ScenarioX_compact

Daniel Gomon

2024-06-26

Scenario information - Adjust here to get appropriate output.

```r
scenario <- 6 #Choose from 1:6
n <- c(500) #Remove one if no files
n_obs <- c(4, 10) #At most c(3,6)
N <- 1000
methods <- c("multinomial", "poisson", "msm") #

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

```r
if(inherits(get(var_names[1])[[1]], "npmsm")){
  n_states <- nrow(get(var_names[1])[[1]]$tmat)
} else if(inherits(get(var_names[1])[[1]], "msm")){
  n_states <- get(var_names[1])[[1]]$qmodel$nstates
}

if(scenario == 3){
  shapes <- c(0.5, 0.5, 2)
  scales <- c(5, 10, 10/gamma(1.5))
} else{
  shapes <- c(1, 1, 1)
  scales <- c(10, 20, 10)
}
from <- 1 #Which state do we consider transitions from?
```

Loading the necessary packages

```
## Loading required package: survival
```

Cumulative Intensity Functions

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

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

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

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")
}
```

Probability Transition Functions

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

#Extract Data

Intensities

```
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_d
}
```

Transition Probabilities

```
oracle_df_pt <- probtrans_weib(transMat = get(var_names[1])[[1]]$tmat, times = eval_times, shapes = sha
```

```r
for(i in 1:nrow(load_names)){
  assign(paste0("summary_df_pt", i), suppressWarnings(create_summary_df_pt(get(var_names[i]), eval_times
}
for(i in 1:nrow(load_names)){
  assign(paste0("stat_df_pt", i), extract_summary_stat_pt(summary_df = get(paste0("summary_df_pt", i)),
}
```

Plotting

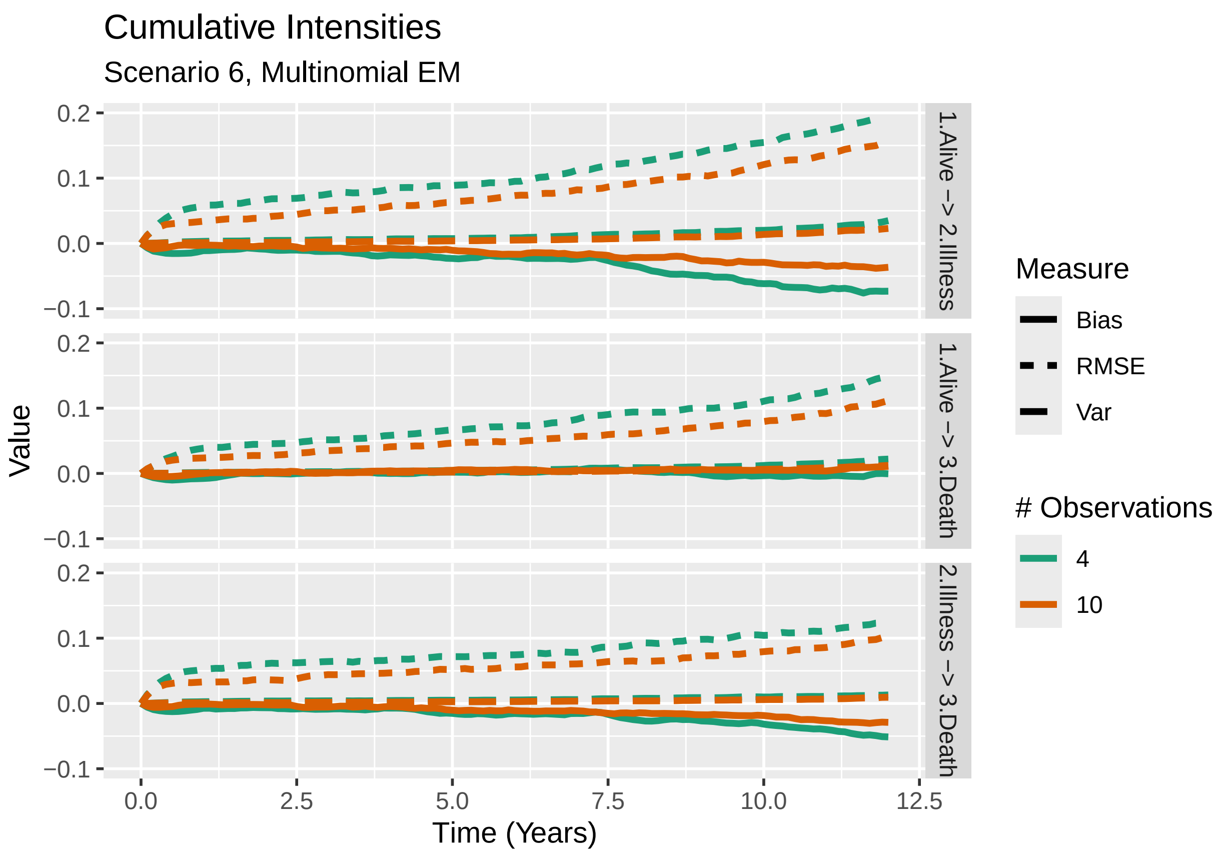
Cumulative Intensities

Compare the statistics as n increases within methods.

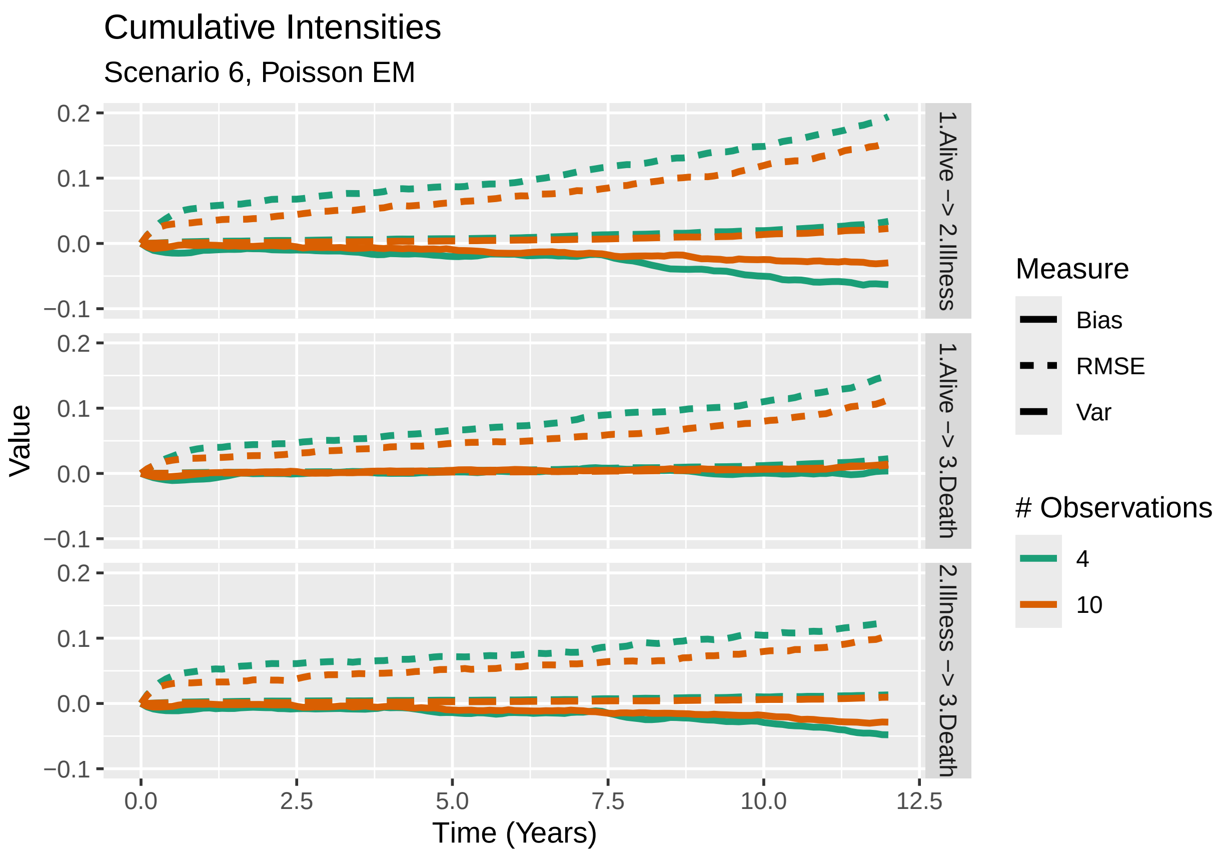
```r
trans_names <- c("1.Alive -> 2.Illness", "1.Alive -> 3.Death", "2.Illness -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous") #
names(method_names) <- c("multinomial", "poisson", "msm") #


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

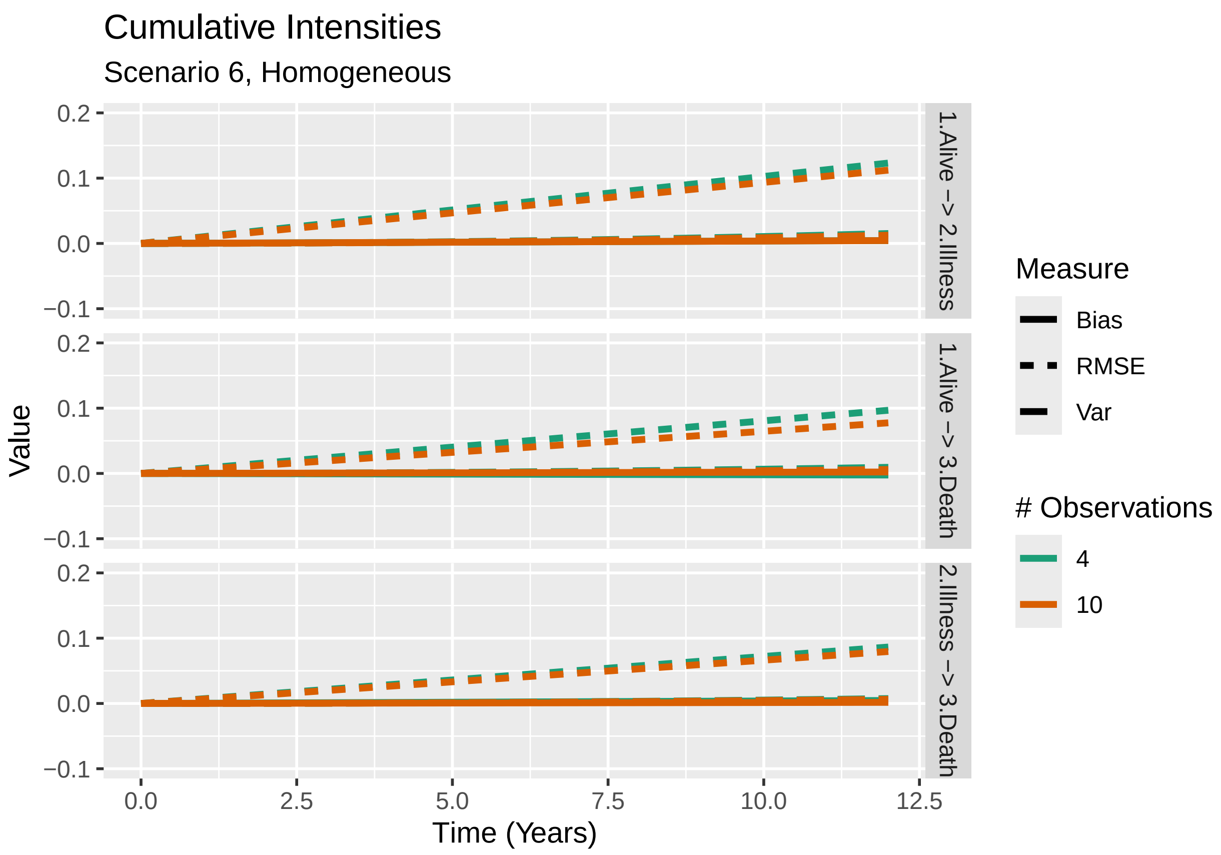
```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
all_stat_plotn <- ggarrange(plot_all_statn1, plot_all_statn2, plot_all_statn3,
          labels = c("A", "B", "C"),
          ncol = 3, nrow = 1, common.legend = TRUE, legend = "bottom")
```

```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```r
ggsave(file = paste0("Scenario", scenario, "gg_compint.eps"), plot = all_stat_plotn, width = 5.5, heigh
```

Compare between methods for fixed n.

```r
trans_names <- c("1.Alive -> 2.Illness", "1.Alive -> 3.Death", "2.Illness -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous")
names(method_names) <- c("multinomial", "poisson", "msm")


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(methods, 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)
  assign(paste0("plot_all_stat", i), plot_all_stat)
}

all_stat_plot <- ggarrange(plot_all_stat1, plot_all_stat2, plot_all_stat3,
          labels = c("A", "B", "C"),
          ncol = 3, nrow = 1, common.legend = TRUE, legend = "bottom")


ggsave(file = paste0("Scenario", scenario, "gg_intensities.eps"), plot = all_stat_plot, width = 5.5, hei
```

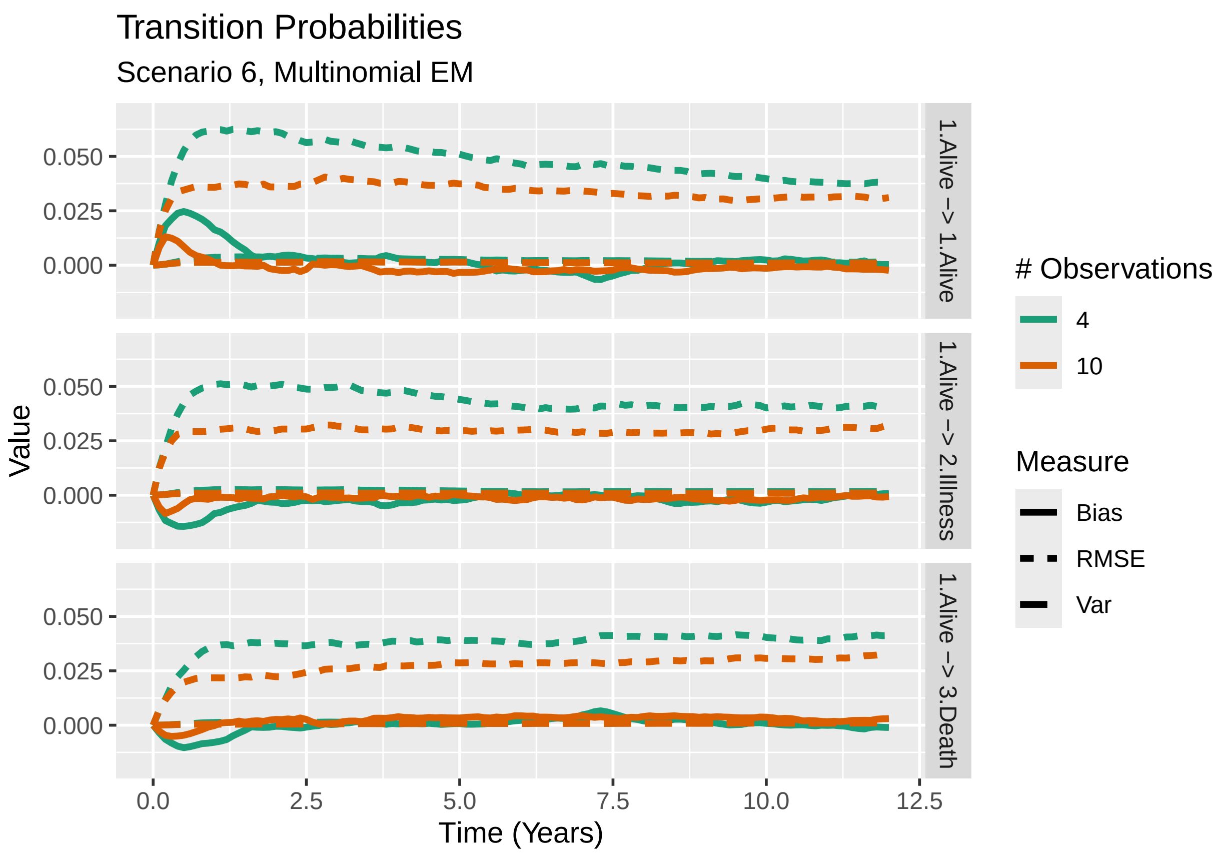
Transition Probabilities

Compare the statistics as n increases within methods.

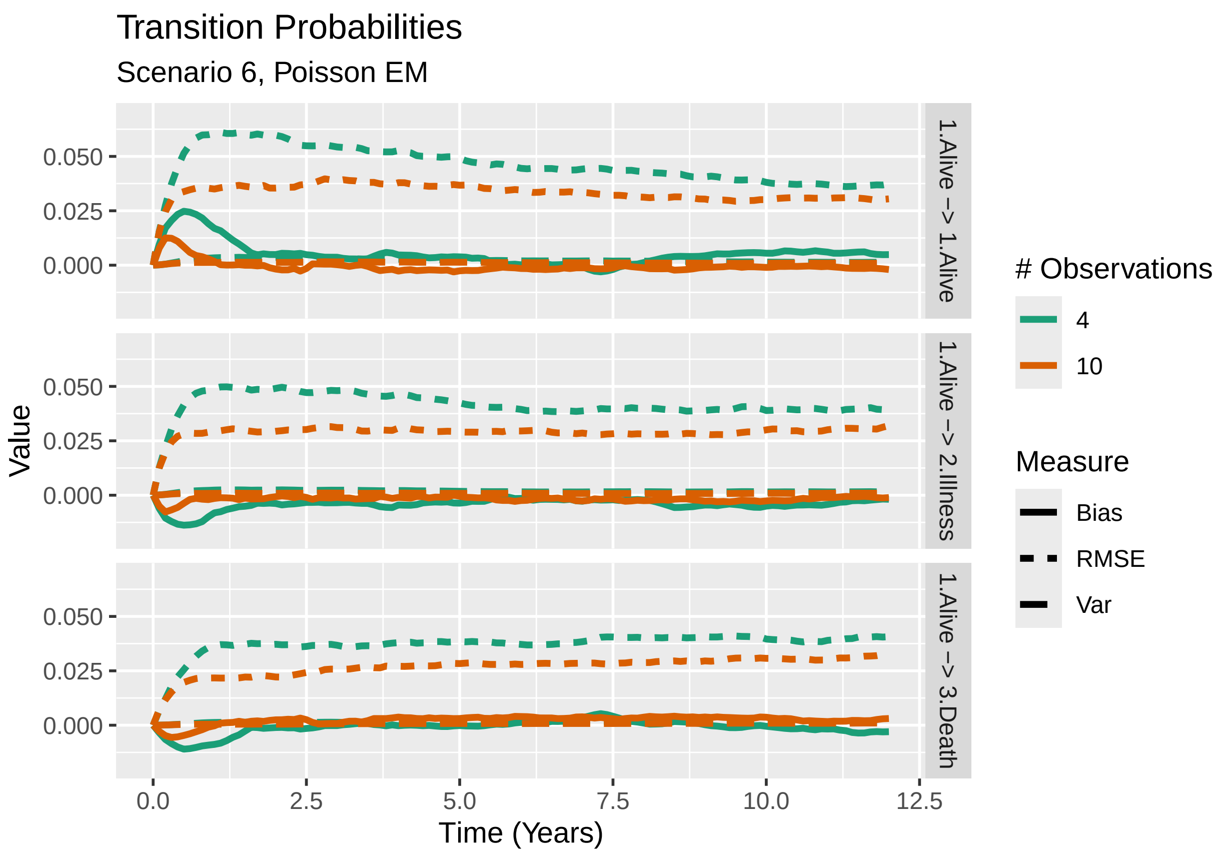
```r
trans_names <- c("1.Alive -> 1.Alive", "1.Alive -> 2.Illness", "1.Alive -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous")#
names(method_names) <- c("multinomial", "poisson", "msm")#

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

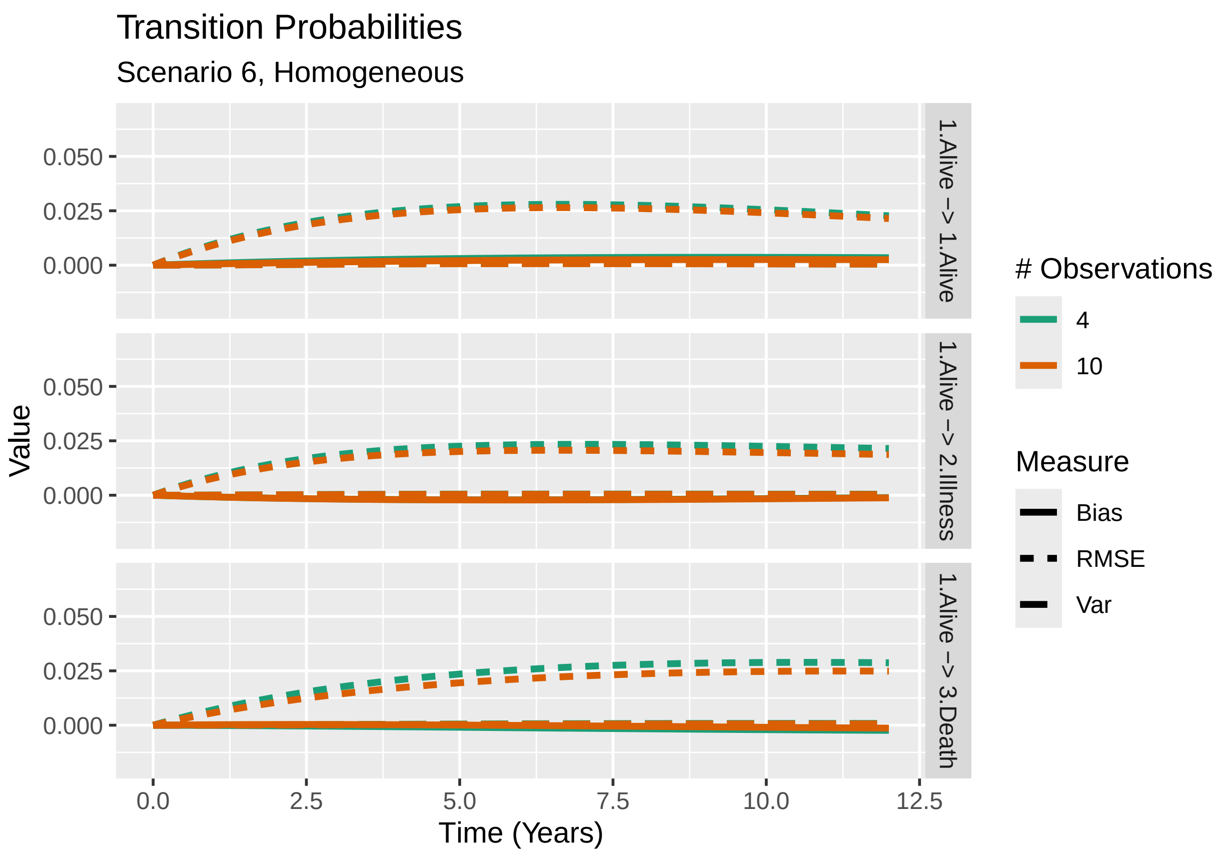
```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
all_stat_plot_ptn <- ggarrange(plot_all_stat_ptn1, plot_all_stat_ptn2, plot_all_stat_ptn3,
          labels = c("A", "B", "C"),
          ncol = 3, nrow = 1, common.legend = TRUE, legend = "bottom")
```

```
## Warning: Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 180 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```r
ggsave(file = paste0("Scenario", scenario, "gg_compprobs.eps"), plot = all_stat_plot_ptn, width = 5.5, h
```

Compare between methods for fixed n.

```r
trans_names <- c("1.Alive -> 1.Alive", "1.Alive -> 2.Illness", "1.Alive -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous")
names(method_names) <- c("multinomial", "poisson", "msm")



for(i in 1:length(n)){
  stat_multiple <- NULL
  for(j in 1:length(methods)){
    stat_multiple <- rbind(stat_multiple, get(paste0("stat_df_pt", i + length(n)*(j-1))))
  }
  stat_multiple <- cbind(stat_multiple, c(rep(methods, each = length(eval_times) * n_states * 3)))
  colnames(stat_multiple)[5] <- "Method"
  plot_all_stat_pt <- ggplot(data = stat_multiple, aes(x = time, y = stats, group = interaction(type, a
  print(plot_all_stat_pt)
  assign(paste0("plot_all_stat_pt", i), plot_all_stat_pt)
}

all_stat_plot_pt <- ggarrange(plot_all_stat_pt1, plot_all_stat_pt2, plot_all_stat_pt3,
          labels = c("A", "B", "C"),
          ncol = 3, nrow = 1, common.legend = TRUE, legend = "bottom")


ggsave(file = paste0("Scenario", scenario, "gg_probs.eps"), plot = all_stat_plot_pt, width = 5.5, heigh
```

Combination plots

for n = 500, plot cumulative intensities and transition probabilities

```
trans_names <- c("1.Alive -> 1.Alive", "1.Alive -> 2.Illness", "1.Alive -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous")
names(method_names) <- c("multinomial", "poisson", "msm")


#only for n = 500
k <- 3


stat_multiple <- NULL
for(j in 1:length(methods)){
  stat_multiple <- rbind(stat_multiple, get(paste0("stat_df_pt", k + length(n)*(j-1))))
}
stat_multiple <- cbind(stat_multiple, c(rep(methods, each = length(eval_times) * n_states * 3)))
colnames(stat_multiple)[5] <- "Method"
comb_plot <- ggplot(data = stat_multiple, aes(x = time, y = stats, group = interaction(type, as.factor(



trans_names <- c("1.Alive -> 2.Illness", "1.Alive -> 3.Death", "2.Illness -> 3.Death")
names(trans_names) <- c(1, 2, 3)
method_names <- c("Multinomial EM", "Poisson EM", "Homogeneous")
names(method_names) <- c("multinomial", "poisson", "msm")



stat_multiple <- NULL
for(j in 1:length(methods)){
  stat_multiple <- rbind(stat_multiple, get(paste0("stat_df", k + length(n)*(j-1))))
```

```
}
stat_multiple <- cbind(stat_multiple, c(rep(methods, each = length(eval_times) * 9)))
colnames(stat_multiple)[5] <- "Method"
comb_plot2 <- ggplot(data = stat_multiple, aes(x = time, y = stats, group = interaction(type, as.factor






all_comb_plot <- ggarrange(comb_plot, comb_plot2,
          labels = c("A", "B"),
          ncol = 2, nrow = 1, common.legend = TRUE, legend = "bottom")

ggsave(file = paste0("Scenario", scenario, "gg_combplot.eps"), plot = all_comb_plot, width = 5.5, heigh
```