Eindoprdracht deel 2

Mohammed Al Hor

2023-02-12

# 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(path = '~/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),  
 total\_throughput = working\_hours(`Eind lossen`, `Eind admin`, saturday = FALSE))  
  
summary(data$admin\_time)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -10.3500 0.1517 0.1667 0.7985 0.3561 31.2167

summary(data$check\_time)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -10.71667 0.08333 0.16667 0.29957 0.29167 7.37778

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] 0.360472

mean\_admin\_time

## [1] 0.8573892

# 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),  
 total\_throughput = working\_hours(`Eind lossen`, `Eind admin`, saturday = FALSE))  
  
summary(data$admin\_time)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -10.3500 0.1517 0.1667 0.7985 0.3561 31.2167

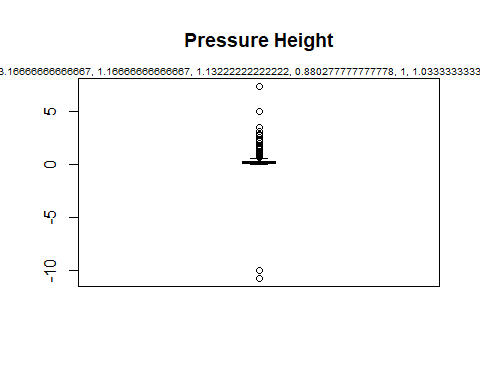
summary(data$check\_time)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -10.71667 0.08333 0.16667 0.29957 0.29167 7.37778

# In deze stap wordt de mean berekend van beide activiteiten, deze wordt later gebruikt om te imputen in de data  
mean\_check\_time <- mean(data$check\_time[data$check\_time > 0])  
mean\_admin\_time <- mean(data$admin\_time[data$admin\_time > 0])

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)

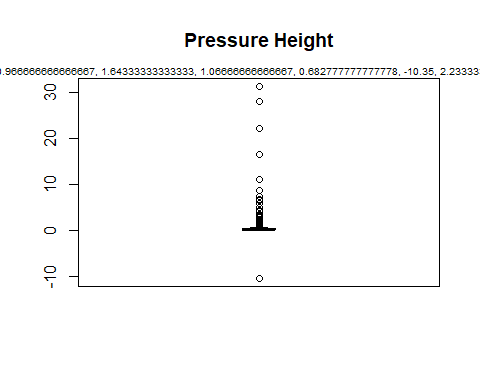


outlier(data$check\_time)

## [1] -10.71667

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)

# De uiteindelijke gemiddelden worden als volgt berekend.  
mean(df\_final$check\_time)

## [1] 0.2697075

mean(df\_final$admin\_time)

## [1] 0.4632331

Als laatste behandelen we de vraag of de KPI van 90% is behaald. Hiervoor berekenen we de totale throughput van de pakketjes (check\_time + admin\_time) en berekenen we de fractie van pakketjes dat binnen de tijd zijn behandeld.

shipping\_time <- 2\*11\*60 # parcels should be shipped within 2 days, 11 working hours in a day, 60 minutes in an hour.  
shipped\_in\_time <- df\_final %>% filter(total\_throughput<= shipping\_time)  
# To calculate the amount of parcels shipped in time we do the following  
in\_time <- nrow(shipped\_in\_time)/nrow(df\_final)\*100  
in\_time

## [1] 100

Alle pakketjes zijn op tijd en de KPI is dus behaald. Dit is wel berekend op basis van data waar de outliers zijn vervangen door het gemiddelde.

# 4. (2 points) Determine the utilisation (= fraction of time a worker is busy) of the express workers (between 07h00 and 18h00). Do the same for the admin workers. Om de fractie te berekenen dat een medewerker bezig is met een pakketje berekenen we eerst de totale werktijd. Dit is 6 dagen per week, 11 uur per dag. De periode over de hele dataset is van 3 oktober 2016 tot 30 november 2016. Dit zijn 59 werkdagen.

total\_working\_hours <- 59\*11

# De totale tijd dat iedere werknemer bezig is hebben we al berekend. Laten we eerst de berekening doen voor express workers.  
utilization\_check <- sum(df\_final$check\_time)/total\_working\_hours  
  
# Er is maar 1 express worker, maar er zijn 2 admin workers. De beschikbare werktijd moeten we dus verdubbelen.  
utilization\_admin <- sum(df\_final$admin\_time)/(total\_working\_hours\*2)  
  
utilization\_check

## [1] 0.1757878

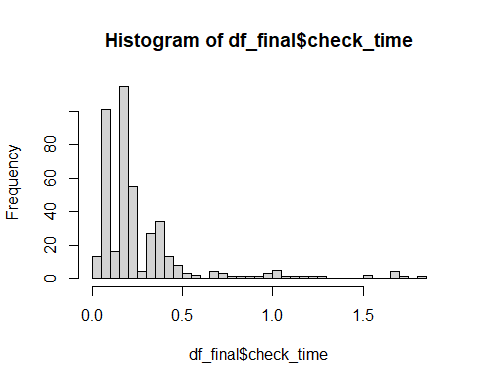
utilization\_admin

## [1] 0.1509612

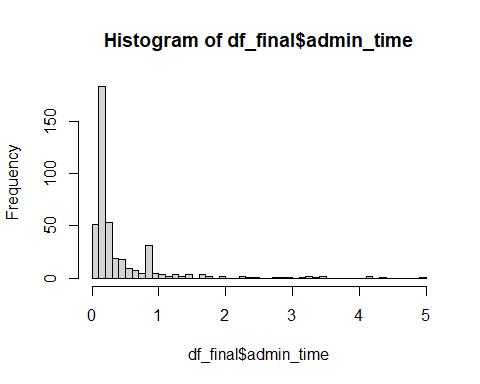
# 5. (3 points) Determine for each phase (checking and admin) the best fitting distribution (including the fitted parameters) and explain your choice.

Laten we eerst naar wat histogrammen kijken.

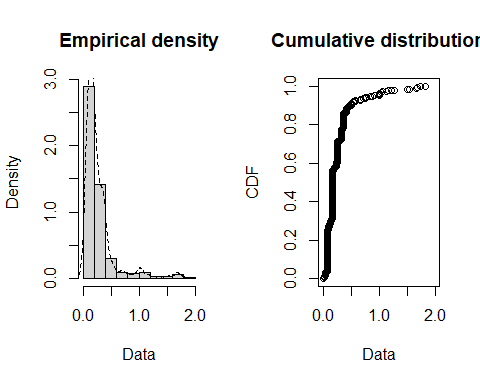
#Laten we eerst kijken naar wat histogrammen  
hist(df\_final$check\_time, breaks = 50)



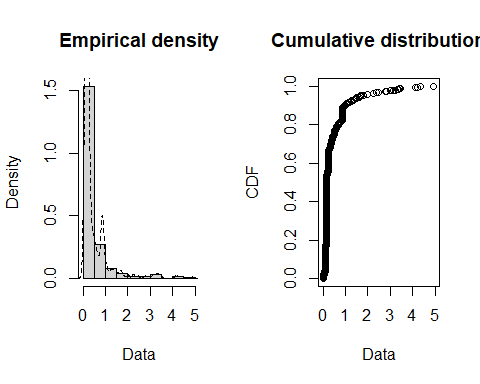
hist(df\_final$admin\_time, breaks = 50)



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



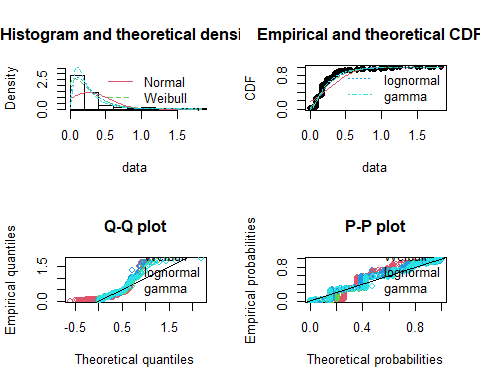
plotdist(df\_final$admin\_time, histo = TRUE, demp=TRUE)



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.

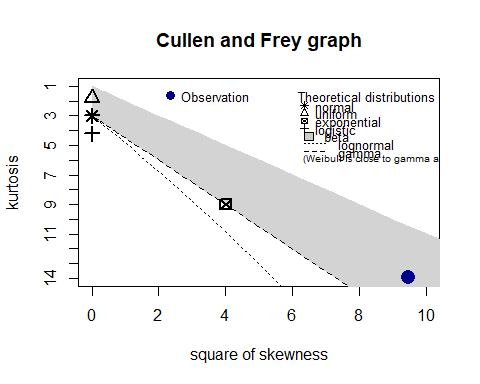
# Fit some other distributions  
fit\_n <- fitdist(df\_final$check\_time, "norm")  
fit\_w <- fitdist(df\_final$check\_time, "weibull")  
fit\_g <- fitdist(df\_final$check\_time, "gamma")  
fit\_ln <- fitdist(df\_final$check\_time, "lnorm")

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)



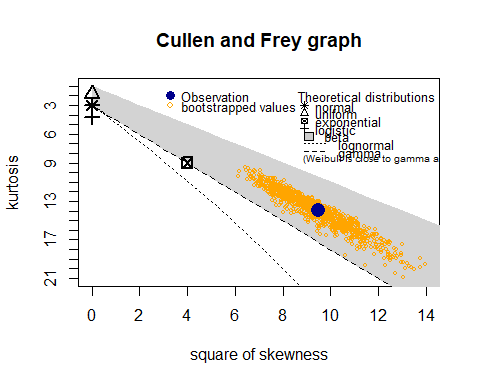
Aan de hand van deze grafieken kunnen we zien dat de normale verdeling geen hele goede fit heeft. Als we kijken naar de CDF lijken de Gamma, Weibull en Lognormale verdeling het beste te passen. We checken vervolgens de Cullen and Frey graphs voor checking time, wellicht dat we visueel kunnen afleiden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

descdist(df\_final$check\_time)



## summary statistics  
## ------  
## min: 0.006666667 max: 1.833611   
## median: 0.1666667   
## mean: 0.2697075   
## estimated sd: 0.2844704   
## estimated skewness: 3.076956   
## estimated kurtosis: 13.95203

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



## summary statistics  
## ------  
## min: 0.006666667 max: 1.833611   
## median: 0.1666667   
## mean: 0.2697075   
## estimated sd: 0.2844704   
## estimated skewness: 3.076956   
## estimated kurtosis: 13.95203

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 =" "139.892024143688"

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

## [1] "AIC weibull =" "-279.434164829018"

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

## [1] "AIC gamma =" "-308.284623453416"

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

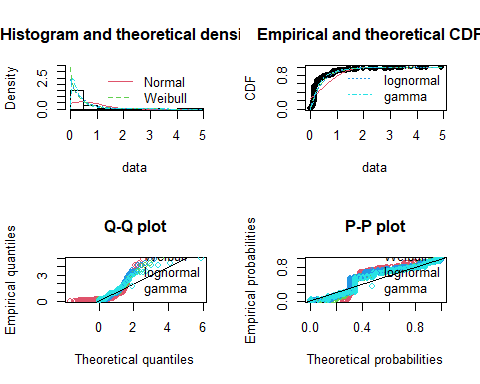
## [1] "AIC lnorm =" "-383.281910908791"

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

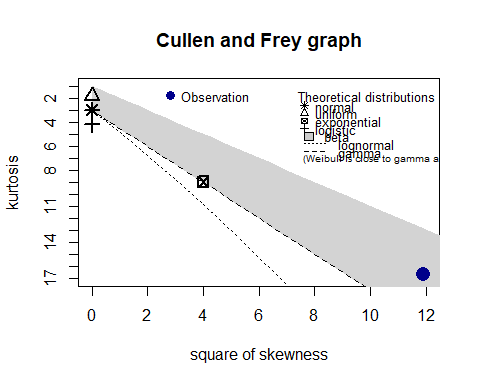
In dit onderdeel doen we hetzelfde voor admin time. We gebruiken wederom dezelfde verdelingen als hiervoor.

# 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)

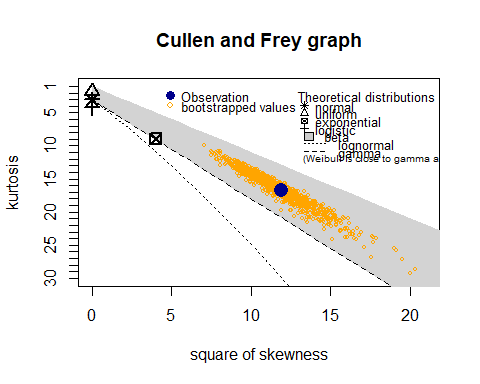
 Wederom zien we dat de normale verdeling niet geschikt is voor deze data. De Gamma, Weibull en lognormale verdeling komen beter in de buurt. Laten we kijken naar de Cullen en Frey graphs.

descdist(df\_final$admin\_time)



## summary statistics  
## ------  
## min: 0.002777778 max: 4.905278   
## median: 0.1666667   
## mean: 0.4632331   
## estimated sd: 0.6909581   
## estimated skewness: 3.44835   
## estimated kurtosis: 16.6898

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



## summary statistics  
## ------  
## min: 0.002777778 max: 4.905278   
## median: 0.1666667   
## mean: 0.4632331   
## estimated sd: 0.6909581   
## estimated skewness: 3.44835   
## estimated kurtosis: 16.6898

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 =" "890.674878111596"

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

## [1] "AIC weibull =" "188.612087600184"

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

## [1] "AIC gamma =" "198.723886635092"

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

## [1] "AIC lnorm =" "75.09700750941"

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 -1.366801 0.05019776  
## sdlog 1.032416 0.03549503

fit\_ln

## Fitting of the distribution ' lnorm ' by maximum likelihood   
## Parameters:  
## estimate Std. Error  
## meanlog -1.6535597 0.03889674  
## sdlog 0.7999877 0.02750395

# 6. Replace the statistical distributions in the simmer script with the fitted distributions from the previous question. For the arrivals use the exact ‘Eind Lossen’ time stamps. Run the simulation for 10 working days, and repeat 100 times. Recompute the performance measures from question 1.

env <- simmer("parcel depot")  
env

## simmer environment: parcel depot | now: 0 | next:   
## { Monitor: in memory }

lambda = 6  
rate = 1/lambda  
parcel <- trajectory("parcel's path") %>%  
 seize("express worker", 1) %>% # checking in of parcels by express worker  
 timeout(function() rlnorm(1, fit\_ln$estimate, fit\_ln$sd)) %>% # estimate and sd from fit\_ln we previously estimated  
 release("express worker", 1) %>%  
   
 seize("admin worker", 1) %>% # preparing the parcel for shipping by admin worker  
 timeout(function() rlnorm(1, fit\_lnn$estimate, fit\_lnn$sd)) %>% # mean of 8 with sd of 2  
 release("admin worker", 1)  
  
env %>%  
 add\_resource("express worker", 1) %>%  
 add\_resource("admin worker", 2) %>%  
 add\_generator("parcel", parcel, function() rexp(1, 1/6))

## simmer environment: parcel depot | now: 0 | next: 0  
## { Monitor: in memory }  
## { Resource: express worker | monitored: TRUE | server status: 0(1) | queue status: 0(Inf) }  
## { Resource: admin worker | monitored: TRUE | server status: 0(2) | queue status: 0(Inf) }  
## { Source: parcel | monitored: 1 | n\_generated: 0 }

# Dit is wat ik zelf heb geprobeerd, kunnen jullie me hier terugkoppeling over geven?   
# add\_dataframe('arrivals', parcel, data = df\_final, col\_time = 'Eind lossen', time = 'interarrival')  
for (i in 1:100){  
 env %>% run(until=110) # we run the simulation for 110 hours (10 working days)  
}  
  
env %>% get\_mon\_arrivals()

## name start\_time end\_time activity\_time finished replication  
## 1 parcel0 10.74990 11.20179 0.4518932 TRUE 1  
## 2 parcel1 13.90852 14.34330 0.4347829 TRUE 1  
## 3 parcel2 35.68269 36.12182 0.4391313 TRUE 1  
## 4 parcel3 35.78812 36.31798 0.4403992 TRUE 1  
## 5 parcel4 40.24769 40.71291 0.4652174 TRUE 1  
## 6 parcel5 47.63245 48.07597 0.4435174 TRUE 1  
## 7 parcel6 59.20513 59.64679 0.4416627 TRUE 1  
## 8 parcel7 64.13705 64.59788 0.4608326 TRUE 1  
## 9 parcel8 67.13658 67.60749 0.4709163 TRUE 1  
## 10 parcel9 67.38082 67.82813 0.4473080 TRUE 1  
## 11 parcel10 75.86846 76.31146 0.4429970 TRUE 1  
## 12 parcel11 77.69810 78.14789 0.4497969 TRUE 1  
## 13 parcel12 78.35268 78.81385 0.4611758 TRUE 1  
## 14 parcel13 85.48509 85.95426 0.4691620 TRUE 1  
## 15 parcel14 86.39741 86.84993 0.4525203 TRUE 1  
## 16 parcel15 86.98749 87.41902 0.4315318 TRUE 1  
## 17 parcel16 98.04202 98.49040 0.4483835 TRUE 1

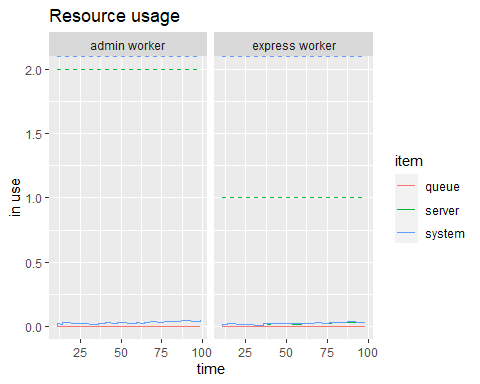
env %>% get\_mon\_resources()

## resource time server queue capacity queue\_size system limit  
## 1 express worker 10.74990 1 0 1 Inf 1 Inf  
## 2 express worker 10.95455 0 0 1 Inf 0 Inf  
## 3 admin worker 10.95455 1 0 2 Inf 1 Inf  
## 4 admin worker 11.20179 0 0 2 Inf 0 Inf  
## 5 express worker 13.90852 1 0 1 Inf 1 Inf  
## 6 express worker 14.09449 0 0 1 Inf 0 Inf  
## 7 admin worker 14.09449 1 0 2 Inf 1 Inf  
## 8 admin worker 14.34330 0 0 2 Inf 0 Inf  
## 9 express worker 35.68269 1 0 1 Inf 1 Inf  
## 10 express worker 35.78812 1 1 1 Inf 2 Inf  
## 11 express worker 35.87758 1 0 1 Inf 1 Inf  
## 12 admin worker 35.87758 1 0 2 Inf 1 Inf  
## 13 express worker 36.06138 0 0 1 Inf 0 Inf  
## 14 admin worker 36.06138 2 0 2 Inf 2 Inf  
## 15 admin worker 36.12182 1 0 2 Inf 1 Inf  
## 16 admin worker 36.31798 0 0 2 Inf 0 Inf  
## 17 express worker 40.24769 1 0 1 Inf 1 Inf  
## 18 express worker 40.44247 0 0 1 Inf 0 Inf  
## 19 admin worker 40.44247 1 0 2 Inf 1 Inf  
## 20 admin worker 40.71291 0 0 2 Inf 0 Inf  
## 21 express worker 47.63245 1 0 1 Inf 1 Inf  
## 22 express worker 47.81847 0 0 1 Inf 0 Inf  
## 23 admin worker 47.81847 1 0 2 Inf 1 Inf  
## 24 admin worker 48.07597 0 0 2 Inf 0 Inf  
## 25 express worker 59.20513 1 0 1 Inf 1 Inf  
## 26 express worker 59.39836 0 0 1 Inf 0 Inf  
## 27 admin worker 59.39836 1 0 2 Inf 1 Inf  
## 28 admin worker 59.64679 0 0 2 Inf 0 Inf  
## 29 express worker 64.13705 1 0 1 Inf 1 Inf  
## 30 express worker 64.32495 0 0 1 Inf 0 Inf  
## 31 admin worker 64.32495 1 0 2 Inf 1 Inf  
## 32 admin worker 64.59788 0 0 2 Inf 0 Inf  
## 33 express worker 67.13658 1 0 1 Inf 1 Inf  
## 34 express worker 67.33765 0 0 1 Inf 0 Inf  
## 35 admin worker 67.33765 1 0 2 Inf 1 Inf  
## 36 express worker 67.38082 1 0 1 Inf 1 Inf  
## 37 express worker 67.57474 0 0 1 Inf 0 Inf  
## 38 admin worker 67.57474 2 0 2 Inf 2 Inf  
## 39 admin worker 67.60749 1 0 2 Inf 1 Inf  
## 40 admin worker 67.82813 0 0 2 Inf 0 Inf  
## 41 express worker 75.86846 1 0 1 Inf 1 Inf  
## 42 express worker 76.05090 0 0 1 Inf 0 Inf  
## 43 admin worker 76.05090 1 0 2 Inf 1 Inf  
## 44 admin worker 76.31146 0 0 2 Inf 0 Inf  
## 45 express worker 77.69810 1 0 1 Inf 1 Inf  
## 46 express worker 77.88414 0 0 1 Inf 0 Inf  
## 47 admin worker 77.88414 1 0 2 Inf 1 Inf  
## 48 admin worker 78.14789 0 0 2 Inf 0 Inf  
## 49 express worker 78.35268 1 0 1 Inf 1 Inf  
## 50 express worker 78.55068 0 0 1 Inf 0 Inf  
## 51 admin worker 78.55068 1 0 2 Inf 1 Inf  
## 52 admin worker 78.81385 0 0 2 Inf 0 Inf  
## 53 express worker 85.48509 1 0 1 Inf 1 Inf  
## 54 express worker 85.67036 0 0 1 Inf 0 Inf  
## 55 admin worker 85.67036 1 0 2 Inf 1 Inf  
## 56 admin worker 85.95426 0 0 2 Inf 0 Inf  
## 57 express worker 86.39741 1 0 1 Inf 1 Inf  
## 58 express worker 86.58175 0 0 1 Inf 0 Inf  
## 59 admin worker 86.58175 1 0 2 Inf 1 Inf  
## 60 admin worker 86.84993 0 0 2 Inf 0 Inf  
## 61 express worker 86.98749 1 0 1 Inf 1 Inf  
## 62 express worker 87.17490 0 0 1 Inf 0 Inf  
## 63 admin worker 87.17490 1 0 2 Inf 1 Inf  
## 64 admin worker 87.41902 0 0 2 Inf 0 Inf  
## 65 express worker 98.04202 1 0 1 Inf 1 Inf  
## 66 express worker 98.24508 0 0 1 Inf 0 Inf  
## 67 admin worker 98.24508 1 0 2 Inf 1 Inf  
## 68 admin worker 98.49040 0 0 2 Inf 0 Inf  
## replication  
## 1 1  
## 2 1  
## 3 1  
## 4 1  
## 5 1  
## 6 1  
## 7 1  
## 8 1  
## 9 1  
## 10 1  
## 11 1  
## 12 1  
## 13 1  
## 14 1  
## 15 1  
## 16 1  
## 17 1  
## 18 1  
## 19 1  
## 20 1  
## 21 1  
## 22 1  
## 23 1  
## 24 1  
## 25 1  
## 26 1  
## 27 1  
## 28 1  
## 29 1  
## 30 1  
## 31 1  
## 32 1  
## 33 1  
## 34 1  
## 35 1  
## 36 1  
## 37 1  
## 38 1  
## 39 1  
## 40 1  
## 41 1  
## 42 1  
## 43 1  
## 44 1  
## 45 1  
## 46 1  
## 47 1  
## 48 1  
## 49 1  
## 50 1  
## 51 1  
## 52 1  
## 53 1  
## 54 1  
## 55 1  
## 56 1  
## 57 1  
## 58 1  
## 59 1  
## 60 1  
## 61 1  
## 62 1  
## 63 1  
## 64 1  
## 65 1  
## 66 1  
## 67 1  
## 68 1

summary(get\_mon\_resources(env))

## resource time server queue   
## Length:68 Min. :10.75 Min. :0.0000 Min. :0.00000   
## Class :character 1st Qu.:40.39 1st Qu.:0.0000 1st Qu.:0.00000   
## Mode :character Median :67.34 Median :1.0000 Median :0.00000   
## Mean :60.84 Mean :0.5735 Mean :0.01471   
## 3rd Qu.:78.62 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :98.49 Max. :2.0000 Max. :1.00000   
## capacity queue\_size system limit replication  
## Min. :1.0 Min. :Inf Min. :0.0000 Min. :Inf Min. :1   
## 1st Qu.:1.0 1st Qu.:Inf 1st Qu.:0.0000 1st Qu.:Inf 1st Qu.:1   
## Median :1.5 Median :Inf Median :1.0000 Median :Inf Median :1   
## Mean :1.5 Mean :Inf Mean :0.5882 Mean :Inf Mean :1   
## 3rd Qu.:2.0 3rd Qu.:Inf 3rd Qu.:1.0000 3rd Qu.:Inf 3rd Qu.:1   
## Max. :2.0 Max. :Inf Max. :2.0000 Max. :Inf Max. :1

plot(get\_mon\_resources(env))



Het is me helaas niet gelukt om met echte datums te werken voor de tijdsintervallen, zou hier graag feedback op willen ontvangen. Online kon ik moeilijk documentatie vinden voor het simmer package. Desalniettemin, gaan we wat performance metrics uit de vorige vragen berekenen. Laten we eerst kijken naar de checker.

env\_per\_user <- get\_mon\_arrivals(env, per\_resource = TRUE)  
# We keep dividing by 60 to get the output in hours  
# Mean and standard deviation of checking time  
  
# filter for only express workers  
express\_worker <- env\_per\_user %>% filter(resource == 'express worker')  
#calculate the mean and sd:  
print(c("Mean activity time =",mean(express\_worker$activity\_time)))

## [1] "Mean activity time =" "0.191341860888656"

print(c("SD activity time =",sd(express\_worker$activity\_time)))

## [1] "SD activity time =" "0.0071714689874333"

# calculate the wait time for express worker and then the mean and sd  
express\_worker\_wait <- express\_worker$end\_time - express\_worker$start\_time - express\_worker$activity\_time  
mean(express\_worker\_wait)

## [1] 0.005262387

print(c("Mean wait time =",mean(express\_worker\_wait)))

## [1] "Mean wait time =" "0.00526238685685938"

print(c("SD wait time =",sd(express\_worker\_wait)))

## [1] "SD wait time =" "0.0216973768536922"

Nu kijken we naar administratie werkers.

# filter for only admin workers  
admin\_worker <- env\_per\_user %>% filter(resource == 'admin worker')  
  
#calculate the mean and sd:  
print(c("Mean activity time =",mean(admin\_worker$activity\_time)))

## [1] "Mean activity time =" "0.258730400293704"

print(c("SD activity time =",sd(admin\_worker$activity\_time)))

## [1] "SD activity time =" "0.0117450484660652"

# calculate the wait time for admin workers and then the mean and sd  
admin\_worker\_wait <- admin\_worker$end\_time - admin\_worker$start\_time - admin\_worker$activity\_time  
print(c("Mean wait time =",mean(express\_worker\_wait)))

## [1] "Mean wait time =" "0.00526238685685938"

print(c("SD wait time =",sd(express\_worker\_wait)))

## [1] "SD wait time =" "0.0216973768536922"

Als laatste nog even naar de totale throughput van de pakketjes kijken.

# we calculate the total throughput of a parcel by first taking the end time of administration and subtracting the start time of checking.  
total\_throughput <- env\_per\_user %>% group\_by(name) %>%   
 summarise(min\_start\_time = min(start\_time), max\_end\_time = max(end\_time)) %>%  
 mutate(throughput = max\_end\_time - min\_start\_time)  
  
print(c("Mean total throughput =",mean(total\_throughput$throughput)))

## [1] "Mean total throughput =" "0.455334648039219"

print(c("SD total throughput =",sd(total\_throughput$throughput)))

## [1] "SD total throughput =" "0.022421323039129"

# 7. Now replace the statistical distributions in the simmer script with the empirical distributions. How do the different simulations compare?

env <- simmer("parcel depot")  
env

## simmer environment: parcel depot | now: 0 | next:   
## { Monitor: in memory }

parcel <- trajectory("parcel's path") %>%  
 seize("express worker", 1) %>% # checking in of parcels by express worker  
 timeout(function() rnorm(1, mean(df\_final$check\_time), sd(df\_final$check\_time))) %>% # empirical distributions  
 release("express worker", 1) %>%  
   
 seize("admin worker", 1) %>% # preparing the parcel for shipping by admin worker  
 timeout(rnorm(1, mean(df\_final$admin\_time), sd(df\_final$admin\_time))) %>% # mean of 8 with sd of 2  
 release("admin worker", 1)  
  
env %>%  
 add\_resource("express worker", 1) %>%  
 add\_resource("admin worker", 2) %>%  
 add\_generator("parcel", parcel, function() rexp(1, 1/6))

## simmer environment: parcel depot | now: 0 | next: 0  
## { Monitor: in memory }  
## { Resource: express worker | monitored: TRUE | server status: 0(1) | queue status: 0(Inf) }  
## { Resource: admin worker | monitored: TRUE | server status: 0(2) | queue status: 0(Inf) }  
## { Source: parcel | monitored: 1 | n\_generated: 0 }

# Dit is wat ik zelf heb geprobeerd, kunnen jullie me hier terugkoppeling over geven?   
# add\_dataframe('arrivals', parcel, data = df\_final, col\_time = 'Eind lossen', time = 'interarrival')  
for (i in 1:100){  
 env %>% run(until=110) # we run the simulation for 110 hours (10 working days)  
}  
  
env %>% get\_mon\_arrivals()

## name start\_time end\_time activity\_time finished replication  
## 1 parcel0 3.466266 4.254623 0.7883566 TRUE 1  
## 2 parcel1 6.863200 7.211695 0.3484944 TRUE 1  
## 3 parcel2 14.220190 14.796135 0.5759450 TRUE 1  
## 4 parcel3 37.868626 38.342329 0.4737034 TRUE 1  
## 5 parcel4 41.262376 41.973259 0.7108827 TRUE 1  
## 6 parcel5 59.181329 59.776297 0.5949682 TRUE 1  
## 7 parcel6 81.766189 82.475286 0.7090974 TRUE 1  
## 8 parcel7 83.566547 84.496000 0.9294533 TRUE 1  
## 9 parcel8 83.638392 84.544760 0.1983157 TRUE 1  
## 10 parcel9 83.726656 84.797945 0.4027407 TRUE 1  
## 11 parcel10 99.577813 99.878736 0.3009237 TRUE 1

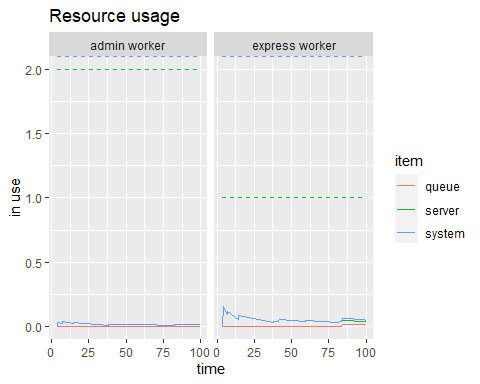
env %>% get\_mon\_resources()

## resource time server queue capacity queue\_size system limit  
## 1 express worker 3.466266 1 0 1 Inf 1 Inf  
## 2 express worker 4.105067 0 0 1 Inf 0 Inf  
## 3 admin worker 4.105067 1 0 2 Inf 1 Inf  
## 4 admin worker 4.254623 0 0 2 Inf 0 Inf  
## 5 express worker 6.863200 1 0 1 Inf 1 Inf  
## 6 express worker 7.062139 0 0 1 Inf 0 Inf  
## 7 admin worker 7.062139 1 0 2 Inf 1 Inf  
## 8 admin worker 7.211695 0 0 2 Inf 0 Inf  
## 9 express worker 14.220190 1 0 1 Inf 1 Inf  
## 10 express worker 14.646579 0 0 1 Inf 0 Inf  
## 11 admin worker 14.646579 1 0 2 Inf 1 Inf  
## 12 admin worker 14.796135 0 0 2 Inf 0 Inf  
## 13 express worker 37.868626 1 0 1 Inf 1 Inf  
## 14 express worker 38.192774 0 0 1 Inf 0 Inf  
## 15 admin worker 38.192774 1 0 2 Inf 1 Inf  
## 16 admin worker 38.342329 0 0 2 Inf 0 Inf  
## 17 express worker 41.262376 1 0 1 Inf 1 Inf  
## 18 express worker 41.823703 0 0 1 Inf 0 Inf  
## 19 admin worker 41.823703 1 0 2 Inf 1 Inf  
## 20 admin worker 41.973259 0 0 2 Inf 0 Inf  
## 21 express worker 59.181329 1 0 1 Inf 1 Inf  
## 22 express worker 59.626741 0 0 1 Inf 0 Inf  
## 23 admin worker 59.626741 1 0 2 Inf 1 Inf  
## 24 admin worker 59.776297 0 0 2 Inf 0 Inf  
## 25 express worker 81.766189 1 0 1 Inf 1 Inf  
## 26 express worker 82.325731 0 0 1 Inf 0 Inf  
## 27 admin worker 82.325731 1 0 2 Inf 1 Inf  
## 28 admin worker 82.475286 0 0 2 Inf 0 Inf  
## 29 express worker 83.566547 1 0 1 Inf 1 Inf  
## 30 express worker 83.638392 1 1 1 Inf 2 Inf  
## 31 express worker 83.726656 1 2 1 Inf 3 Inf  
## 32 express worker 84.346445 1 1 1 Inf 2 Inf  
## 33 admin worker 84.346445 1 0 2 Inf 1 Inf  
## 34 express worker 84.395204 1 0 1 Inf 1 Inf  
## 35 admin worker 84.395204 2 0 2 Inf 2 Inf  
## 36 admin worker 84.496000 1 0 2 Inf 1 Inf  
## 37 admin worker 84.544760 0 0 2 Inf 0 Inf  
## 38 express worker 84.648389 0 0 1 Inf 0 Inf  
## 39 admin worker 84.648389 1 0 2 Inf 1 Inf  
## 40 admin worker 84.797945 0 0 2 Inf 0 Inf  
## 41 express worker 99.577813 1 0 1 Inf 1 Inf  
## 42 express worker 99.729180 0 0 1 Inf 0 Inf  
## 43 admin worker 99.729180 1 0 2 Inf 1 Inf  
## 44 admin worker 99.878736 0 0 2 Inf 0 Inf  
## replication  
## 1 1  
## 2 1  
## 3 1  
## 4 1  
## 5 1  
## 6 1  
## 7 1  
## 8 1  
## 9 1  
## 10 1  
## 11 1  
## 12 1  
## 13 1  
## 14 1  
## 15 1  
## 16 1  
## 17 1  
## 18 1  
## 19 1  
## 20 1  
## 21 1  
## 22 1  
## 23 1  
## 24 1  
## 25 1  
## 26 1  
## 27 1  
## 28 1  
## 29 1  
## 30 1  
## 31 1  
## 32 1  
## 33 1  
## 34 1  
## 35 1  
## 36 1  
## 37 1  
## 38 1  
## 39 1  
## 40 1  
## 41 1  
## 42 1  
## 43 1  
## 44 1

summary(get\_mon\_resources(env))

## resource time server queue   
## Length:44 Min. : 3.466 Min. :0.0000 Min. :0.00000   
## Class :character 1st Qu.:14.759 1st Qu.:0.0000 1st Qu.:0.00000   
## Mode :character Median :59.627 Median :1.0000 Median :0.00000   
## Mean :54.534 Mean :0.5909 Mean :0.09091   
## 3rd Qu.:84.359 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :99.879 Max. :2.0000 Max. :2.00000   
## capacity queue\_size system limit replication  
## Min. :1.0 Min. :Inf Min. :0.0000 Min. :Inf Min. :1   
## 1st Qu.:1.0 1st Qu.:Inf 1st Qu.:0.0000 1st Qu.:Inf 1st Qu.:1   
## Median :1.5 Median :Inf Median :1.0000 Median :Inf Median :1   
## Mean :1.5 Mean :Inf Mean :0.6818 Mean :Inf Mean :1   
## 3rd Qu.:2.0 3rd Qu.:Inf 3rd Qu.:1.0000 3rd Qu.:Inf 3rd Qu.:1   
## Max. :2.0 Max. :Inf Max. :3.0000 Max. :Inf Max. :1

plot(get\_mon\_resources(env))



env\_per\_user <- get\_mon\_arrivals(env, per\_resource = TRUE)  
# We keep dividing by 60 to get the output in hours  
# Mean and standard deviation of checking time  
  
# filter for only express workers  
express\_worker <- env\_per\_user %>% filter(resource == 'express worker')  
#calculate the mean and sd:  
print(c("Mean activity time =",mean(express\_worker$activity\_time)))

## [1] "Mean activity time =" "0.398887888645582"

print(c("SD activity time =",sd(express\_worker$activity\_time)))

## [1] "SD activity time =" "0.225498190107898"

# calculate the wait time for express worker and then the mean and sd  
express\_worker\_wait <- express\_worker$end\_time - express\_worker$start\_time - express\_worker$activity\_time  
  
print(c("Mean wait time =",mean(express\_worker\_wait)))

## [1] "Mean wait time =" "0.125145547930934"

print(c("SD wait time =",sd(express\_worker\_wait)))

## [1] "SD wait time =" "0.278571358056705"

Nu kijken we naar administratie werkers.

# filter for only admin workers  
admin\_worker <- env\_per\_user %>% filter(resource == 'admin worker')  
  
#calculate the mean and sd:  
print(c("Mean activity time =",mean(admin\_worker$activity\_time)))

## [1] "Mean activity time =" "0.149555862119045"

print(c("SD activity time =",sd(admin\_worker$activity\_time)))

## [1] "SD activity time =" "0"

# calculate the wait time for admin workers and then the mean and sd  
admin\_worker\_wait <- admin\_worker$end\_time - admin\_worker$start\_time - admin\_worker$activity\_time  
print(c("Mean wait time =",mean(express\_worker\_wait)))

## [1] "Mean wait time =" "0.125145547930934"

print(c("SD wait time =",sd(express\_worker\_wait)))

## [1] "SD wait time =" "0.278571358056705"

Als laatste nog even naar de totale throughput van de pakketjes kijken.

# we calculate the total throughput of a parcel by first taking the end time of administration and subtracting the start time of checking.  
total\_throughput <- env\_per\_user %>% group\_by(name) %>%   
 summarise(min\_start\_time = min(start\_time), max\_end\_time = max(end\_time)) %>%  
 mutate(throughput = max\_end\_time - min\_start\_time)  
  
print(c("Mean total throughput =",mean(total\_throughput$throughput)))

## [1] "Mean total throughput =" "0.67358929869556"

print(c("SD total throughput =",sd(total\_throughput$throughput)))

## [1] "SD total throughput =" "0.243695923108491"