1.Linear Programming Problem(Graphical method)

```
cons.1 = function(x) (36 - 6*x)/6
cons.2 = function(x) (10 - x)/2
cons.3 = function(x) 4
cons.4 = function(x) 0
cons.5 = function(x) 0
library(ggplot2)
p = ggplot(data=data.frame(x=0), aes(x=x)) +
  stat_function(colour="red", fun=cons.1) +
  stat_function(colour="blue", fun=cons.2) +
  geom_vline(xintercept=4, colour="Green") +
  scale_x_continuous(breaks=seq(0, 10, 1), lim=c(0, 10)) +
  scale_y_continuous(breaks=seq(0, 10, 1), lim=c(0, 10)) +
  labs(title="Optimization problem", subtitle="Graphical method", x="x1", y="x2")+
  theme_bw()
print(p)
f_region = data.frame(x=c(0,4,4,2,0),y=c(0,0,2,4,5))
p = p + geom_polygon(data = f_region,mapping = aes(x = x,y = y),fill =
"#dddddd")+geom_point(data = f_region,aes(x = x,y = y),color = "black")
print(p)
library(lpSolve)
f.obj = c(4,5)
f.con = matrix(c(1,2,
                 6,6,
                 1,0),nrow = 3,byrow = TRUE)
f.dir = c("<=","<=","<=")
f.rhs = c(10,36,4)
lpresult = lp("max",f.obj,f.con,f.dir,f.rhs)
solution = lpresult$solution
p = p + geom_point(aes(x = solution[1],y = solution[2]),color = "yellow",size = 3)
print(p)
```

2.Simplex method

```
# Define the objective function
obj_fn = c(4, 10)

# Define the constraints
f1_cons = c(2, 1)
f2_cons = c(2, 5)
f3_cons = c(2, 3)

# Define the rhs values
c1 = 50
c2 = 100
c3 = 90
```

```
library(boot)
# Solve using the simplex method
simplex(a=obj_fn, A1=rbind(f1_cons, f2_cons, f3_cons), b1=c(c1, c2, c3), maxi=F)
```

3.Big-M method

```
library(lpSolve)
M = 1000
obj = c(6, 4, 0, 0, 0, -M)
mat = matrix(c(2, 3, 1, 0, 0, 0,
               3, 2, 0, 1, 0, 0,
               1, 1, 0, 0, -1, 1), ncol=6, byrow=TRUE)
rhs = c(30, 24, 3)
direction = c("<=", "<=", ">=")
lp_solution = lp("max", obj, mat, direction, rhs)
print("Optimal solution:\n")
cat("x1: ", lp_solution$solution[1], "\n")
cat("x2: ", lp_solution$solution[2], "\n")
cat("Maximum cost: ", lp_solution$objval, "\n")
```

4.Two phase simplex method

```
library(lpSolve)
c<-c(3,-1)
A<-matrix(c(2,1,
            1,3,
            0,1),ncol=2,byrow=TRUE)
b<-c(2,2,4)
const.dir<-c(">=","<=","<=")
num_artificial<-ncol(A)
c_phase1<-c(rep(0,length(c)),rep(1,num_artificial))
result_phase1<-lp(direction="min",objective.in = c_phase1,
                  const.mat=A,const.dir=const.dir,const.rhs=b)
if(result_phase1$status!=0||any(is.na(result_phase1$solution))){
  cat("No feasible solution found is Phase 1.\n")
}else{
  cat("Feasible solution found in Phase 1.\n")
}
x_phase1<-result_phase1$solution[1:length(c)]
c_phase2<-c
A_phase2<-A[,1:length(c)]
b_phase2<-b
c_phase2[1:length(c)]<-c
result_phase2<-lp(direction ="max",objective.in=c_phase2,
                  const.mat=A_phase2,const.dir=const.dir,const.rhs=b_phase2)
```

```
if(result_phase2$status!=0||any(is.na(result_phase2$solution))){
  cat("No feasible solution found in Phase 2.\n")
} else{
  cat("Optimal solution found in Phase 2.\n")
  cat("\nFinal solution:\n")
  cat("objective value:",result_phase2$objval,"\n")
  cat("optimal values of variables:",result_phase2$solution[1:length(c)],"\n")
}
```

5.Dual&Primal of LPP

```
library(lpSolve)

obj.coef<- c(3,2)

mat<-matrix(c(-1,-1,
              1,1,
              1,2,
              0,1),ncol=2,byrow=TRUE)

dir<- c("<=","<=","<=","<=")

rhs<- c(-1,7,10,3)


primal_solution <- lp("max",obj.coef,mat,dir,rhs)

cat("Primal solution:\n")
print(primal_solution)

primal_variables <- primal_solution$solution

print(primal_variables)

dual_coefs <- primal_solution$solution

dual_mat <- matrix(c(-1,1,1,0,
                     -1,1,2,1),ncol= 4, byrow=TRUE)

dual_solution <-lp("min",rhs,dual_mat,c(">=",">="),obj.coef)

cat("Dual solution:\n")
print(dual_solution)

dual_variables <- dual_solution$solution
cat("\nDual variables values:\n")
print(dual_variables)
```

6.Implementing problems on sensitivity analysis

```
library(lpSolve)
obj=c(2,1,4,-1,0,0)
mat=matrix(c(1,2,1,-3,1,0,0,0,-1,1,2,0,1,0,2,7,-5,-10,0,0,1),ncol=7,byrow=TRUE)
rhs=c(8,0,21)
direction=c("<=","<=","<=")
original_solution=solve_lp(obj,mat,direction,rhs)
solve_lp=function(obj,mat,direction,rhs)
lp_solution=lp("max",obj,mat,direction,rhs)
if (lp_solution$status == 0) {
   cat("Optimal solution:\n")
   cat("x1: ", lp_solution$solution[1], "\n")
   cat("x2: ", lp_solution$solution[2], "\n")
   cat("x3: ", lp_solution$solution[3], "\n")
   cat("x4:",lp_solution$solution[4],"\n")
   cat("maximum cost:",lp_solution$objval,"\n")
   cat(sensitivity analysis for rhs:\n")
for(i in 1:length(rhs){
     cat("constraint",i,":",lp_solution$pi[i],"\n")
}
}else if(lp_solution$status==2){
     cat("the problem is infeasible\n")
}else{
     cat("error in solving LPP\n")
}
return(lp_solution)
}
original_solution=solve_lp(obj,mat,direction,rhs)
new_rhs=c(8,11,21)
new_solution=solve_lp(obj,mat,direction,new_rhs)
new_rhs=c(3,-2,4)
new_solution=solve_lp(obj,mat,direction,new_rhs)
```

7.Effect of changes in cost coefficient

```
library(lpSolve)
solve_lp = function(obj, mat, direction, rhs){
  lp_solution = lp("max", obj, mat, direction, rhs)
  if (lp_solution$status == 0) {
    cat("Optimal solution:\n")
    cat("x1: ", lp_solution$solution[1], "\n")
    cat("x2: ", lp_solution$solution[2], "\n")
    cat("x3: ", lp_solution$solution[3], "\n")
    cat("Maximum value of Z: ", lp_solution$objval, "\n")
    cat("Shadow prices (Sensitivity analysis for RHS): \n")
```

```
    for (i in 1:length(rhs)) {
      cat("Constraint ", i, ": ", lp_solution$pi[i], "\n")
    }
    cat("Reduced costs (Sensitivity analysis for objective function coefficients): \n")
    for(j in 1:length(obj)) {
      cat("Coefficient ", j, ": ", lp_solution$rc[j], "\n")
    }
  } else if (lp_solution$status == 2){
    cat("The problem is infeasible.\n")
  } else {
    cat("Error in solving the linear programming problem.\n")
  }
  return(lp_solution)
}
obj = c(4, 6, 2, 0, 0)
mat = matrix(c(1, 1, 1, 1, 0,
               1, 4, 7, 0, 1), ncol=5, byrow=T)
rhs = c(3, 9)
direction = c("<=", "<=")
original_solution = solve_lp(obj, mat, direction, rhs)
new_obj_c3 = c(4, 6, 12, 0, 0)
new_solution_c3 = solve_lp(new_obj_c3, mat, direction, rhs)
cat("Change in maximum value of Z for C3: ", new_solution_c3$objval -
original_solution$objval, "\n")
new_obj_all = c(2, 8, 4, 0, 0)
new_solution_all = solve_lp(new_obj_all, mat, direction, rhs)
cat("Change in maximum value of Z for (c1, c2, c3): ", new_solution_all$objval -
original_solution$objval, "\n")
```

8.Transportation Problem

```
library(lpSolve)
cm=matrix(0,nrow=3,ncol=3)
cm[1,1]=2
cm[1,2]=2
cm[1,3]=3
cm[2,1]=4
cm[2,2]=1
cm[2,3]=2
cm[3,1]=1
cm[3,2]=3
cm[3,3]=1
print("cost matrix")
print(cm)
row_signs=rep("<=",3)
row_rhs=c(10,15,40)
col_signs=rep(">=",3)
```

```
col_rhs=c(20,15,30)
lp_solution=lp.transport(cm,"min",row_signs,row_rhs,col_signs,col_rhs)
print("optimal transportation matrix")
print(lp_solution$solution)
print(paste("total cost:",lp_solution$objval))
```

9.Assignment Problem

```
library(lpSolve)
assign.costs=matrix(c(11,8,9,14,10,6,6,11,7),3,3,byrow=TRUE)
Assign.costs
lp.assign(assign.costs)
lp.assign(assign.costs)$solution
```

10.Queueing Theory(Model 1)

```
lambda=10/60
mu=1/3
traffic_intensity=function(lambda,mu){
  rho=lambda/mu
  return(rho)
}
avg_num_customers_queue=function(lambda,mu){
  rho=traffic_intensity(lambda,mu)
  lq=(rho^2/(1-rho))
  return(lq)
}
rho=traffic_intensity(lambda,mu)
lq=avg_num_customers_queue(lambda,mu)
ls=lq+lambda/mu
ws=ls/lambda
wq=lq/lambda
cat("traffic intensity(rho):",round(rho,4),"\n")
cat("avg no of customers waiting in queue(lq):",round(lq,4),"\n")
cat("avg no of customers in the system(ls):",round(ls,4),"\n")
cat("avg time a customer spends in the system(ws):",round(ws,4),"minutes\n")
cat("avg time a customer spends waiting in the queue(wq):",round(wq,4),"minutes\n")
```

11.Queueing Theory(Model 2)

```
lambda <- 20
mu <- 11
s <- 2
rho <- lambda / (s * mu)
```

```
if (rho >= 1)
  stop('Stability measure is >= 1')
find_ps <- function(lambda, mu, s, rho) {
  fact <- 1
  sum <- 0
  for (n in 0 : (s - 1)) {
    sum <- sum + (((s * rho) ^ n) / fact)
    fact <- fact * (n + 1)
  }
  p0 <- sum + (((s * rho) ^ s) / (fact * (1 - rho)))
  p0 <- 1 / p0
  cat('Probabilty of server being idle (P0)', p0, '\n')
  ps <- ((lambda / mu) ^ s) * p0 / fact
  cat('Probabilty of server being idle (Ps)', ps, '\n')
  return(ps)
}
lq <- find_ps(lambda, mu, s, rho) * rho / ((1 - rho) ^ 2)
ls <- (lambda / mu) + lq
ws <- ls / lambda
wq <- lq / lambda
cat(
  'Traffic intensity (rho):', rho,
  '\nAvg no of customers waiting in queue (lq):', lq,
  '\nAvg no of customers in the system (ls):', ls,
  '\nAvg time spent by customer in the system (ws):', ws,
  '\nAvg time spent waiting in the queue by customer (wq):', wq, '\n'
)
```

12.Replacement model

```
ini_cost=6000
scrap=c(3000,1500,750,375,200,200)
main.cost=c(1000,1200,1400,1800,2300,2800)
total=ini_cost-scrap+cumsum(main.cost)
avg_cost=total/(1:length(total))
optimal_year=which.min(avg_cost)
cat("year : ",optimal_year,"\n")
cat("average cost Rs. ",avg_cost[optimal_year])
```

13.Inventory Model

```
annual_demand=200
inventory_cost=1
holding_cost=25
EOQ=sqrt((2annual_demandinventory_cost)/holding_cost)
```

```
time_between_orders=EOQ/annual_demand
orders_per_year=annual_demand/EOQ
c_min=sqrt(2inventory_cost *holding_costannual_demand)
total_cost=(annual_demand)+c_min
cat("economic order ",round(EOQ),”\n”)
cat("time b/w two consecutive orders ",round(time_between_orders,2),"years\n")
cat("minimum average yearly cost Rs.",round(c_min),”\n”)
cat("no of orders per year ",round(orders_per_year,2),”\n”)
cat("optimal total cost: ",round(total_cost))
```