INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Planeación y Control de la Producción

Computational Application 1: Forecast and APP

https://github.com/adeandak/PCP/tree/master/CompApp_ForecastAPP

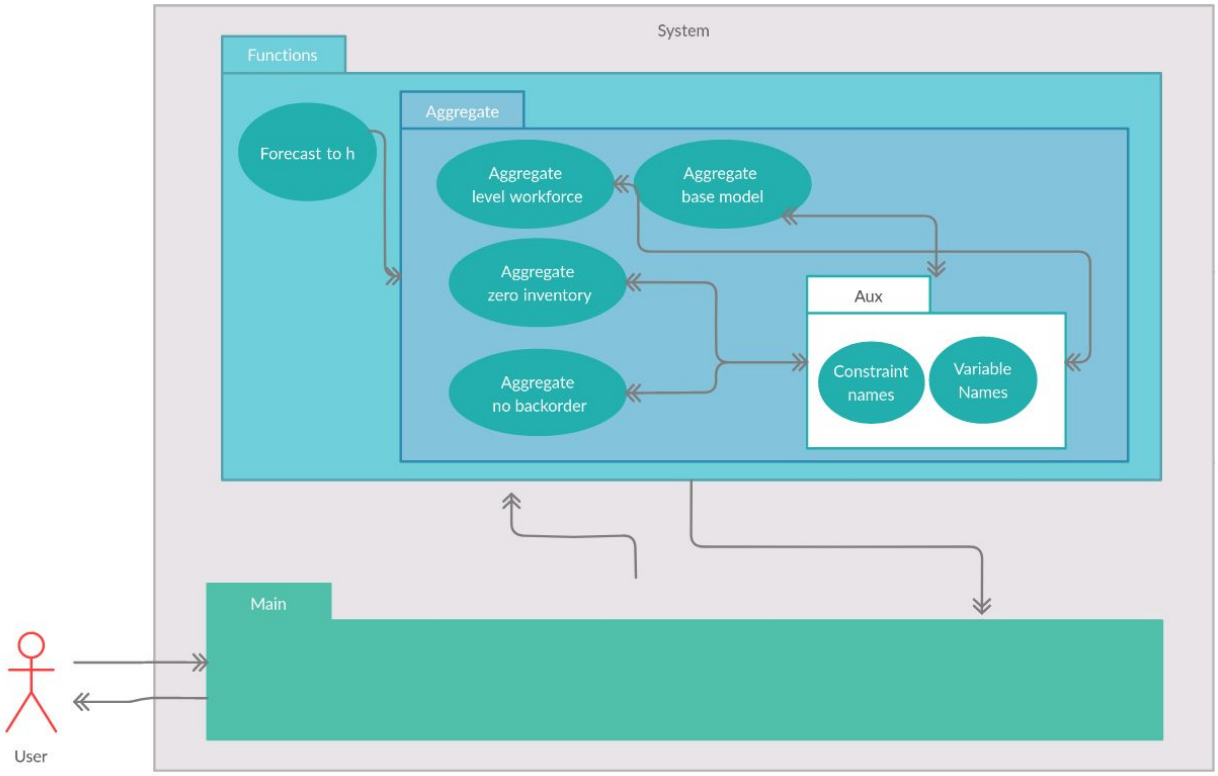
Andrea de Anda Kuri	173347
Juliana Hernández Ottalengo	171405
Silvestre Leonardo Gónzalez Abreu	170404

24 Septiembre 2020

Estructura de la aplicación

La aplicación está conformada por tres partes, la primera es capaz de hacer un pronóstico de la demanda para “h” periodos a partir de un archivo de excel dado y la segunda desarrolla un plan agregado de producción que minimiza los costos a partir de la demanda pronosticada anteriormente y la última convierte ambos componentes en una aplicación con interfaz amigable al usuario.

Diagrama UML



Pronóstico

Para la parte del pronóstico la aplicación recibe un documento de excel, una fecha final de los datos y cuantos periodos se desea pronosticar. Con los datos, la función “forecastToh” se encarga de analizar si la serie de tiempo no es aleatoria, en caso de que no lo sea convierte los datos a una serie de tiempo y define dos subgrupos: el primero llamado “Training data”, que toma en cuenta todos los datos exceptuando los de los últimos h periodos y se utiliza para pronosticar los siguientes valores. El segundo, llamado “Test data”, conformado por los datos de los últimos h periodos, es utilizado para medir la precisión de cada método y de esta forma poder elegir el mejor.

```r
forecastToh <- function(ts,endTs=c(2020,12),f,hF){

  pValue <- Box.test(ts, lag = 1, type = "Ljung")$p.value

  if(pValue <= 0.05){

    endTrainData=endTs
    for(i in 1:hF){
      if(endTrainData[2]<=1){
        endTrainData[1]=endTrainData[1]-1
        endTrainData[2]=f
      }else{
        endTrainData[2]=endTrainData[2]-1
      }
      if(i==hF-1){
        startTestData=endTrainData
      }
    }

    trainData=window(ts,end=endTrainData)
    testData=window(ts,start = startTestData)
```

La función crea pronósticos a partir de trainingData con 6 métodos distintos (Promedio, Naive, Naive Estacional, Exponencial simple, Holt, Holt-Winters Aditiva y Holt-Winters Multiplicativa), en seguida corre la función accuracy con los parámetros de trainingData y testData para medir la exactitud del pronóstico respecto a los valores reales.

```r
# #---------Hacer pronósticos-----
    averageMethod <- meanf(trainData, h=hF)
    naiveMethod <- naive(trainData, h= hF)
    seasonalNaiveMethod <- snaive(trainData, h=hF)
    fcSimpleExpSmoo <- ses(trainData, initial = c("optimal"), h= hF)
    holtForecast <- holt(trainData, h=hF)
    hwAditive <- hw(trainData, seasonal = "additive", h= hF)
    hwMultiplicative <- hw(trainData, seasonal = "multiplicative", h = hF)
    #
    # #-------Checar la exactitud de los métodos
    accAve=accuracy(averageMethod, testData)
```

```
    accNai=accuracy(naiveMethod, testData)
    accSNai=accuracy(seasonalNaiveMethod, testData)
    accExp=accuracy(fcSimpleExpSmoo, testData)
    accHolt=accuracy(holtForecast, testData)
    accHWA=accuracy(hwAditive, testData)
    accHWM=accuracy(hwMultiplicative, testData)


meths=c(accAve[1,2],accNai[1,2],accSNai[1,2],accExp[1,2],accHolt[1,2],accHW
A[1,2],accHWM[1,2])
```

Luego selecciona el mejor método basado en la raíz del error cuadrático medio (RMSE), utiliza el método seleccionado y la serie de tiempo original para hacer el pronóstico de los siguientes “h” periodos para la demanda y checa que los residuales sean aleatorios, para decidir si es el método adecuado.

```
cond=TRUE
    while(cond){
      minRMSE=which.min(meths)

      selectedMethod=switch (minRMSE,
                             meanf(ts,h=hF),
                             naive(ts,h=hF),
                             snaive(ts,h=hF),
                             ses(ts,initial = c("optimal"),h=hF),
                             holt(ts,h=hF),
                             hw(ts, seasonal = "additive", h= hF),
                             hw(ts, seasonal = "multiplicative", h= hF)
      )

      pValueSelected=Box.test(selectedMethod$residuals, lag = 1, type =
"Ljung")$p.value

      if(pValueSelected<=0.05){
        meths[minRMSE]=Inf
      }else{
        cond=FALSE
      }
    }
    return(selectedMethod)
```

En caso de que los datos sean aleatorios, es decir el valor p de la prueba Ljung-Box es mayor a 0.05, la función manda el mensaje de que no se puede correr el pronóstico debido a que la serie parece ser aleatoria.

```
 }else{
    warning("La serie de tiempo es aleatoria.")
  }
}
```

Plan agregado de producción

Para calcular el plan agregado de producción se crearon cuatro funciones, para cada uno de los casos:

1. Modelo base
   La función "aggBaseModel" recibe los valores de la demanda pronosticada (calculada con la función forecastToh), el número de periodos, un vector con los días laborales de cada trabajador, el número de trabajadores inicial, el inventario inicial, backorders iniciales, el salario, el costo de contratar, el costo de despedir, el costo de tener inventario en el almacén, el costo de generar una backorder, el costo de producción, el costo de producción del periodo anterior, el salario del periodo anterior y días del periodo anterior y lo convierte en un problema de minimización que se resuelve con programación lineal.
   Primero, calcula el número de variables y restricciones del problema y les asigna nombre con ayuda de las funciones "varNames" y "resNames".

```
aggBaseModel <- function(lambda=c(0,0,0,0,0,0),tPeriods=6,
lDays=c(22,22,22,22,22,22),W0=0,I0=0,B0=0,wages=0,hire=0,fire=0,
hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLastPeriod=0,dLastPeriod=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*3+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=1)
```

   Después calcula el salario de cada trabajador y crea la matriz de función objetivo

```
monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
    objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)
```

En seguida crea la matriz de las restricciones, y añade las restricciones de trabajadores, inventario y backorders iniciales

```
lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)
```

Asigna a la matriz los valores de las restricciones de número de trabajadores, inventario y producción

```
 lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
    lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }

  for(k in 1:tPeriods){
    lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
    lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
    lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
```

```
    lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
    lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
    lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("=", tPeriods)
  productionRHS = rep(0,tPeriods)
```

Finalmente, encuentra la solución óptima al problema por medio de la función “lp” e imprime el resultado

```
 # Find the optimal solution
  aggPlan = lp(direction = "min", objective.in = objF, const.mat =
lpMatrix, const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns), const.rhs =
c(initialsRHS, workersRHS, inventoryRHS, productionRHS))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)

  return(aggPlan)
```

2. Fuerza de trabajo constante
   La función “aggLevelWorkforce” recibe los valores de la demanda pronosticada (calculada con la función forecastToh), el número de periodos, un vector con los días laborales de cada trabajador, el número de trabajadores inicial, el inventario inicial, backorders iniciales, el salario, el costo de contratar, el costo de despedir, el costo de tener inventario en el almacén, el costo de generar una backorder, el costo de producción, el costo de producción del

periodo anterior, el salario del periodo anterior, días del periodo anterior y el nivel de fuerza de trabajo deseado y lo convierte en un problema de minimización que se resuelve con programación lineal.
Primero, calcula el número de variables y restricciones del problema y les asigna nombre con ayuda de las funciones “varNames” y “resNames”.

```
aggLevelWorkforce <- function(lambda=c(0,0,0,0,0,0),tPeriods=6,
lDays=c(22,22,22,22,22,22),W0=0,I0=0,B0=0,wages=0,hire=0,fire=0,
hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLastPeriod=0,
dLastPeriod=0,levelWf=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*4+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=2)
```

Después calcula el salario de cada trabajador y crea la matriz de función objetivo.

```
  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
    objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)
```

En seguida crea la matriz de las restricciones, y añade las restricciones de trabajadores, inventario y backorders iniciales.

```
  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0
```

```
  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)
```

Asigna a la matriz los valores de las restricciones de número de trabajadores, inventario, producción y nivel de fuerza de trabajo.

```
  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
    lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }

  for(k in 1:tPeriods){
    lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
    lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
    lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
    lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
    lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
    lpMatrix[rest[k+3+3*tPeriods],vars[4+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns = rep("=", tPeriods)
  invBackRHS = rep(levelWf,tPeriods)
```

Finalmente, encuentra la solución óptima al problema por medio de la función “lp” e imprime el resultado.

```
  # Find the optimal solution
  aggPlan = lp(direction = "min", objective.in = objF, const.mat =
lpMatrix, const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
const.rhs = c(initialsRHS, workersRHS, inventoryRHS, productionRHS,
invBackRHS))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)

  return(aggPlan)
```

3. Fuerza de trabajo constante y sin backorders
   La función “aggNoBackorder” recibe los valores de la demanda pronosticada (calculada con la función forecastToh), el número de periodos, un vector con los días laborales de cada trabajador, el número de trabajadores inicial, el inventario inicial, backorders iniciales, el salario, el costo de contratar, el costo de despedir, el costo de tener inventario en el almacén, el costo de generar una backorder, el costo de producción, el costo de producción del periodo anterior, el salario del periodo anterior, días del periodo anterior y el nivel de fuerza de trabajo deseado y lo convierte en un problema de minimización que se resuelve con programación lineal.
   Primero, calcula el número de variables y restricciones del problema y les asigna nombre con ayuda de las funciones “varNames” y “resNames”.

```
aggNoBackorder <- function(lambda=c(0,0,0,0,0,0),tPeriods=6,
lDays=c(22,22,22,22,22,22),W0=0,I0=0,B0=0,wages=0,hire=0,fire=0,
hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLastPeriod=0,
dLastPeriod=0,levelWf=0)

  numVar=tPeriods*6+3
  numRes=tPeriods*5+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=3)
```

Después calcula el salario de cada trabajador y crea la matriz de función objetivo.

```
  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
    objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)
```

En seguida crea la matriz de las restricciones, y añade las restricciones de trabajadores, inventario y backorders iniciales.

```
  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)
```

Asigna a la matriz los valores de las restricciones de número de trabajadores, inventario, producción y nivel de fuerza de trabajo.

```
  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
    lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }
```

```
  for(k in 1:tPeriods){
    lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
    lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
    lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
    lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
    lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
    lpMatrix[rest[k+3+3*tPeriods],vars[4+6*(k-1)]]=1
    lpMatrix[rest[k+3+4*tPeriods],vars[8+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep(">=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns <- rep("=", 2*tPeriods)
  invBackRHS <- c(rep(levelWf,tPeriods), rep(0,tPeriods))

    lpMatrix[rest[k+3+3*tPeriods],vars[4+6*(k-1)]]=1
  }
```

Finalmente, encuentra la solución óptima al problema por medio de la función “lp” e imprime el resultado.

```
  # Find the optimal solution
  aggPlan = lp(direction = "min", objective.in = objF, const.mat =
lpMatrix, const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
const.rhs = c(initialsRHS, workersRHS, inventoryRHS, productionRHS,
invBackRHS))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)
```

```
  return(aggPlan)
```

4. SIn inventario ni backorders
   La función “aggZeroInvBack” recibe los valores de la demanda pronosticada (calculada con la función forecastToh), el número de periodos, un vector con los días laborales de cada trabajador, el número de trabajadores inicial, el inventario inicial, backorders iniciales, el salario, el costo de contratar, el costo de despedir, el costo de tener inventario en el almacén, el costo de generar una backorder, el costo de producción, el costo de producción del periodo anterior, el salario del periodo anterior y días del periodo anterior y lo convierte en un problema de minimización que se resuelve con programación lineal.
   Primero, calcula el número de variables y restricciones del problema y les asigna nombre con ayuda de las funciones “varNames” y “resNames”.

```
aggZeroInvBack <- function(lambda=c(0,0,0,0,0,0),tPeriods=6,
lDays=c(22,22,22,22,22,22),W0=0,I0=0,B0=0,wages=0,hire=0,fire=0,hold=0,
backOrder=0,pCost=0,pLastPeriod=0,wLastPeriod=0,dLastPeriod=0){
 numVar=tPeriods*6+3
 numRes=tPeriods*5+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=4)
```

   Después calcula el salario de cada trabajador y crea la matriz de función objetivo.

```
  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
    objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)
```

   En seguida crea la matriz de las restricciones, y añade las restricciones de trabajadores, inventario y backorders iniciales.

```
  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
```

```
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)
```

Asigna a la matriz los valores de las restricciones de número de trabajadores, inventario, producción y nivel de fuerza de trabajo.

```
 lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
    lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
    lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }

  for(k in 1:tPeriods){
    lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
    lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
    lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
    lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
    lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
    lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
    lpMatrix[rest[k+3+3*tPeriods],vars[7+6*(k-1)]]=1
    lpMatrix[rest[k+3+4*tPeriods],vars[8+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
```

```
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("<=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns <- rep("=", 2*tPeriods)
  invBackRHS <- rep(0,2*tPeriods)
```

Finalmente, encuentra la solución óptima al problema por medio de la función “lp” e imprime el resultado.

```
  # Find the optimal solution
  aggPlan = lp(direction = "min", objective.in = objF, const.mat =
lpMatrix, const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
const.rhs = c(initialsRHS, workersRHS, inventoryRHS, productionRHS,
invBackRHS))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)

  return(aggPlan)
```

Funciones auxiliares

Para poder plantear los problemas de programación lineal se utilizaron dos funciones auxiliares.

La función “varNames” se encarga de asignarle nombre a las variables de decisión (fuerza de trabajo, contrataciones, despidos, inventario, backorders y producción) para cada periodo.

```
varNames <- function(tP=6){
  varLet=c('W','H','F','I','B','P')
  varNames=c('W0','I0','B0')
  for(i in 1:tP){
    for(j in 1:6){
      varNames=c(varNames,paste(varLet[j],i,sep=""))
    }
  }
  return(varNames)
}
```

La función “restNames” asigna etiquetas a las restricciones dependiendo del periodo y del tipo de modelo de plan agregado de producción.

```
restNames <- function(tP=6,mode=1){
  #mode:
  #1=base model
  #2=level workforce
  #3=no backorders
  #4=zero inventory and backorders

  varLet=c('Wo','In','Pr')


varLet=switch(mode,varLet,c(varLet,'Wk'),c(varLet,'Wk','Bkz'),c(varLet,'Inz
','Bkz'))

  varNames=c('InitialWorkers','InitialInventory','InitialBackorders')
  for(i in 1:length(varLet)){
    for(j in 1:tP){
      varNames=c(varNames,paste(varLet[i],j,sep=""))
    }
  }
  return(varNames)
}
```

Manual de usuario

Crear un proyecto que contenga los dos archivos .R que se encuentran en la secci'on de anexos.

Abrir el proyecto

1. Abrir el proyecto (**CompApp1_ForecastApp.proj**)
2. Dentro del proyecto se encuentra un archivo llamado **forecastAndAggProdApp.R** que es el que necesita ser modificado.

Datos de entrada

3. Introducir los datos correspondientes para poder realizar el pronóstico (los datos que están después del "=" son marcadores de posición y deben de modificarse)
   ¿Qué datos se deben introducir?
   - Para cargar el excel, en la línea 10:

```
#Inputs para carga de Excel
pathArchivo="DAUPSA.xlsx"
saltos=10
columnaDatos=2
conTitulos=TRUE
```

   - Para la serie de tiempo, en la línea 24:

```
#Inputs para creacion de serie de tiempo
frecuencia=12 #ie numero de periodos en un año
anoFin=2020
mesFin=7
```

   - Para poder realizar el pronóstico, en la línea 36:

```
#Inputs para realizar el pronóstico
horizonte=6 #ie periodos a futuro que se desea pronosticar
```

4. Modificar los datos de entrada necesarios para crear el plan de producción agregada.
   - A partir de la línea 49:

```
#Inputs para realizar el plan agregado de produccion
modo=1  #el tipo de plan a realizar,
      #1=base model
      #2=level workforce
      #3=no backorders
      #4=no inventory/backorders
diasLaboralesPAnterior=260
```

```
produccionPAnterior=41383
trabajadoresPAnterior=40
vectorDiasLaborales=c(21,20,23,21,22,22)
trabajadoresIniciales=35
inventarioInicial=0
backordersInicial=0
salarios=120
costoContratar=450
costoDespedir=600
costoAlmacen=5
costoBackorder=15
costoProducir=0
```

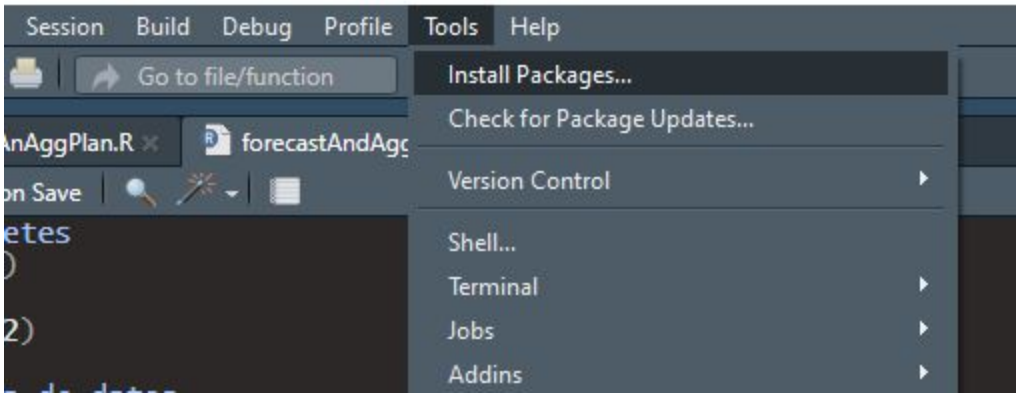
- Adicionalmente, en caso de seleccionar el modo 2 o 3 se debe de especificar la fuerza laboral, a partir de la línea 69:

```
#en el caso de usar el modo 2 o 3 se debe especificar la fuerza laboral
fuerzaLaboralConstante=40
```

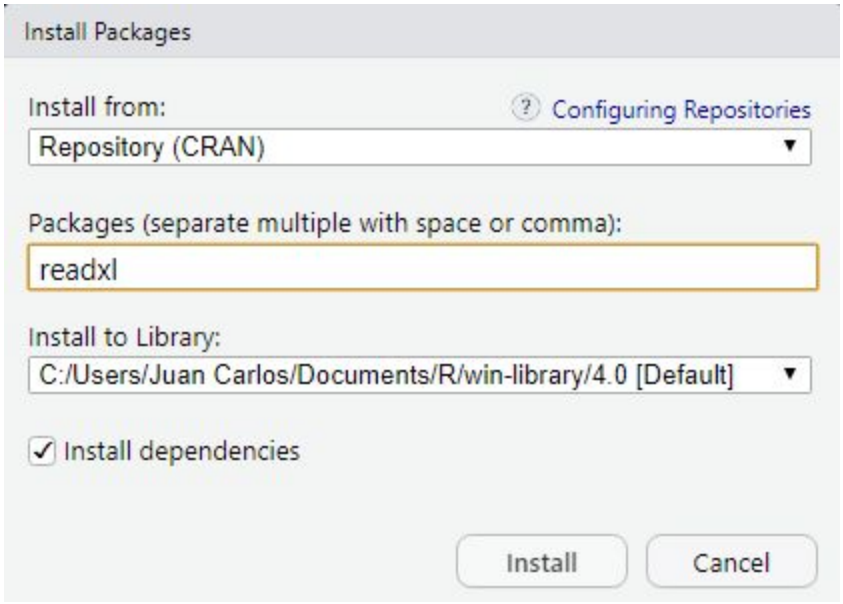
Paquetes de R necesarios

5. Antes de correr el programa es necesario instalar los paquetes necesarios:
    - fpp2
    - lpsolve
    - Ggplot2

Para instalar los paquetes utiliza la herramienta de “tool” en en el menu superior de R y luego haz click en la opción “Install Packages”.

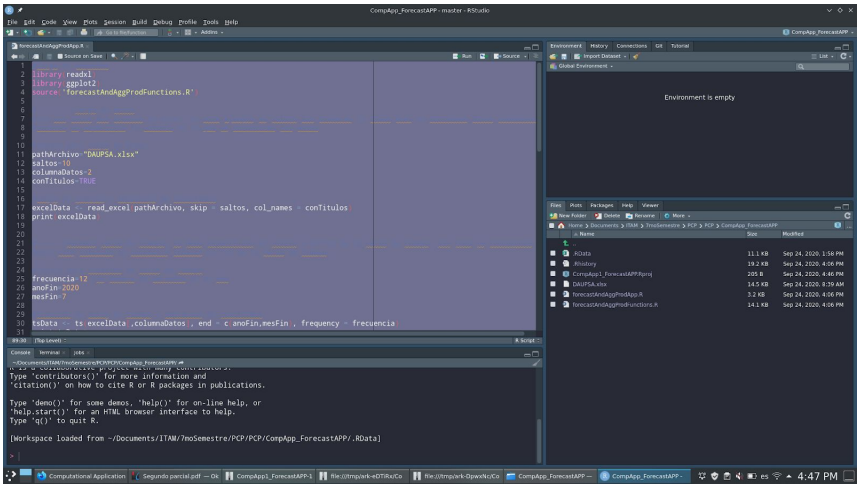


En seguida busca los paquetes necesarios y selecciona la opción de install.



Correr el código

6. Seleccionar todo el código y correrlo
    a. Ctrl+enter
    b. Dar click en **Run**



Anexos

forecastAndAggProdApp.R

```
#Carga de paquetes
library(readxl)
library(ggplot2)
source('forecastAndAggProdFunctions.R')

#---------Carga de datos
#Lee los datos del archivo de excel, en este ejemplo se saltan las primeras
10 filas pues no contienen datos relevantes,
#tambien se especifica la columna en la que se encuentran los datos

#Inputs para carga de Excel
pathArchivo="DAUPSA.xlsx"
saltos=10
columnaDatos=2
conTitulos=TRUE

#Carga de archivo e impresion de resultados
excelData <- read_excel(pathArchivo, skip = saltos, col_names = conTitulos)
print(excelData)


#-------Hacemos serie de tiempo, se debe especificar el inicio o el fin de
la serie de tiempo, asi como su frecuencia,
#para esta solucion es necesario indicar el fin de la serie de tiempo.

#Inputs para creacion de serie de tiempo
frecuencia=12 #ie numero de periodos en un año
anoFin=2020
mesFin=7

#Creacion de serie de tiempo e impresion de resultados
tsData <- ts(excelData[,columnaDatos], end = c(anoFin,mesFin), frequency =
frecuencia)
print(tsData)

#-------Realizacion del pronostico, se identifica automaticamente si es
posible pronosticar con los datos cargados,
```

```
#en ese caso se selecciona el mejor metodo para realizarlo y se devuelve la
serie de tiempo del pronostico

#Inputs para realizar el pronostico
horizonte=6 #ie periodos a futuro que se desean pronosticar

#Realizacion del pronostico y graficacion de resultados
fcData=forecastToh(ts=tsData,endTs =
c(anoFin,mesFin),f=frecuencia,hF=horizonte)
autoplot(tsData) + xlab("Year") + ylab("Demand") +
  autolayer(fcData, PI=FALSE, series = paste(fcData$method,"forecast"))
print(fcData)


#-------Construccion del plan agregado de produccion, se deben especificar
los datos de produccion del periodo anterior,
#al igual que las condiciones iniciales de trabajadores, inventario y
backorders, y los datos de costos de cada variable

#Inputs para realizar el plan agregado de produccion
modo=1  #el tipo de plan a realizar,
      #1=base model
      #2=level workforce
      #3=no backorders
      #4=no inventory/backorders
diasLaboralesPAnterior=260
produccionPAnterior=41383
trabajadoresPAnterior=40
vectorDiasLaborales=c(21,20,23,21,22,22)
trabajadoresIniciales=35
inventarioInicial=0
backordersInicial=0
salarios=120
costoContratar=450
costoDespedir=600
costoAlmacen=5
costoBackorder=15
costoProducir=0

#en el caso de usar el modo 2 o 3 se debe especificar la fuerza laboral
fuerzaLaboralConstante=40

#Solucion del plan de produccion e impresion
```

```
aggPlan=aggProdPlan(lambda=unlist(fcData[2], use.names=FALSE),
                    tPeriods=horizonte,
                    lDays=vectorDiasLaborales,
                    W0=trabajadoresIniciales,
                    I0=inventarioInicial,
                    B0=backordersInicial,
                    wages=salarios,
                    hire=costoContratar,
                    fire=costoDespedir,
                    hold=costoAlmacen,
                    backOrder=costoBackorder,
                    pCost=costoProducir,
                    pLastPeriod=produccionPAnterior,
                    wLastPeriod=trabajadoresPAnterior,
                    dLastPeriod=diasLaboralesPAnterior,
                    levelWf = fuerzaLaboralConstante,
                    mode = modo)
```

forecastAndAggProdFunctions.R

```
library(fpp2)

forecastToh <- function(ts,endTs=c(2020,12),f,hF){

  pValue <- Box.test(ts, lag = 1, type = "Ljung")$p.value

  if(pValue <= 0.05){

      endTrainData=endTs
      for(i in 1:hF){
      if(endTrainData[2]<=1){
      endTrainData[1]=endTrainData[1]-1
      endTrainData[2]=f
      }else{
      endTrainData[2]=endTrainData[2]-1
      }
      if(i==hF-1){
      startTestData=endTrainData
      }
      }

      trainData=window(ts,end=endTrainData)
```

```
      testData=window(ts,start = startTestData)


      # #-----------------------------------------
      # #Encontrar el mejor método para pronosticar
      # #-----------------------------------------

      # #---------Hacer pronósticos-----
      averageMethod <- meanf(trainData, h=hF)
      naiveMethod <- naive(trainData, h= hF)
      seasonalNaiveMethod <- snaive(trainData, h=hF)
      fcSimpleExpSmoo <- ses(trainData, initial = c("optimal"), h= hF)
      holtForecast <- holt(trainData, h=hF)
      hwAditive <- hw(trainData, seasonal = "additive", h= hF)
      hwMultiplicative <- hw(trainData, seasonal = "multiplicative", h =
hF)
      #
      # #-------Checar la exactitud de los métodos
      accAve=accuracy(averageMethod, testData)
      accNai=accuracy(naiveMethod, testData)
      accSNai=accuracy(seasonalNaiveMethod, testData)
      accExp=accuracy(fcSimpleExpSmoo, testData)
      accHolt=accuracy(holtForecast, testData)
      accHWA=accuracy(hwAditive, testData)
      accHWM=accuracy(hwMultiplicative, testData)


meths=c(accAve[1,2],accNai[1,2],accSNai[1,2],accExp[1,2],accHolt[1,2],accHW
A[1,2],accHWM[1,2])

      cond=TRUE
      while(cond){
      minRMSE=which.min(meths)

      selectedMethod=switch (minRMSE,
                             meanf(ts,h=hF),
                             naive(ts,h=hF),
                             snaive(ts,h=hF),
                             ses(ts,initial = c("optimal"),h=hF),
                             holt(ts,h=hF),
                             hw(ts, seasonal = "additive", h= hF),
                             hw(ts, seasonal = "multiplicative", h= hF)
      )
```

```
      pValueSelected=Box.test(selectedMethod$residuals, lag = 1, type =
"Ljung")$p.value

      if(pValueSelected<=0.05){
      meths[minRMSE]=Inf
      }else{
      cond=FALSE
      }
      }
      return(selectedMethod)

  }else{
      warning("La serie de tiempo es aleatoria.")
  }

}

require(lpSolve)

varNames <- function(tP=6){
  varLet=c('W','H','F','I','B','P')
  varNames=c('W0','I0','B0')
  for(i in 1:tP){
      for(j in 1:6){
      varNames=c(varNames,paste(varLet[j],i,sep=""))
      }
  }
  return(varNames)
}

restNames <- function(tP=6,mode=1){
  #mode:
  #1=base model
  #2=level workforce
  #3=no backorders
  #4=zero inventory and backorders

  varLet=c('Wo','In','Pr')


varLet=switch(mode,varLet,c(varLet,'Wk'),c(varLet,'Wk','Bkz'),c(varLet,'Inz
','Bkz'))
```

```
  varNames=c('InitialWorkers','InitialInventory','InitialBackorders')
  for(i in 1:length(varLet)){
      for(j in 1:tP){
      varNames=c(varNames,paste(varLet[i],j,sep=""))
      }
  }
  return(varNames)
}

aggBaseModel <-
function(lambda=c(0,0,0,0,0,0),tPeriods=6,lDays=c(22,22,22,22,22,22),W0=0,I
0=0,B0=0,wages=0,hire=0,fire=0,hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLa
stPeriod=0,dLastPeriod=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*3+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=1)

  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
      objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)

  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)
```

```
  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
      lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }

  for(k in 1:tPeriods){
      lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
      lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
      lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
      lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
      lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("=", tPeriods)
  productionRHS = rep(0,tPeriods)

  # Find the optimal solution
  aggPlan = lp(direction = "min",
               objective.in = objF,
               const.mat = lpMatrix,
               const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns),
               const.rhs = c(initialsRHS, workersRHS, inventoryRHS,
productionRHS),
               int.vec = c(4+6*c(0:tPeriods-1)))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
```

```
  print(best_sol)

  return(aggPlan)
}

aggLevelWorkforce <-
function(lambda=c(0,0,0,0,0,0),tPeriods=6,lDays=c(22,22,22,22,22,22),W0=0,I
0=0,B0=0,wages=0,hire=0,fire=0,hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLa
stPeriod=0,dLastPeriod=0,levelWf=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*4+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=2)

  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
      objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)

  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)

  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
      lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
```

```
      lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }

  for(k in 1:tPeriods){
      lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
      lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
      lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
      lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
      lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
      lpMatrix[rest[k+3+3*tPeriods],vars[4+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns = rep("=", tPeriods)
  invBackRHS = rep(levelWf,tPeriods)

  # Find the optimal solution
  aggPlan = lp(direction = "min",
            objective.in = objF,
            const.mat = lpMatrix,
            const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
            const.rhs = c(initialsRHS, workersRHS, inventoryRHS,
productionRHS, invBackRHS),
            int.vec = c(4+6*c(0:tPeriods-1)))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)
```

```
  return(aggPlan)
}

aggNoBackorder <-
function(lambda=c(0,0,0,0,0,0),tPeriods=6,lDays=c(22,22,22,22,22,22),W0=0,I
0=0,B0=0,wages=0,hire=0,fire=0,hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLa
stPeriod=0,dLastPeriod=0,levelWf=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*5+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=3)

  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
      objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)

  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)

  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
      lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
```

```
  }

  for(k in 1:tPeriods){
      lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
      lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
      lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
      lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
      lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
      lpMatrix[rest[k+3+3*tPeriods],vars[4+6*(k-1)]]=1
      lpMatrix[rest[k+3+4*tPeriods],vars[8+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns <- rep("=", 2*tPeriods)
  invBackRHS <- c(rep(levelWf,tPeriods), rep(0,tPeriods))

  # Find the optimal solution
  aggPlan = lp(direction = "min",
               objective.in = objF,
               const.mat = lpMatrix,
               const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
               const.rhs = c(initialsRHS, workersRHS, inventoryRHS,
productionRHS, invBackRHS),
               int.vec = c(4+6*c(0:tPeriods-1)))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)

  return(aggPlan)
```

```
}

aggZeroInvBack <-
function(lambda=c(0,0,0,0,0,0),tPeriods=6,lDays=c(22,22,22,22,22,22),W0=0,I
0=0,B0=0,wages=0,hire=0,fire=0,hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLa
stPeriod=0,dLastPeriod=0){

  numVar=tPeriods*6+3
  numRes=tPeriods*5+3

  vars=varNames(tP = tPeriods)
  rest=restNames(tP = tPeriods,mode=4)

  monthWage=wages*lDays
  objF=c(0,0,0)
  for(i in 1:tPeriods)
      objF=c(objF,monthWage[i],hire,fire,hold,backOrder,pCost)

  lpMatrix = matrix(0, nrow = numRes, ncol = numVar, dimnames = list(rest,
vars))

  lpMatrix['InitialWorkers', 'W0'] = 1
  lpMatrix['InitialInventory', 'I0'] = 1
  lpMatrix['InitialBackorders', 'B0'] = 1

  initialSigns = rep("=", 3)

  InitialNumberOfWorkers = W0
  InitialNumberInventory = I0
  InitialNumberBackorders = B0

  initialsRHS = c(InitialNumberOfWorkers,
                  InitialNumberInventory,
                  InitialNumberBackorders)

  lpMatrix['Wo1','W0']=-1
  lpMatrix['In1','I0']=-1
  lpMatrix['In1','B0']=1
  for(j in 2:tPeriods){
      lpMatrix[rest[j+3],vars[4+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[7+6*(j-2)]]=-1
      lpMatrix[rest[j+3+tPeriods],vars[8+6*(j-2)]]=1
  }
```

```
  for(k in 1:tPeriods){
      lpMatrix[rest[k+3],vars[5+6*(k-1)]]=-1
      lpMatrix[rest[k+3],vars[4+6*(k-1)]]=1
      lpMatrix[rest[k+3],vars[6+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[7+6*(k-1)]]=1
      lpMatrix[rest[k+3+tPeriods],vars[8+6*(k-1)]]=-1
      lpMatrix[rest[k+3+tPeriods],vars[9+6*(k-1)]]=-1
      lpMatrix[rest[k+3+2*tPeriods],vars[9+6*(k-1)]]=1

lpMatrix[rest[k+3+2*tPeriods],vars[4+6*(k-1)]]=-(pLastPeriod/(dLastPeriod*w
LastPeriod))*lDays[k]
      lpMatrix[rest[k+3+3*tPeriods],vars[7+6*(k-1)]]=1
      lpMatrix[rest[k+3+4*tPeriods],vars[8+6*(k-1)]]=1
  }

  workersSigns = rep("=", tPeriods)
  workersRHS = rep(0,tPeriods)
  inventorySigns = rep("=", tPeriods)
  inventoryRHS = -1*lambda
  productionSigns = rep("<=", tPeriods)
  productionRHS = rep(0,tPeriods)
  invBackSigns <- rep("=", 2*tPeriods)
  invBackRHS <- rep(0,2*tPeriods)

  # Find the optimal solution
  aggPlan = lp(direction = "min",
               objective.in = objF,
               const.mat = lpMatrix,
               const.dir =
c(initialSigns,workersSigns,inventorySigns,productionSigns,invBackSigns),
               const.rhs = c(initialsRHS, workersRHS, inventoryRHS,
productionRHS, invBackRHS),
               int.vec = c(4+6*c(0:tPeriods-1)))

  #Print solution
  print(paste("Total cost = ", aggPlan$objval))
  best_sol = aggPlan$solution
  names(best_sol) = vars
  print(best_sol)

  return(aggPlan)
}
```

```
aggProdPlan <-
function(lambda=c(0,0,0,0,0,0),tPeriods=6,lDays=c(22,22,22,22,22,22),W0=0,I
0=0,B0=0,wages=0,hire=0,fire=0,hold=0,backOrder=0,pCost=0,pLastPeriod=0,wLa
stPeriod=0,dLastPeriod=0,levelWf=0,mode=1){

  aggPlan=switch (mode,

aggBaseModel(lambda,tPeriods,lDays,W0,I0,B0,wages,hire,fire,hold,backOrder,
pCost,pLastPeriod,wLastPeriod,dLastPeriod),

aggLevelWorkforce(lambda,tPeriods,lDays,W0,I0,B0,wages,hire,fire,hold,backO
rder,pCost,pLastPeriod,wLastPeriod,dLastPeriod,levelWf),

aggNoBackorder(lambda,tPeriods,lDays,W0,I0,B0,wages,hire,fire,hold,backOrde
r,pCost,pLastPeriod,wLastPeriod,dLastPeriod,levelWf),

aggZeroInvBack(lambda,tPeriods,lDays,W0,I0,B0,wages,hire,fire,hold,backOrde
r,pCost,pLastPeriod,wLastPeriod,dLastPeriod)
  )

  return(aggPlan)
}
```

Intento (fallido) de la implementación de una app Shiny

Inicialmente se

```
library(shiny)
library(readxl)
source('forecastAnAggPlan.R')

ui <- fluidPage(
  titlePanel("Pronóstico"),
  sidebarLayout(
      sidebarPanel(
      fileInput('archivo', 'Carga un archivo de Excel',
                multiple = FALSE,
                accept = c(
                ".xls",
                ".xlsx"
                )
      ),
      tags$hr(),
      radioButtons('nombre', '¿Los datos tienen un título?',
                c(Si=1,
                No=0
                )),
      tags$hr(),
      numericInput('year','Indique el año final', value =2020),
      tags$hr(),
      numericInput('month','Indique el periodo final',value=12),
      tags$hr(),
      numericInput('freq','Indique la frecuencia',value=12),
      tags$hr(),
      numericInput('horizon','Indique el horizonte de pronóstico',value=6),
      tags$hr(),
      numericInput('colExcel','Indique la columna de los datos',value=2),
      tags$hr(),
      numericInput('skipper','En caso de que los datos no inicien en la
primera celda, indicar los saltos necesarios',value=0)

      ),

      mainPanel(
      tags$style(type="text/css",
                ".shiny-output-error { visibility: hidden; }",
                ".shiny-output-error:before { visibility: hidden; }"
```

```
      ),
      fluidRow("Forecast",
            splitLayout(cellWidths = c("50%", "50%"),
                        plotOutput("contents1"), plotOutput("contents2"))

      ))
  )
)

server <- function(input, output) {

  data_1<-reactive({
      inFile <- input$archivo
      if (is.null(inFile))
      return(NULL)
      excelData <- read_excel(inFile$name, skip = input$skipper, col_names
= (input$nombre==1))
      data<-ts(excelData[,input$colExcel],end =
c(input$year,input$month),frequency = input$freq)
      return(data)
  })

  output$contents1 <- renderPlot({
      data<-data_1()
      theForecast=forecastToh(data,endTs =
c(input$year,input$month),f=input$freq,hF=input$horizon)
      if(is.null(theForecast))
      plot.default(main = "Data is random or insufficient")
      else
      plot(theForecast$mean,main=paste("Forecast with",theForecast$method))
      print(theForecast$mean)

  })
   output$contents2 <- renderPlot({
      data<-data_1()
      tsdisplay(data,main = "Time series")
  })
}

shinyApp(ui = ui, server = server)
```