Eindopdracht 3

Mohammed Al Hor

2023-03-12

# Inladen van de benodigde libraries  
library(simmer)

## Warning: package 'simmer' was built under R version 4.2.2

library(simmer.plot)

## Warning: package 'simmer.plot' was built under R version 4.2.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

##   
## Attaching package: 'simmer.plot'

## The following objects are masked from 'package:simmer':  
##   
## get\_mon\_arrivals, get\_mon\_attributes, get\_mon\_resources

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:simmer':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyverse)

## ── Attaching packages  
## ───────────────────────────────────────  
## tidyverse 1.3.2 ──

## ✔ tibble 3.1.8 ✔ purrr 0.3.4  
## ✔ tidyr 1.2.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.2

## Warning: package 'stringr' was built under R version 4.2.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ dplyr::select() masks simmer::select()  
## ✖ tidyr::separate() masks simmer::separate()

library(tidyr)  
library(readxl)  
library(writexl)

## Warning: package 'writexl' was built under R version 4.2.2

library(fitdistrplus)

## Warning: package 'fitdistrplus' was built under R version 4.2.2

## Loading required package: MASS  
##   
## Attaching package: 'MASS'  
##   
## The following object is masked from 'package:dplyr':  
##   
## select  
##   
## The following object is masked from 'package:simmer':  
##   
## select  
##   
## Loading required package: survival

library(outliers)

# 3. Now load the actual data into R and transform the data into an appropriate format for analysis using the scripts we will provide. Clean for outliers.Determine the average processing time for each phase (checking and admin) and determine the proportion of parcels sent out in time. Is the KPI target of 90% fulfilled?

set.seed(42)  
# inladen dataset  
setwd("~/Data-Science-Business-Analytics/Predictive Modeling Simulation/Eindopdracht")  
df <- read\_excel('~/Data-Science-Business-Analytics/Data/parcel processing data clean-crossed pakketjes.xlsx', sheet = "Data", skip = 1) %>%   
 drop\_na()

# Sourcen van de functies die we later nodig hebben  
source("./compute\_working\_hours.R")

# In dit onderdeel berekenen we de duratie van verschillende activiteiten mbv de working hours functie  
data <- df %>%  
 mutate(check\_time = working\_hours(`Aangepast Begin checken`, `Eind checken`, saturday = FALSE),  
 admin\_time = working\_hours(`Begin admin`, `Eind admin`,saturday = FALSE),  
 throughput\_time = working\_hours(`Eind lossen`, `Eind admin`,saturday = FALSE),  
 min\_eind\_lossen = str\_c(lubridate::date(min(df$`Eind lossen`)), " 07:00:00 UTC") %>% as.POSIXct(tz="utc"),  
 time\_stamp = working\_hours(min\_eind\_lossen, `Eind lossen`, saturday = FALSE),  
 check\_time\_waiting = working\_hours(`Eind lossen`, `Aangepast Begin checken`, saturday = FALSE),  
 admin\_time\_waiting = working\_hours(`Eind checken`, `Begin admin`, saturday = FALSE),  
 activity\_time = check\_time + admin\_time) %>%  
 as.data.frame()  
# We vervangen de 0 waarden met de mean  
data <- data %>%   
 mutate(  
 check\_time = replace(check\_time, check\_time <= 0, mean(check\_time)),  
 admin\_time = replace(admin\_time, admin\_time <= 0, mean(admin\_time))  
 )  
mean\_check\_time <- mean(data$check\_time[data$check\_time > 0])  
mean\_admin\_time <- mean(data$admin\_time[data$admin\_time > 0])  
mean\_check\_time

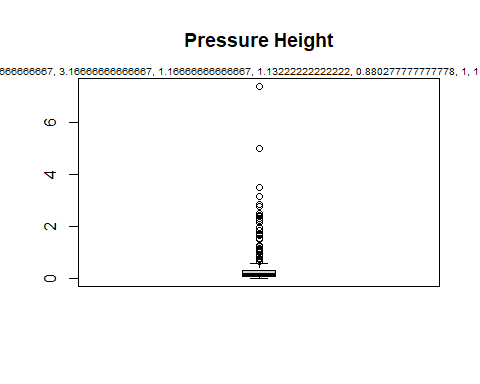
## [1] 0.3584562

mean\_admin\_time

## [1] 0.855021

Outlier analyse. De outliers worden opgespoord en in een vector gezet, deze wordt later gebruikt om die observaties te vervangen met het gemiddelde.

# Check for outliers  
# Checking time  
outlier\_values\_1 <- boxplot.stats(data$check\_time)$out # outlier values.  
boxplot(data$check\_time, main="Pressure Height", boxwex=0.2)  
mtext(paste("Outliers: ", paste(outlier\_values\_1, collapse=", ")), cex=0.6)

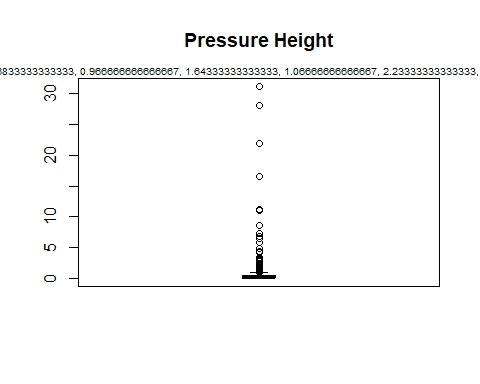


outlier(data$check\_time)

## [1] 7.377778

check\_time\_outliers <-scores(data$check\_time, type="z", prob=0.95) # logicals with cut-off

# Admin time  
outlier\_values\_2 <- boxplot.stats(data$admin\_time)$out # outlier values.  
boxplot(data$admin\_time, main="Pressure Height", boxwex=0.2)  
mtext(paste("Outliers: ", paste(outlier\_values\_2, collapse=", ")), cex=0.6)



outlier(data$admin\_time)

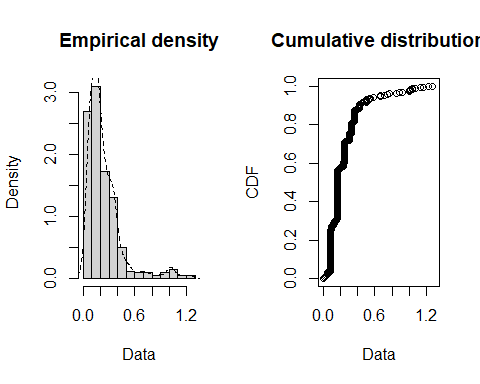
## [1] 31.21667

admin\_time\_outliers <-scores(data$admin\_time, type="z", prob=0.95) # logicals with cut-off

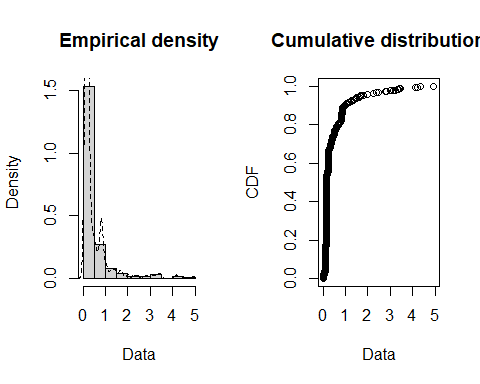
Voor zowel checking time als admin time zien we outliers, deze worden vervangen door het gemiddele in het volgende code:

# In dit onderdeel doen we wat data manipulatie, de waarden die kleiner of gelijk aan nul zijn worden vervangen door gemiddelden. Hetzelfde geldt voor de outliers.  
df\_final <- data %>%  
 mutate(check\_time = ifelse(check\_time\_outliers,mean\_check\_time, check\_time),  
 admin\_time = ifelse(admin\_time\_outliers,mean\_admin\_time, admin\_time)) %>%  
 mutate(check\_time = ifelse(check\_time <= 0, mean\_check\_time, check\_time),  
 admin\_time = ifelse(admin\_time <= 0, mean\_admin\_time, admin\_time),  
 total\_throughput = check\_time + admin\_time)

# Emperische verdeling & CDF  
plotdist(df\_final$check\_time, histo = TRUE, demp=TRUE)



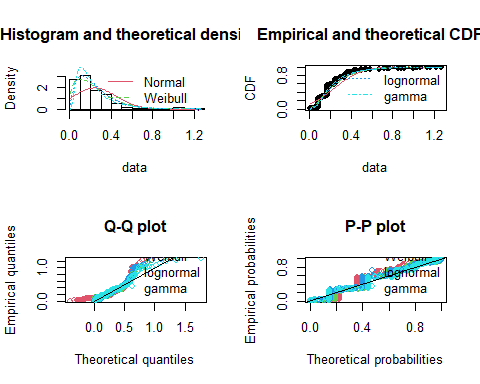
plotdist(df\_final$admin\_time, histo = TRUE, demp=TRUE)



Kijkende naar deze histogrammen, de empirical density en de CDF kunnen we wel stellen dat beide activiteiten niet normaal zijn verdeeld. Laten we een aantal verdelingen proberen en op zoek gaan naar degene met de beste fit. We bekijken de normale, weibull, gamma en de lognormale verdelingen. We doen dit eerst voor checking time.

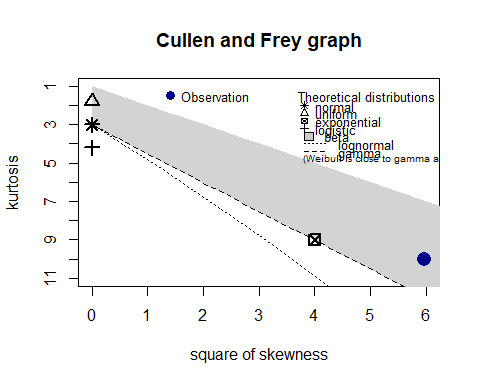
# Fit some other distributions  
fit\_n <- fitdist(df\_final$check\_time, "norm")  
fit\_w <- fitdist(df\_final$check\_time, "weibull")  
fit\_g <- fitdist(df\_final$check\_time, "gamma")  
fit\_ln <- fitdist(df\_final$check\_time, "lnorm")

plot.legend <- c("Normal", "Weibull", "lognormal", "gamma")  
par(mfrow=c(2,2))  
denscomp(list(fit\_n, fit\_w, fit\_g, fit\_ln), legendtext = plot.legend)  
cdfcomp (list(fit\_n, fit\_w, fit\_g, fit\_ln), legendtext = plot.legend)  
qqcomp (list(fit\_n, fit\_w, fit\_g, fit\_ln), legendtext = plot.legend)  
ppcomp (list(fit\_n, fit\_w, fit\_g, fit\_ln), legendtext = plot.legend)



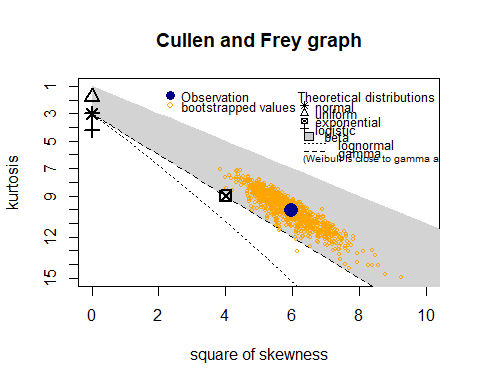
Aan de hand van deze grafieken kunnen we zien dat de normale verdeling geen hele goede fit heeft. Als we kijken naar de CDF lijken de Gamma, Weibull en Lognormale verdeling het beste te passen. We checken vervolgens de Cullen and Frey graphs voor checking time, wellicht dat we visueel kunnen afleiden wat de beste verdeling is.

descdist(df\_final$check\_time)



## summary statistics  
## ------  
## min: 0.006666667 max: 1.266667   
## median: 0.1666667   
## mean: 0.2429038   
## estimated sd: 0.2073224   
## estimated skewness: 2.44264   
## estimated kurtosis: 10.03937

descdist(df\_final$check\_time, boot = 1000)



## summary statistics  
## ------  
## min: 0.006666667 max: 1.266667   
## median: 0.1666667   
## mean: 0.2429038   
## estimated sd: 0.2073224   
## estimated skewness: 2.44264   
## estimated kurtosis: 10.03937

Om een definitieve keuze te maken over de verdeling kunnen we kijken naar de AIC (Akaike Information Criterion). De laagste waaarde heeft de beste fit.

print(c("AIC normal =",fit\_n$aic))

## [1] "AIC normal =" "-127.743494529332"

print(c("AIC weibull =",fit\_w$aic))

## [1] "AIC weibull =" "-409.5362485545"

print(c("AIC gamma =",fit\_g$aic))

## [1] "AIC gamma =" "-442.369612630227"

print(c("AIC lnorm =",fit\_ln$aic))

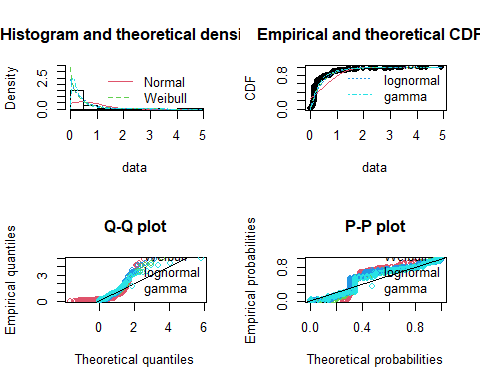
## [1] "AIC lnorm =" "-476.608877716932"

De lognormale verdeling heeft voor checking time de laagste AIC en dus de beste fit. Deze zullen we in het volgende onderdeel gebruiken.

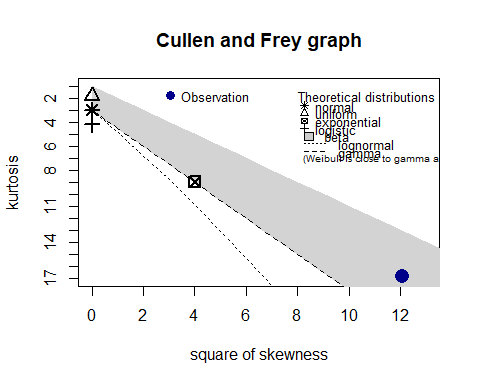
In dit onderdeel doen we hetzelfde voor admin time. We gebruiken wederom dezelfde verdelingen als hiervoor.

# Fit some other distributions  
fit\_nn <- fitdist(df\_final$admin\_time, "norm")  
fit\_ww <- fitdist(df\_final$admin\_time, "weibull")  
fit\_gg <- fitdist(df\_final$admin\_time, "gamma")  
fit\_lnn <- fitdist(df\_final$admin\_time, "lnorm")

plot.legend <- c("Normal", "Weibull", "lognormal", "gamma")  
par(mfrow=c(2,2))  
denscomp(list(fit\_nn, fit\_ww, fit\_gg, fit\_lnn), legendtext = plot.legend)  
cdfcomp (list(fit\_nn, fit\_ww, fit\_gg, fit\_lnn), legendtext = plot.legend)  
qqcomp (list(fit\_nn, fit\_ww, fit\_gg, fit\_lnn), legendtext = plot.legend)  
ppcomp (list(fit\_nn, fit\_ww, fit\_gg, fit\_lnn), legendtext = plot.legend)

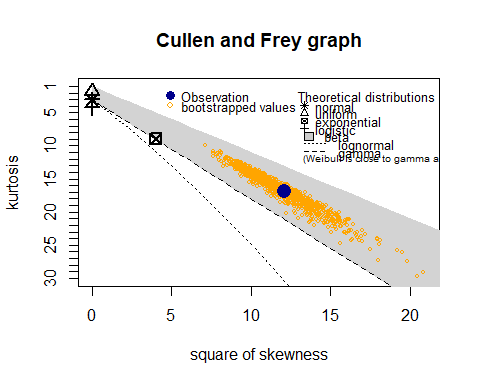
 Wederom zien we dat de normale verdeling niet geschikt is voor deze data. De Gamma, Weibull en lognormale verdeling komen beter in de buurt. Laten we kijken naar de Cullen en Frey graphs.

descdist(df\_final$admin\_time)



## summary statistics  
## ------  
## min: 0.002777778 max: 4.905278   
## median: 0.1666667   
## mean: 0.4607978   
## estimated sd: 0.6896614   
## estimated skewness: 3.475479   
## estimated kurtosis: 16.86317

descdist(df\_final$admin\_time, boot = 1000)



## summary statistics  
## ------  
## min: 0.002777778 max: 4.905278   
## median: 0.1666667   
## mean: 0.4607978   
## estimated sd: 0.6896614   
## estimated skewness: 3.475479   
## estimated kurtosis: 16.86317

Voor checking time is de Cullen and Frey graph lastiger te interpreteren. Laten we dus wederom een blik werpen op de verschillende AIC waarden en op basis daarvan een keuze maken.

print(c("AIC normal =",fit\_nn$aic))

## [1] "AIC normal =" "889.085696707452"

print(c("AIC weibull =",fit\_ww$aic))

## [1] "AIC weibull =" "184.350780302785"

print(c("AIC gamma =",fit\_gg$aic))

## [1] "AIC gamma =" "194.320463282346"

print(c("AIC lnorm =",fit\_lnn$aic))

## [1] "AIC lnorm =" "69.8478100541442"

De lognormale verdeling heeft de laagste AIC en dus de beste fit. Dit zullen we gebruiken in de volgende vraag waarin we de simulatie gaan doen. De parameters zijn als volgt:

fit\_lnn

## Fitting of the distribution ' lnorm ' by maximum likelihood   
## Parameters:  
## estimate Std. Error  
## meanlog -1.369741 0.05003415  
## sdlog 1.029051 0.03537934

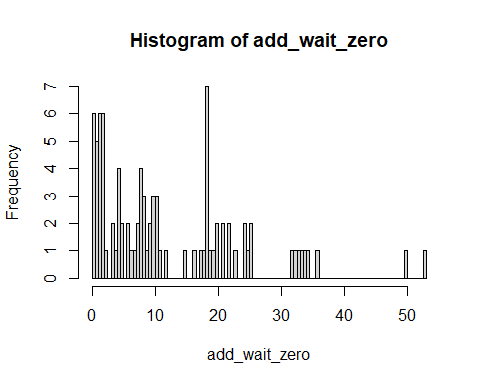
fit\_ln

## Fitting of the distribution ' lnorm ' by maximum likelihood   
## Parameters:  
## estimate Std. Error  
## meanlog -1.6888910 0.03608676  
## sdlog 0.7421951 0.02551699

# 8. (3 points) Include the additional waiting time, as described above, in the simulation script. Then perform at least 100 simulation runs. How is the proportion of parcels sent out in time affected?

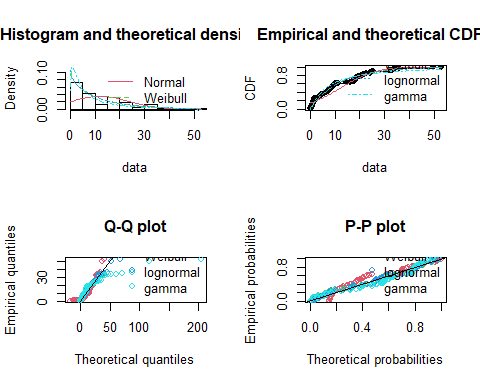
Voordat we kunnen simuleren moeten we de ‘additional’ waiting time berekenen. Hiervoor hebben we de daadwerkelijke wachttijd en de empirische wachttijd voor nodig. De empirische wachttijd wordt opgehaald uit de volgende simulatie:

# Opslaan resulaten empirische simulatie  
result\_emp <- env\_emp %>%  
 get\_mon\_arrivals(per\_resource = TRUE) %>%  
 transform(waiting\_time = (end\_time - start\_time -activity\_time),  
 activity\_time = activity\_time)  
actual\_wait <- df\_final$check\_time\_waiting[1:91]  
sim\_wait <- result\_emp %>% dplyr::filter(resource == "express worker") %>% .$waiting\_time %>% .[1:91]  
add\_wait <- actual\_wait - sim\_wait  
# Verwijderen van 0 waarden, waiting time mag geen 0 zijn  
add\_wait\_zero <- ifelse(add\_wait <0, actual\_wait, add\_wait)  
hist(add\_wait\_zero, breaks = 100)



# Fit some other distributions  
fit\_wt\_nn <- fitdist(add\_wait\_zero, "norm")  
fit\_wt\_ww <- fitdist(add\_wait\_zero, "weibull")  
fit\_wt\_gg <- fitdist(add\_wait\_zero, "gamma")  
fit\_wt\_lnn <- fitdist(add\_wait\_zero, "lnorm")

# Plotting  
plot.legend <- c("Normal", "Weibull", "lognormal", "gamma")  
par(mfrow=c(2,2))  
denscomp(list(fit\_wt\_nn, fit\_wt\_ww, fit\_wt\_gg, fit\_wt\_lnn), legendtext = plot.legend)  
cdfcomp (list(fit\_wt\_nn, fit\_wt\_ww, fit\_wt\_gg, fit\_wt\_lnn), legendtext = plot.legend)  
qqcomp (list(fit\_wt\_nn, fit\_wt\_ww, fit\_wt\_gg, fit\_wt\_lnn), legendtext = plot.legend)  
ppcomp (list(fit\_wt\_nn, fit\_wt\_ww, fit\_wt\_gg, fit\_wt\_lnn), legendtext = plot.legend)



# Print AIC's  
print(c("AIC normal =",fit\_wt\_nn$aic))

## [1] "AIC normal =" "706.195700791146"

print(c("AIC weibull =",fit\_wt\_ww$aic))

## [1] "AIC weibull =" "640.652291740917"

print(c("AIC gamma =",fit\_wt\_gg$aic))

## [1] "AIC gamma =" "640.30988570074"

print(c("AIC lnorm =",fit\_wt\_lnn$aic))

## [1] "AIC lnorm =" "655.586098220569"

Nu we een verdeling hebben gekozen, in dit geval de gamma verdeling (laagste AIC), kunnen we gaan simuleren en bepalen wat voor impact deze teoevoeging heeft op het aantal pakketjes wat op tijd wordt verzonden.

# Arrival data gebaseerd op echte waarden.  
df\_sim <- data %>%  
 arrange(time\_stamp) %>%  
 dplyr::select(time\_stamp)  
# opslaan van gamma verdeling  
wait\_shape <- fit\_wt\_gg$estimate[1]  
wait\_rate <- fit\_wt\_gg$estimate[2]  
  
env <- simmer("parcel depot")  
  
# Define process trajectory based on flow chart  
parcel <- trajectory("parcel' path") %>%  
 seize("wait time") %>%  
 timeout(function() rgamma(1, shape =wait\_shape,rate =wait\_rate)) %>%  
 release("wait time", 1) %>%  
 seize("express worker", 1) %>%  
 timeout(function() rlnorm(1, fit\_ln$estimate, fit\_ln$sd)) %>%  
 release("express worker", 1) %>%  
 seize("admin worker", 1) %>% # every parcel seizes 1 of the 2 admins  
 timeout(function() rlnorm(1, fit\_lnn$estimate, fit\_lnn$sd)) %>%  
 release("admin worker", 1)  
  
set.seed(42) #!!!! set seed for reproduction of numbers  
env <- lapply(1:100, function(i) {  
 simmer("parcel depot") %>%  
 add\_resource("wait time", capacity = Inf) %>%  
 add\_resource("express worker", 1) %>%  
 add\_resource("admin worker", 2) %>%  
 add\_dataframe("parcel", parcel, df\_sim, col\_time = "time\_stamp", time="absolute") %>%  
 run(until=10\*11)  
})

# Get throughput of the process based on parcels  
result\_arrival <- env %>%  
 get\_mon\_arrivals() %>%  
 mutate(throughput = (end\_time - start\_time))  
  
sum(result\_arrival$throughput < 22) / length(result\_arrival$throughput) # 22 uur voor de KPI

## [1] 0.8541842

Het percentage pakketjes wat binnen de KPI worden verstuurd ligt nu op 85%, voorheen was dit 92% (zie resultaten vorige opdracht). Het aantal pakketjes wat binnen de afgesproken tijd worden vestuurd, als we ‘additioanl wait time’ mee modelleren, is dus met ongeveer 7 procent gedaald.

# 9. (2 points) Perform a sensitivity analysis on the number of express workers and the number of admin workers. Explain how these numbers affect the KPI target. Assume an express worker costs EUR 30 per hour and an admin worker costs EUR 35 per hour. How can the KPI target be achieved with the lowest possible costs?

expres\_workers <- c(1,2,3,4,5,6,7,8,9,10)  
admin\_workers <- c(1,2,3,4,5,6,7,8,9,10)  
  
sensitivity\_results <- tibble(express = numeric(),  
 admin = numeric(),  
 ontime = numeric())  
  
for (a in expres\_workers) {  
 for (i in admin\_workers) {  
 print("calculating scenario for ")  
 print(a)  
 expres\_workers <- a  
 print("Express and ")  
 admin\_workers <- i  
 print(i)  
 print("admin workers ")  
  
 env <- simmer("parcel depot")  
   
 # Define process trajectory based on flow chart  
 parcel <- trajectory("parcel' path") %>%  
 seize("wait time") %>%  
 timeout(function() rgamma(1, shape =wait\_shape,rate =wait\_rate)) %>%  
 release("wait time", 1) %>%  
 seize("express worker", 1) %>%  
 timeout(function() rlnorm(1, fit\_ln$estimate, fit\_ln$sd)) %>%  
 release("express worker", 1) %>%  
 seize("admin worker", 1) %>% # every parcel seizes 1 of the 2 admins  
 timeout(function() rlnorm(1, fit\_lnn$estimate, fit\_lnn$sd)) %>%  
 release("admin worker", 1)  
   
 set.seed(42) #!!!! set seed for reproduction of numbers  
 env <- lapply(1:100, function(i) {  
 simmer("parcel depot") %>%  
 add\_resource("wait time", capacity = Inf) %>%  
 add\_resource("express worker", expres\_workers) %>%  
 add\_resource("admin worker", admin\_workers) %>%  
 add\_dataframe("parcel", parcel, df\_sim, col\_time = "time\_stamp", time="absolute") %>%  
 run(until=10\*11)  
 })  
  
 # Get throughput of the process based on parcels  
 result\_arrival <- env %>%  
 get\_mon\_arrivals() %>%  
 mutate(throughput = (end\_time - start\_time))  
  
 on\_time <- sum(result\_arrival$throughput < 22) / length(result\_arrival$throughput)  
 print(on\_time)  
 sensitivity\_results <- sensitivity\_results %>%  
 add\_row(express =expres\_workers,  
 admin = admin\_workers,  
 ontime = on\_time)  
  
 }  
}

## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 1  
## [1] "admin workers "  
## [1] 0.8609159  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 2  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 3  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 4  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 5  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 6  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 7  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 8  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 9  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 1  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8541842  
## [1] "calculating scenario for "  
## [1] 2  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8572291  
## [1] "calculating scenario for "  
## [1] 3  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8556751  
## [1] "calculating scenario for "  
## [1] 4  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 5  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 6  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 7  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 8  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 9  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448  
## [1] "calculating scenario for "  
## [1] 10  
## [1] "Express and "  
## [1] 10  
## [1] "admin workers "  
## [1] 0.8557448

sensitivity\_results <- sensitivity\_results %>%  
 mutate(express\_costs = express \* 30,  
 admin\_costs = admin \* 35,  
 total\_costs = express\_costs + admin\_costs)  
sensitivity\_results

## # A tibble: 19 × 6  
## express admin ontime express\_costs admin\_costs total\_costs  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 1 0.861 30 35 65  
## 2 1 2 0.854 30 70 100  
## 3 1 3 0.854 30 105 135  
## 4 1 4 0.854 30 140 170  
## 5 1 5 0.854 30 175 205  
## 6 1 6 0.854 30 210 240  
## 7 1 7 0.854 30 245 275  
## 8 1 8 0.854 30 280 310  
## 9 1 9 0.854 30 315 345  
## 10 1 10 0.854 30 350 380  
## 11 2 10 0.857 60 350 410  
## 12 3 10 0.856 90 350 440  
## 13 4 10 0.856 120 350 470  
## 14 5 10 0.856 150 350 500  
## 15 6 10 0.856 180 350 530  
## 16 7 10 0.856 210 350 560  
## 17 8 10 0.856 240 350 590  
## 18 9 10 0.856 270 350 620  
## 19 10 10 0.856 300 350 650

In bovenstaand figuur worden de resultaten van de sensitivity analyse gevisualizeerd. Het is me niet gelukt om een combinatie express worker en admin worker te vinden waarbij de KPI wordt behaald. Nadat ik hier tevergeefs heel wat uren in heb gestop om het probleem te vinden heb ik besloten om deze resultaten toch te uploaden. Hoogst haalbare percentage is 85%.