Методы понижения размерности

Елена Ставровская

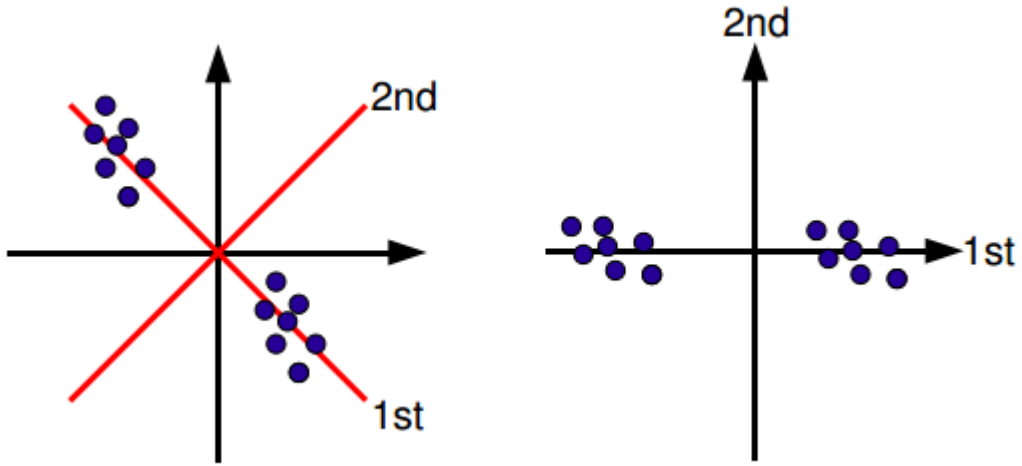
Цель

- Способ смотреть на данные из многомерного пространства
- Выявление значимых координат

Метод главных компонент
(*Principal component analysis, PCA*)

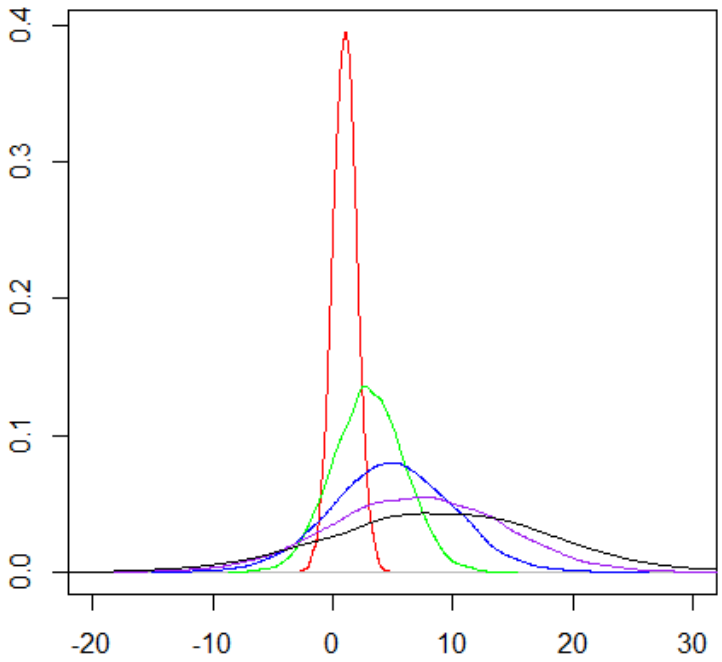
- ✓ один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации
- ✓ Преобразование некоторого набора возможно скоррелированных переменных в набор линейно нескоррелированных переменных – главных компонент
- ✓ Число главных компонент всегда меньше или равно количеству начальных переменных
- ✓ Первая компонента имеет наибольшую возможную дисперсию (т.е. объясняет вариабильность данных настолько, насколько это возможно)
- ✓ Каждая следующая компонента имеет наибольшую дисперсию при условии, что она ортогональна предыдущим

PCA – по сути ортогональное линейное преобразование, переводящее данные в новую систему координат.



Principal Component	**1st**	**2nd**	**3rd**	**Other**
Proportion of Variance	62%	34%	3%	rest

1st and 2nd principal components explain 96% of variance.

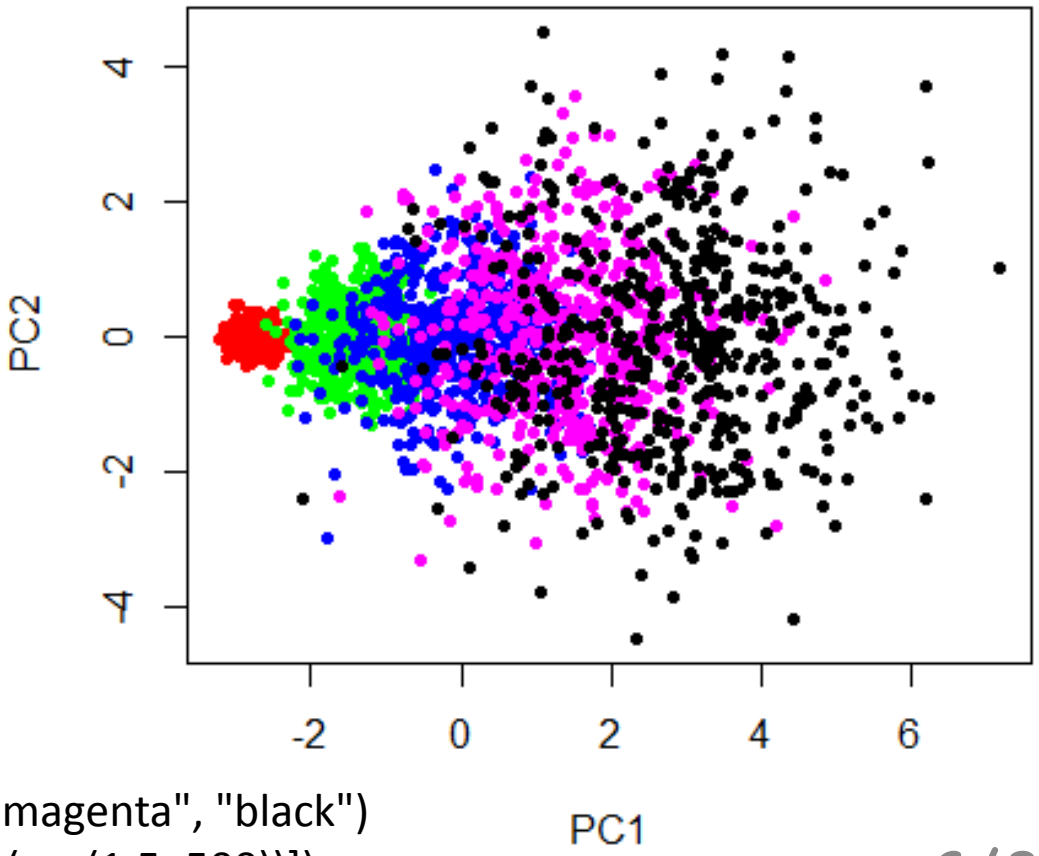



```
z1 <- rnorm(10000, mean=1, sd=1)
z2 <- rnorm(10000, mean=3, sd=3)
z3 <- rnorm(10000, mean=5, sd=5)
z4 <- rnorm(10000, mean=7, sd=7)
z5 <- rnorm(10000, mean=9, sd=9)
mydata <- matrix(c(z1, z2, z3, z4, z5), 2500, 20, byrow=T, dimnames=list(paste("R",
1:2500, sep=""), paste("C", 1:20, sep="")))
  # генерируем матрицу, 2500 строк и 20 столбцов

pca <- prcomp(mydata, scale=T)
```

summary(pca) # Выводит на экран summary для всех компонент
summary(pca)$importance[, 1:6] # для первых шести компонент

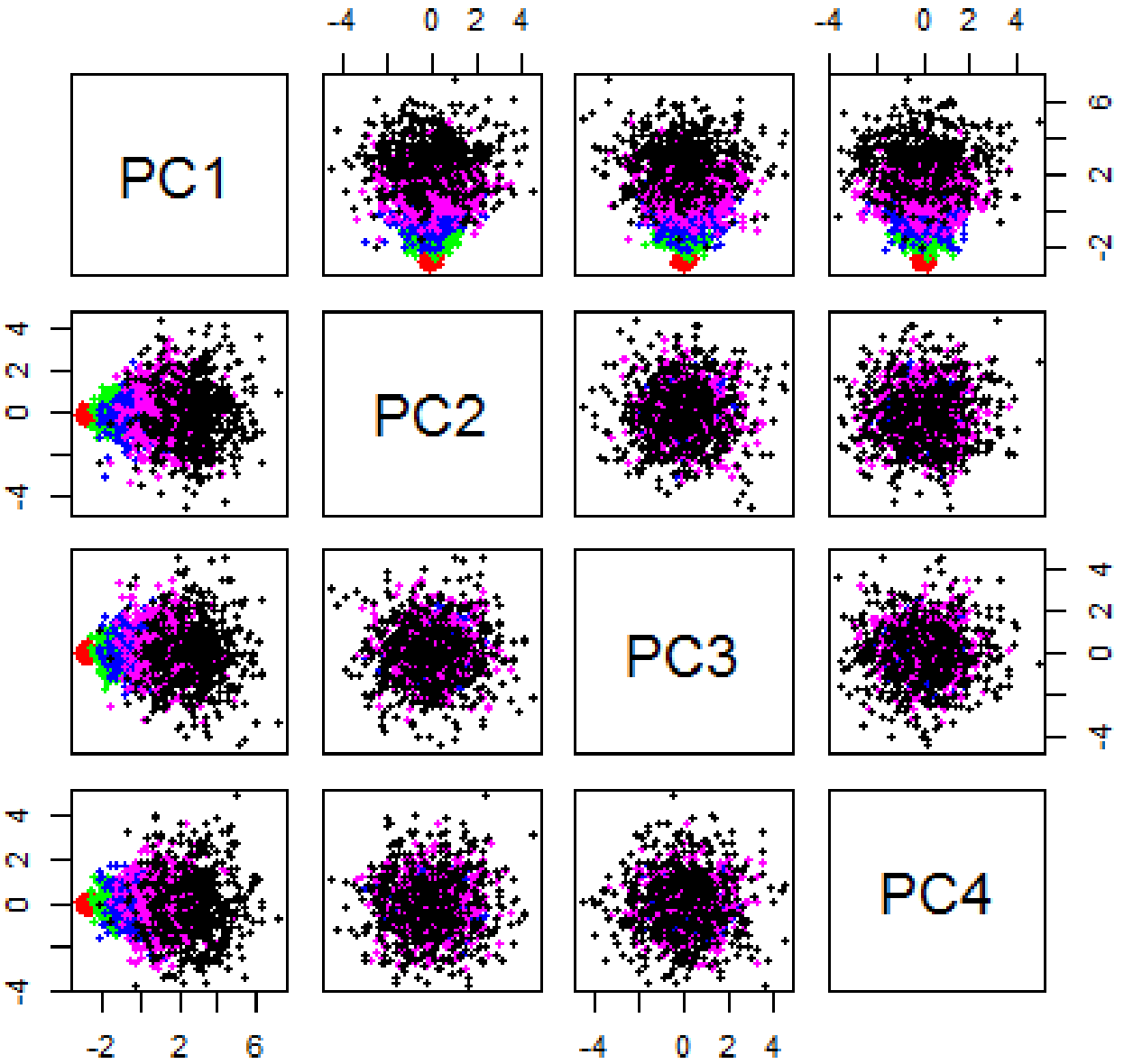
```
                            PC1       PC2       PC3       PC4       PC5       PC6
Standard deviation     2.151885 0.9901776 0.9758579 0.9586941 0.9454844 0.9412403
Proportion of Variance 0.231530 0.0490200 0.0476100 0.0459500 0.0447000 0.0443000
Cumulative Proportion  0.231530 0.2805500 0.3281700 0.3741200 0.4188200 0.4631200
```

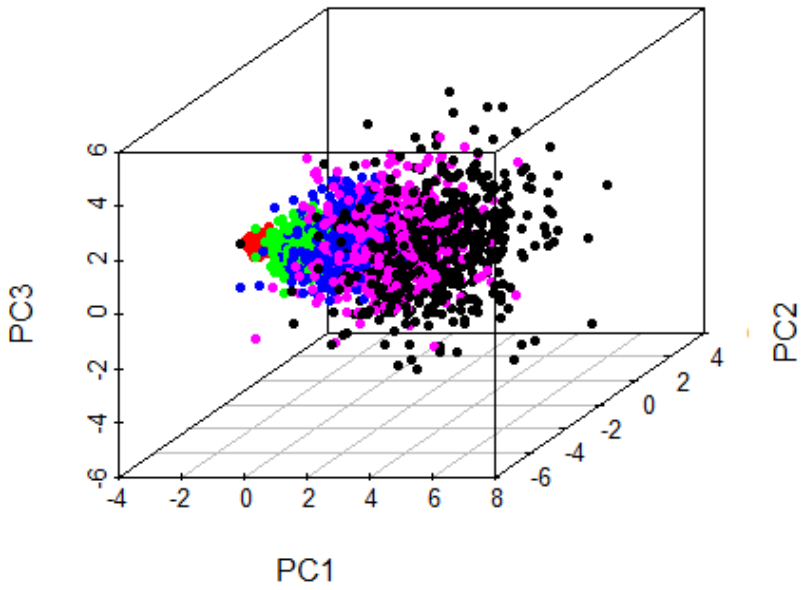


mycolors <- c("red", "green", "blue", "magenta", "black")
plot(pca$x, pch=20, col=mycolors[sort(rep(1:5, 500))])

Перевод старых координат в новые

```
> head(pca$rotation)
          PC1         PC2         PC3         PC4
C1 0.2331362 -0.06723522  0.07145997 -0.1251237
C2 0.2201653  0.32161194 -0.11223440  0.2130697
C3 0.2374093  0.14604708  0.27549168  0.0388385
C4 0.2222868  0.14031927  0.22169891 -0.2411913
C5 0.2310132 -0.11770617 -0.12619603  0.3079368
C6 0.2343129  0.02973445 -0.22135328 -0.0362403
```


PC1
PC2
PC3
PC4
-4
0
2
4
-2
2
6



pca$x[,1:3]

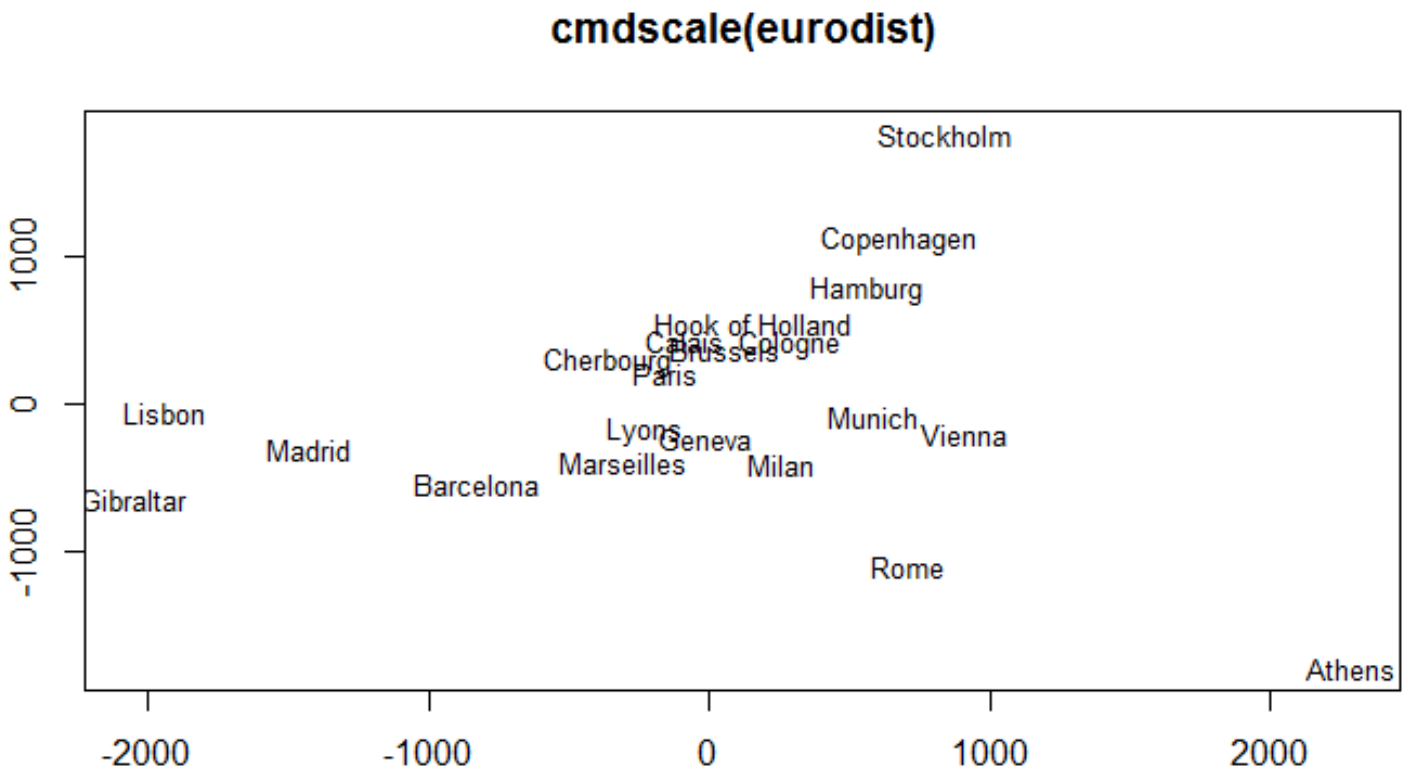
- здесь содержатся коэффициенты для перехода в новую систему координат, по первым 3 компонентам: можно использовать для дальнейшей кластеризации

Многомерное шкалирование

- ✓ На входе - попарные сходства/различия анализируемых объектов (матрица расстояний)

- ✓ На выходе числовые значения координат, которые приписываются каждому объекту в некоторой новой системе координат, с сохранением (насколько это возможно) попарных расстояний между объектами

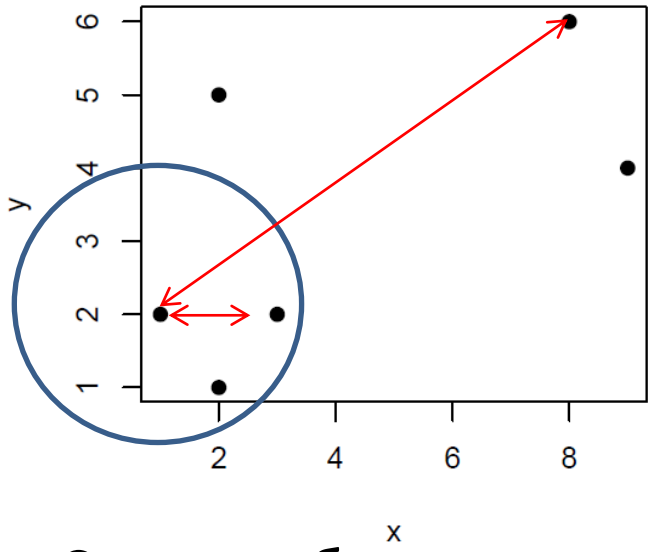
```
loc <- cmdscale(eurodist)
# MDS для географических расстояний между европейскими городами
plot(loc[,1], -loc[,2], type="n", xlab="", ylab="", main="cmdscale(eurodist)")
text(loc[,1], -loc[,2], rownames(loc), cex=0.8)
# строим график, минус нужен для того, чтобы перевернуть график в нужную ориентацию
```



tSNE

- t-distributed stochastic neighbor embedding
- Метод понижения размерности (до 2 или 3)
- Замеряет меру сходства точек – насколько точки близки при Гауссовом распределении расстояний вокруг точки с параметром сигма (сигма для каждой точки своя).
- Преобразует сходство между двумя точками в условные вероятности (при условии остальных наблюдаемых расстояний между точками)

tSNE



$$p_{j|i} = \frac{exp(- \parallel x_i - x_j \parallel^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(- \parallel x_i - x_k \parallel^2 / 2\sigma_i^2)}$$

Сигма выбирается так, выбирается так, чтобы точки в областях с большей плотностью имели меньшую дисперсию. Для этого оценивается **перплексия:**

$$Perp(P_i) = 2^{H(P_i)}$$

Перплексия - сглаженная оценка эффективного количества «соседей» для точки X_i, параметр метода. Рекомендованные значения от 5 до 50 (можно подбирать бинарным поиском)

tSNE

Близки точки – близки и образы точек

$$q_{j|i} = \frac{exp(- \parallel y_i - y_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel y_i - y_k \parallel^2)}$$

Если точки отображения Y_i и Y_j корректно моделируют сходство между исходными точками высокой размерности X_i и X_j, то соответствующие условные вероятности $p_{j|i}$ и $q_{j|i}$ будут эквивалентны

$$Cost = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

tSNE

- Оптимизируем Cost методом градиентного спуска
- В результате близкие точки в проекции сближаются, далекие растаскиваются

tSNE

- Недостатки:
  - Для больших выборок работает долго
  - Градиентный спуск находит локальный минимум
  - При добавлении новых точек нужно все пересчитывать

tSNE

- B R:

```
install.packages(“Rtsne”)
library(“Rtsne”)
```

tSNE

tSNE работает с уникальными объектами – убираем дубли

```
> iris_unique <- unique(iris)
```

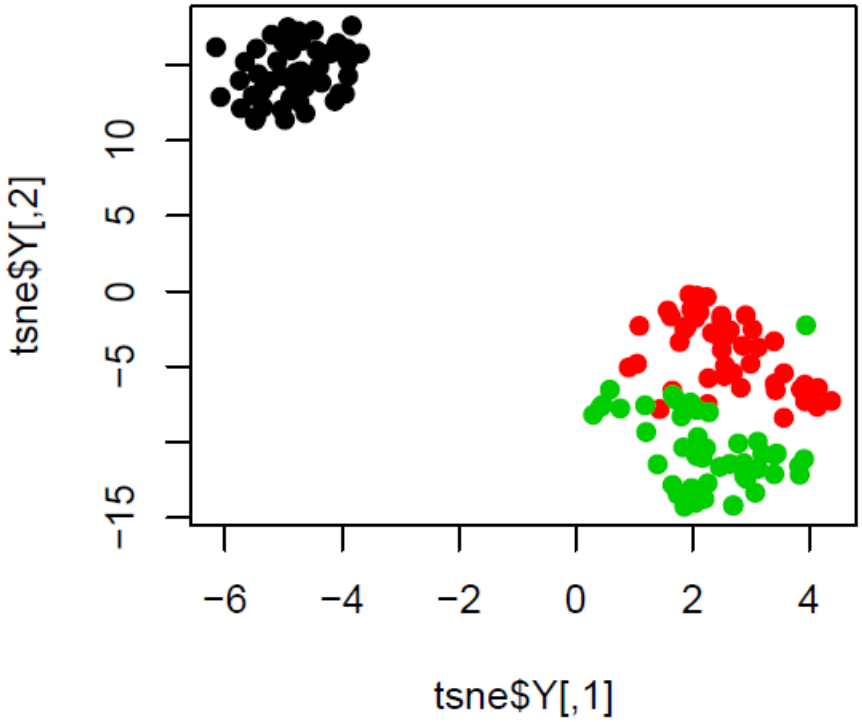
По-умолчанию размерность 2 (для 3 надо указывать dim=3)

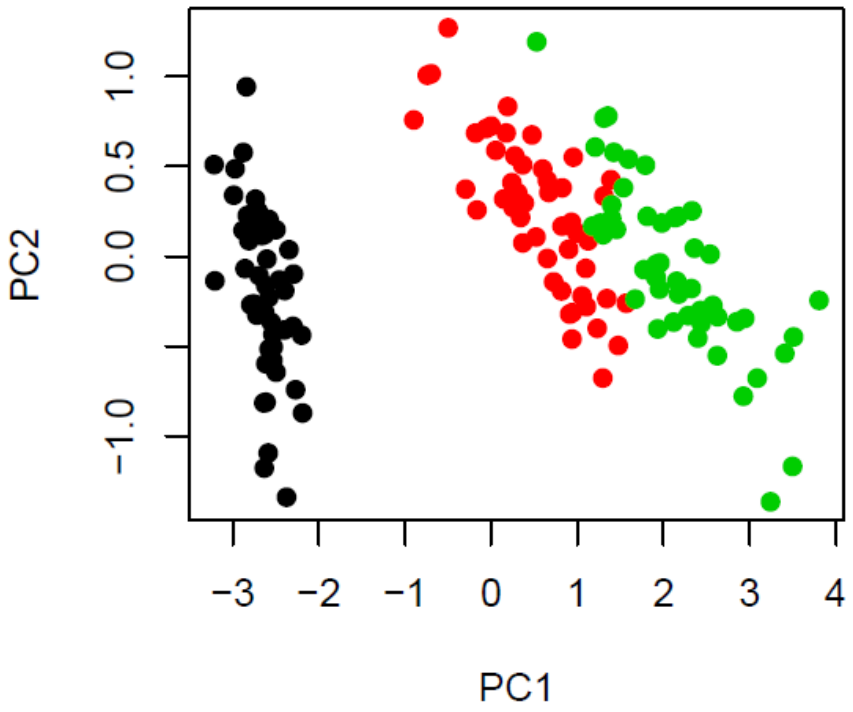
```
> tsne<-Rtsne(iris_unique[, c(-5)])
> plot(tsne$Y, col=iris_unique[,5], pch=19)
```

Сделаем для сравнения PCA

```
> pca<-prcomp(iris_unique[, -5])
> plot(pca$x, col=iris_unique[,5], pch=19)
```

tSNE





Классификация раковых образцов на основании зависимостей мутационных процессов от состояний хроматина.