聚类有效性指标的比较分析——基于5种距离度量方法

Yang Le

2020-04-08

产生数据

函数GenData

输入：

1. n: 向量，每个类簇的样本个数，向量长度作为类簇数量k
2. p: 有效变量个数
3. sigma: 向量，每个类簇的标准差，长度与k一致
4. outlier: 离群点的个数
5. centers: 给定类簇中心（若不赋值将随机产生）

```
library(mvtnorm)
GenData <- function(n = c(50,50), p = 2, sigma = c(0.06,0.06), outlier = 0, centers = NA){
  # 类簇个数
  k <- length(n)

  # 判断参数输入是否有效
  if(k!=length(sigma)){
    print("类别数量与标准差参数的长度不匹配！")
    return(NULL)
  }
  # 给定类簇中心（若不赋值将随机产生）
  if(is.na(centers)[1])
    # 从等差数列中抽样产生类簇中心（保证“完全分离”）
    centers <- replicate(p,sample(c(-3:3),k,replace = F))
  else if(k!=dim(centers)[1]){
    print("类别数量与类中心参数的长度不匹配！")
    return(NULL)
  }

  data <- c() # 生成的数据
  clustLab <- c() # 类簇类别的标签
  for(i in 1:k){
    center <- centers[i,]
    covar <- matrix(0,p,p)
    # 协方差矩阵
    diag(covar) <- sigma[i]
    cluster <- rmvnorm(n[i], center, covar)
    # 产生一个类簇
    data <- rbind(data, cluster)
    # 合并已生成的数据和标签
    clustLab <- c(clustLab,rep(i,n[i]))
  }

  # 产生噪音点 （不属于任何一类）
  if(outlier > 0){
    # 从均匀分布中产生噪音点
```

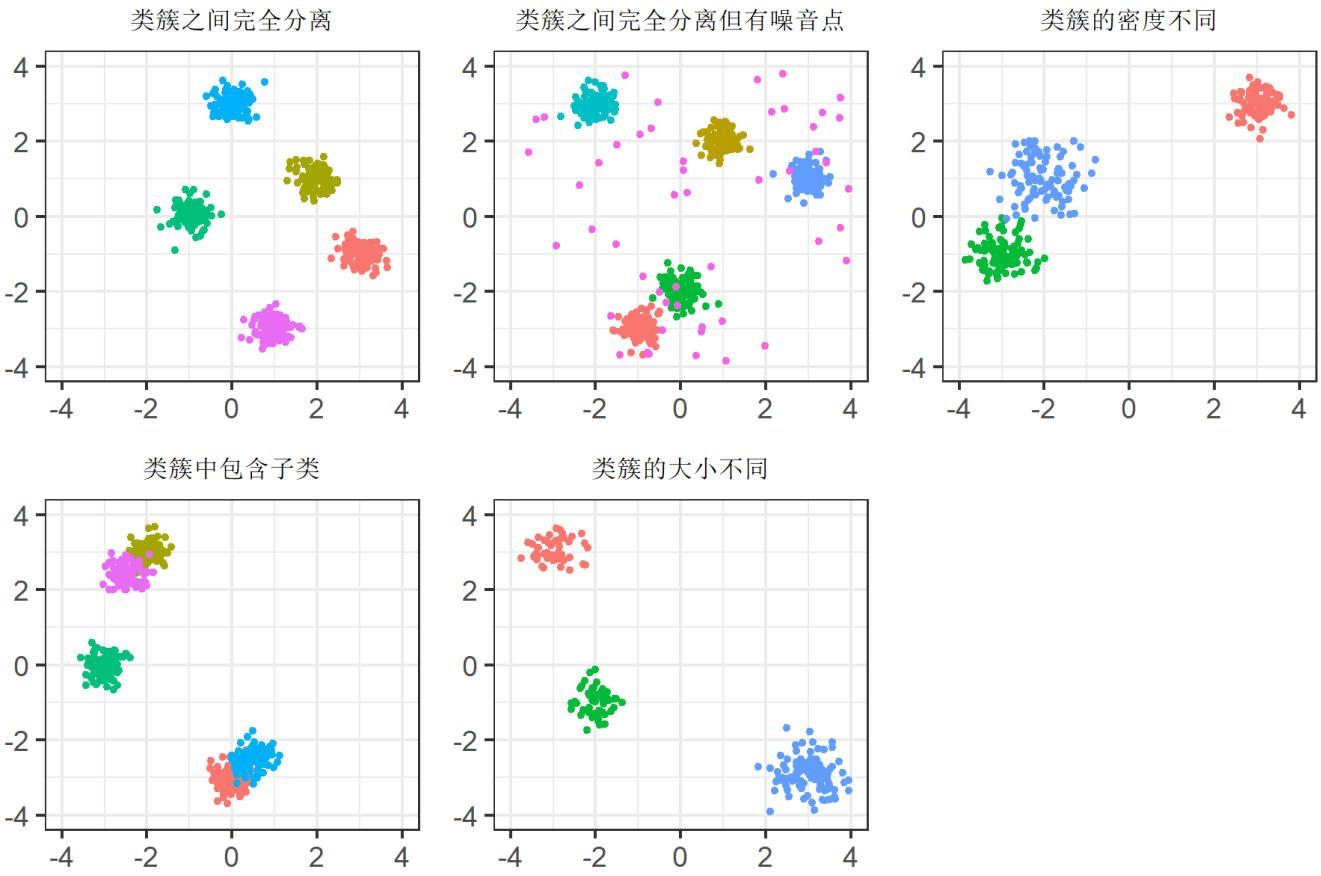
```
    outdata <- t(replicate(outlier, runif(p,-4,4)))
    # 合并数据和标签
    data <- rbind(data,outdata)
    clustLab <- c(clustLab,rep(k+1,outlier))
  }

  # 返回结果（数据和类簇标签）
  out <- list(data=data, clustLab=clustLab)
  return(out)
}
```

以生成2维的数据为例

1. 完全分离
2. 完全分离有噪音点
3. 类簇的密度不同
4. 包含子类
5. 类簇的大小不同

```
# 作图函数
library(ggplot2)
library(ggthemes)
clustPlot <- function(data,title=NA){
  data <- data.frame(data$data, data$clustLab)
  colnames(data) <- c(paste("X",1:p,sep = ""), "clustLab")
  data$clustLab <- factor(data$clustLab)
  p<-ggplot(data,aes(X1,X2,color=clustLab))+
    geom_point(show.legend = FALSE,size=0.8)+
    ggtitle(title)+xlim(-4,4)+ylim(-4,4)+
    theme_bw()+
    theme(axis.title=element_blank(),
          plot.title = element_text(hjust = 0.5,size = 8))
  return(p)
}

# 1.    完全分离
p = 2
n <- rep(100,5)
sigma <- rep(0.06,5)
data_1 <- GenData(n = n, p = p, sigma = sigma, outlier = 0)
p1 <- clustPlot(data_1,"类簇之间完全分离")

# 2.    完全分离有噪音点
data_2 <- GenData(n = n, p = p, sigma = sigma, outlier = 50)
p2 <- clustPlot(data_2,"类簇之间完全分离但有噪音点")

# 3.    类簇的密度不同
n <- rep(100,3)
sigma <- c(0.08,0.15,0.3)
data_3 <- GenData(n = n, p = p, sigma = sigma, outlier = 0)
p3 <- clustPlot(data_3,"类簇的密度不同")

# 4.    包含子类
n <- rep(100,5)
sigma <- rep(0.06,5)
k <- 3 # 三个大类
k_sub <- 2 # 其中两类含有子类
centers <-replicate(p,sample(c(-3:3),k,replace = F))
bias <- sample(c(-0.5,0.5),4,replace = T) # 设置子类的偏移
dim(bias)<-c(2,2)
```

```r
centers <- rbind(centers, centers[sample(1:k,k_sub),] + bias)
data_4 <- GenData(n = n, p = p, sigma = sigma, outlier = 0, centers = centers)
p4 <- clustPlot(data_4,"类簇中包含子类")

# 5.    类簇的大小不同
n <- c(50,50,100)
sigma <- c(0.1,0.1,0.2)
data_5 <- GenData(n = n, p = p, sigma = sigma, outlier = 0)
p5 <- clustPlot(data_5,"类簇的大小不同")

library(ggpubr)
ggarrange(p1, p2, p3, p4, p5, nrow = 2, ncol = 3)
```



聚类及其有效性指标的计算

F1 score

R的MLmetrics包中的F1_score仅针对二分类，这里是个多分类问题，因此重新定义了函数f1_fun用于计算多分类的F1_score.
另外，直接计算F1_score会有标签不匹配的问题，这里将F1_score封装为一个新的函数，将聚类结果的类簇标签和真实的标签进行匹配。

```r
# 多分类的F1_score
f1_fun <- function(y_true,y_pred){
  class <- sort(unique(y_true))
  tp=NA
  fp=NA
  fn=NA
  for(i in 1:length(class)){
    tp[i] = sum(y_pred==class[i] & y_true==class[i])
    fp[i] = sum(y_pred==class[i] & y_true!=class[i])
    fn[i] = sum(y_pred!=class[i] & y_true==class[i])
```

```
  }
  f1 = 2*tp/(2*tp+fp+fn)
  return(mean(f1))
}

# 针对于聚类结果的F1_score（解决标签匹配问题）
computeF1_score <- function(TrueClust,EstmateClust){
  cluster_label <- unique(TrueClust)
  new_EstmateClust <- rep(length(cluster_label)+1,length(EstmateClust))
  for(i in cluster_label){
    true_i <- EstmateClust[TrueClust==i]
    # 以真实类簇的聚类标签的众数作为该类簇的标签
    label_i <- as.numeric(names(table(true_i))[table(true_i) == max(table(true_i))])
    new_EstmateClust[EstmateClust==label_i] <- i
  }
  # print(new_EstmateClust)
  return(f1_fun(new_EstmateClust,TrueClust))
}
```

内部有效性指标与其他外部有效性

NbClust函数根据内部有效性给出最佳聚类数。
? 聚类算法中使用的距离计算方法是?
? 模糊聚类cmeans的有效性指标xb?

```
library(NbClust)
library(fclust)
library(e1071)
library(ClusterR)
computeIndeices<-function(x,TrueClust,distance){
  # 内部有效性计算方法*9
  in_ind<-c("kl", "ch", "mcclain","db","silhouette", "dunn","ccc","sdindex","sdbw")

  re<-c()
  # NbClust中不支持马氏距离，事先进行计算
  if(distance=="mahalanobis"){
    distc<-matrix(nrow = nrow(x),ncol = nrow(x))
    for(i in 1:nrow(x)){
      for(j in 1:i){
        distij<-(((x[i,]-x[j,])%*% t(t(x[i,]-x[j,])))) / cov(x[i,],x[j,]))
        distc[i,j]<-distij
      }
    }
    dist<-as.dist(distc)
  }
  for(ind in in_ind){
    if(distance=="mahalanobis"){
      res_i <- NbClust(x, diss=dist,distance = NULL, min.nc=2, max.nc=12, # 内部有效性
                       method = "kmeans", index = ind)
    }else{
      res_i <- NbClust(x, distance = distance, min.nc=2, max.nc=12,
                       method = "kmeans", index = ind)
    }
    # 外部有效性
    jaccard <- external_validation(TrueClust, res_i$Best.partition ,method = "jaccard_index")
    purity <- external_validation(TrueClust, res_i$Best.partition ,method = "purity")
    F1_score <-computeF1_score(TrueClust,res_i$Best.partition)
    re<-cbind(re,c(res_i$Best.nc,jaccard,purity,F1_score))
    #assign(paste("res_",ind,sep = ""),res_i)
  }
```

```
  re<-data.frame(re)
  colnames(re)<-in_ind
  row.names(re)<-c("best_k","value_index","jaccard","purity","F1_score")
  return(re)
}
```

测试距离计算方法的影响

实验设计

对于每一种数据情形，随机生成100个数据集，对每一个数据集进行聚类（给定聚类数目的范围），采用5种距离计算方法计算聚类的内部有效性指标，以选择最优的聚类数目，并计算最优聚类的外部有效性指标。将100次重复试验的结果进行汇总作为最终的实验结果。
Data_compute函数：一轮实验，产生数据并计算指标。
res_combine函数： 对重复试验的结果进行汇总，并计算最优聚类数的方差。

```
# 一轮实验：产生数据并计算指标
Data_compute<-function(t,n = n, p = p, sigma = sigma, outlier = outlier, centers = NA){
  print(t)
  set.seed(t)
  Data <- GenData(n = n, p = p, sigma = sigma, outlier = outlier, centers = centers)
  out<-c()
  for(dist_m in distance){ # 测试不同的距离计算方式
    out <- rbind(out,computeIndeices(Data$data,Data$clustLab,dist_m))
  }
  return(out)
}

# 汇总结果：指标的平均值&最优类簇个数的方差
library(stringr)
res_combine<-function(res_all){
  res_sum<-Reduce("+",res_all)/length(times)
  res_sum<-cbind(rep(distance,each=5),
                 rep(c("best k","index value","jaccard","purity","F1 score"),5),
                 res_sum)
  colnames(res_sum)[1:2]<-c('distance','ex_index')
  best_kdf<-c()
  for(i in times){
    res<-res_all[[i]]
    best_ks<-res[str_detect(row.names(res),"best_k"),]
    row.names(best_ks)<-distance
    best_kdf<-rbind(best_kdf,best_ks)
  }
  best_kvar<-c()
  for(dist_m in distance){
    dist_bestk<-best_kdf[str_detect(row.names(best_kdf),dist_m),]
    best_kvar<-rbind(best_kvar,apply(dist_bestk, 2, var))
  }
  best_kvar<-cbind(distance,ex_index=rep("k_var",5),best_kvar)
  res_comb<-rbind(res_sum,best_kvar)
  res_comb<-res_comb[order(res_comb$distance),]
  return(res_comb)
}
```

1. 完全分离

```
# 重复100次，控制随机性的影响
times <- c(1:100)
distance<-c("mahalanobis","euclidean", "maximum", "manhattan", "canberra")

# 1. 完全分离
p = 6 # 6维
n <- rep(100,5) # 5类 每类100个
sigma <- rep(0.04,5)
outlier <- 0
# 重复实验
res_all1 <- lapply(times,Data_compute,n = n, p = p, sigma = sigma, outlier = outlier)
# 汇总结果
res_comb1<-res_combine(res_all1)
```

结果如下：

类簇完全分离的情况

	distance	ex_index	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	canberra	best k	7.02	6.92	2.03	3.78	4.90	3.21	6.96	3.74	11.92
3	canberra	jaccard	0.83	0.84	0.36	0.67	0.79	0.59	0.84	0.67	0.53
4	canberra	purity	1.00	1.00	0.41	0.75	0.88	0.64	1.00	0.75	1.00
5	canberra	F1 score	0.78	0.79	0.62	0.88	0.86	0.79	0.79	0.88	0.58
6	canberra	k_var	3.17	3.29	0.09	0.98	1.61	1.38	3.49	0.86	0.07
7	euclidean	best k	7.02	6.92	3.78	3.78	4.97	3.32	6.96	3.74	11.92
9	euclidean	jaccard	0.83	0.84	0.61	0.67	0.80	0.61	0.84	0.67	0.53
10	euclidean	purity	1.00	1.00	0.68	0.75	0.89	0.66	1.00	0.75	1.00
11	euclidean	F1 score	0.78	0.79	0.74	0.88	0.86	0.81	0.79	0.88	0.58
12	euclidean	k_var	3.17	3.29	3.87	0.98	1.81	1.31	3.49	0.86	0.07
13	mahalanobis	best k	7.02	6.92	5.82	3.78	2.03	2.13	6.96	3.74	11.92
15	mahalanobis	jaccard	0.83	0.84	0.53	0.67	0.36	0.37	0.84	0.67	0.53
16	mahalanobis	purity	1.00	1.00	0.73	0.75	0.40	0.42	1.00	0.75	1.00
17	mahalanobis	F1 score	0.78	0.79	0.64	0.88	0.61	0.62	0.79	0.88	0.58
18	mahalanobis	k_var	3.17	3.29	9.79	0.98	0.09	0.30	3.49	0.86	0.07
19	manhattan	best k	7.02	6.92	4.09	3.78	5.05	3.34	6.96	3.74	11.92
21	manhattan	jaccard	0.83	0.84	0.63	0.67	0.80	0.61	0.84	0.67	0.53
22	manhattan	purity	1.00	1.00	0.72	0.75	0.89	0.67	1.00	0.75	1.00
23	manhattan	F1 score	0.78	0.79	0.75	0.88	0.86	0.81	0.79	0.88	0.58
24	manhattan	k_var	3.17	3.29	4.43	0.98	1.77	1.34	3.49	0.86	0.07
25	maximum	best k	7.02	6.92	3.75	3.78	5.11	3.31	6.96	3.74	11.92
27	maximum	jaccard	0.83	0.84	0.60	0.67	0.81	0.61	0.84	0.67	0.53
28	maximum	purity	1.00	1.00	0.67	0.75	0.90	0.66	1.00	0.75	1.00
29	maximum	F1 score	0.78	0.79	0.74	0.88	0.86	0.81	0.79	0.88	0.58
30	maximum	k_var	3.17	3.29	3.91	0.98	1.65	1.27	3.49	0.86	0.07

2. 完全分离有噪音点

```
# 2. 完全分离有噪音点
p = 6 # 6维
n <- rep(100,5) # 5类 每类100个
sigma <- rep(0.04,5)
outlier <- 50 # 50个噪音点
# 重复实验
res_all2 <- lapply(times,Data_compute,n = n, p = p, sigma = sigma, outlier = outlier)
# 汇总结果
res_comb2<-res_combine(res_all2)
```

结果如下：

类簇完全分离有噪音点的情况

	distance	ex_index	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	canberra	best k	5.89	6.56	2.00	4.11	7.41	5.67	7.43	4.96	5.85
3	canberra	jaccard	0.78	0.88	0.30	0.62	0.89	0.62	0.89	0.73	0.82
4	canberra	purity	0.88	0.95	0.36	0.75	0.96	0.70	0.96	0.84	0.91
5	canberra	F1 score	0.79	0.84	0.45	0.74	0.80	0.65	0.82	0.79	0.82
6	canberra	k_var	3.51	1.72	0.00	0.73	2.65	12.10	3.34	1.49	2.47
7	euclidean	best k	5.89	6.56	5.41	4.11	8.20	7.96	7.43	4.96	5.85
9	euclidean	jaccard	0.78	0.88	0.54	0.62	0.89	0.82	0.89	0.73	0.82
10	euclidean	purity	0.88	0.95	0.62	0.75	0.96	0.90	0.96	0.84	0.91
11	euclidean	F1 score	0.79	0.84	0.60	0.74	0.78	0.74	0.82	0.79	0.82
12	euclidean	k_var	3.51	1.72	18.08	0.73	3.41	6.62	3.34	1.49	2.47
13	mahalanobis	best k	5.89	6.56	5.01	4.11	2.27	2.18	7.43	4.96	5.85
15	mahalanobis	jaccard	0.78	0.88	0.62	0.62	0.34	0.32	0.89	0.73	0.82
16	mahalanobis	purity	0.88	0.95	0.74	0.75	0.41	0.40	0.96	0.84	0.91
17	mahalanobis	F1 score	0.79	0.84	0.71	0.74	0.48	0.48	0.82	0.79	0.82
18	mahalanobis	k_var	3.51	1.72	4.68	0.73	0.40	0.29	3.34	1.49	2.47
19	manhattan	best k	5.89	6.56	5.64	4.11	7.91	8.01	7.43	4.96	5.85
21	manhattan	jaccard	0.78	0.88	0.56	0.62	0.89	0.82	0.89	0.73	0.82
22	manhattan	purity	0.88	0.95	0.64	0.75	0.96	0.90	0.96	0.84	0.91
23	manhattan	F1 score	0.79	0.84	0.61	0.74	0.79	0.74	0.82	0.79	0.82
24	manhattan	k_var	3.51	1.72	18.29	0.73	3.11	7.12	3.34	1.49	2.47
25	maximum	best k	5.89	6.56	5.03	4.11	8.69	6.68	7.43	4.96	5.85
27	maximum	jaccard	0.78	0.88	0.51	0.62	0.90	0.73	0.89	0.73	0.82
28	maximum	purity	0.88	0.95	0.58	0.75	0.97	0.82	0.96	0.84	0.91
29	maximum	F1 score	0.79	0.84	0.58	0.74	0.77	0.73	0.82	0.79	0.82
30	maximum	k_var	3.51	1.72	18.19	0.73	3.51	9.71	3.34	1.49	2.47

3. 类簇的密度不同

```
# 3.    类簇的密度不同
p = 6 # 6维
n <- rep(100,5) # 5类 每类100个
sigma <- c(0.04,0,04,0.1,0.15,0.15) # sigma不同
outlier <- 0
```

```
# 重复实验
res_all3 <- lapply(times,Data_compute,n = n, p = p, sigma = sigma, outlier = outlier)
# 汇总结果
res_comb3<-res_combine(res_all3)
```

结果如下：

类簇的密度不同的情况

	distance	ex_index	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	canberra	best k	7.00	6.78	2.00	3.80	4.90	3.06	6.82	3.79	10.52
3	canberra	jaccard	0.83	0.85	0.36	0.68	0.80	0.57	0.85	0.68	0.61
4	canberra	purity	1.00	1.00	0.40	0.76	0.89	0.61	1.00	0.76	1.00
5	canberra	F1 score	0.78	0.80	0.62	0.89	0.87	0.77	0.80	0.89	0.63
6	canberra	k_var	3.25	3.14	0.00	0.93	1.69	1.37	3.14	0.87	2.07
7	euclidean	best k	7.00	6.78	2.21	3.80	4.83	3.42	6.82	3.79	10.52
9	euclidean	jaccard	0.83	0.85	0.40	0.68	0.80	0.63	0.85	0.68	0.61
10	euclidean	purity	1.00	1.00	0.44	0.76	0.88	0.68	1.00	0.76	1.00
11	euclidean	F1 score	0.78	0.80	0.65	0.89	0.87	0.83	0.80	0.89	0.63
12	euclidean	k_var	3.25	3.14	0.55	0.93	1.78	1.28	3.14	0.87	2.07
13	mahalanobis	best k	7.00	6.78	5.72	3.80	2.04	2.05	6.82	3.79	10.52
15	mahalanobis	jaccard	0.83	0.85	0.56	0.68	0.36	0.36	0.85	0.68	0.61
16	mahalanobis	purity	1.00	1.00	0.75	0.76	0.41	0.41	1.00	0.76	1.00
17	mahalanobis	F1 score	0.78	0.80	0.67	0.89	0.62	0.62	0.80	0.89	0.63
18	mahalanobis	k_var	3.25	3.14	8.73	0.93	0.06	0.17	3.14	0.87	2.07
19	manhattan	best k	7.00	6.78	2.38	3.80	4.84	3.40	6.82	3.79	10.52
21	manhattan	jaccard	0.83	0.85	0.43	0.68	0.79	0.62	0.85	0.68	0.61
22	manhattan	purity	1.00	1.00	0.47	0.76	0.88	0.68	1.00	0.76	1.00
23	manhattan	F1 score	0.78	0.80	0.66	0.89	0.87	0.82	0.80	0.89	0.63
24	manhattan	k_var	3.25	3.14	1.05	0.93	1.97	1.29	3.14	0.87	2.07
25	maximum	best k	7.00	6.78	2.23	3.80	5.03	3.35	6.82	3.79	10.52
27	maximum	jaccard	0.83	0.85	0.40	0.68	0.81	0.62	0.85	0.68	0.61
28	maximum	purity	1.00	1.00	0.45	0.76	0.90	0.67	1.00	0.76	1.00
29	maximum	F1 score	0.78	0.80	0.65	0.89	0.86	0.81	0.80	0.89	0.63
30	maximum	k_var	3.25	3.14	0.60	0.93	1.65	1.26	3.14	0.87	2.07

4. 包含子类

```
# 4.     包含子类
p = 6 # 6维
n <- rep(100,5)
sigma <- rep(0.04,5)
k <- 3 # 三个大类
k_sub <- 2 # 其中两类含有子类
centers <-replicate(p,sample(c(-3:3),k,replace = F)) # 随机选取类中心
bias <- sample(c(-0.5,0.5),k_sub*p,replace = T) # 设置子类的偏移
dim(bias)<-c(k_sub,p)
centers <- rbind(centers, centers[sample(1:k,k_sub),] + bias)
outlier <- 0
```

```
# 重复实验
res_all4 <- lapply(times,Data_compute,n = n, p = p, sigma = sigma,
                   outlier = outlier, centers=centers)
# 汇总结果
res_comb4<-res_combine(res_all4)
```

结果如下：

包含子类的情况

	distance	ex_index	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	canberra	best k	4.31	4.89	2.37	2.37	2.37	2.37	6.00	2.37	11.98
3	canberra	jaccard	0.59	0.77	0.45	0.45	0.45	0.45	0.90	0.45	0.51
4	canberra	purity	0.68	0.85	0.47	0.47	0.47	0.47	1.00	0.47	1.00
5	canberra	F1 score	0.66	0.78	0.66	0.66	0.66	0.66	0.79	0.66	0.58
6	canberra	k_var	7.61	2.12	0.24	0.24	0.24	0.24	0.00	0.24	0.02
7	euclidean	best k	4.31	4.89	2.36	2.37	2.37	2.36	6.00	2.37	11.98
9	euclidean	jaccard	0.59	0.77	0.44	0.45	0.45	0.44	0.90	0.45	0.51
10	euclidean	purity	0.68	0.85	0.47	0.47	0.47	0.47	1.00	0.47	1.00
11	euclidean	F1 score	0.66	0.78	0.65	0.66	0.66	0.65	0.79	0.66	0.58
12	euclidean	k_var	7.61	2.12	0.23	0.24	0.24	0.23	0.00	0.24	0.02
13	mahalanobis	best k	4.31	4.89	4.84	2.37	2.00	2.00	6.00	2.37	11.98
15	mahalanobis	jaccard	0.59	0.77	0.51	0.45	0.38	0.38	0.90	0.45	0.51
16	mahalanobis	purity	0.68	0.85	0.70	0.47	0.40	0.40	1.00	0.47	1.00
17	mahalanobis	F1 score	0.66	0.78	0.61	0.66	0.58	0.58	0.79	0.66	0.58
18	mahalanobis	k_var	7.61	2.12	4.56	0.24	0.00	0.00	0.00	0.24	0.02
19	manhattan	best k	4.31	4.89	2.35	2.37	2.37	2.32	6.00	2.37	11.98
21	manhattan	jaccard	0.59	0.77	0.44	0.45	0.45	0.44	0.90	0.45	0.51
22	manhattan	purity	0.68	0.85	0.47	0.47	0.47	0.46	1.00	0.47	1.00
23	manhattan	F1 score	0.66	0.78	0.65	0.66	0.66	0.65	0.79	0.66	0.58
24	manhattan	k_var	7.61	2.12	0.23	0.24	0.24	0.22	0.00	0.24	0.02
25	maximum	best k	4.31	4.89	2.37	2.37	2.37	2.36	6.00	2.37	11.98
27	maximum	jaccard	0.59	0.77	0.45	0.45	0.45	0.44	0.90	0.45	0.51
28	maximum	purity	0.68	0.85	0.47	0.47	0.47	0.47	1.00	0.47	1.00
29	maximum	F1 score	0.66	0.78	0.66	0.66	0.66	0.65	0.79	0.66	0.58
30	maximum	k_var	7.61	2.12	0.24	0.24	0.24	0.23	0.00	0.24	0.02

5. 类簇的大小不同

```
# 5.    类簇的大小不同
p = 6 # 6维
n <- c(25,50,50,100,150)
sigma <- c(0.02,0.04,0.04,0.08,0.12)
outlier <- 0
# 重复实验
res_all5 <- lapply(times,Data_compute,n = n, p = p, sigma = sigma, outlier = outlier)
# 汇总结果
res_comb5<-res_combine(res_all5)
```

结果如下：

类簇大小不同的情况

	distance	ex_index	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	canberra	best k	6.06	5.60	2.08	3.60	4.10	3.07	5.88	3.70	9.63
3	canberra	jaccard	0.72	0.76	0.52	0.79	0.86	0.70	0.74	0.82	0.51
4	canberra	purity	0.96	0.96	0.68	0.86	0.90	0.80	0.97	0.87	0.96
5	canberra	F1 score	0.80	0.83	0.81	0.92	0.92	0.88	0.82	0.93	0.68
6	canberra	k_var	3.75	2.85	0.22	0.99	1.04	1.20	2.71	0.80	8.70
7	euclidean	best k	6.06	5.60	2.76	3.60	4.03	3.31	5.88	3.70	9.63
9	euclidean	jaccard	0.72	0.76	0.64	0.79	0.86	0.75	0.74	0.82	0.51
10	euclidean	purity	0.96	0.96	0.76	0.86	0.90	0.82	0.97	0.87	0.96
11	euclidean	F1 score	0.80	0.83	0.87	0.92	0.92	0.90	0.82	0.93	0.68
12	euclidean	k_var	3.75	2.85	1.17	0.99	0.92	1.21	2.71	0.80	8.70
13	mahalanobis	best k	6.06	5.60	6.91	3.60	2.01	2.21	5.88	3.70	9.63
15	mahalanobis	jaccard	0.72	0.76	0.48	0.79	0.51	0.51	0.74	0.82	0.51
16	mahalanobis	purity	0.96	0.96	0.85	0.86	0.67	0.68	0.97	0.87	0.96
17	mahalanobis	F1 score	0.80	0.83	0.68	0.92	0.81	0.80	0.82	0.93	0.68
18	mahalanobis	k_var	3.75	2.85	15.19	0.99	0.01	1.10	2.71	0.80	8.70
19	manhattan	best k	6.06	5.60	2.80	3.60	4.03	3.27	5.88	3.70	9.63
21	manhattan	jaccard	0.72	0.76	0.65	0.79	0.86	0.74	0.74	0.82	0.51
22	manhattan	purity	0.96	0.96	0.76	0.86	0.90	0.82	0.97	0.87	0.96
23	manhattan	F1 score	0.80	0.83	0.87	0.92	0.92	0.90	0.82	0.93	0.68
24	manhattan	k_var	3.75	2.85	1.25	0.99	0.88	1.17	2.71	0.80	8.70
25	maximum	best k	6.06	5.60	2.64	3.60	4.00	3.36	5.88	3.70	9.63
27	maximum	jaccard	0.72	0.76	0.62	0.79	0.85	0.76	0.74	0.82	0.51
28	maximum	purity	0.96	0.96	0.74	0.86	0.89	0.83	0.97	0.87	0.96
29	maximum	F1 score	0.80	0.83	0.86	0.92	0.92	0.91	0.82	0.93	0.68
30	maximum	k_var	3.75	2.85	1.06	0.99	1.05	1.20	2.71	0.80	8.70

6. 考虑一类特殊的数据：特征之间具有相关性

可以通过特征提取消除相关性，但会带来信息损失，是一个trade off的问题。在特征具有相关性的情形下，距离计算方式如何影响内部有效性？

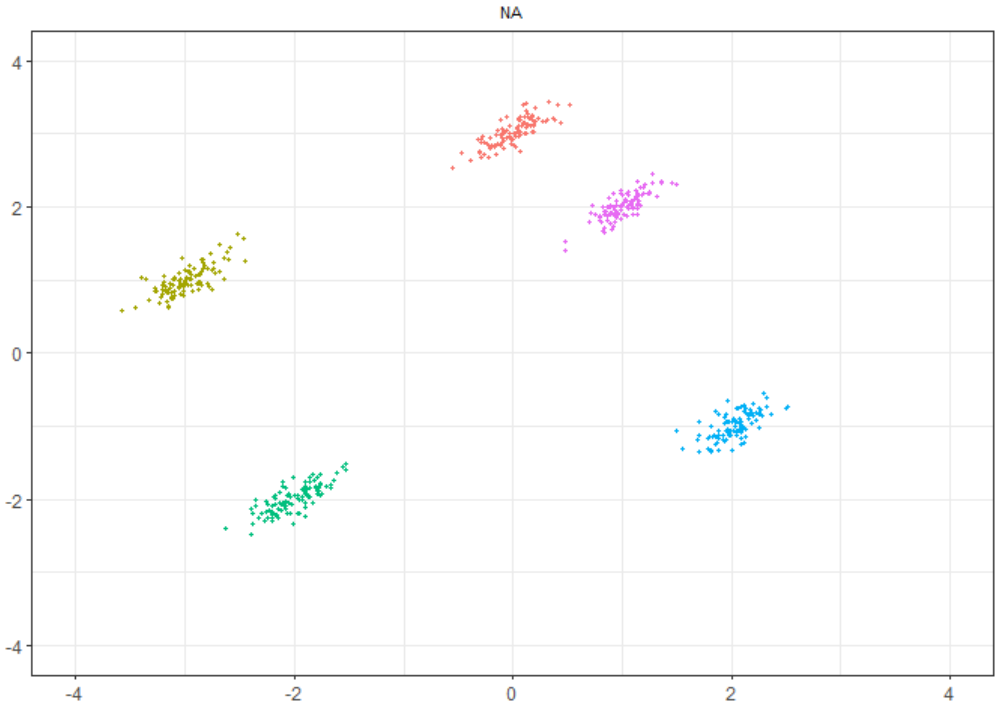
生成数据

```
# 生成数据：特征之间具有相关性
# 修改GenData函数，pr表示有相关性的特征维数（<=p），c表示协方差
library(Matrix)
GenDataR <- function(n = c(50,50), p = 2, pr = 2, c = 0.03,
                     sigma = c(0.04,0.04), outlier = 0, centers = NA){
  # 类簇个数
  k <- length(n)

  # 判断参数输入是否有效
  if(k!=length(sigma)){
```

```
    print("类别数量与标准差参数的长度不匹配！")
    return(NULL)
  }
  # 给定类簇中心（若不赋值将随机产生）
  if(is.na(centers)[1])
    # 从等差数列中抽样产生类簇中心（保证“完全分离”）
    centers <- replicate(p,sample(c(-3:3),k,replace = F))
  else if(k!=dim(centers)[1]){
    print("类别数量与类中心参数的长度不匹配！")
    return(NULL)
  }

  data <- c() # 生成的数据
  clustLab <- c() # 类簇类别的标签
  for(i in 1:k){
    center <- centers[i,]
    # 协方差矩阵
    covar <- as.matrix(sparseMatrix(i=rep(c(1:pr),each=pr),j=rep(c(1:pr),pr),
                            x=rep(c,pr^2),dims = c(p,p)))
    diag(covar) <- sigma[i]
    cluster <- rmvnorm(n[i], center, covar)
    # 产生一个类簇
    data <- rbind(data, cluster)
    # 合并已生成的数据和标签
    clustLab <- c(clustLab,rep(i,n[i]))
  }

  # 产生噪音点 （不属于任何一类）
  if(outlier > 0){
    # 从均匀分布中产生噪音点
    outdata <- t(replicate(outlier, runif(p,-3,3)))
    # 合并数据和标签
    data <- rbind(data,outdata)
    clustLab <- c(clustLab,rep(k+1,outlier))
  }

  # 返回结果（数据和类簇标签）
  out <- list(data=data, clustLab=clustLab)
  return(out)
}

Data_compute_r<-function(t,n = n, p = p, pr=pr , c=0.03 ,sigma = sigma, outlier = outlier, centers =
  print(t)
  set.seed(t)
  Data <- GenDataR(n = n, p = p, sigma = sigma,pr=pr , c=c,outlier = outlier, centers = centers)
  out<-c()
  for(dist_m in distance){ # 测试不同的距离计算方式
    out <- rbind(out,computeIndeices(Data$data,Data$clustLab,dist_m))
  }
  return(out)
}
```

以2维为例：

```
p = 2
n <- rep(100,5)
sigma <- rep(0.04,5)
data_r <- GenDataR(n = n, p = p, sigma = sigma, outlier = 0)
clustPlot(data_r)
```



距离计算方式的影响

```
# 重复100次，控制随机性的影响
times <- c(1:100)
distance<-c("mahalanobis","euclidean", "maximum", "manhattan", "canberra")

p = 6 # 6维
pr = 3 # 3个特征之间相关
n <- rep(100,5) # 5类 每类100个
sigma <- rep(0.04,5)
outlier <- 0
# 重复实验
res_all6 <- lapply(times,Data_compute_r,n = n, p = p, pr=pr,
                   sigma = sigma, outlier = outlier)
# 汇总结果
res_comb6<-res_combine(res_all6)
write.csv(res_comb6,"res_comb6.csv",row.names = F)
```

汇总结果

```
res_comb6<-read.csv("res_comb6.csv")
data_class <- c('完全分离','完全分离有噪音点','类簇的密度不同','包含子类','类簇的大小不同','特征之间有相关'
bestk_comb<-c()
for(i in 1:6){
  res_temp<-get(paste("res_comb",i,sep = ""))
  res_temp<-res_temp[res_temp$ex_index=="best k",]
  res_temp<-cbind(`data set`=rep(data_class[i],5),res_temp)
  bestk_comb<-rbind(bestk_comb,res_temp)
}
knitr::kable(bestk_comb[,-3], digits = 2, caption = "最优聚类数") %>%
  kable_styling(font_size = 12)
```

最优聚类数

	data set	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
1	完全分离	canberra	7.02	6.92	2.03	3.78	4.90	3.21	6.96	3.74	11.92

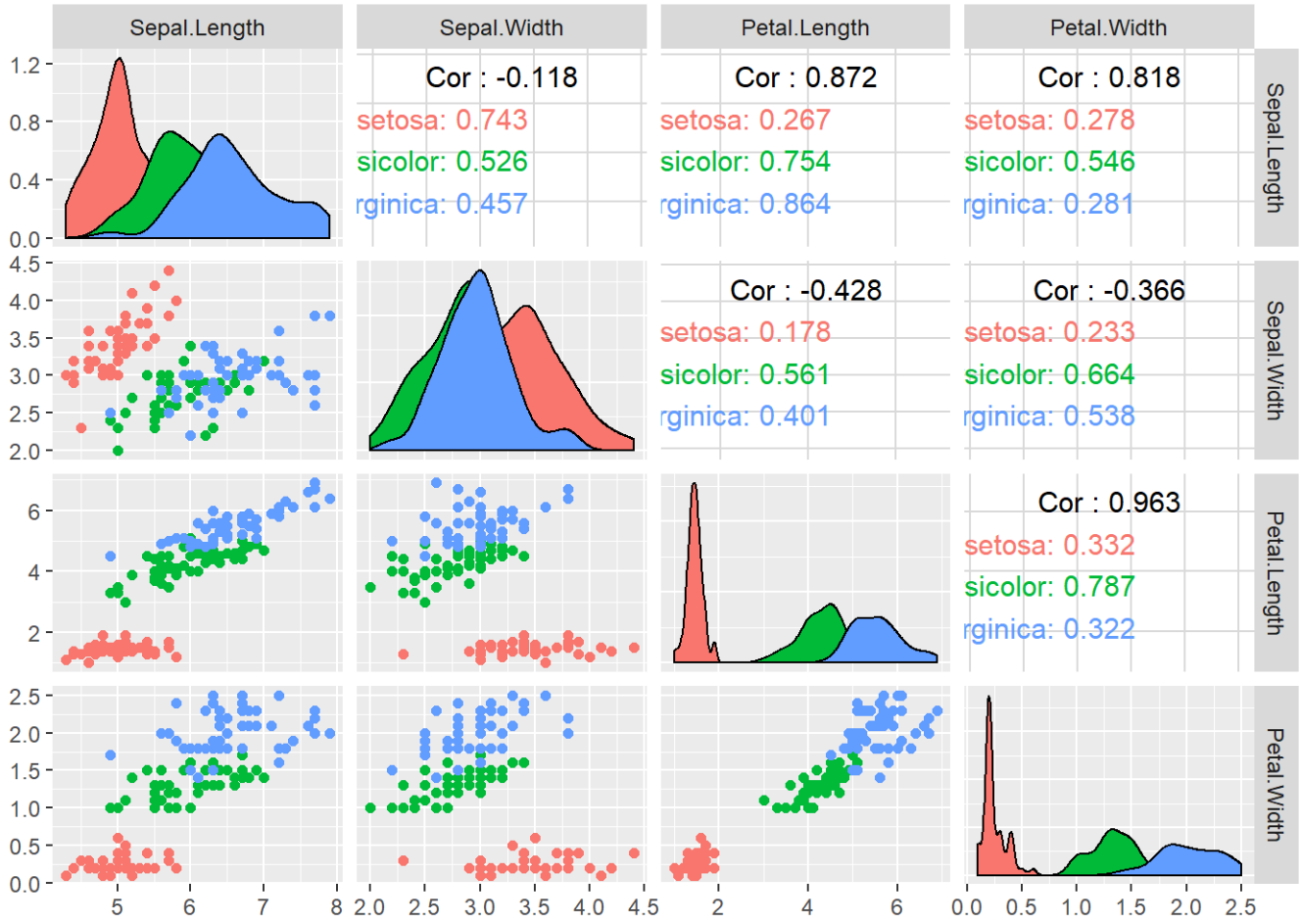
	data set	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
7	完全分离	euclidean	7.02	6.92	3.78	3.78	4.97	3.32	6.96	3.74	11.92
13	完全分离	mahalanobis	7.02	6.92	5.82	3.78	2.03	2.13	6.96	3.74	11.92
19	完全分离	manhattan	7.02	6.92	4.09	3.78	5.05	3.34	6.96	3.74	11.92
25	完全分离	maximum	7.02	6.92	3.75	3.78	5.11	3.31	6.96	3.74	11.92
11	完全分离有噪音点	canberra	5.89	6.56	2.00	4.11	7.41	5.67	7.43	4.96	5.85
71	完全分离有噪音点	euclidean	5.89	6.56	5.41	4.11	8.20	7.96	7.43	4.96	5.85
131	完全分离有噪音点	mahalanobis	5.89	6.56	5.01	4.11	2.27	2.18	7.43	4.96	5.85
191	完全分离有噪音点	manhattan	5.89	6.56	5.64	4.11	7.91	8.01	7.43	4.96	5.85
251	完全分离有噪音点	maximum	5.89	6.56	5.03	4.11	8.69	6.68	7.43	4.96	5.85
12	类簇的密度不同	canberra	7.00	6.78	2.00	3.80	4.90	3.06	6.82	3.79	10.52
72	类簇的密度不同	euclidean	7.00	6.78	2.21	3.80	4.83	3.42	6.82	3.79	10.52
132	类簇的密度不同	mahalanobis	7.00	6.78	5.72	3.80	2.04	2.05	6.82	3.79	10.52
192	类簇的密度不同	manhattan	7.00	6.78	2.38	3.80	4.84	3.40	6.82	3.79	10.52
252	类簇的密度不同	maximum	7.00	6.78	2.23	3.80	5.03	3.35	6.82	3.79	10.52
14	包含子类	canberra	4.31	4.89	2.37	2.37	2.37	2.37	6.00	2.37	11.98
73	包含子类	euclidean	4.31	4.89	2.36	2.37	2.37	2.36	6.00	2.37	11.98
133	包含子类	mahalanobis	4.31	4.89	4.84	2.37	2.00	2.00	6.00	2.37	11.98
193	包含子类	manhattan	4.31	4.89	2.35	2.37	2.37	2.32	6.00	2.37	11.98
253	包含子类	maximum	4.31	4.89	2.37	2.37	2.37	2.36	6.00	2.37	11.98
15	类簇的大小不同	canberra	6.06	5.60	2.08	3.60	4.10	3.07	5.88	3.70	9.63
74	类簇的大小不同	euclidean	6.06	5.60	2.76	3.60	4.03	3.31	5.88	3.70	9.63
134	类簇的大小不同	mahalanobis	6.06	5.60	6.91	3.60	2.01	2.21	5.88	3.70	9.63
194	类簇的大小不同	manhattan	6.06	5.60	2.80	3.60	4.03	3.27	5.88	3.70	9.63
254	类簇的大小不同	maximum	6.06	5.60	2.64	3.60	4.00	3.36	5.88	3.70	9.63
16	特征之间有相关	canberra	7.07	6.89	2.03	4.20	5.16	3.21	6.95	3.76	12.00
75	特征之间有相关	euclidean	7.07	6.89	4.14	4.20	5.11	3.33	6.95	3.76	12.00
135	特征之间有相关	mahalanobis	7.07	6.89	5.83	4.20	2.01	2.22	6.95	3.76	12.00
195	特征之间有相关	manhattan	7.07	6.89	4.21	4.20	5.15	3.30	6.95	3.76	12.00
255	特征之间有相关	maximum	7.07	6.89	4.03	4.20	5.24	3.28	6.95	3.76	12.00

真实数据

鸢尾花数据

类簇的分布以及变量之间的关系

```
library(MASS)
library(GGally)
ggpairs(iris, columns=1:4,mapping = aes(color=Species))
```



内部有效性测试结果

```
distance<-c("mahalanobis","euclidean", "maximum", "manhattan", "canberra")
iris_numeric<-apply(iris,2,as.numeric)

res_iris<-c()
  for(dist_m in distance){
    res_iris <- rbind(res_iris,computeIndeices(iris_numeric[,1:4],as.numeric(iris$Species),dist_m))
  }
res_iris<-cbind(distance=rep(distance,each=5),res_iris)
knitr::kable(res_iris, digits = 2, caption = "鸢尾花数据集上的结果") %>% kable_styling(font_size = 12)
```

鸢尾花数据集上的结果

	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
best_k	mahalanobis	3.00	3.00	2.00	2.00	2.00	2.00	3.00	2.00	12.00
value_index	mahalanobis	12.49	561.63	0.08	0.47	0.89	0.01	37.67	1.24	0.02
jaccard	mahalanobis	0.70	0.70	0.57	0.57	0.57	0.57	0.70	0.57	0.28
purity	mahalanobis	0.89	0.89	0.67	0.67	0.67	0.67	0.89	0.67	0.97
F1_score	mahalanobis	0.89	0.89	0.83	0.83	0.83	0.83	0.89	0.83	0.39
best_k1	euclidean	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
value_index1	euclidean	12.49	561.63	0.27	0.47	0.68	0.10	37.67	1.24	0.02
jaccard1	euclidean	0.70	0.70	0.57	0.57	0.57	0.70	0.70	0.57	0.28
purity1	euclidean	0.89	0.89	0.67	0.67	0.67	0.89	0.89	0.67	0.97
F1_score1	euclidean	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39

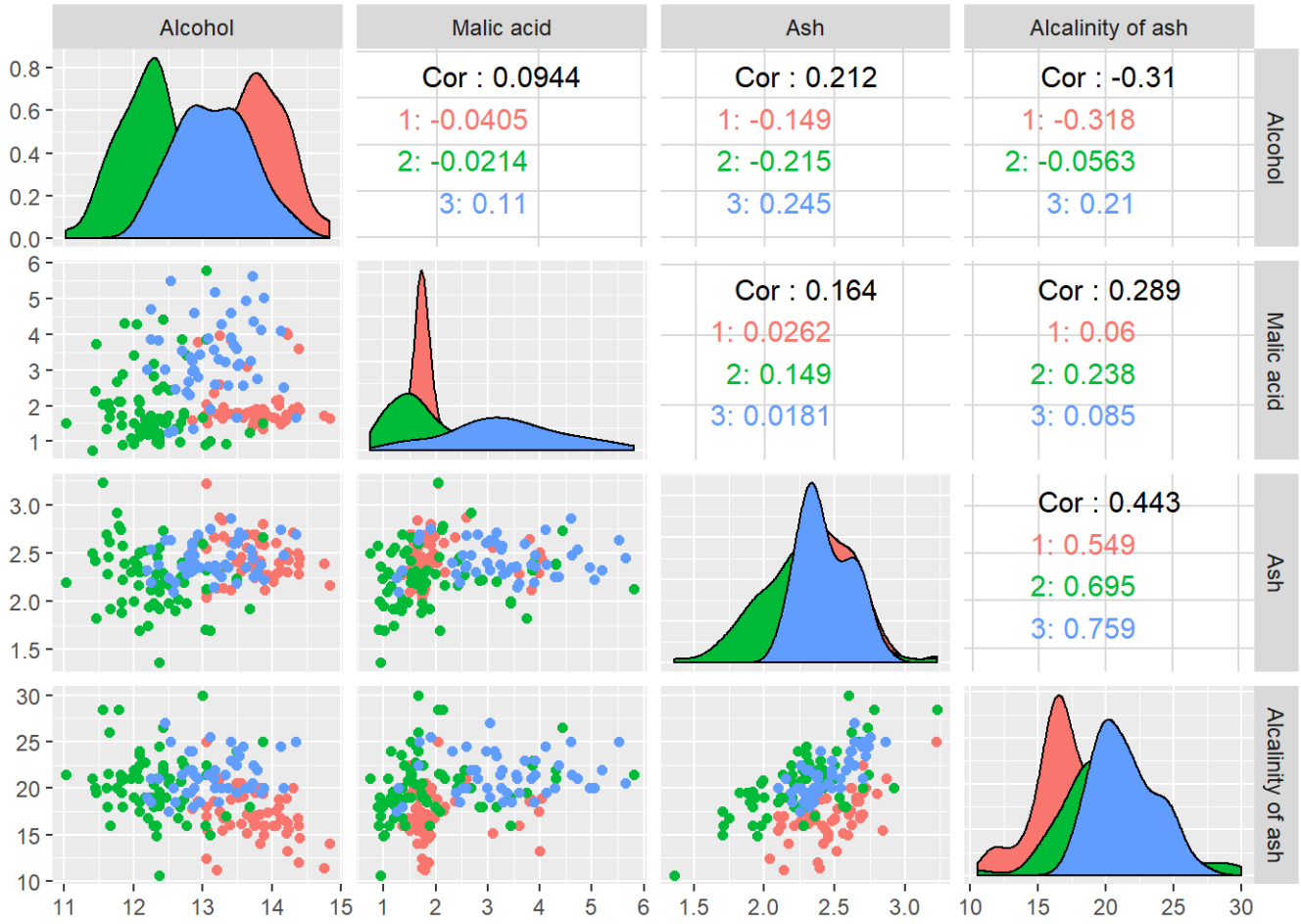
	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
best_k2	maximum	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
value_index2	maximum	12.49	561.63	0.24	0.47	0.71	0.10	37.67	1.24	0.02
jaccard2	maximum	0.70	0.70	0.57	0.57	0.57	0.70	0.70	0.57	0.28
purity2	maximum	0.89	0.89	0.67	0.67	0.67	0.89	0.89	0.67	0.97
F1_score2	maximum	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39
best_k3	manhattan	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
value_index3	manhattan	12.49	561.63	0.28	0.47	0.67	0.08	37.67	1.24	0.02
jaccard3	manhattan	0.70	0.70	0.57	0.57	0.57	0.70	0.70	0.57	0.28
purity3	manhattan	0.89	0.89	0.67	0.67	0.67	0.89	0.89	0.67	0.97
F1_score3	manhattan	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39
best_k4	canberra	3.00	3.00	2.00	2.00	2.00	2.00	3.00	2.00	12.00
value_index4	canberra	12.49	561.63	0.22	0.47	0.72	0.06	37.67	1.24	0.02
jaccard4	canberra	0.70	0.70	0.57	0.57	0.57	0.57	0.70	0.57	0.28
purity4	canberra	0.89	0.89	0.67	0.67	0.67	0.67	0.89	0.67	0.97
F1_score4	canberra	0.89	0.89	0.83	0.83	0.83	0.83	0.89	0.83	0.39

红酒数据

类簇的分布以及变量之间的关系

这里只展示了4个变量。

```
wine<-read.csv("红酒数据//wine.data",header = FALSE)
colnames(wine)<-c('class','Alcohol','Malic acid','Ash','Alcalinity of ash' ,'Magnesium','Total pheno
ggpairs(wine, columns=2:5,mapping = aes(color=as.factor(wine$class)))
```



内部有效性测试结果

```
wine<-apply(wine,2,as.numeric)
res_wine<-c()
  for(dist_m in distance){
    res_wine <- rbind(res_wine,computeIndeices(wine[,2:ncol(wine)],wine[,1],dist_m))
  }
res_wine<-cbind(distance=rep(distance,each=5),res_wine)
knitr::kable(res_wine, digits = 2, caption = "红酒数据集上的结果") %>% kable_styling(font_size = 12)
```

红酒数据集上的结果

	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
best_k	mahalanobis	2.00	12.00	10.00	7.00	2.00	4.00	2.00	4.00	12.00
value_index	mahalanobis	5.58	1519.63	0.10	0.56	0.72	0.00	24.85	0.02	0.06
jaccard	mahalanobis	0.47	0.15	0.18	0.24	0.47	0.35	0.47	0.35	0.15
purity	mahalanobis	0.66	0.73	0.72	0.72	0.66	0.72	0.66	0.72	0.73
F1_score	mahalanobis	0.71	0.30	0.33	0.40	0.71	0.47	0.71	0.47	0.30
best_k1	euclidean	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
value_index1	euclidean	5.58	1519.63	0.23	0.56	0.66	0.05	24.85	0.02	0.06
jaccard1	euclidean	0.47	0.15	0.47	0.24	0.47	0.24	0.47	0.35	0.15
purity1	euclidean	0.66	0.73	0.66	0.72	0.66	0.72	0.66	0.72	0.73
F1_score1	euclidean	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k2	maximum	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
value_index2	maximum	5.58	1519.63	0.22	0.56	0.66	0.04	24.85	0.02	0.06
jaccard2	maximum	0.47	0.15	0.47	0.24	0.47	0.24	0.47	0.35	0.15

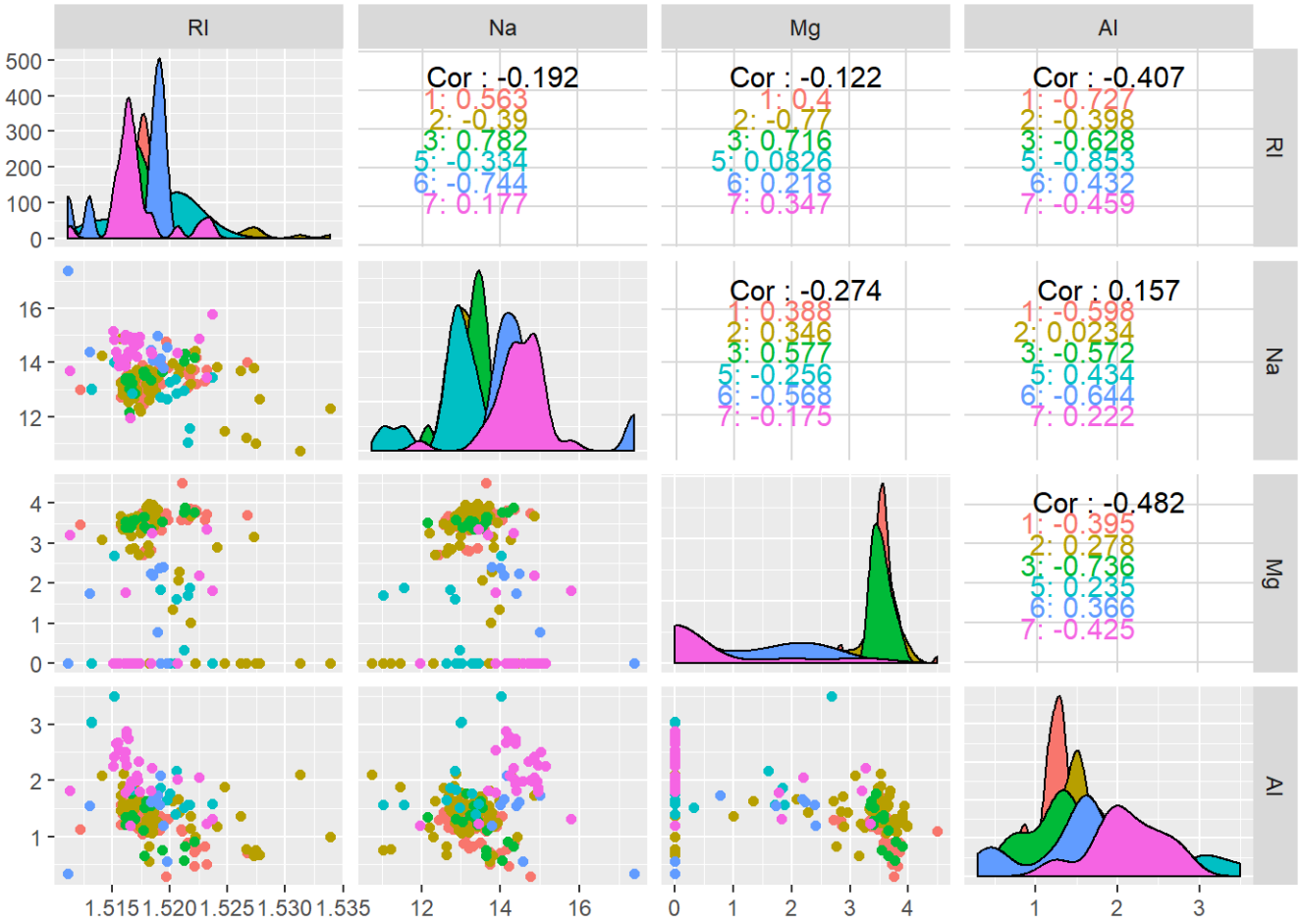
	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
purity2	maximum	0.66	0.73	0.66	0.72	0.66	0.72	0.66	0.72	0.73
F1_score2	maximum	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k3	manhattan	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
value_index3	manhattan	5.58	1519.63	0.25	0.56	0.63	0.07	24.85	0.02	0.06
jaccard3	manhattan	0.47	0.15	0.47	0.24	0.47	0.24	0.47	0.35	0.15
purity3	manhattan	0.66	0.73	0.66	0.72	0.66	0.72	0.66	0.72	0.73
F1_score3	manhattan	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k4	canberra	2.00	12.00	2.00	7.00	2.00	8.00	2.00	4.00	12.00
value_index4	canberra	5.58	1519.63	0.63	0.56	0.20	0.11	24.85	0.02	0.06
jaccard4	canberra	0.47	0.15	0.47	0.24	0.47	0.21	0.47	0.35	0.15
purity4	canberra	0.66	0.73	0.66	0.72	0.66	0.72	0.66	0.72	0.73
F1_score4	canberra	0.71	0.30	0.71	0.40	0.71	0.36	0.71	0.47	0.30

玻璃数据

类簇的分布以及变量之间的关系

这里只展示了4个变量。

```
glass<-read.csv("玻璃数据//glass.data",header = FALSE)
colnames(glass)<-c('Id','RI','Na','Mg', 'Al','Si','K','Ca', 'Ba','Fe','Type')
ggpairs(glass, columns=2:5,mapping = aes(color=as.factor(glass$Type)))
```



```
glass<-apply(glass,2,as.numeric)

res_glass<-c()
  for(dist_m in distance){
    res_glass<- rbind(res_glass,computeIndeices(glass[,2:10],glass[,11],dist_m))
  }
res_glass<-cbind(distance=rep(distance,each=5),res_glass)
knitr::kable(res_glass, digits = 2, caption = "玻璃数据集上的结果") %>% kable_styling(font_size = 12)
```

玻璃数据集上的结果

	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
best_k	mahalanobis	12.00	3.00	3.00	9.00	3.00	3.00	2.00	4.00	10.00
value_index	mahalanobis	17.59	135.00	0.09	0.96	0.74	0.02	85.62	3.52	0.27
jaccard	mahalanobis	0.20	0.34	0.34	0.25	0.34	0.34	0.32	0.32	0.27
purity	mahalanobis	0.65	0.50	0.50	0.58	0.50	0.50	0.43	0.55	0.67
F1_score	mahalanobis	0.42	0.43	0.43	0.48	0.43	0.43	0.33	0.40	0.47
best_k1	euclidean	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
value_index1	euclidean	17.59	135.00	0.21	0.96	0.58	0.12	85.62	3.52	0.27
jaccard1	euclidean	0.20	0.34	0.32	0.25	0.34	0.34	0.32	0.32	0.27
purity1	euclidean	0.65	0.50	0.43	0.58	0.50	0.50	0.43	0.55	0.67
F1_score1	euclidean	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k2	maximum	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
value_index2	maximum	17.59	135.00	0.19	0.96	0.61	0.12	85.62	3.52	0.27
jaccard2	maximum	0.20	0.34	0.32	0.25	0.34	0.34	0.32	0.32	0.27
purity2	maximum	0.65	0.50	0.43	0.58	0.50	0.50	0.43	0.55	0.67
F1_score2	maximum	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k3	manhattan	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
value_index3	manhattan	17.59	135.00	0.22	0.96	0.57	0.12	85.62	3.52	0.27
jaccard3	manhattan	0.20	0.34	0.32	0.25	0.34	0.34	0.32	0.32	0.27
purity3	manhattan	0.65	0.50	0.43	0.58	0.50	0.50	0.43	0.55	0.67
F1_score3	manhattan	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k4	canberra	12.00	3.00	2.00	9.00	2.00	2.00	2.00	4.00	10.00
value_index4	canberra	17.59	135.00	0.26	0.96	0.46	0.09	85.62	3.52	0.27
jaccard4	canberra	0.20	0.34	0.32	0.25	0.32	0.32	0.32	0.32	0.27
purity4	canberra	0.65	0.50	0.43	0.58	0.43	0.43	0.43	0.55	0.67
F1_score4	canberra	0.42	0.43	0.33	0.48	0.33	0.33	0.33	0.40	0.47

真实数据结果的汇总

真实数据的特征之间可能具有相关性，我们的研究中没考虑这一点。——不足

```
dataname<-c("iris","wine","glass")
res_comb7<-c()
for(dataset in dataname){
  res <- get(paste("res_",dataset,sep = ""))
```

```
  res <- cbind(dataset = rep(dataset,10),
    res[str_detect(row.names(res),'best_k|F1_score'),])
  res_comb7<-rbind(res,res_comb7)
}
knitr::kable(res_comb7, digits = 2, caption = "真实数据集上的结果") %>% kable_styling(font_size = 12)
```

真实数据集上的结果

	dataset	distance	kl	ch	mcclain	db	silhouette	dunn	ccc	sdindex	sdbw
best_k	glass	mahalanobis	12.00	3.00	3.00	9.00	3.00	3.00	2.00	4.00	10.00
F1_score	glass	mahalanobis	0.42	0.43	0.43	0.48	0.43	0.43	0.33	0.40	0.47
best_k1	glass	euclidean	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
F1_score1	glass	euclidean	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k2	glass	maximum	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
F1_score2	glass	maximum	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k3	glass	manhattan	12.00	3.00	2.00	9.00	3.00	3.00	2.00	4.00	10.00
F1_score3	glass	manhattan	0.42	0.43	0.33	0.48	0.43	0.43	0.33	0.40	0.47
best_k4	glass	canberra	12.00	3.00	2.00	9.00	2.00	2.00	2.00	4.00	10.00
F1_score4	glass	canberra	0.42	0.43	0.33	0.48	0.33	0.33	0.33	0.40	0.47
best_k6	wine	mahalanobis	2.00	12.00	10.00	7.00	2.00	4.00	2.00	4.00	12.00
F1_score6	wine	mahalanobis	0.71	0.30	0.33	0.40	0.71	0.47	0.71	0.47	0.30
best_k12	wine	euclidean	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
F1_score12	wine	euclidean	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k22	wine	maximum	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
F1_score22	wine	maximum	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k32	wine	manhattan	2.00	12.00	2.00	7.00	2.00	7.00	2.00	4.00	12.00
F1_score32	wine	manhattan	0.71	0.30	0.71	0.40	0.71	0.40	0.71	0.47	0.30
best_k42	wine	canberra	2.00	12.00	2.00	7.00	2.00	8.00	2.00	4.00	12.00
F1_score42	wine	canberra	0.71	0.30	0.71	0.40	0.71	0.36	0.71	0.47	0.30
best_k5	iris	mahalanobis	3.00	3.00	2.00	2.00	2.00	2.00	3.00	2.00	12.00
F1_score5	iris	mahalanobis	0.89	0.89	0.83	0.83	0.83	0.83	0.89	0.83	0.39
best_k11	iris	euclidean	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
F1_score11	iris	euclidean	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39
best_k21	iris	maximum	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
F1_score21	iris	maximum	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39
best_k31	iris	manhattan	3.00	3.00	2.00	2.00	2.00	3.00	3.00	2.00	12.00
F1_score31	iris	manhattan	0.89	0.89	0.83	0.83	0.83	0.89	0.89	0.83	0.39
best_k41	iris	canberra	3.00	3.00	2.00	2.00	2.00	2.00	3.00	2.00	12.00
F1_score41	iris	canberra	0.89	0.89	0.83	0.83	0.83	0.83	0.89	0.83	0.39