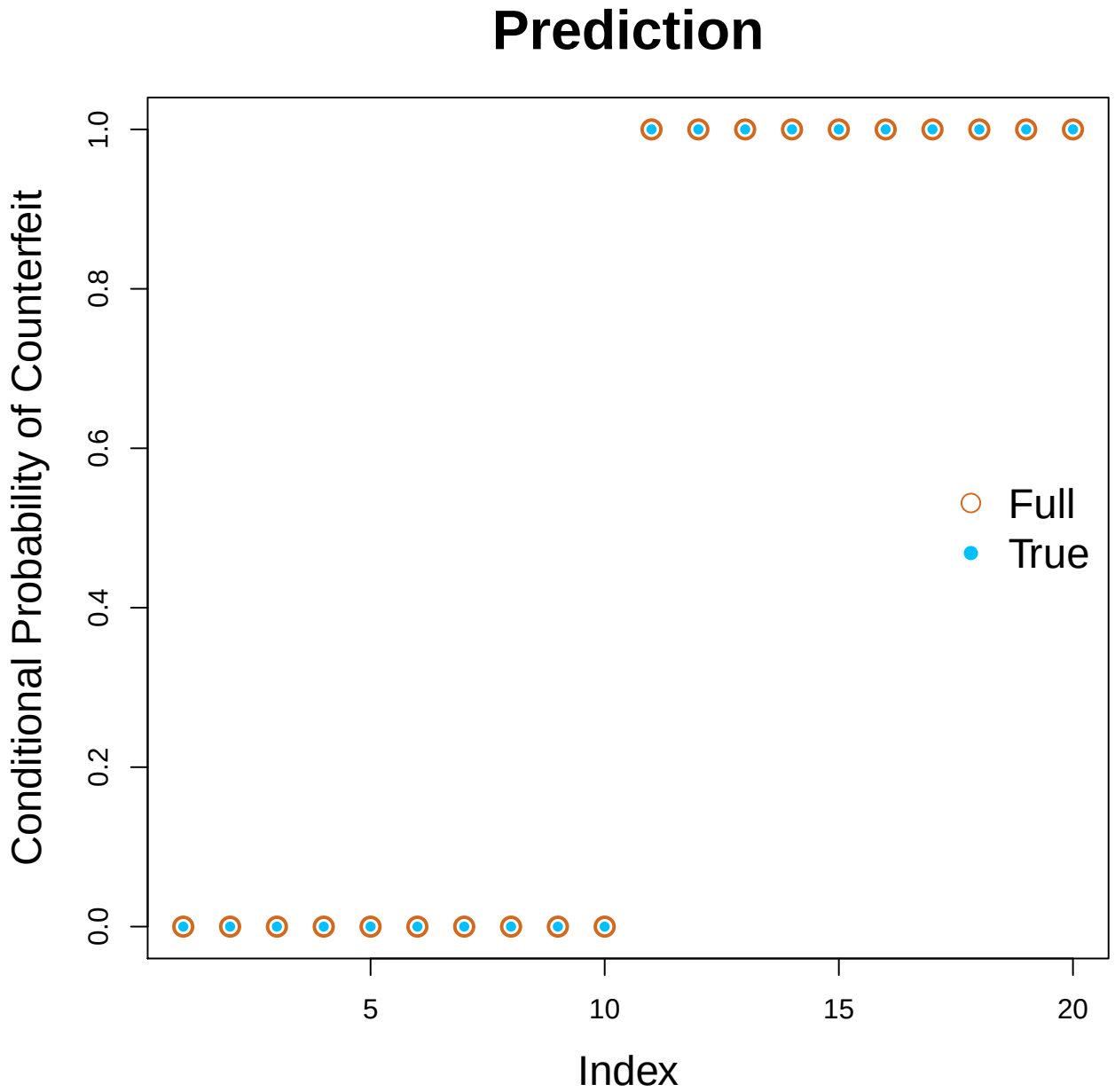
where $\mathbf{w}_1$ a unit eigenvector associated to the largest eigenvalue, and $\mathbf{w}_2$ is a unit eigenvector associated to the second largest eigenvalue, etc. They are known as the first, second, third principal components, etc.

- Consider the following dataset which is about Swiss bank notes

`Y`	Genuine (0) or counterfeit (1)	
`Length`	Length	mm
`Left`	Width of left edge	mm
`Right`	Width of right edge	mm
`Bottom`	Bottom margin width	mm
`Top`	Top margin width	mm
`Diagonal`	Length of diagonal	mm

```
> sm.df = read.table("~/Desktop/swiss_money.txt",
+                    sep = "", header = TRUE)

> # Identify rows corresponding to genuine
> row_id_0 = which(sm.df$Y == 0)
> row_id_1 = which(sm.df$Y == 1)
```

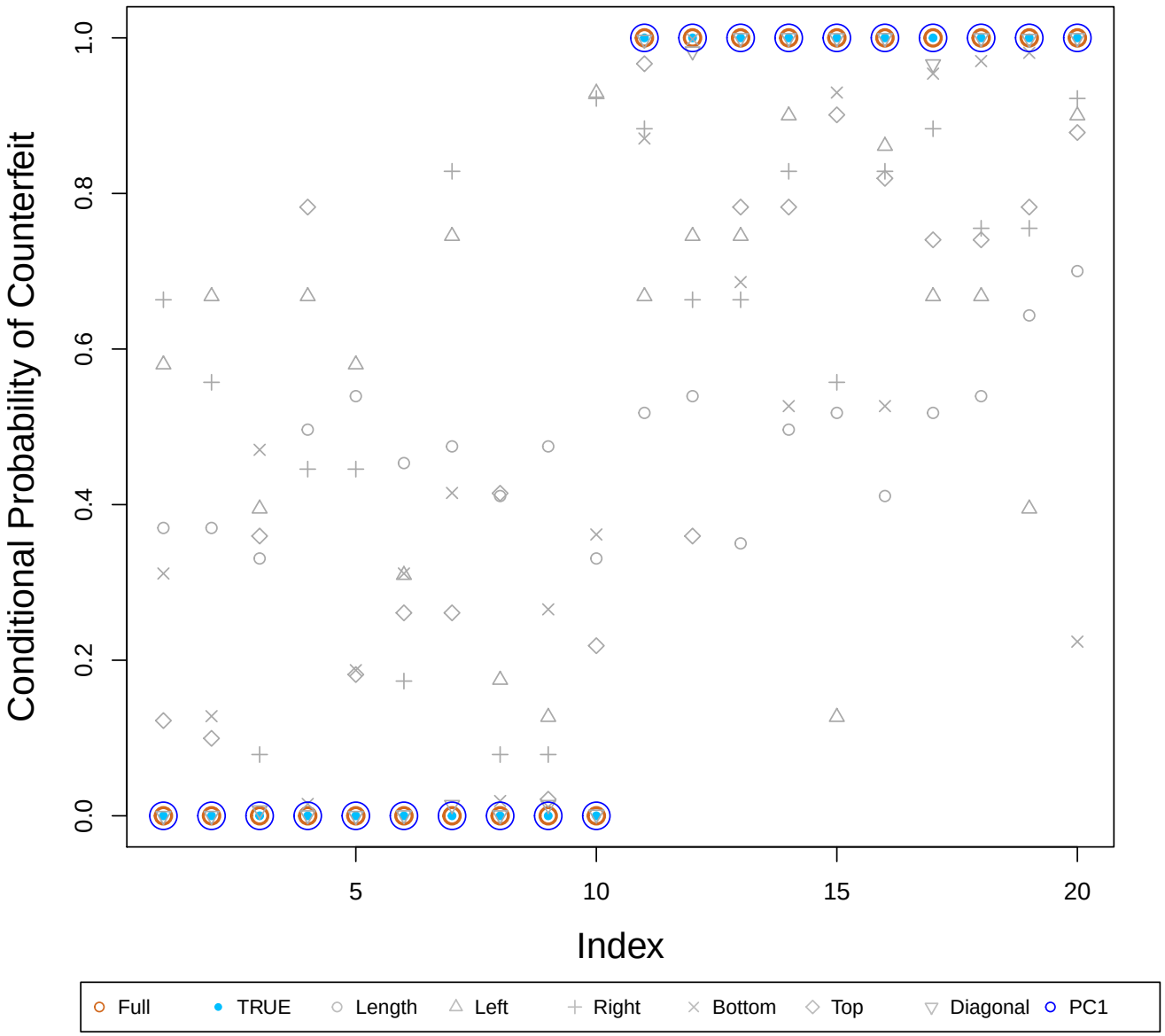
```
> n = 10 # leave out 10 genuine and 10 counterfeit
>
> # randomly generate those 20 observations
> set.seed(1)
>
> index_0 = sample(row_id_0, size = n)
> index_1 = sample(row_id_1, size = n)

> library(gam)
> # Full model
> sm.LG =
+   gam(Y~s(Length) + s(Left) + s(Right) + s(Bottom)
+       + s(Top) + s(Diagonal), family = binomial,
+       data = sm.df[-c(index_0, index_1),])
> # Prediction for those 20 observations
> sm_full =
+   predict(sm.LG, sm.df[c(index_0, index_1),],
+           type = "response")
```


Prediction
Conditional Probability of Counterfeit
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
Index
Full
True

```
> # models with one variable at a time
> vname = names(sm.df)
> m = 6 # number of independent variables
> pred = matrix(nrow = 2*n, ncol = m)

> for (i in 1:m){
+   f = as.formula(paste("Y~", vname[i], sep = ""))
+
+   sm.tmp.LG = gam(f, family = binomial,
+                   data = sm.df[-c(index_0, index_1),])
+
+   tmp = predict(
+     sm.tmp.LG, sm.df[c(index_0, index_1),],
+     type = "response")
+
+   pred[, i] = tmp
+
+ }
```

```
> # Centering the data matrix
> X = scale(sm.df[, -7], scale = FALSE)

> e = eigen(t(X) %*% X) # solve eigen problem

> z1 = X %*% e$vectors[,1] # first principal comp

> # remove testing cases from the ``new data''
> z1.m = z1[-c(index_0, index_1),]

> sm.pc1.LG = gam(
+   Y~s(z1.m), family = binomial,
+   data = sm.df[-c(index_0, index_1),])

> z1.m = z1[c(index_0, index_1),]
> sm_pc1 =
+   predict(sm.pc1.LG, data.frame(z1.m = z1.m),
+           type = "response")
```

Prediction


Conditional Probability of Counterfeit
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
Index
Full
TRUE
Length
Left
Right
Bottom
Top
Diagonal
PC1

- Notice we have converted the minimisation problem

$$\min_{\{\mathbf{w}:\ \|\mathbf{w}\|=1\}} \sum_{i=1}^{n} \left\|\mathbf{r}_i - \left(\mathbf{r}_i^{\mathrm{T}}\mathbf{w}\right)\mathbf{w}\right\|^2$$

  to the maximisation problem merely for computation reasons

$$\max_{\{\mathbf{w}:\ \|\mathbf{w}\|=1\}} \|\mathbf{X}\mathbf{w}\|^2$$

- However, it can be understood as maximising the sample variance of z since

$$\frac{1}{n-1}\|\mathbf{X}\mathbf{w}\|^2 = s_z^2$$

  where s_z^2 denotes the sample variance of z.

- Note we have centred the data before PCA is performed, when the variables are measured in different units, we need to scaled the data as well.

- To illustrate the scaling effect, consider the the dataset for the 50 US states

`Murder`	numeric murder arrests	per 100,000
`Assault`	numeric assault arrests	per 100,000
`UrbanPop`	numeric percent urban population	%
`Rape`	numeric Rape arrests	per 100,000

```
> var(USArrests) # Covariance matrix
```

	Murder	Assault	UrbanPop	Rape
Murder	18.970465	291.0624	4.386204	22.99141
Assault	291.062367	6945.1657	312.275102	519.26906
UrbanPop	4.386204	312.2751	209.518776	55.76808
Rape	22.991412	519.2691	55.768082	87.72916

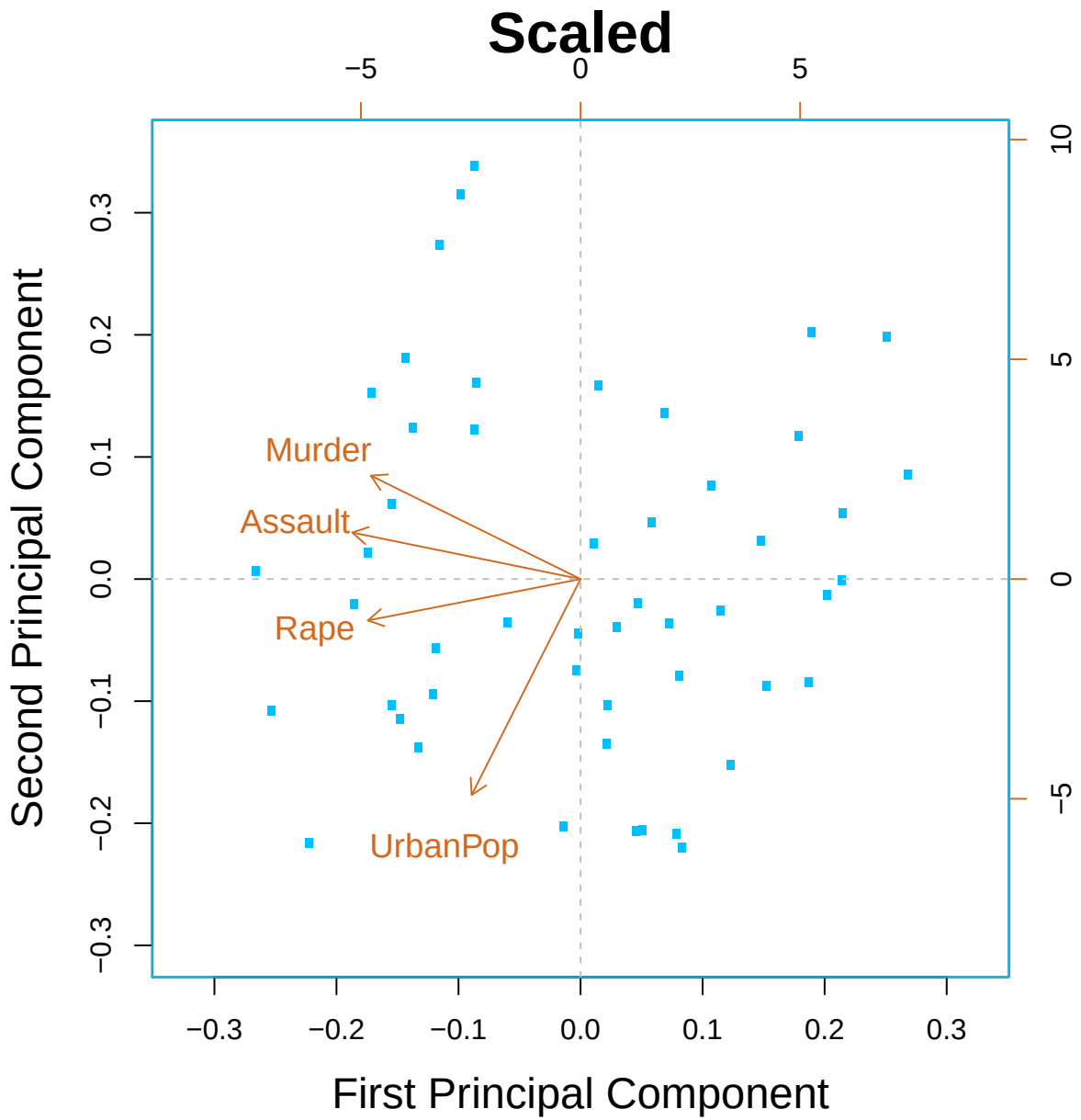
- Notice the variance of `Assault` is dominantly large, followed by `UrbanPop`.

- Of course, R has functions for PCA

```
> USArrests.PCA = prcomp(
+    USArrests, center = TRUE, scale. = TRUE)
```

- Typically, PCA is visualised using a biplot, which depict the plane defined by the first and second eigenvectors with projections of the original data points.

```
> biplot(USArrests.PCA, main = "Scaled",
+         xlab = "First Principal Component",
+         ylab = "Second Principal Component",
+         asp = TRUE, cex.lab = 1.5,
+         cex.main = 2, cex=c(3, 1.2),
+         xlim = c(-0.3, 0.3), ylim = c(-0.3, 0.35),
+         col=c("deepskyblue", "chocolate"),
+         xlabs=rep(" ", nrow(USArrests)))
>
> abline(h = 0, lty = 2, col = "grey")
> abline(v = 0, lty = 2, col = "grey")
```


Scaled
−5
0
5
10
5
0
−5
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Second Principal Component
Murder
Assault
Rape
UrbanPop
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
First Principal Component

- According to the biplot, the first eigenvector places roughly equal weight on

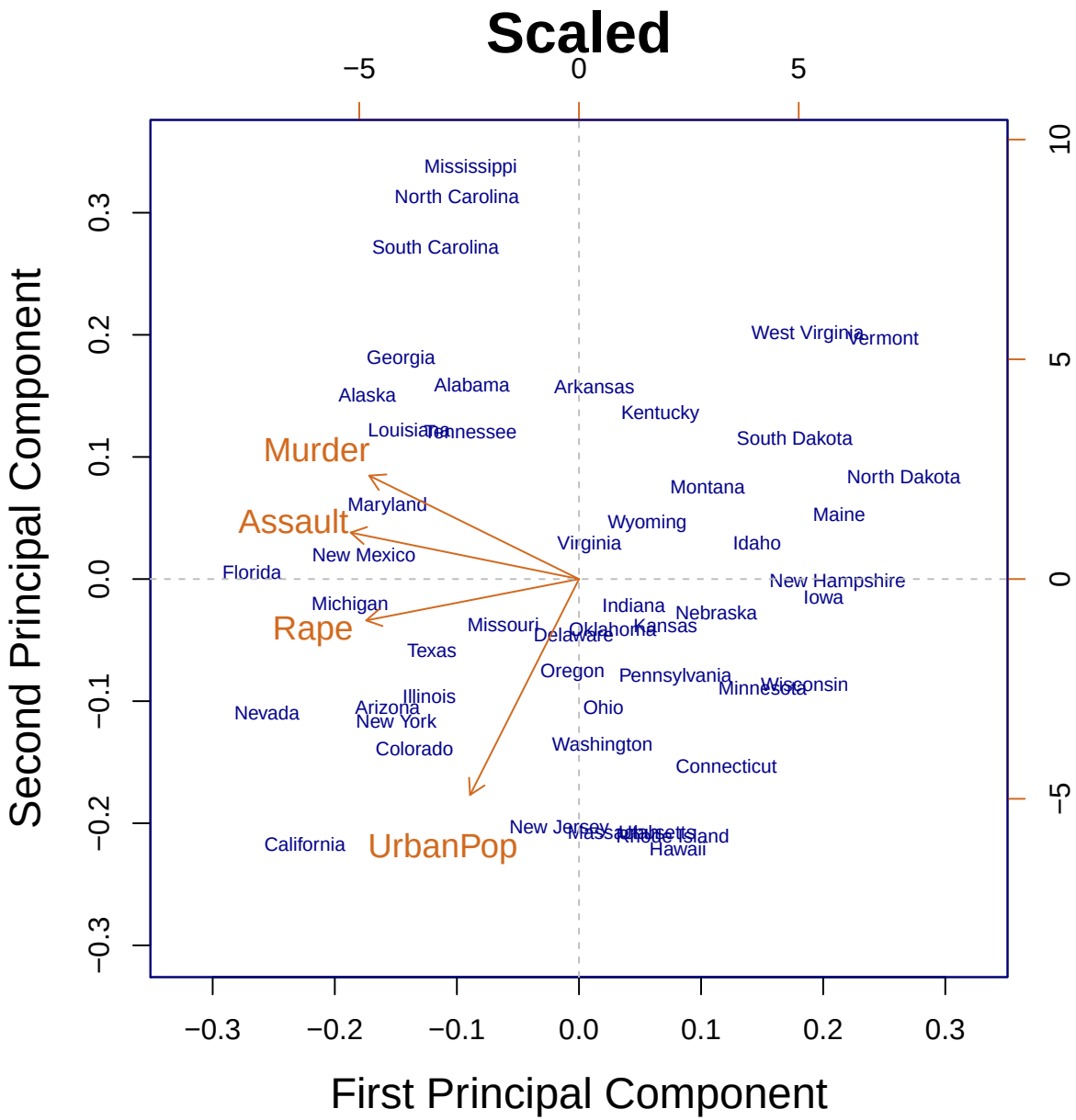
  `Assault`, `Murder` and `Rape`

  with less weight on `UrbanPop`.
- So the 1st-pc roughly corresponds to a measure of overall serious crimes.
- The 2nd eigenvector places most of its weight on

  `UrbanPop`,

  so the 2nd-pc roughly corresponds to the level of urbanisation.
- Furthermore, we see that the crime-related variables are positioned close to each other, which indicates crime-related variables are correlated, that is, states with high murder rates tend to have high assault and rate rates.
- By default, row names are used for each observations in biplot.

Q: What can you conclude?


Scaled
−5
0
5
10
5
0
−5
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Second Principal Component
First Principal Component
Murder
Assault
Rape
UrbanPop
Mississippi
North Carolina
South Carolina
Georgia
Alabama
Alaska
Louisiana
Tennessee
Arkansas
Kentucky
West Virginia
Vermont
South Dakota
North Dakota
Montana
Maine
Wyoming
Idaho
Virginia
Maryland
New Mexico
Florida
Michigan
Missouri
Texas
Indiana
Nebraska
Kansas
Oklahoma
Delaware
New Hampshire
Iowa
Oregon
Pennsylvania
Minnesota
Wisconsin
Illinois
Nevada
Arizona
New York
Colorado
Ohio
Washington
Connecticut
New Jersey
Rhode Island
Hawaii
California

- If we don't scale, that is, we don't divide the data by the standard deviations

```
> USArrests.u.PCA = prcomp(
+    USArrests, center = TRUE, scale. = FALSE)

> sapply(USArrests, var)
```

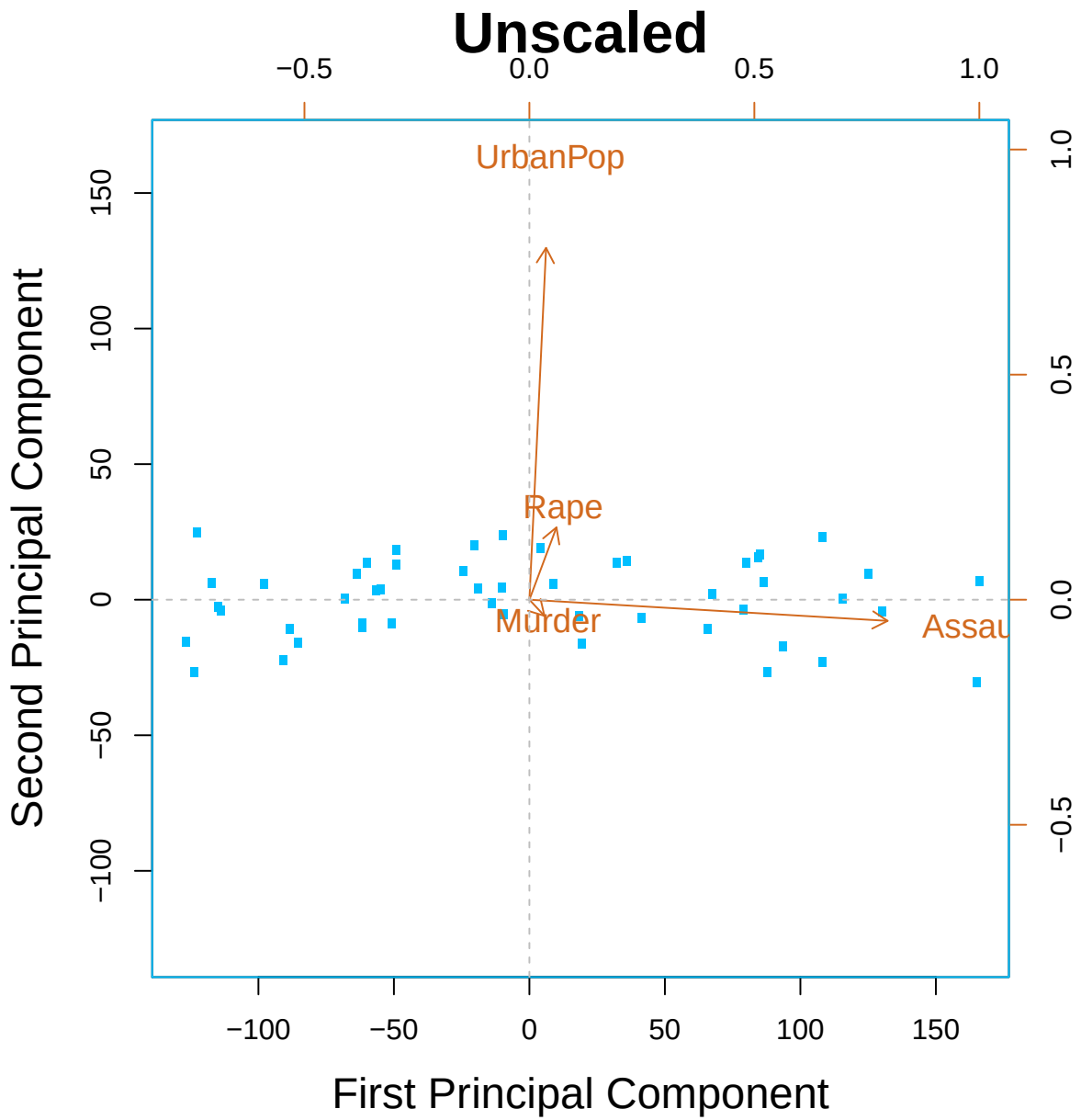
Murder	Assault	UrbanPop	Rape
18.97047	6945.16571	209.51878	87.72916

- If we put it in the context of maximising variance, it is not surprising that

  `Assault` and `UrbanPop`

  will have the largest weights.

- So we usually scale our data as well as centring them unless they are all measured in the same units, like our Swiss bank note example.


Unscaled
−0.5
0.0
0.5
1.0
UrbanPop
Rape
Murder
Assau
150
100
50
0
−50
−100
1.0
0.5
0.0
−0.5
−100
−50
0
50
100
150
Second Principal Component
First Principal Component

- It is natural to ask how much of the information in a given dataset is lost by using the first ℓ principal components, because it is usually used to decide how many principal components should be included in the investigation.
- Since $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is symmetric, there exists k orthonormal eigenvectors in $\mathbb{R}^k$

$$\mathcal{B} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_k\}$$

so the data matrix of all principal components captures all information

$$\mathbf{Z} = \mathbf{X}\mathbf{W}$$

where $\mathbf{W}$ is a matrix with vectors in $\mathcal{B}$ as its columns.

- The proportion of variability explained by the first ℓ-pc is given

$$R^2 = \frac{\sum_{j=1}^{\ell} s_{z_j}^2}{\sum_{j=1}^{k} s_{z_j}^2} = \frac{\sum_{j=1}^{\ell} \lambda_j}{\sum_{j=1}^{k} \lambda_j}$$

- Consider the following data which contains expression levels on 6830 genes from 64 cancer cell lines. Cancer type is also recorded.

```
> gene.df = read.table("~/Desktop/NCI60.csv",
+                      sep = ",", header = TRUE)
```

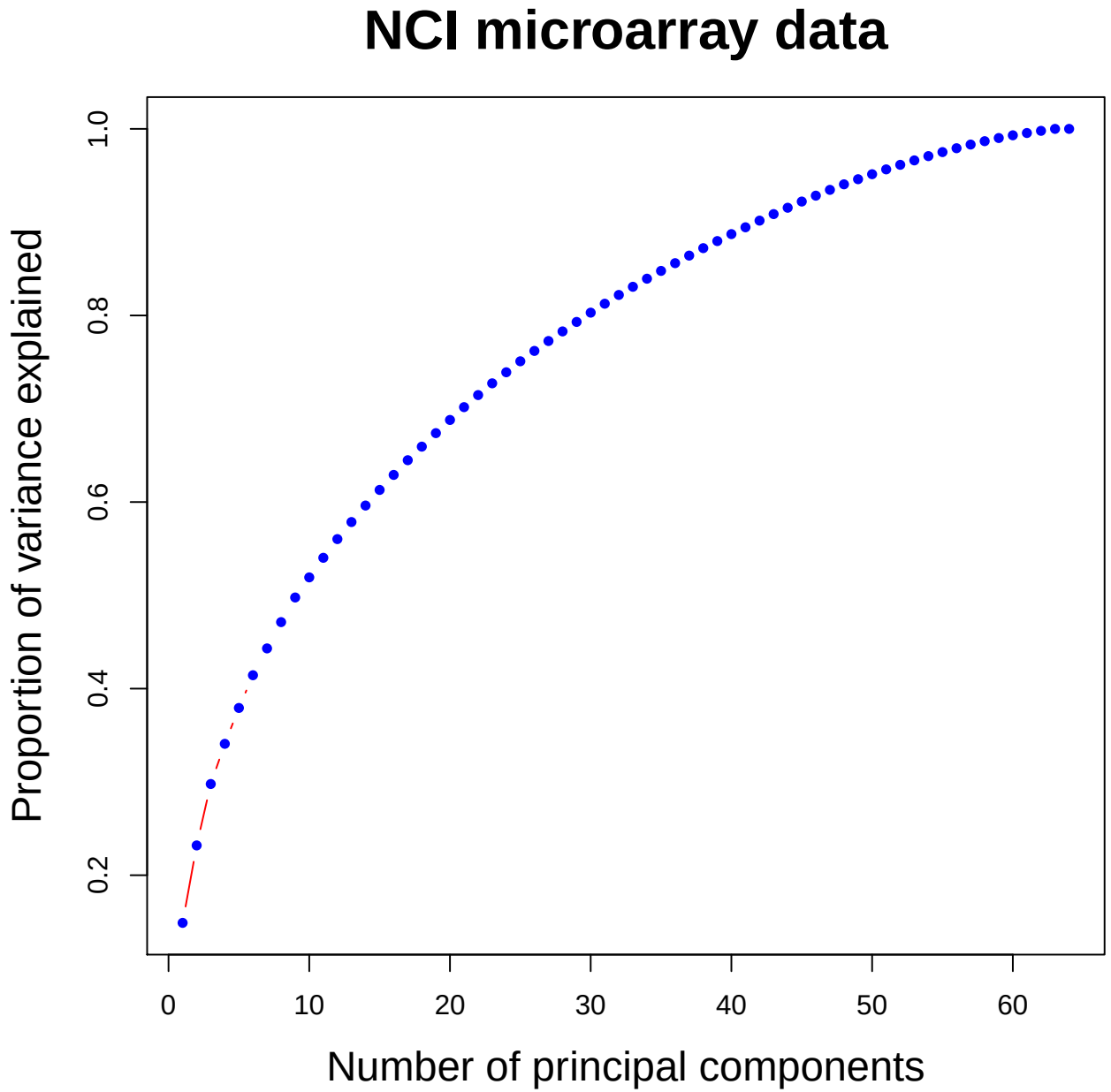
```
> dim(gene.df)
```

```
[1]   64 6832
```

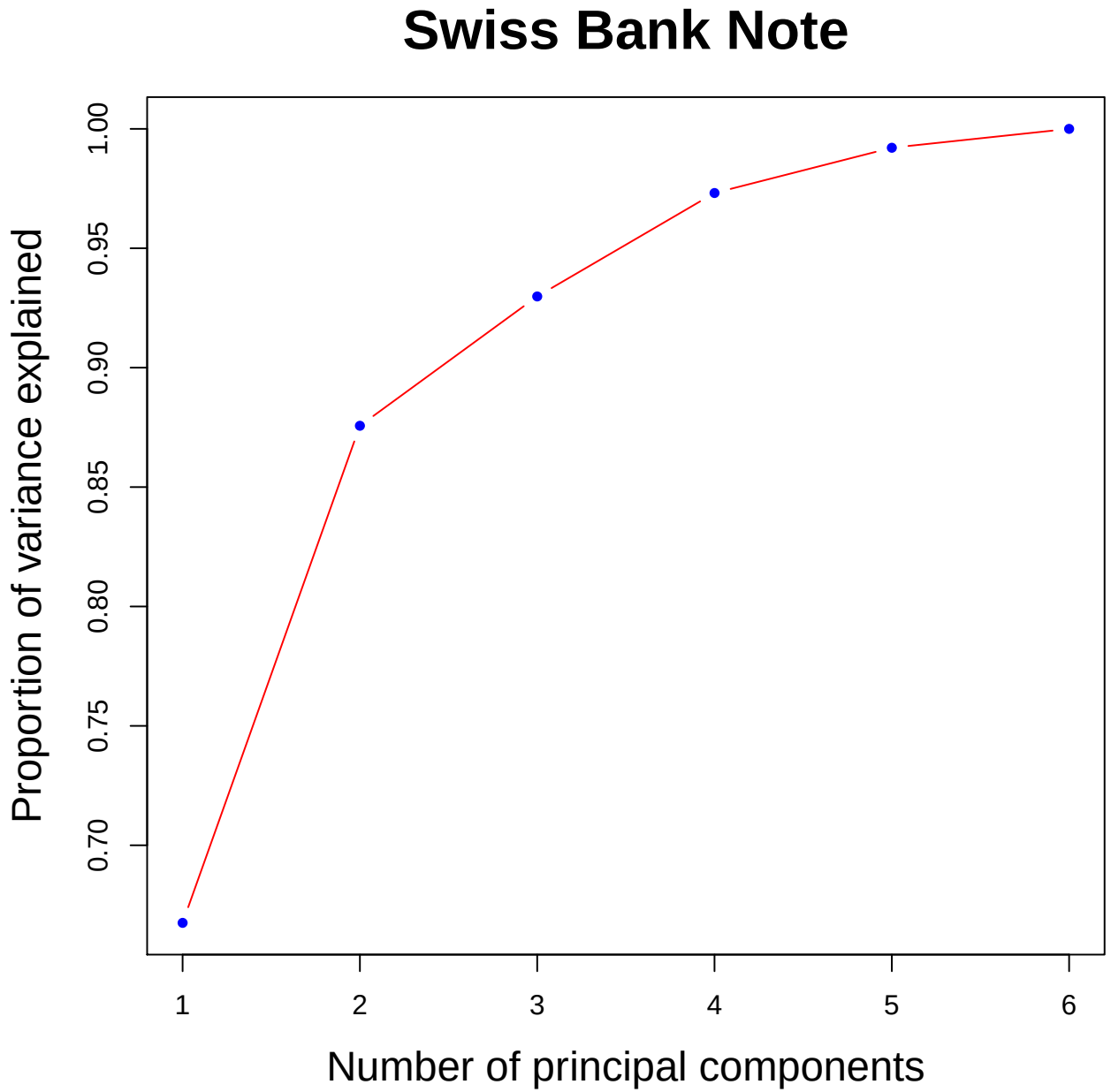
```
> class.vec = sapply(gene.df, class)
> table(class.vec)
```

```
class.vec
 factor numeric
      2    6830
```

```
> index.rm = which(class.vec == "factor")
```


NCI microarray data
Proportion of variance explained
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
Number of principal components

```
> # Remove factor or character variables
> gene.PCA = prcomp(
+   gene.df[, -index.rm], center = TRUE)
>
>
> prop_variance =
+   cumsum(gene.PCA$sdev^2)/sum(gene.PCA$sdev^2)

> plot(prop_variance, col = 2,
+      type = "c", main = "NCI microarray data",
+      xlab = "Number of principal components",
+      ylab = "Proportion of variance explained",
+      cex.lab = 1.5, cex.main = 2)
>
>
> points(prop_variance, pch = 20, col = 4)
```


Swiss Bank Note
Proportion of variance explained
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1
2
3
4
5
6
Number of principal components