Sistem Antrian dengan Dua Server dan Cycle

Zefanya-01112200028

2022-12-12

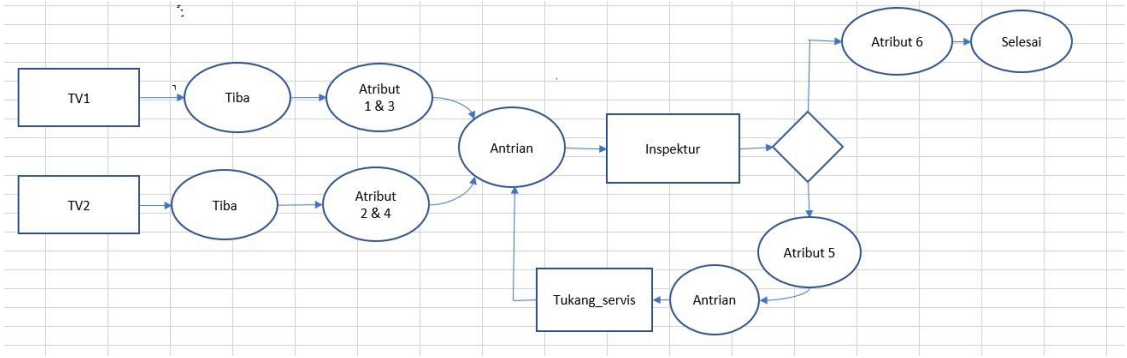
Introduction

Pada kesempatan kali ini, saya akan membahas halaman 110 (Pritsker, 1997), namun akan ditambahkan beberapa kondisi yaitu prioritas karena untuk kasus ini, terdapat dua jenis TV yang masuk yaitu TV jenis 1 dan TV jenis 2. TV jenis 1 juga akan didahulukan walaupun sudah ada entitas dengan prioritas yang lebih rendah yang sudah mengambil server (*preemption*). Tujuannya adalah membuat sistem jika terdapat dua jenis entitas.

Deskripsi Sistem

Asumsi yang digunakan pada sistem ini adalah TV jenis 1 merupakan servis premium, sehingga membutuhkan penanganan yang lebih cepat dibanding TV jenis 2. Namun, TV jenis 1 ini lebih jarang terjadi waktu antar kedatangannya adalah berdistribusi uniform dengan parameter minimum 30 dan maksimum 35, dibanding parameter minimum 3.5 dan maksimum 7.5 untuk TV jenis 2. Setelah kedua jenis televisi masuk ke sistem melalui dua jalur yang berbeda, akan ada antrian untuk dicek oleh dua inspektor. setelah selesai inspektor akan menentukan apakah TV layak atau tidak, dari pengamatan ternyata 85% TV layak sedangkan 15% tidak layak. TV yang layak akan masuk menyudahi aktivitasnya, sedangkan TV yang tidak layak akan masuk ke jalur servis, dimana dia akan mengantri sebelum diurus oleh seorang tukang servis. Setelah diservis akan TV akan mengantri lagi sebelum diperiksa oleh inspektor kembali. Lama aktivitas saat diperiksa oleh inspektor adalah berdistribusi uniform dengan parameter minimum 6 dan maximum 12, sedangkan lama servis adalah berdistribusi uniform dengan parameter minimum 20 dan maximum 40 .Begitupun seterusnya hingga TV ini diterima. Sistem ini akan dijalankan selama 500 unit waktu.

Gambar



Penjelasan

Berdasarkan Yudistira (n.d.), tanda kotak menandakan aktivitas dan tanda bulat adalah status sistem, sedangkan ketupat menandakan percabangan (*branch*). Pada sistem ini atribut 1 merupakan atribut *start_time* dan atribut 2 merupakan atribut jenis TV yaitu TV jenis 1,

begitu pula dengan 3 dan 4 untuk atribut TV jenis 2. merupakan atribut berapa kali sudah melewati lintasan yang ditolak. atribut 5 menghitung berapa kali entitas ditolak dan mengulang masuk antrian. Sedangkan, atribut 6 mencatat atribut time in system , namun ini akan langsung diambil dari perbedaan `get_mon_arrival()$start_time` dan `get_mon_arrival()$end_time` untuk menghemat penggunaan `set_attribute()`.

Script R

Pendefinisian library yang akan digunakan

```
library(simmer) library(dplyr)
library(ggplot2)


env <- simmer("sistem TV")
```

Pertama diinisialisasi dahulu seednya

```
set.seed(28)
```

Setelah itu diinisalisasi variabel-variabel yang akan digunakan dalam sistem yaitu lama untuk `timeout()` dan waktu antar kedatangan. Fungsi diterima akan menentukan apakah entitas TV akan masuk `lintasan_diterima` atau tidak.

```
diterima = function(){
  ifelse(runif(1)<0.85, return(1), return(2))} AK = function(){round(runif(1,
min = 30, max = 35),3)}
AK2 = function(){round(runif(1, min = 3.5, max = 7.5), 3)}
Lama_aktivitas = function(){round(runif(1, min = 6, max = 12),3)}
Lama_servis = function(){round(runif(1, min = 20, max = 40), 3)}
```

Didefinisikan sistem antrian akhir dahulu karena sebelum bisa menggunakan fungsi `join()` harus sudah didefinisikan dahulu lintasan selanjutnya, maka tidak bisa didefinisikan dari lintasan awal (lintasan entitas baru masuk).

```
lintasan_diterima <- trajectory() %>%
log_("TV diterima...")

lintasan_selesai <- trajectory() %>%
log_("TV sudah selesai diperiksa")
```

Sekarang didefinisikan lintasan awal. Pada `lintasan_ditolak` digunakan fungsi `join()` untuk menghindari pengulangan tak hingga kali jika digunakan fungsi `rollback()`

```
antrian <- trajectory() %>%
log_("kumpulan TV masuk...") %>%
seize("inspektor") %>%  log_("terjadi
pengecekan") %>%  timeout(Lama_aktivitas)
%>%  release("inspektor") %>%
log_("selesai diperiksa oleh
inspektor...")
```

set_attribute ditambahkan untuk menghitung berapa banyak TV telah diservis

```
lintasan_ditolak <- trajectory() %>%
  set_attribute("mengulang",1, mod = "+", init = 0) %>%
log_("TV perlu diservis...") %>%
seize("tukang_servis") %>%  timeout(Lama_servis) %>%
release("tukang_servis") %>%
  log_("kembali ke lintasan antrian...") %>%
join(antrian)

lintasan_selesai <- trajectory() %>%
log_("TV sudah selesai diperiksa")

antrian %>%  branch(
diterima,    continue = FALSE,
join(lintasan_diterima) %>%
join(lintasan_selesai),
join(lintasan_ditolak))

traj_A = trajectory() %>%
log_("TV jenis 1 tiba") %>%
set_attribute("jenis",1) %>%
  set_attribute("tiba", function(){now(env)}) %>%
join(antrian)

traj_B = trajectory() %>%
log_("TV jenis 2 tiba") %>%
set_attribute("jenis",2) %>%
  set_attribute("tiba", function(){now(env)}) %>%
join(antrian)
```

Terakhir dilakukan pembangkitan server dan entiti. Digunakan `preemptive = TRUE` agar terjadi *preemption* yaitu entitas dengan prioritas lebih tinggi akan menginterupsi aktivitas pada *resource* oleh entitas dengan prioritas lebih rendah. Juga digunakan *mon* = 2 agar dapat dilacak atributnya nantinya.

Hasil

Dapat mengetahui ringkasan statistiknya

- **Pengolahan data**

Data perlu diolah karena masih bersifat berantakan. Pertama digunakan `left_join()` untuk menggabungkan variabel dari `get_mon_atribute()` dan `get_mon_arrival()` agar dapat lebih mudah diolah nantinya. Kemudian dipisah antara TV jenis 1 dan TV jenis 2.

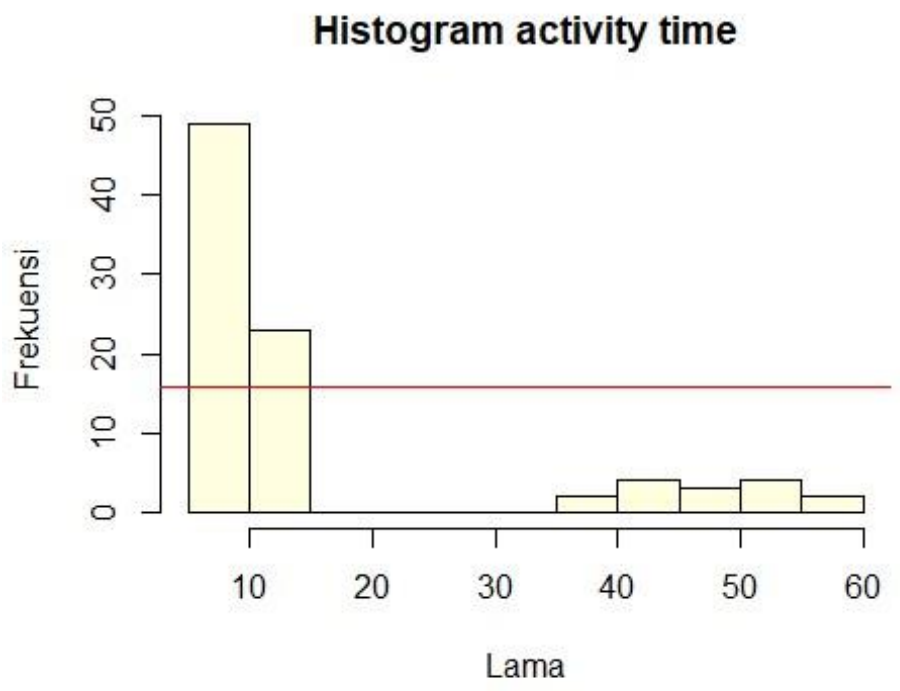
```
data_waktu = get_mon_arrivals(env) %>%
  transform(time_in_system =
get_mon_arrivals(env)$end_timeget_mon_arrivals(env)$start_time) %>%
  .[order(.$start_time),]

data_atribut = get_mon_attributes(env)

data_atribut_jenis = data_atribut %>%
filter(key == "jenis") %>%  arrange(time)

data_gabungan = left_join(data_waktu, data_atribut_jenis) %>%
arrange(value)

## Joining, by = c("name", "replication")

data_jenis_1 <- data_gabungan %>%
filter(value == 1)

data_jenis_2 <- data_gabungan %>%
filter(value == 2)

data_server <- get_mon_resources(env)

data_inspektor <- data_server %>%
filter(resource == "inspektor")

data_tukang_service <- data_server %>%
filter(resource == "tukang_servis")

data_antrian_inspektor <- data_inspektor %>%

select(server)
```

- **Visualisasi**

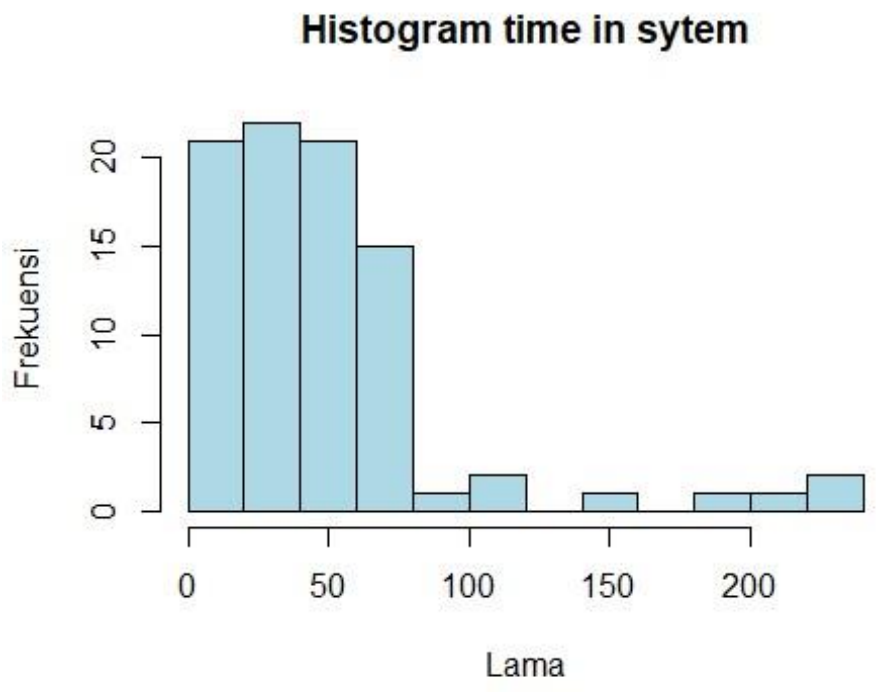
akan dicari visualisasi berupa:

1. activity_time
2. time in system
3. Panjang antrian di server inspektor
4. Panjang antrian di server tukang servis

```
hist(data_waktu$activity_time, main = "Histogram activity time", col =
"lightyellow",xlab = "Lama", ylab = "Frekuensi")
abline(mean(data_waktu$activity_time),0, col = "red")
```



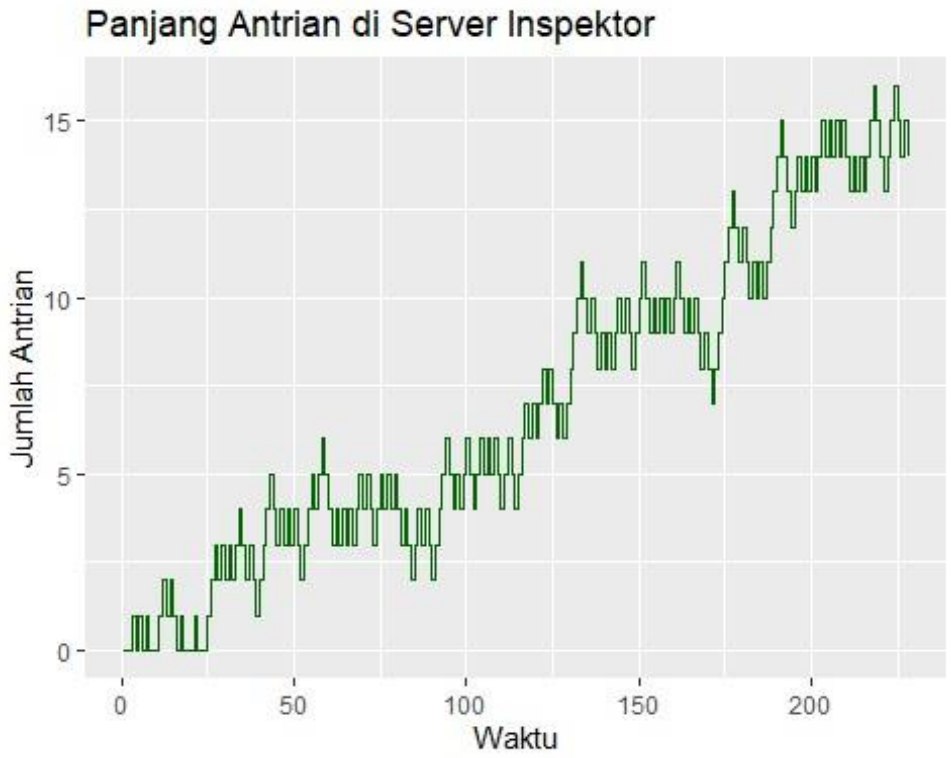

```
hist(data_waktu$time_in_system, main = "Histogram time in sytem", col
= "lightblue", xlab= "Lama", ylab = "Frekuensi")
abline(mean(data_waktu$time_in_system),0, col = "red")
```



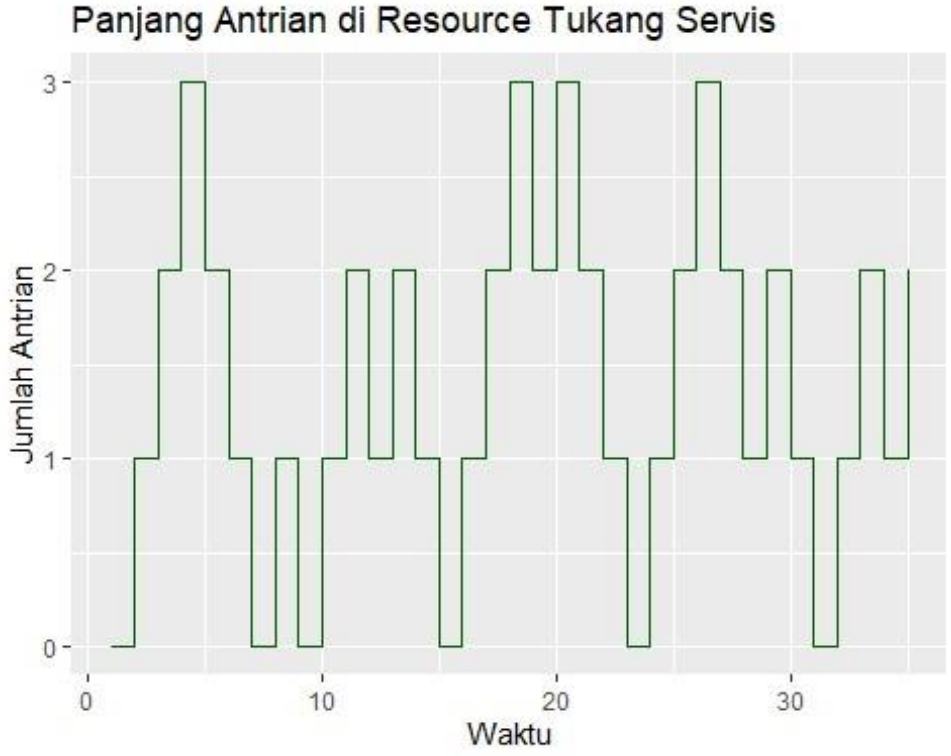
Server

```
data <- data.frame(x = 1 : nrow(data_inspektor), y = data_inspektor$queue)
grafik_inspektor = ggplot(data, aes(x,y)) + geom_step(direction = "hv", col =
"darkgreen") +
  ylab("Jumlah Antrian") + xlab("Waktu") + labs(title = "Panjang Antrian di
Server Inspektor")
```

```
grafik_inspektor
```




```
data1 <- data.frame(x = 1 : nrow(data_tukang_service), y =
data_tukang_service$queue)
grafik_tukang_service = ggplot(data1, aes(x,y)) + geom_step(direction = "hv",
col = "darkgreen") +
  ylab("Jumlah Antrian") + xlab("Waktu") + labs(title = "Panjang Antrian di
Resource Tukang Servis")
grafik_tukang_service
```



- **Perulangan**

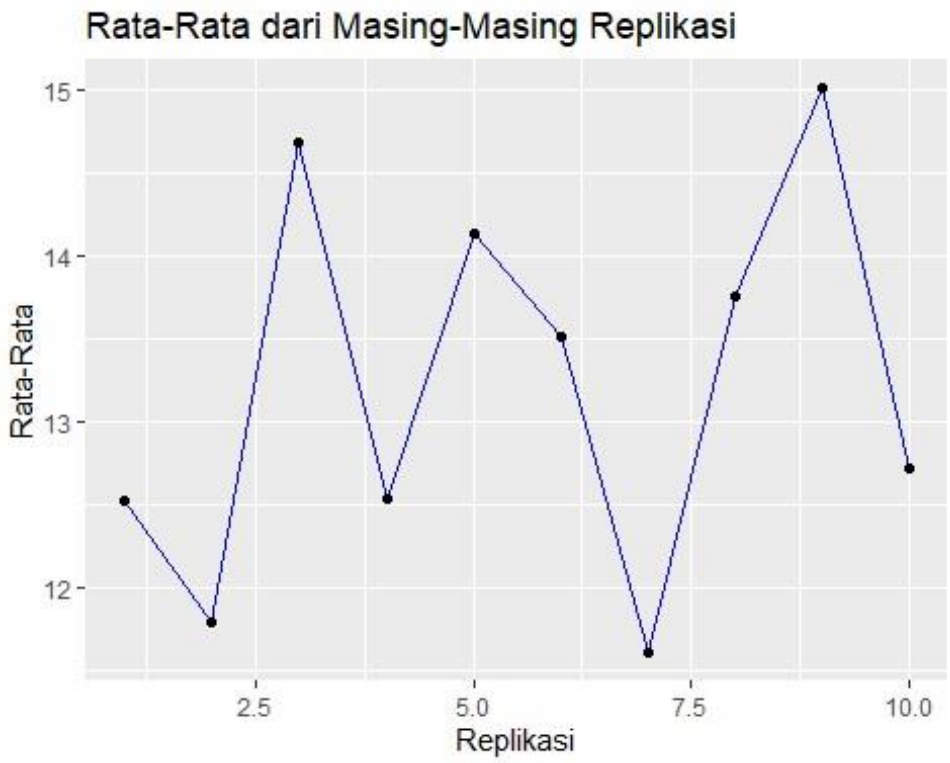
- **Pencarian parameter statistik** digunakan mclapply untuk mereplikasi simulasi sehingga nanti bisa mendapatkan parameter:

1. Rata-rata *activity-time* untuk TV langsung diterima dan yang diservis 2. Selang kepercayaan untuk rata-rata *activity-time* keseluruhan.

```
library(reshape)

replikasi = 10
replikasi_env <- lapply(1:replikasi, function(i){
simmer("sistem TV") %>%
  add_resource("inspektor", 2, preemptive = TRUE) %>%
add_resource("tukang_servis", 1, preemptive = TRUE) %>%
  add_generator("TV_jenis_1 ", traj_A, AK, mon = 2, priority = 1) %>%
add_generator("TV_jenis_2 ", traj_B, AK2, mon = 2) %>%  run(500) %>%
invisible })
```

Data yang didapatkan masih perlu diolah.

```
rata_rata <- NULL for(i
in 1:replikasi){
rata_rata =
c(rata_rata,mean(get_mon_arrivals(replikasi_env[[i]])$activity_time))
}
rata_rata

##  [1] 12.52015 11.79694 14.67583 12.53443 14.12728 13.50741 11.61072
13.74944
##  [9] 15.00749 12.71399

data2 <- data.frame(x = 1 : replikasi, y = rata_rata)
grafik_rata_rata_replikasi = ggplot(data2, aes(x,y)) +
  geom_line(col = "blue") + ylab("Rata-Rata") + xlab("Replikasi") +
labs(title = "Rata-Rata dari Masing-Masing Replikasi") +
geom_point(col = "black") grafik_rata_rata_replikasi
```



Selang kepercayaan dengan menggunakan rumus dari selang kepercayaan Walpole et al. (2012):

$$\bar{x} - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} < \mu < \bar{x} + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}$$

sedangkan untuk variansinya (untuk standar deviasinya cukup diakarkan)

$$\frac{(n-1)s^2}{\chi^2_{1-\frac{\alpha}{2}}} < \sigma^2 < \frac{(n-1)s^2}{\chi^2_{\frac{\alpha}{2}}}$$

dan dipilih $\alpha = 0.05$ untuk selang kepercayaan rata-ratanya:

```
alpha = 0.05
batas_bawah = mean(rata_rata)-qnorm(0.975, lower.tail =
TRUE)*(sd(rata_rata)/sqrt(replikasi))
batas_atas = mean(rata_rata)+qnorm(0.975, lower.tail =
TRUE)*(sd(rata_rata)/sqrt(replikasi))
```

untuk selang kepercayaan variansinya:

```
batas_bawah_var = ((replikasi- 1)*var(rata_rata))/(qchisq(1-alpha/2,df =
replikasi-1,
                                                         lower.tail = TRUE))
```

```
          batas_atas_var =  ((replikasi - 1)*var(rata_rata))/(qchisq(alpha/2,
                                                              df = replikasi - 1,
                                                              lower.tail = TRUE))
```

Analisis Keluaran Simulasi

```
paste("Diduga parameter rata-rata berada diantara: ", batas_bawah, "dan",
batas_atas)
```

```
## [1] "Diduga parameter rata-rata berada diantara:  12.4988821648136 dan
13.9498541718332"
```

```
paste("Diduga parameter variansi berada diantara: ", batas_bawah_var, "dan",
batas_atas_var)
```

```
## [1] "Diduga parameter variansi berada diantara:  0.64823234461158 dan
4.56644249698232"
```

Dari parameter ini dapat dilihat bahwa rata-ratanya namun variansinya tidak stabil. Berarti mungkin terdapat perbedaan antara waktu yang satu dengan yang lain. Perusahaan perlu memikirkan cara agar stabil `activity_time`nya.

```
paste("rata rata antrian pada tukang servis adalah:",
mean(data_tukang_service$queue))
```

```
## [1] "rata rata antrian pada tukang servis adalah: 1.4"
```

Untuk tukang servis terlihat bahwa dia cukup dapat menghandle pekerjaannya

```
paste("antrian terpanjang adalah:",max(data_inspektor$queue))
```

```
## [1] "antrian terpanjang adalah: 16"
```

Namun terdapat kesenjangan pada dari antrian inspektor yang justru kewalahan menghadapi TV yang berdatangan. Juga dari grafik terlihat bahwa trennya naik. Sehingga bisa jadi perlu ditambah lagi inspektornya.

Dari data grafik `system_time` juga terlihat bahwa jika entitas TV ditolak oleh inspektor, maka `activity_time` dari entitas tersebut akan jauh lebih lama, maka perusahaan perlu memikirkan cara agar waktu servis lebih rendah dari `runif(min = 20, max = 40)` bisa dengan menambahkan `tukang_servis` sehingga kapasitasnya bertambah atau melatih yang sudah ada agar bisa lebih cepat.

Penutup

- **Kesimpulan**

Dari pengamatan, disarankan perusahaan menambah lagi inspektor dan tukang servis.

- **Pengembangan**

Untuk pengembangan lebih lanjut dapat menambahkan kasus *balking* dan *reneging* yang belum ada di sistem ini. Bisa juga ditambahkan jam buka agar lebih sesuai dengan kenyataan perusahaan.

Referensi

Pritsker, Alan. (1997). Simulation with Visual SLAM and AweSim. John Wiley & Sons.

Walpole, R., Myers, R., Myers, S., Ye, K. (2012). Probability & Statistics for Engineers & Scientists (9th ed.). Pearson Education

Yudistira, Anom. (n.d.). Lecturer Note Simmer.