Сравнение подходов к ранжированию

Тушавин В. А.

4 января 2016 г.

Целью настоящего исследования является сравнение алгоритмов ранжирования объектов с точки зрения их эффективности. Сравниваются три алгоритма. Два из них были ранее описаны в статье Тушавин В. А. Ранжирование показателей качества с использованием методов Кемени-Янга и Шульце //Экономика и менеджмент систем управления. 2005. № 4.4. Третий - быстрый алгоритм нахождения медианы Кемени из пакета ConsRank.

### Определение библиотек и функций

library(gtools)  
library(ConsRank)

## Warning: package 'ConsRank' was built under R version 3.2.3

## Loading required package: MASS  
## Loading required package: proxy

## Warning: package 'proxy' was built under R version 3.2.3

##   
## Attaching package: 'proxy'  
##   
## The following objects are masked from 'package:stats':  
##   
## as.dist, dist  
##   
## The following object is masked from 'package:base':  
##   
## as.matrix  
##   
## Loading required package: rgl  
##   
## Attaching package: 'ConsRank'  
##   
## The following object is masked from 'package:base':  
##   
## labels

library(lpSolve)  
library(irr)  
  
# Функции для нахождения медианы Кемени  
# Нахождение расстояния между оценками  
kendall\_tau<-function(rank.a,rank.b) {  
 tau<-0  
 n<-length(rank.a)  
 for(k in 1:ncol(z<-combn(n,2))) {  
 i=z[1,k]  
 j=z[2,k]  
 tau<-tau+(sign(rank.a[i]-rank.a[j]) == -sign(rank.b[i]-rank.b[j]))  
 }   
 return(tau)  
}  
  
  
# Построение графа  
build\_graph<-function(ranks) {  
 n\_voters<-nrow(ranks)  
 n\_candidates<-ncol(ranks)  
 edge\_weights<-matrix(0,nrow=n\_candidates,ncol=n\_candidates)  
 for(k in 1:ncol(z<-combn(n\_candidates,2))) {  
 i=z[1,k]  
 j=z[2,k]  
 preference<-ranks[, i] - ranks[, j]  
 h.ij <- sum(preference < 0)   
 h.ji <- sum(preference > 0)  
 if(h.ij > h.ji) edge\_weights[i, j] <- h.ij - h.ji else if(h.ij < h.ji) edge\_weights[j, i] <- h.ji - h.ij  
 }  
 return(edge\_weights)  
}  
  
# Нахождение медианы Кемени посредством решения задачи ЛП  
rank\_solve<-function(ranks) {  
 tic = proc.time()[3]  
 n\_voters<-nrow(ranks)  
 n\_candidates<-ncol(ranks)  
 # Строим граф  
 edge\_weights<-build\_graph(ranks)  
 # Задаем параметры.   
 # Коээфициенты при целевой функции  
 objective.in<- as.vector(t(edge\_weights))  
 # Коэффициенты для каждой пары  
 pairwise\_constraints <- matrix(0,  
 n\_candidates \* (n\_candidates - 1) / 2, n\_candidates ^ 2)  
 for(k in 1:nrow(z<-combinations(n\_candidates,2))) {  
 i=z[k,1]  
 j=z[k,2]   
 pairwise\_constraints[k,c((i-1)\*n\_candidates+j,(j-1)\*n\_candidates+i)]<-1  
 }  
 # Коэффициенты для каждой тройки  
 triangle\_constraints <-matrix(0,n\_candidates \*  
 (n\_candidates - 1) \*  
 (n\_candidates - 2), n\_candidates ^ 2)  
   
 for(m in 1:nrow(z<-permutations(n\_candidates,3))) {  
 i=z[m,1]  
 j=z[m,2]  
 k=z[m,3]  
 triangle\_constraints[m,c((i-1)\*n\_candidates+j,(j-1)\*n\_candidates+k,(k-1)\*n\_candidates+i)]<-1  
 }  
 constraints<-rbind(pairwise\_constraints,triangle\_constraints)  
 constraint\_rhs<-rep(1,nrow(pairwise\_constraints)+nrow(triangle\_constraints))  
 constraint\_signs<-c(rep("==",nrow(pairwise\_constraints)),rep(">=",nrow(triangle\_constraints)))  
 z<-lp("min",objective.in, constraints, constraint\_signs, constraint\_rhs,all.int=T)   
 x<-matrix(z$solution,nrow=n\_candidates,ncol=n\_candidates,byrow=T)  
 best\_rank<-apply(x,1,sum)  
 tau<-sum(apply(ranks,1,function(x){kendall\_tau(x,best\_rank)}))  
 toc = proc.time()[3]  
 eltime = toc - tic  
 Consensus=best\_rank+1  
 names(Consensus)<-LETTERS[1:n\_candidates]  
 return(list(min\_dist=tau,best\_rank=best\_rank,Consensus=Consensus,Eltime=eltime))  
}

Поскольку предыдущая версия функции нахождения ранжирования методом Шульце требовала полное ранжирование заданное в виде последовательности номеров элементов, то необходимо её немного преобразовать для работы с таблицей рангов

# Модифицированния функция нахождения итогового  
# ранжирования методом Шульце  
Schulze.m<-function(ranks,Wk=NULL) {  
 tic = proc.time()[3]  
 if (class(ranks) == "data.frame") {  
 ranks = as.matrix(ranks)  
 }   
 n\_voters<-nrow(ranks)  
 n\_candidates<-ncol(ranks)  
 ranks[is.na(ranks)]<-Inf  
 if (n\_voters == 1) {  
 consensus = ranks  
 } else {  
 mtx<-matrix(data=0,nrow=n\_candidates,ncol=n\_candidates)  
 rownames(mtx)<-colnames(ranks)  
 colnames(mtx)<-colnames(ranks)  
 index<-combinations(n\_candidates,2)  
 for(i in 1:n\_voters){  
 temp<-matrix(data=0,nrow=n\_candidates,ncol=n\_candidates)  
 for(idx in 1:nrow(index)) {  
 x1=index[idx,1]  
 x2=index[idx,2]  
 if(ranks[i,x1]<ranks[i,x2]) temp[x1,x2]<-temp[x1,x2]+1  
 if(ranks[i,x1]>ranks[i,x2]) temp[x2,x1]<-temp[x2,x1]+1  
 }  
 if(!is.null(Wk)) temp<-temp\*Wk[i]  
 mtx<-mtx+temp  
 }  
 result<-matrix(data=0,nrow=n\_candidates,ncol=n\_candidates)  
 for(i in 1:n\_candidates)  
 for(j in 1:n\_candidates)  
 if(i!=j) result[i,j]<-ifelse(mtx[i,j] > mtx[j,i],mtx[i,j],0)  
 for(i in 1:n\_candidates)  
 for(j in 1:n\_candidates)  
 if(i!=j) for(k in 1:n\_candidates)  
 if(i!=k & j !=k) result[j,k]<-max(result[j,k],  
 min(result[j,i],result[i,k]))  
 vec<-rep(0,n\_candidates)  
 for(k in 1:nrow(z<-combinations(n\_candidates,2))) {  
 i=z[k,1]  
 j=z[k,2]  
 if(result[i,j]>result[j,i])   
 vec[j]<-vec[j]+1  
 else if(result[i,j]<result[j,i])  
 vec[i]<-vec[i]+1  
 }  
 }   
 consensus<-matrix(vec,nrow=1,ncol=n\_candidates)  
 colnames(consensus)<-colnames(ranks)  
toc = proc.time()[3]  
eltime = toc - tic  
return(list(Consensus=consensus+1,Eltime=eltime))  
}

Проведем тестирование на примере из Википедии

test<-data.frame(order=c("ACBED","ADECB",  
 "BEDAC",  
 "CABED",  
 "CAEBD",  
 "CBADE",  
 "DCEBA",  
 "EBADC"),  
 wk=c(5,5,8,3,7,2,7,8))   
ranks<-matrix(0,nrow=nrow(test),ncol=5)  
colnames(ranks)<-LETTERS[1:5]  
for(i in 1:nrow(test))   
 ranks[i,t(asc(as.character(test$order[i]))-64)]<-1:5  
Wk<-test$wk  
Schulze.m(ranks,Wk)

## $Consensus  
## A B C D E  
## [1,] 2 4 3 5 1  
##   
## $Eltime  
## elapsed   
## 0

Результаты свопали полностью. Функция работает. Проверим результат с помощью пакета ConsRank

FASTcons(ranks,Wk)

## $Consensus  
## A B C D E  
## [1,] 3 2 5 4 1  
##   
## $Tau  
## [,1]  
## [1,] 0.1555556  
##   
## $Eltime  
## elapsed   
## 0.87

Имеется расхождение, поскольку пример является несбалансированным по рангам.

Сравним результаты тестового примера из пакета ConsRank

data(sports)  
Schulze.m(sports)

## $Consensus  
## Baseball Football Basketball Tennis Cycling Swimming Jogging  
## [1,] 4 6 3 5 1 2 7  
##   
## $Eltime  
## elapsed   
## 0.02

FASTcons(sports,maxiter=10)

## $Consensus  
## Baseball Football Basketball Tennis Cycling Swimming Jogging  
## [1,] 4 6 3 5 1 2 7  
##   
## $Tau  
## [,1]  
## [1,] 0.1443223  
##   
## $Eltime  
## elapsed   
## 0.31

Результаты удовлетворительны.

### Задание функций тестирования

Пусть имеется матрица рангов n x m, где n - эксперты, а m - ранжируемые показатели и имеется случайный вектор рангов длиной m. Пусть имеется полный конценсус и все эксперты поставили одинаковую оценку равную этому случайному вектору. Внесем искажения в матрицу, поменяв в каждой строке, кроме первой, два случайно выбранных рядом стоящих ранга местами. Попробуем отыскать исходное ранжирование с помощью метода Шульце, медианы Кемени методом ЛП и методом ветвей и границ.

change<-function(vec,cnt=1) {  
 idx<-1:(length(vec)-1)  
 for(i in 1:cnt) {  
 k<-sample(idx,1)  
 x1<-which(vec==k)  
 x2<-which(vec==(k+1))  
 x<-vec[x1]  
 vec[x1]<-vec[x2]  
 vec[x2]<-x  
 }   
 return(vec)  
}  
  
dotest<-function(val=5,experts=10,tests=1,correct=0) {  
 result<-list()  
 # Start  
 for(i in 1:tests) {  
 x0<-sample(1:val,val)  
 names(x0)<-LETTERS[1:val]  
 z<-matrix(rep(x0,experts),byrow=T,ncol=val)  
 colnames(z)<-LETTERS[1:val]  
 z1<-t(apply(z,1,change,cnt=1))  
 if(correct>0) for(j in 1:correct) z1[j,]<-x0  
 result[[i]]<-list(kendall=kendall(t(z1)),  
 TrueValue=x0,  
 Shulze=Schulze.m(z1),  
 FASTcons=QuickCons(z1),  
 LP=rank\_solve(z1-1))  
 }  
return(result)  
}

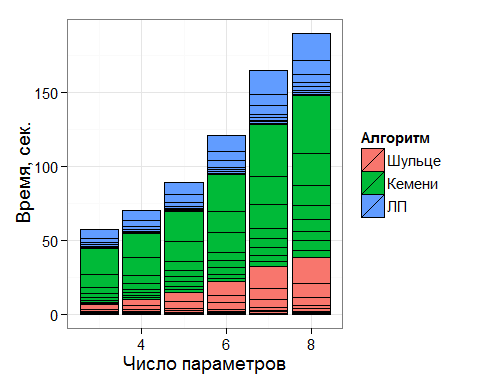
### Тестирование

if(!file.exists("doe.RDs")) {   
 set.seed(2015)  
 doe<-expand.grid(testn=1:100,val=3:8,experts=c(4,8,16,32,64,128,256,512))  
 doe$W<-NA  
 doe$Shulze.time<-NA  
 doe$FASTcons.time<-NA  
 doe$LP.time<-NA  
 doe$TS<-""  
 doe$Sch<-""  
 doe$Ken<-""  
 doe$LP<-""  
 doe$TrueS<-FALSE  
 doe$TrueF<-FALSE  
 doe$TrueLP<-FALSE  
 doe$SF<-FALSE  
 doe$SLP<-FALSE  
 doe$FLP<-FALSE  
   
 pb <- txtProgressBar(min = 0, max = nrow(doe), style = 3,file = stderr())  
 for(i in 1:nrow(doe)) {   
 r1<-dotest(val=doe$val[i],experts=doe$experts[i],tests=1,correct=1)  
 doe$W[i]<-r1[[1]]$kendall$value  
 doe$Shulze.time[i]<-r1[[1]]$Shulze$Eltime  
 doe$FASTcons.time[i]<-r1[[1]]$FASTcons$Eltime  
 doe$LP.time[i]<-r1[[1]]$LP$Eltime  
   
 doe$TS[i]<-paste0(names(r1[[1]]$TrueValue)[order(r1[[1]]$TrueValue)],collapse="")  
 doe$Sch[i]<-paste0(names(r1[[1]]$Shulze$Consensus[1,])[order(r1[[1]]$Shulze$Consensus[1,])],collapse="")  
 doe$Ken[i]<-paste0(names(r1[[1]]$FASTcons$Consensus[1,])[order(r1[[1]]$FASTcons$Consensus[1,])],collapse="")  
 doe$LP[i]<-paste0(names(r1[[1]]$LP$Consensus)[order(r1[[1]]$LP$Consensus)],collapse="")  
   
 doe$TrueS[i]<-(doe$TS[i] == doe$Sch[i])  
 doe$TrueF[i]<-(doe$TS[i] == doe$Ken[i])  
 doe$TrueLP[i]<-(doe$TS[i] == doe$LP[i])  
 doe$SF[i]<-(doe$Sch[i] == doe$Ken[i])  
 doe$SLP[i]<-(doe$Sch[i] == doe$LP[i])  
 doe$FLP[i]<-(doe$Ken[i] == doe$LP[i])  
 setTxtProgressBar(pb, i)  
 }  
 close(pb)  
 ### Сохранение данных  
 saveRDS(doe,"doe.RDs")  
 library(xlsx)  
 write.xlsx(doe, "mydata.xlsx", sheetName="Results",row.names=FALSE)  
} else doe<-readRDS("doe.RDs")  
   
knitr::kable(agg.doe<-aggregate(cbind(TrueS,TrueF,TrueLP,SF,SLP,FLP,Shulze.time,FASTcons.time,LP.time)~val+experts,data=doe,sum))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| val | experts | TrueS | TrueF | TrueLP | SF | SLP | FLP | Shulze.time | FASTcons.time | LP.time |
| 3 | 4 | 37 | 36 | 0 | 67 | 63 | 64 | 0.082 | 0.819 | 0.190 |
| 4 | 4 | 52 | 60 | 37 | 80 | 53 | 45 | 0.113 | 1.113 | 0.223 |
| 5 | 4 | 67 | 64 | 52 | 87 | 57 | 64 | 0.161 | 1.614 | 0.292 |
| 6 | 4 | 72 | 73 | 63 | 93 | 67 | 68 | 0.229 | 2.322 | 0.432 |
| 7 | 4 | 76 | 77 | 68 | 95 | 62 | 65 | 0.309 | 3.347 | 0.696 |
| 8 | 4 | 89 | 86 | 79 | 95 | 72 | 75 | 0.443 | 4.893 | 1.092 |
| 3 | 8 | 26 | 25 | 0 | 69 | 74 | 75 | 0.095 | 0.965 | 0.232 |
| 4 | 8 | 58 | 61 | 40 | 85 | 80 | 77 | 0.153 | 1.413 | 0.307 |
| 5 | 8 | 85 | 85 | 79 | 92 | 86 | 86 | 0.237 | 1.965 | 0.391 |
| 6 | 8 | 93 | 91 | 87 | 98 | 94 | 96 | 0.334 | 2.691 | 0.538 |
| 7 | 8 | 93 | 93 | 92 | 98 | 97 | 97 | 0.447 | 3.927 | 0.909 |
| 8 | 8 | 98 | 97 | 95 | 99 | 97 | 98 | 0.734 | 6.533 | 1.421 |
| 3 | 16 | 15 | 12 | 0 | 83 | 85 | 88 | 0.204 | 1.519 | 0.376 |
| 4 | 16 | 81 | 80 | 74 | 95 | 93 | 94 | 0.250 | 1.848 | 0.473 |
| 5 | 16 | 99 | 99 | 98 | 100 | 99 | 99 | 0.428 | 3.293 | 0.772 |
| 6 | 16 | 99 | 100 | 99 | 99 | 100 | 99 | 0.674 | 4.841 | 1.027 |
| 7 | 16 | 100 | 100 | 99 | 100 | 99 | 99 | 0.909 | 5.447 | 1.191 |
| 8 | 16 | 99 | 100 | 99 | 99 | 100 | 99 | 0.855 | 5.863 | 1.313 |
| 3 | 32 | 10 | 15 | 0 | 79 | 90 | 85 | 0.211 | 1.490 | 0.432 |
| 4 | 32 | 92 | 94 | 91 | 98 | 99 | 97 | 0.462 | 2.331 | 0.648 |
| 5 | 32 | 100 | 100 | 100 | 100 | 100 | 100 | 0.698 | 3.666 | 1.004 |
| 6 | 32 | 100 | 100 | 100 | 100 | 100 | 100 | 0.961 | 4.633 | 1.296 |
| 7 | 32 | 100 | 100 | 100 | 100 | 100 | 100 | 1.240 | 5.937 | 1.622 |
| 8 | 32 | 100 | 100 | 100 | 100 | 100 | 100 | 1.564 | 7.989 | 2.160 |
| 3 | 64 | 8 | 15 | 0 | 89 | 92 | 85 | 0.430 | 2.746 | 0.814 |
| 4 | 64 | 99 | 99 | 99 | 100 | 100 | 100 | 0.790 | 4.173 | 1.227 |
| 5 | 64 | 100 | 100 | 100 | 100 | 100 | 100 | 0.933 | 3.908 | 1.298 |
| 6 | 64 | 100 | 100 | 100 | 100 | 100 | 100 | 1.300 | 5.148 | 1.631 |
| 7 | 64 | 100 | 100 | 100 | 100 | 100 | 100 | 1.963 | 7.143 | 2.351 |
| 8 | 64 | 100 | 100 | 100 | 100 | 100 | 100 | 2.577 | 9.512 | 3.202 |
| 3 | 128 | 7 | 11 | 0 | 92 | 93 | 89 | 0.706 | 4.342 | 1.446 |
| 4 | 128 | 100 | 100 | 100 | 100 | 100 | 100 | 1.197 | 5.150 | 1.775 |
| 5 | 128 | 100 | 100 | 100 | 100 | 100 | 100 | 1.755 | 6.262 | 2.332 |
| 6 | 128 | 100 | 100 | 100 | 100 | 100 | 100 | 4.284 | 13.640 | 5.005 |
| 7 | 128 | 100 | 100 | 100 | 100 | 100 | 100 | 5.174 | 15.745 | 5.701 |
| 8 | 128 | 100 | 100 | 100 | 100 | 100 | 100 | 4.999 | 13.862 | 5.398 |
| 3 | 256 | 4 | 3 | 0 | 95 | 96 | 97 | 1.493 | 8.693 | 3.024 |
| 4 | 256 | 100 | 100 | 100 | 100 | 100 | 100 | 2.862 | 12.151 | 4.355 |
| 5 | 256 | 100 | 100 | 100 | 100 | 100 | 100 | 4.187 | 13.792 | 5.307 |
| 6 | 256 | 100 | 100 | 100 | 100 | 100 | 100 | 4.995 | 14.446 | 5.793 |
| 7 | 256 | 100 | 100 | 100 | 100 | 100 | 100 | 7.514 | 18.975 | 7.993 |
| 8 | 256 | 100 | 100 | 100 | 100 | 100 | 100 | 9.351 | 21.477 | 9.596 |
| 3 | 512 | 1 | 2 | 0 | 99 | 99 | 98 | 3.117 | 17.981 | 6.029 |
| 4 | 512 | 100 | 100 | 100 | 100 | 100 | 100 | 4.340 | 16.396 | 6.351 |
| 5 | 512 | 100 | 100 | 100 | 100 | 100 | 100 | 6.641 | 19.912 | 8.273 |
| 6 | 512 | 100 | 100 | 100 | 100 | 100 | 100 | 9.394 | 24.662 | 10.737 |
| 7 | 512 | 100 | 100 | 100 | 100 | 100 | 100 | 15.015 | 35.548 | 15.773 |
| 8 | 512 | 100 | 100 | 100 | 100 | 100 | 100 | 18.084 | 39.237 | 18.170 |

### Время выполнения скрипта в зависимости от алгоритма

library(ggplot2)  
library(reshape2)  
agg.doe<-agg.doe[,c(1,2,9:11)]  
names(agg.doe)[3:5]<-c("Шульце","Кемени","ЛП")  
md<-melt(agg.doe,id=c("val","experts"))  
ggplot(md,aes(x=val,y=value,fill=variable))+  
 geom\_bar(stat="identity",colour="black")+  
 scale\_fill\_discrete(name = "Алгоритм")+  
 labs(x="Число параметров",y="Время, сек.")+  
 theme\_bw()+theme(text=element\_text(size=14))



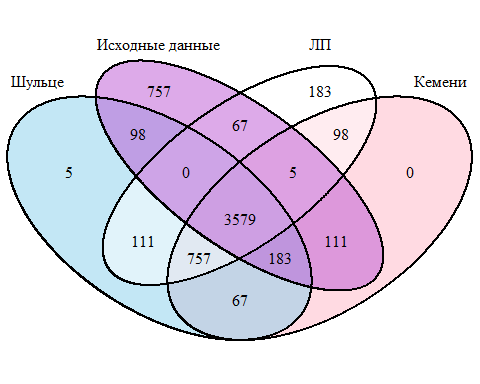
ggsave("Pic03\_00.png",width=6,height=4,dpi=300)

### Построение диаграммы Венна

library(VennDiagram)

## Loading required package: grid

plot.venn<-function(doe) {  
 grid.newpage()  
 draw.quad.venn(area1 = nrow(doe),   
 area2 = nrow(doe),  
 area3 = nrow(doe),  
 area4=nrow(doe),  
 n12 = sum(doe$SF),  
 n13 = sum(doe$TrueS),  
 n14 = sum(doe$SLP),  
 n23 = sum(doe$TrueF),  
 n24 = sum(doe$FLP),  
 n34 = sum(doe$TrueLP),  
 n123 = sum(doe$Sch == doe$Ken & doe$Ken==doe$TS),  
 n124 = sum(doe$Sch == doe$Ken & doe$Ken==doe$LP),  
 n134 = sum(doe$Sch == doe$TS & doe$TS==doe$LP),  
 n234 = sum(doe$Ken == doe$TS & doe$TS==doe$LP),  
 n1234 = sum(doe$Ken == doe$TS & doe$TS==doe$LP & doe$Ken==doe$Sch),   
 category = c("Шульце", "Кемени", "Исходные данные", "ЛП"),   
 fill = c("skyblue", "pink1", "mediumorchid","white"))  
}  
plot.venn(doe)



## (polygon[GRID.polygon.139], polygon[GRID.polygon.140], polygon[GRID.polygon.141], polygon[GRID.polygon.142], polygon[GRID.polygon.143], polygon[GRID.polygon.144], polygon[GRID.polygon.145], polygon[GRID.polygon.146], text[GRID.text.147], text[GRID.text.148], text[GRID.text.149], text[GRID.text.150], text[GRID.text.151], text[GRID.text.152], text[GRID.text.153], text[GRID.text.154], text[GRID.text.155], text[GRID.text.156], text[GRID.text.157], text[GRID.text.158], text[GRID.text.159], text[GRID.text.160], text[GRID.text.161], text[GRID.text.162], text[GRID.text.163], text[GRID.text.164], text[GRID.text.165])

# Анализ зависимости

library(rpart)  
library(rpart.plot)  
library(caret)

## Loading required package: lattice

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.2.3

## Loading required package: gplots  
##   
## Attaching package: 'gplots'  
##   
## The following object is masked \_by\_ '.GlobalEnv':  
##   
## plot.venn  
##   
## The following object is masked from 'package:stats':  
##   
## lowess

set.seed(2015)  
idx<-sample(1:nrow(doe),3800)  
test<-doe[idx,]  
verify<-doe[-idx,]  
  
(mylogit <- glm(TrueS ~ val + experts + W, data = test, family=binomial(link='logit')))

##   
## Call: glm(formula = TrueS ~ val + experts + W, family = binomial(link = "logit"),   
## data = test)  
##   
## Coefficients:  
## (Intercept) val experts W   
## -5.077245 0.707930 0.003041 4.254768   
##   
## Degrees of Freedom: 3799 Total (i.e. Null); 3796 Residual  
## Null Deviance: 3710   
## Residual Deviance: 2097 AIC: 2105

summary(mylogit)

##   
## Call:  
## glm(formula = TrueS ~ val + experts + W, family = binomial(link = "logit"),   
## data = test)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.10587 0.09522 0.19664 0.46476 1.94230   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.0772454 0.2269911 -22.368 < 2e-16 \*\*\*  
## val 0.7079303 0.0910490 7.775 7.53e-15 \*\*\*  
## experts 0.0030411 0.0004047 7.515 5.71e-14 \*\*\*  
## W 4.2547684 0.4587072 9.276 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3709.8 on 3799 degrees of freedom  
## Residual deviance: 2097.3 on 3796 degrees of freedom  
## AIC: 2105.3  
##   
## Number of Fisher Scoring iterations: 6

confint(mylogit)

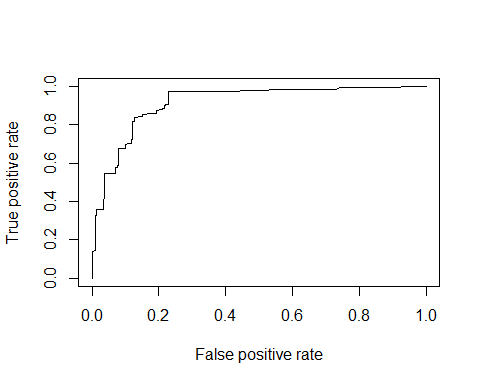
## Waiting for profiling to be done...

## 2.5 % 97.5 %  
## (Intercept) -5.532272880 -4.641835510  
## val 0.533344617 0.890616275  
## experts 0.002261516 0.003848818  
## W 3.360742352 5.160190314

fitted.results<-predict(mylogit,newdata=verify,type='response')  
fitted.results <- (fitted.results > 0.5)  
confusionMatrix(fitted.results,verify$TrueS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 158 21  
## TRUE 55 766  
##   
## Accuracy : 0.924   
## 95% CI : (0.9058, 0.9397)  
## No Information Rate : 0.787   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7593   
## Mcnemar's Test P-Value : 0.0001535   
##   
## Sensitivity : 0.7418   
## Specificity : 0.9733   
## Pos Pred Value : 0.8827   
## Neg Pred Value : 0.9330   
## Prevalence : 0.2130   
## Detection Rate : 0.1580   
## Detection Prevalence : 0.1790   
## Balanced Accuracy : 0.8576   
##   
## 'Positive' Class : FALSE   
##

fitted.results<-predict(mylogit,newdata=verify,type='response')  
pr <- prediction(fitted.results, verify$TrueS)  
prf <- performance(pr, measure = "tpr", x.measure = "fpr")  
plot(prf)



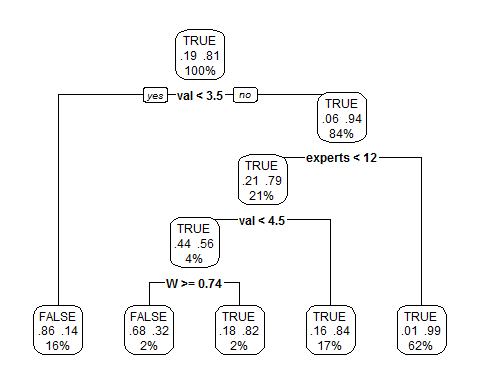
auc <- performance(pr, measure = "auc")  
auc <- auc@y.values[[1]]  
auc

## [1] 0.9147771

(mytree <- rpart(as.factor(TrueS) ~ val + experts + W, data = test))

## n= 3800   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 3800 727 TRUE (0.19131579 0.80868421)   
## 2) val< 3.5 615 87 FALSE (0.85853659 0.14146341) \*  
## 3) val>=3.5 3185 199 TRUE (0.06248038 0.93751962)   
## 6) experts< 12 813 173 TRUE (0.21279213 0.78720787)   
## 12) val< 4.5 162 71 TRUE (0.43827160 0.56172840)   
## 24) W>=0.7375 84 27 FALSE (0.67857143 0.32142857) \*  
## 25) W< 0.7375 78 14 TRUE (0.17948718 0.82051282) \*  
## 13) val>=4.5 651 102 TRUE (0.15668203 0.84331797) \*  
## 7) experts>=12 2372 26 TRUE (0.01096121 0.98903879) \*

prp(mytree,type = 2,extra = 104,fallen.leaves=TRUE)



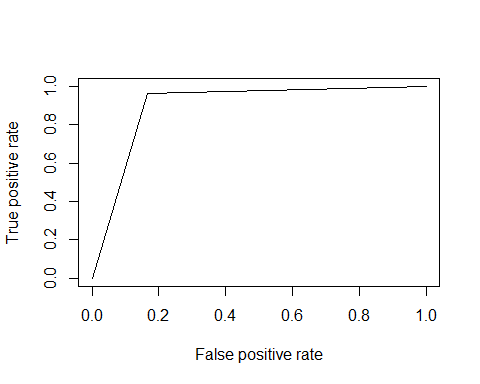
fitted.results<-predict(mytree,newdata=verify,type = "class")  
misClasificError <- mean(fitted.results != verify$TrueS)  
print(paste('Accuracy',1-misClasificError))

## [1] "Accuracy 0.936"

confusionMatrix(fitted.results,verify$TrueS)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 178 29  
## TRUE 35 758  
##   
## Accuracy : 0.936   
## 95% CI : (0.919, 0.9504)  
## No Information Rate : 0.787   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8071   
## Mcnemar's Test P-Value : 0.532   
##   
## Sensitivity : 0.8357   
## Specificity : 0.9632   
## Pos Pred Value : 0.8599   
## Neg Pred Value : 0.9559   
## Prevalence : 0.2130   
## Detection Rate : 0.1780   
## Detection Prevalence : 0.2070   
## Balanced Accuracy : 0.8994   
##   
## 'Positive' Class : FALSE   
##

pr <- prediction(predict(mytree,newdata=verify,type = "vector"), verify$TrueS)  
prf <- performance(pr, measure = "tpr", x.measure = "fpr")  
plot(prf)

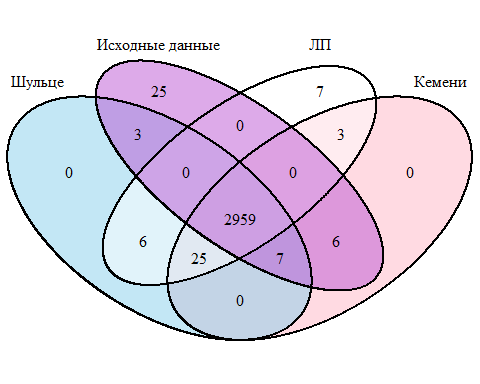


auc <- performance(pr, measure = "auc")  
auc <- auc@y.values[[1]]  
auc

## [1] 0.899416

Построим такую же диаграмму для числа оцениваемых параметров больше 3.5 и число эеспертов больше 11

plot.venn(subset(doe,val>3.5 & experts>11))



## (polygon[GRID.polygon.176], polygon[GRID.polygon.177], polygon[GRID.polygon.178], polygon[GRID.polygon.179], polygon[GRID.polygon.180], polygon[GRID.polygon.181], polygon[GRID.polygon.182], polygon[GRID.polygon.183], text[GRID.text.184], text[GRID.text.185], text[GRID.text.186], text[GRID.text.187], text[GRID.text.188], text[GRID.text.189], text[GRID.text.190], text[GRID.text.191], text[GRID.text.192], text[GRID.text.193], text[GRID.text.194], text[GRID.text.195], text[GRID.text.196], text[GRID.text.197], text[GRID.text.198], text[GRID.text.199], text[GRID.text.200], text[GRID.text.201], text[GRID.text.202])

### Информация о параметрах R

sessionInfo()

## R version 3.2.2 (2015-08-14)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 8 x64 (build 9200)  
##   
## locale:  
## [1] LC\_COLLATE=Russian\_Russia.1251 LC\_CTYPE=Russian\_Russia.1251   
## [3] LC\_MONETARY=Russian\_Russia.1251 LC\_NUMERIC=C   
## [5] LC\_TIME=Russian\_Russia.1251   
##   
## attached base packages:  
## [1] grid stats graphics grDevices utils datasets methods   
## [8] base   
##   
## other attached packages:  
## [1] ROCR\_1.0-7 gplots\_2.17.0 caret\_6.0-62   
## [4] lattice\_0.20-33 rpart.plot\_1.5.3 rpart\_4.1-10   
## [7] VennDiagram\_1.6.9 reshape2\_1.4.1 ggplot2\_1.0.1   
## [10] irr\_0.84 lpSolve\_5.6.13 ConsRank\_1.0.2   
## [13] rgl\_0.95.1367 proxy\_0.4-15 MASS\_7.3-45   
## [16] gtools\_3.5.0   
##   
## loaded via a namespace (and not attached):  
## [1] Rcpp\_0.12.2 nloptr\_1.0.4 formatR\_1.2.1   
## [4] plyr\_1.8.3 highr\_0.5.1 class\_7.3-14   
## [7] bitops\_1.0-6 iterators\_1.0.8 tools\_3.2.2   
## [10] digest\_0.6.8 lme4\_1.1-10 evaluate\_0.8   
## [13] gtable\_0.1.2 nlme\_3.1-122 mgcv\_1.8-9   
## [16] Matrix\_1.2-3 foreach\_1.4.3 parallel\_3.2.2   
## [19] yaml\_2.1.13 SparseM\_1.7 proto\_0.3-10   
## [22] e1071\_1.6-7 stringr\_1.0.0 knitr\_1.11   
## [25] caTools\_1.17.1 MatrixModels\_0.4-1 stats4\_3.2.2   
## [28] nnet\_7.3-11 rmarkdown\_0.8.1 gdata\_2.17.0   
## [31] minqa\_1.2.4 car\_2.1-0 magrittr\_1.5   
## [34] scales\_0.3.0 codetools\_0.2-14 htmltools\_0.2.6   
## [37] splines\_3.2.2 pbkrtest\_0.4-2 colorspace\_1.2-6   
## [40] quantreg\_5.19 labeling\_0.3 KernSmooth\_2.23-15  
## [43] stringi\_1.0-1 munsell\_0.4.2