MSM_Hom_exploration

Gomon, Daniel

17/05/2024

Goal of this file

In this file, we investigate the estimates from our simulation study in the time-homogeneous case. We want to compare the estimates to the oracle on: - MSE at unique time points - supremum norm over the unique time points - Graphically display the cumulative intensities

Scenario information - Adjust here to get appropriate output.

```
scenario <- 1 #Choose from 1:4
n <- c(100, 300, 500) #Remove one if no files
n_obs <- c(6) #At most c(3,6)
N <- 1000
methods <- c("poisson", "poisson")

eval_times <- seq(0, 15, 0.1)
w_shapes <- c(0.5, 0.5, 2)
w_scales <- c(5, 10, 10/gamma(1.5))
```

Loading the data

Loading the necessary packages

Functions

Function to plot cumulative intensities.

Function for MSE over whole time-frame

Function to determine interpolation of cumulative hazard, taking into account the support sets.

Interpol support returns the linear interpolation of estimated cumulative intensities.

Function to plot the interpolated cumulative hazards

Functions for time-specific extraction of statistics (RMSE, Bias, Variance)

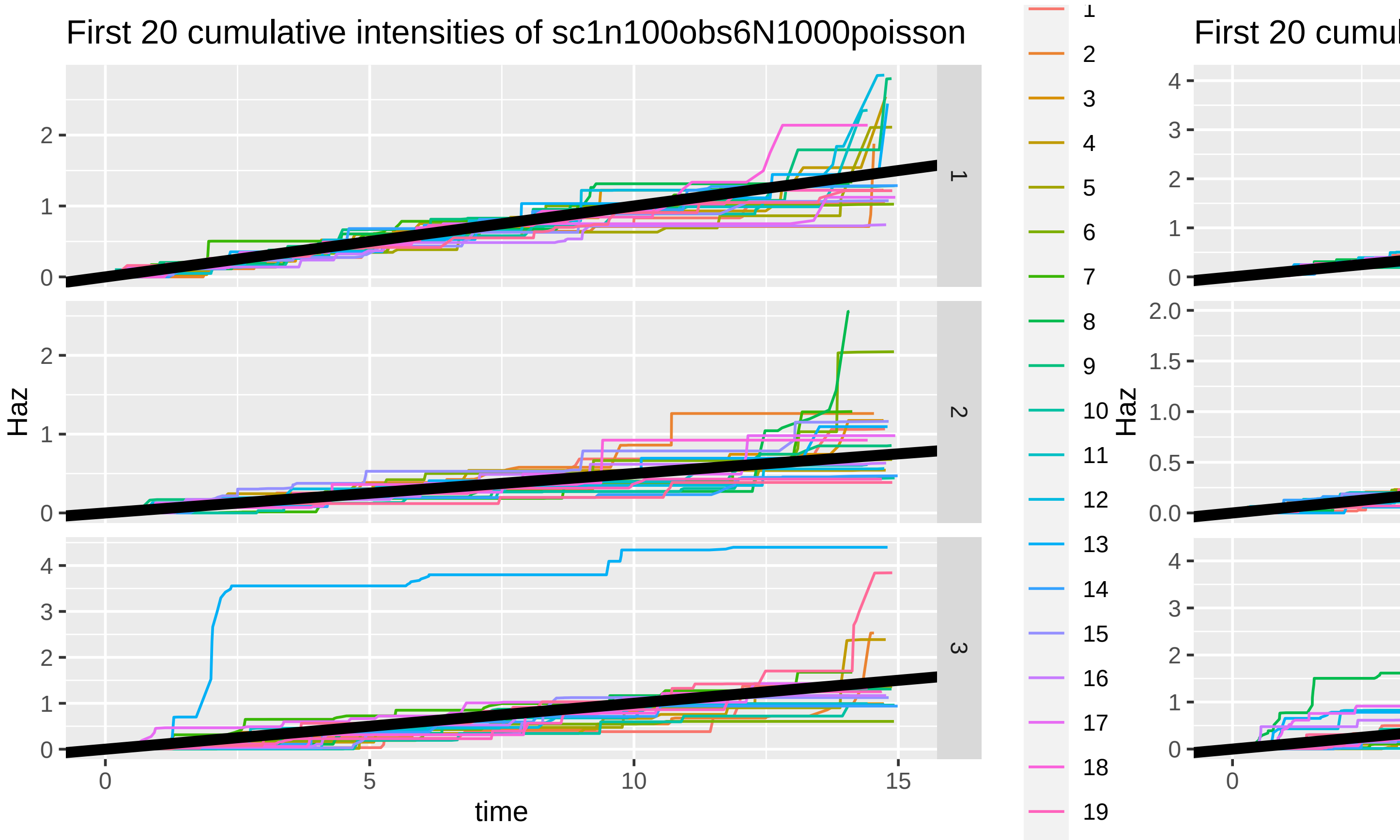
Now we can calculate summary statistics (RMSE, Bias, Variance):

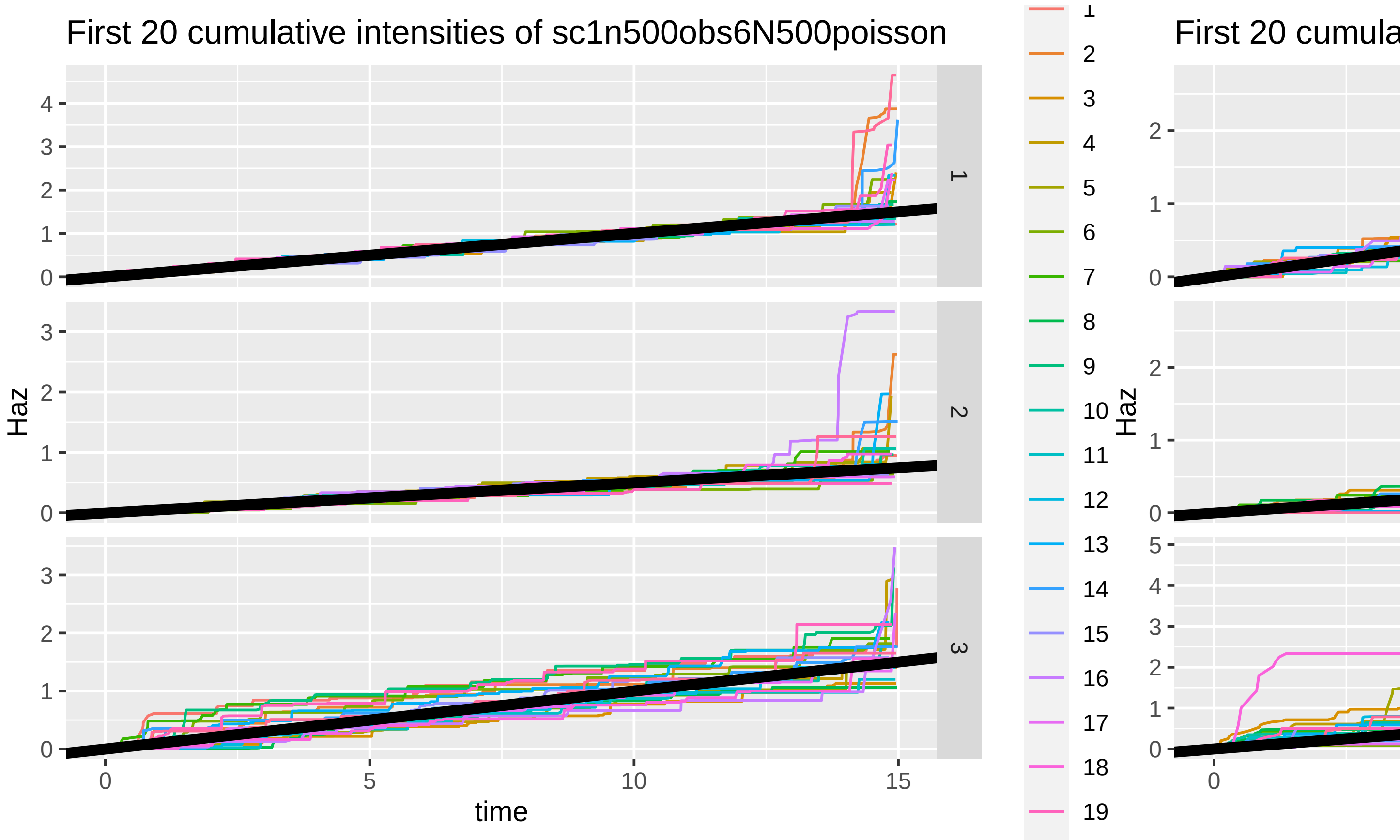
Plotting + Stats

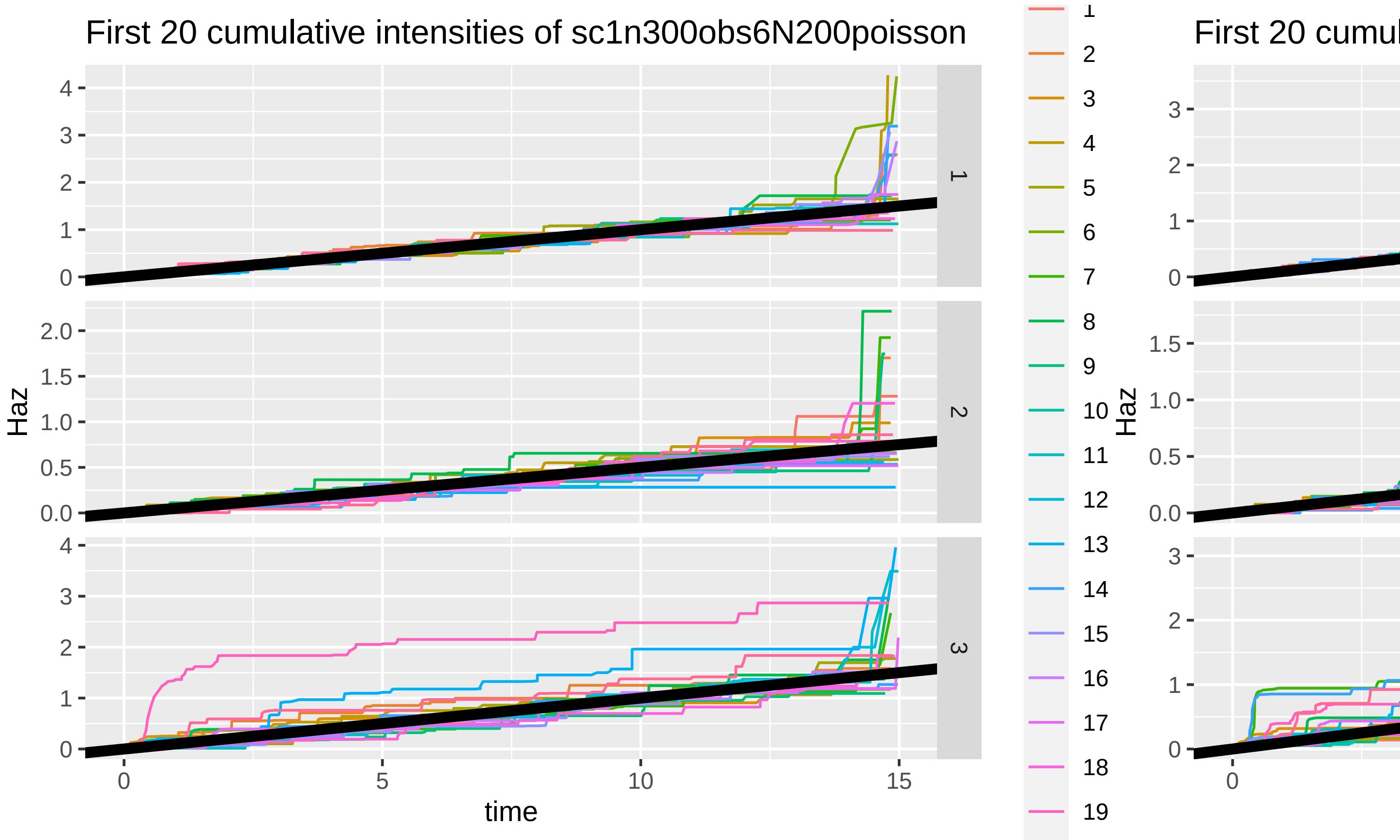
Plot the cumulative intensities of simulated samples

Plot the cumulative intensities of the first 20

```
#oracle if we are in scenarios 1/2/4
if(scenario != 3){
  oracle_plot_df <- data.frame(trans = c(1, 2, 3), slope = c(0.1, 0.05, 0.1))
} else{
  oracle_plot_df <- data.frame(time = rep(eval_times, 3),
                               Haz = c(-pweibull(eval_times, shape = w_shapes[1], scale = w_scales[1], l
                                       -pweibull(eval_times, shape = w_shapes[2], scale = w_scales[2], l
                                       -pweibull(eval_times, shape = w_shapes[3], scale = w_scales[3], l
                               trans = rep(c(1,2,3), each = length(eval_times)),
                               id = -1)
}



for(i in 1:length(var_names)){
  plot_df <- create_plot_df(get(var_names[i])[1:20])
  if(load_names[i, "method"] == "msm"){
    plot_20 <- ggplot(plot_df) + geom_abline( aes(intercept = 0, slope = slope, group = id, col = as.fac
  } else{
    plot_20 <- ggplot(plot_df, aes(x = time, y = Haz, group = id, col = as.factor(id))) + geom_line() +
  }

  #Different oracle for Weibull
  if(scenario != 3){
    plot_20 <- plot_20 + geom_abline(data = oracle_plot_df, aes(intercept = 0, slope = slope), lwd = 2,
  } else{
    plot_20 <- plot_20 + geom_line(data = oracle_plot_df, aes(x = time, y = Haz), lwd = 2, col = "black
  }
  print(plot_20)
}
```


First 20 cumulative intensities of sc1n100obs6N1000poisson
Haz
time
First 20 cumul
Haz


First 20 cumulative intensities of sc1n500obs6N500poisson
First 20 cumula
Haz
time
0
5
10
15
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19



MSE over whole time frame (per sample)

Currently not implemented! Not very interesting I think and a pain to evaluate.

Determine time-specific statistics of data using interpolation

Also no longer implemented, because not very interesting, interpolation doesn't change the visuals much.

Use the written functions to get a visual display

We create an oracle:

```
if(scenario != 3){ #Homogeneous oracle
  oracle_df <- matrix(NA, nrow = length(eval_times), ncol = 5)
  oracle_df[, 1] <- 0.1*eval_times
  oracle_df[, 2] <- 0.05*eval_times
  oracle_df[, 3] <- 0.1*eval_times
  oracle_df[, 4] <- eval_times
  oracle_df[, 5] <- rep(0, length(eval_times))
  colnames(oracle_df) <- c(paste0("trans", 1:3), "time", "id")
} else{ #Weibull oracle
  oracle_df <- matrix(NA, nrow = length(eval_times), ncol = 5)
```

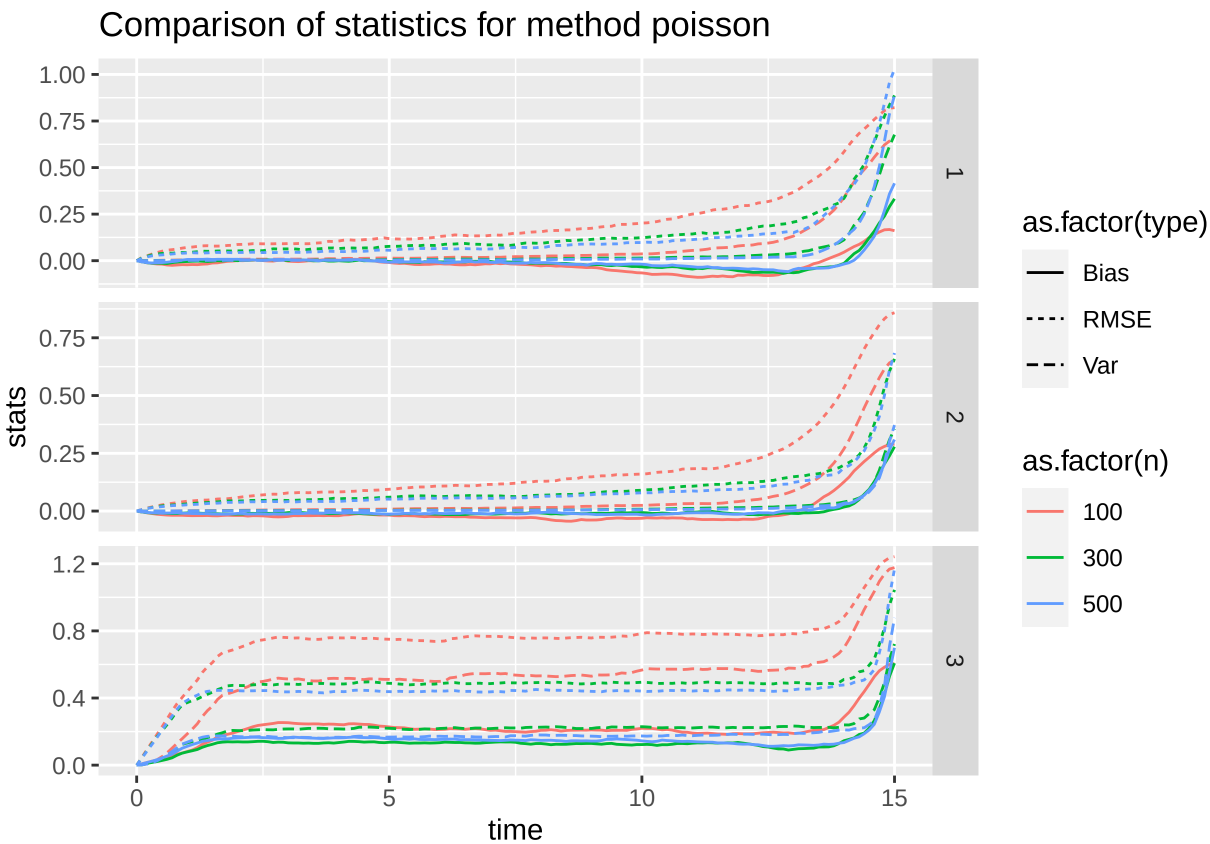
```
  oracle_df[, 1] <- -pweibull(eval_times, shape = w_shapes[1], scale = w_scales[1], lower = FALSE, log =
  oracle_df[, 2] <- -pweibull(eval_times, shape = w_shapes[2], scale = w_scales[2], lower = FALSE, log =
  oracle_df[, 3] <- -pweibull(eval_times, shape = w_shapes[3], scale = w_scales[3], lower = FALSE, log =
  oracle_df[, 4] <- eval_times
  oracle_df[, 5] <- rep(0, length(eval_times))
  colnames(oracle_df) <- c(paste0("trans", 1:3), "time", "id")
}
```

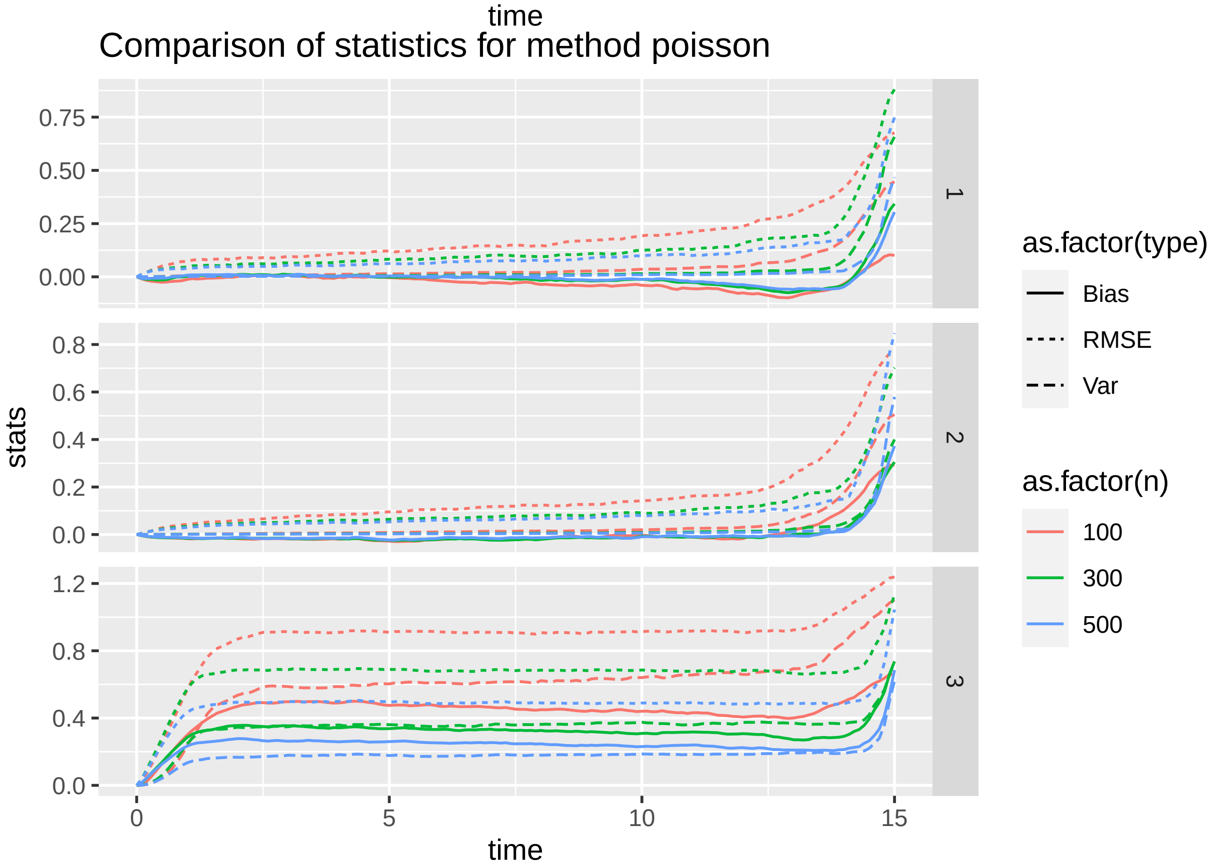
Plot time-dependent summary statistics

```
for(i in 1:nrow(load_names)){
  assign(paste0("summary_df", i), suppressWarnings(create_summary_df(get(var_names[i]), eval_times = eva
}
for(i in 1:nrow(load_names)){
  assign(paste0("stat_df", i), extract_summary_stat(summary_df = get(paste0("summary_df", i)), oracle_df
}
```

Compare the statistics as n increases.

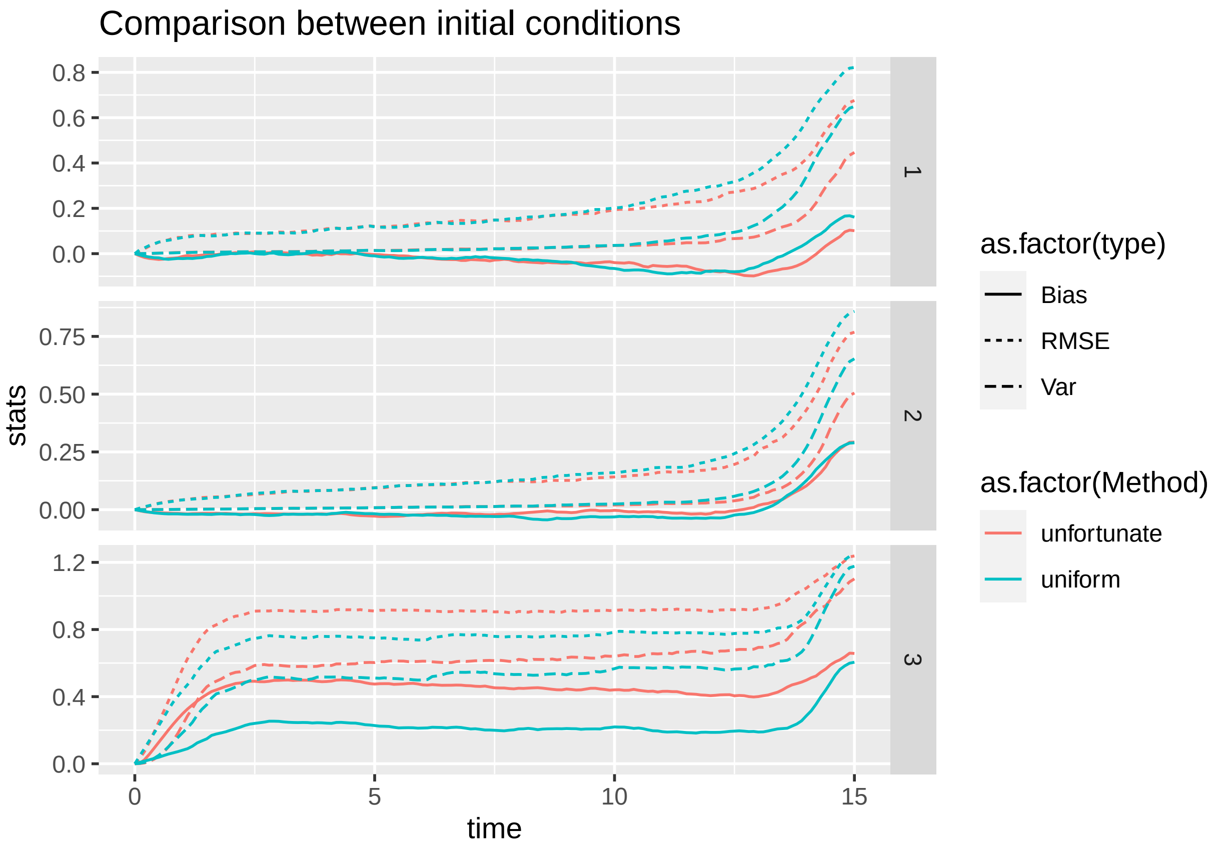
```
for(i in 1:length(methods)){
  stat_multiple <- NULL
  for(j in 1:length(n)){
    stat_multiple <- rbind(stat_multiple, get(paste0("stat_df", (i-1)*length(n)+j)))
  }
  stat_multiple <- cbind(stat_multiple, c(rep(n, each = length(eval_times) * 9)))
  colnames(stat_multiple)[5] <- "n"
  plot_all_stat <- ggplot(data = stat_multiple, aes(x = time, y = stats, group = interaction(type, n), c
  print(plot_all_stat)
}
```

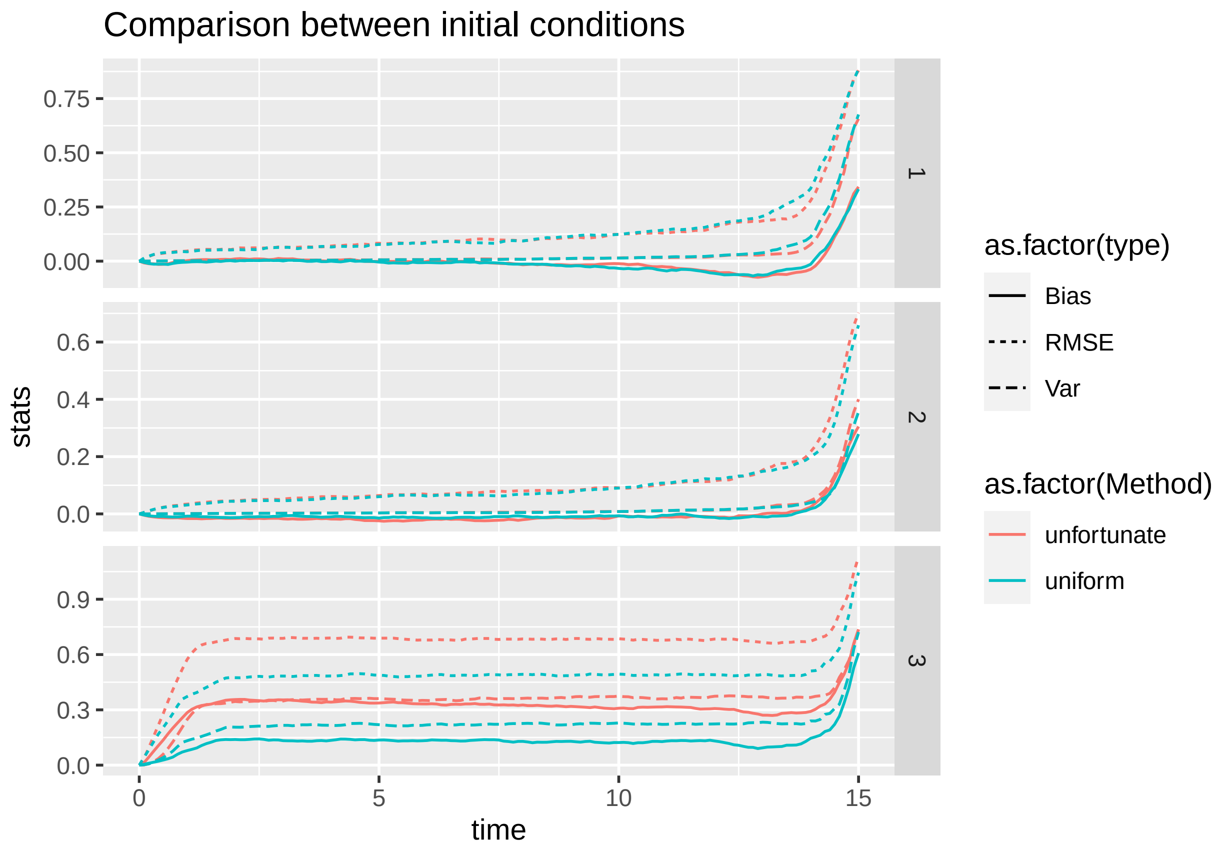

Comparison of statistics for method poisson
stats
time
1
2
3
as.factor(type)
Bias
RMSE
Var
as.factor(n)
100
300
500

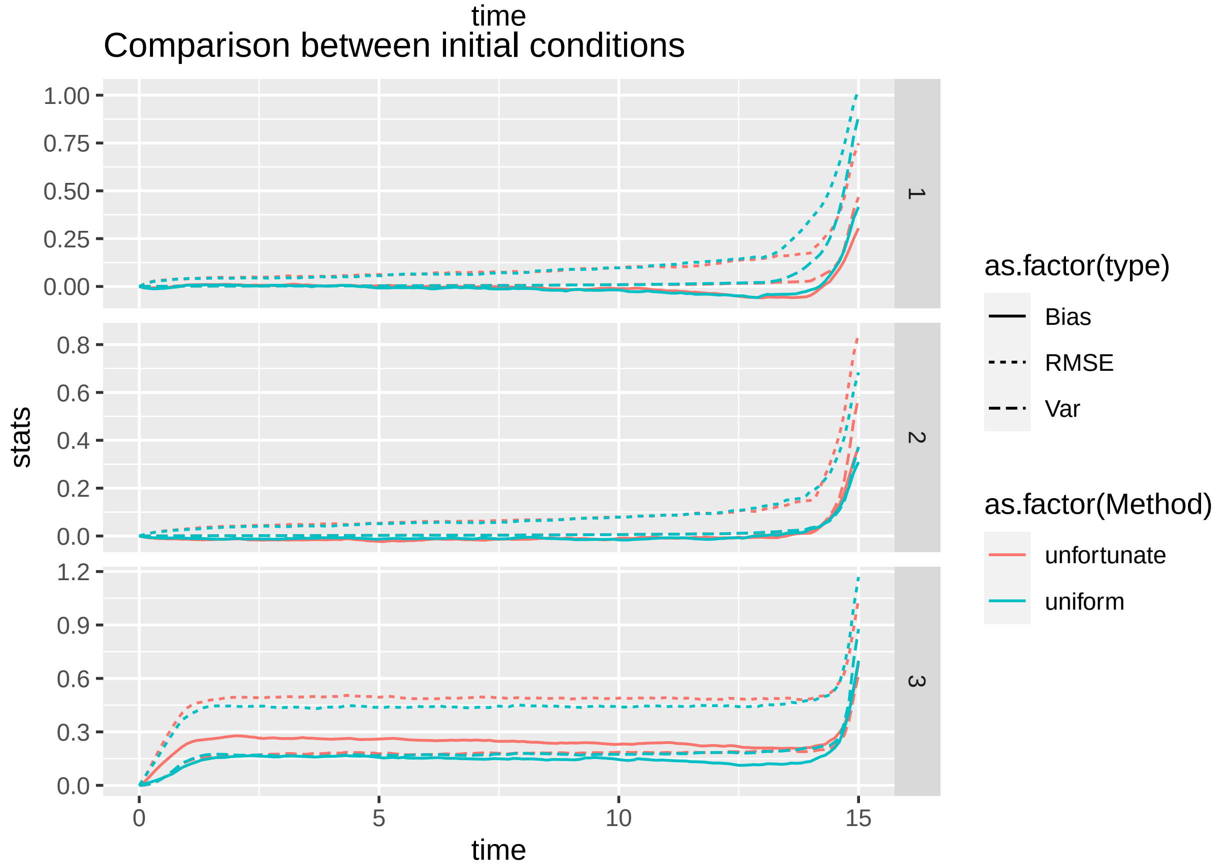

Comparison of statistics for method poisson
stats
time
1
2
3
as.factor(type)
Bias
RMSE
Var
as.factor(n)
100
300
500

Compare between methods

```
init_cond <- c("uniform", "unfortunate")
for(i in 1:length(n)){
  stat_multiple <- NULL
  for(j in 1:length(methods)){
    stat_multiple <- rbind(stat_multiple, get(paste0("stat_df", i + length(n)*(j-1))))
  }
  stat_multiple <- cbind(stat_multiple, c(rep(init_cond, each = length(eval_times) * 9)))
  colnames(stat_multiple)[5] <- "Method"
  plot_all_stat <- ggplot(data = stat_multiple, aes(x = time, y = stats, group = interaction(type, as.fa
  print(plot_all_stat)
}
```







We clearly see that with unfortunate initial conditions we can get very high biases for Poisson. So it's not really doing better.