Standard exponential spacings with missing values - an extreme value theory simulation study to evaluate priors

Sean van der Merwe, Jan Beirlant, Andréhette Verster

2023-10-17

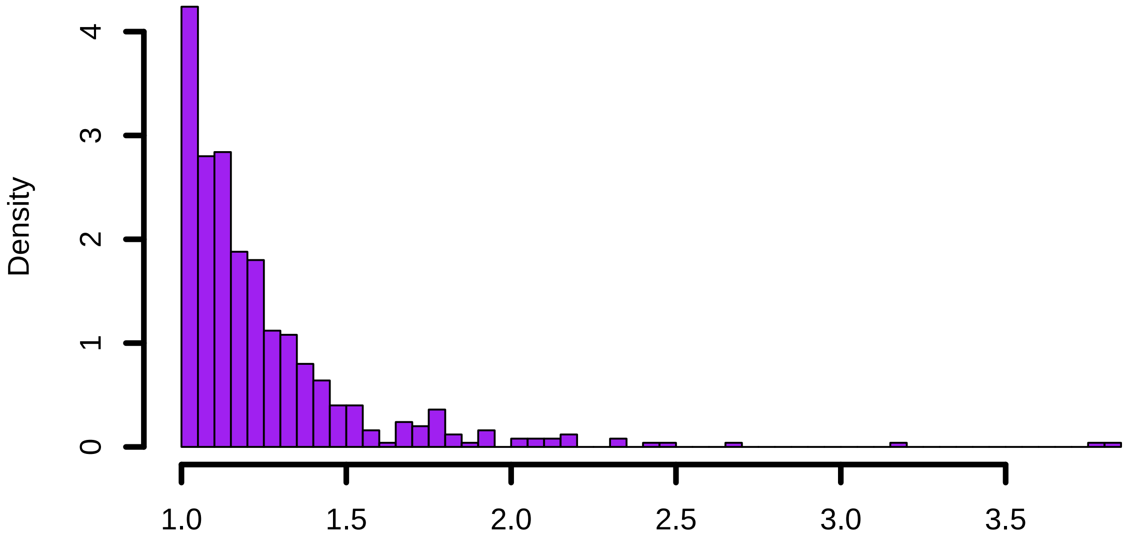
Table of contents

Introduction

Consider the following sample from an extreme value distribution with explicitly positive extreme value index (EVI):

```
set.seed(123456)
y_full <- actuar::rpareto1(500, 5, 1)
# y_full <- actuar::rinvweibull(500, 5, 1)
```

```
par(mar = c(4.4, 4.4, 0.4, 0.4))
y_full <- y_full |> sort(decreasing = TRUE)
y_full |> hist(100, main = '', xlab = '',
               freq = FALSE, lwd = 3, col = 'purple')
```

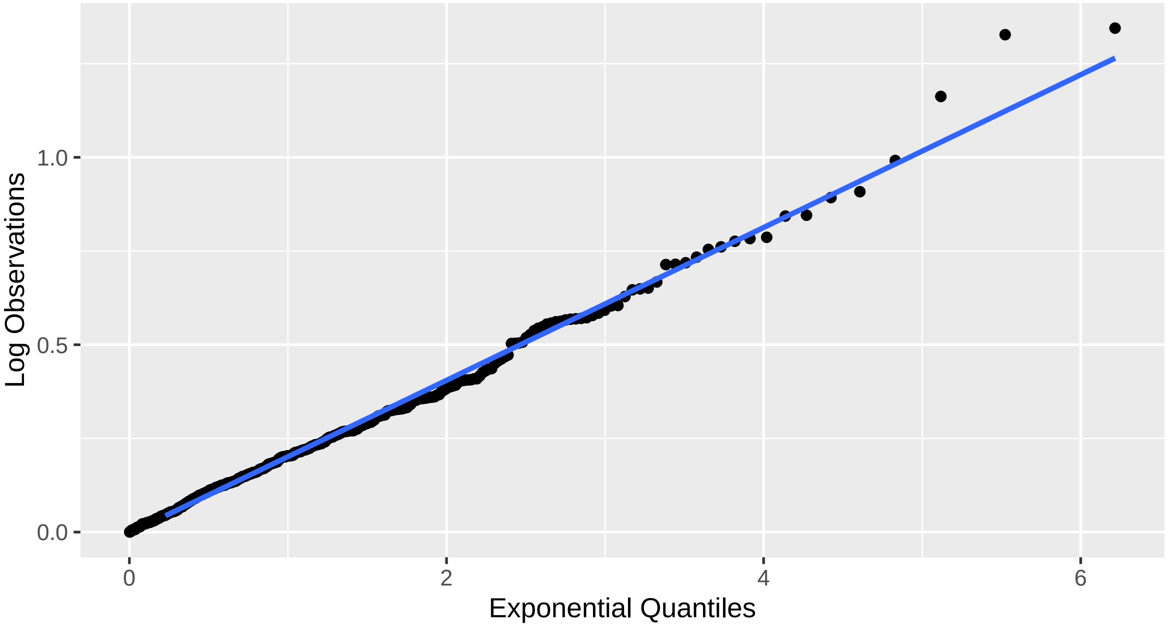


```
options(scipen = 12)
library(tidyverse)
library(knitr)
library(openxlsx)
```

```r
qqplotfunc <- function(y_full, k = length(y_full)%/%2, ylabel = "Log Observations") {
  y_full <- sort(y_full)
  plot_data <- data.frame(log_observations = log(y_full),
      exponential_quantiles = -log(1-(1:length(y_full))/(length(y_full)+1))
  )
  plot_data |> ggplot(aes(x = exponential_quantiles, y = log_observations)) +
    geom_point() +
    geom_smooth(data = subset(plot_data, (length(y_full):1) < k),
                method = 'lm', formula = 'y~x') +
    ylab(ylabel) + xlab('Exponential Quantiles')
}
```

Let us have a look at the tail via the Pareto QQ Plot:

```r
y_full |> qqplotfunc(k = 400)
```



It is clear that we have a good fit to the top part of the observations.

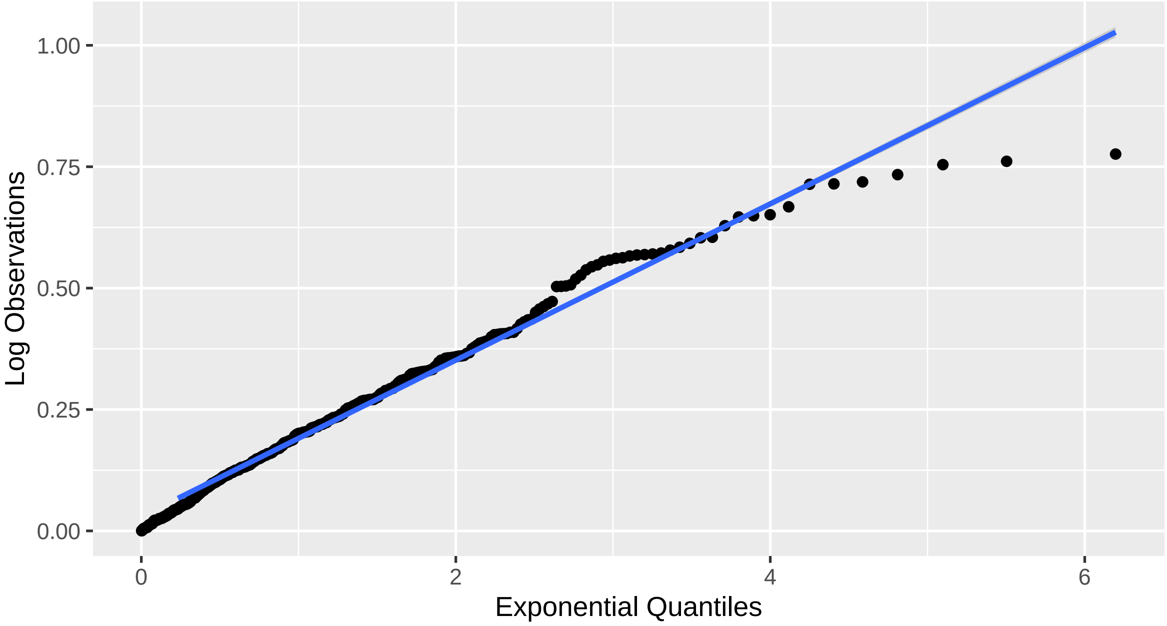
Missing extremes

Consider now the case where the top values are missing.

```
m_example <- 10
```

We explicitly exclude the top 10 values from the sample.

```
y_full[-(1:m_example)] -> y_reduced
y_reduced |> qqplotfunc(k = 400-m_example)
```



```
k_example <- 400 - 1 - m_example;
```

Now we observe a rather distorted plot.

The goal now is to attempt to recover the correct fit in spite of these missing observations.

Model

Assuming that the underlying distribution satisfies the max-domain of attraction condition:

i.e. assuming that the maximum of independent and identically distributed observations $X_1, X_2, \dots, X_n$ can be approximated by the generalized extreme value distribution: as $n \to \infty$

$$\mathbb{P}\left(a_n^{-1}\left(\max_{i=1,\ldots,n} X_i - b_n\right) \leq y\right) \quad \longrightarrow \quad G_\gamma(y) = \exp\left(-(1+\gamma y)^{-1/\gamma}\right) \qquad \text{for } 1+\gamma\, y > 0$$

where $b_n \in \mathbb{R}$, $a_n > 0$ and $\gamma \in \mathbb{R}$ are the location, scale and shape parameters, respectively. The EVI γ is a measure of the tail-heaviness of the distribution of X with a larger value of γ implying a heavier tail of F.

For this study we specifically consider only the case where $\gamma > 0$.

Then the method developed in *[cite previous paper]* can be defined as follows:

First define

$$V_{j,n} := \log \frac{X_{(n-j+1)}}{X_{(n-j)}} \; j = 1, \ldots, k,$$

then, if $\gamma > 0$,

$$V_{j,n} \sim Exp\left(\frac{j+m}{\gamma}\right)$$

The log-likelihood function is then given by

$$\ell(\gamma, m) = \sum_{j=1}^{k} \log(j+m) - k\log\gamma - \left(\frac{j+m}{\gamma}\right) v_{j,n}$$

Additionally, we can assign convenient priors on the parameters, the effect of which can be evaluated as the primary focus of this exercise.

Then, hierarchically, our model becomes

$$\begin{aligned} V_{j,n} &\sim Exp\left(\lambda_j\right) \\ \lambda_j &= (j+m)\gamma^{-1} \\ \gamma &\sim U(0,1) \\ \log\pi(m) &= -\lambda m^a + c \text{ OR } -\lambda\log(m+a) + c \\ &\text{where } \lambda, a > 0 \text{ are prior hyper-parameters} \\ &\text{and } c \text{ is an unknown constant} \end{aligned}$$

Typically, for the first prior on m, λ is set to 0.1, but this can be varied. For the second prior λ is typically set to 1 for a scale parameter and 2 for a variance parameter (Jeffreys' priors), but m does not cleanly fall in these categories so we may evaluate both and more. For the first prior we might want to test a slope adjustment a; while for the second prior adding a is necessary to avoid the prior diverging to infinity at $m = 0$. In all cases we create a prior that starts at a finite value and gradually decreases to zero.

Implementation

The model is implemented using the STAN interface. We will attempt to use this interface via rstan to implement our models.

```r
library(rstan)
mycores <- max(1,floor(parallel::detectCores(logical = FALSE)*0.8))
options(mc.cores = mycores)
rstan_options(auto_write = TRUE)
```

Model definitions

```stan
// This Stan block defines a fit using standard spacings, by Sean van der Merwe, UFS
data {
  int<lower=2> k; // number of observations in total
  real<lower = 0> y[k]; // sorted and pre-transformed observations
  real<lower = 0> lambda; // prior scale hyper-parameter on missing observations
  real<lower = 0> a; // prior shape hyper-parameter on missing observations
}
// The parameters of the model
parameters {
  real<lower = 0> m; // missing information parameter
  real<lower = 0, upper = 1> evi; // extreme value index
}
transformed parameters {
  real<lower = 0> rate[k]; // expected values
  for (j in 1:k) {
    rate[j] = (j+m)/evi;
  }
}
model {
  y ~ exponential(rate);
  target += -lambda*(m^a);
  evi ~ uniform(0, 1);
}
```

```
// This Stan block defines a fit using standard spacings, by Sean van der Merwe, UFS
data {
  int<lower=2> k; // number of observations in total
  real<lower = 0> y[k]; // sorted and pre-transformed observations
  real<lower = 0> lambda; // prior scale hyper-parameter on missing observations
  real<lower = 0> a; // prior shape hyper-parameter on missing observations
}
// The parameters of the model
parameters {
  real<lower = 0> m; // missing information parameter
  real<lower = 0, upper = 1> evi; // extreme value index
}
transformed parameters {
  real<lower = 0> rate[k]; // expected values
  for (j in 1:k) {
    rate[j] = (j+m)/evi;
  }
}
model {
  y ~ exponential(rate);
  target += -lambda*log(m+a);
  evi ~ uniform(0, 1);
}
```

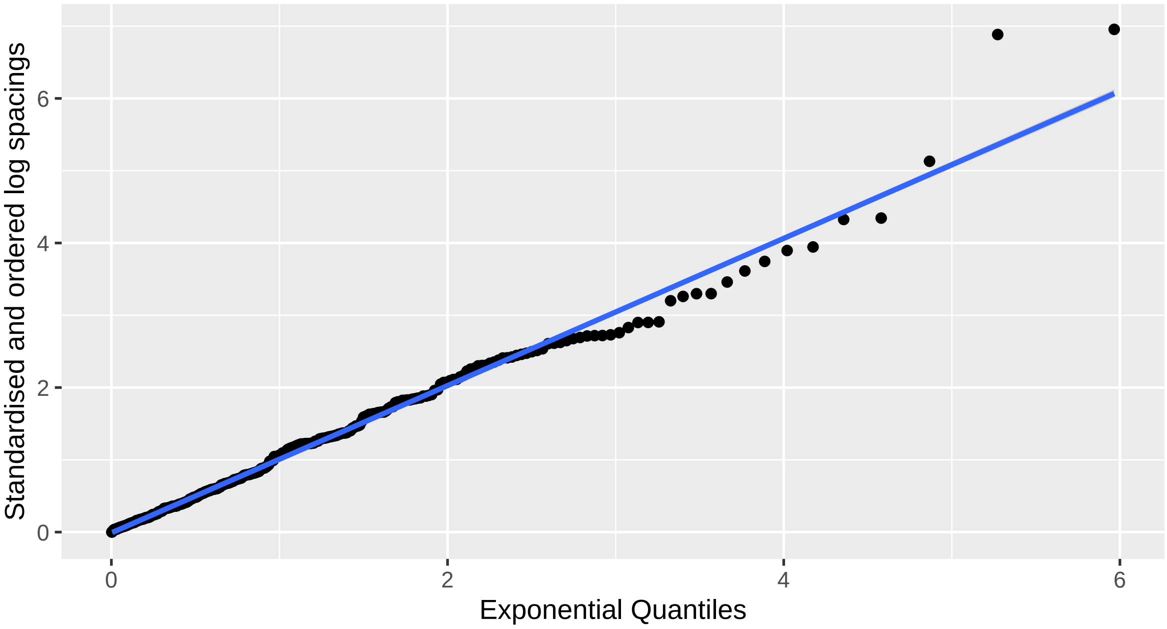
Test fit

We transform the data to exponentials using the log spacings.

```
j <- seq_len(k_example)
threshold <- y_reduced[k_example+1]
v <- -diff(log(y_reduced[seq_len(k_example+1)]))
evi_guess <- 0.2
pars_of_interest <- c('evi', 'm')
init_rates <- (j+m_example)/evi_guess
```

As an intermediate check we transform the new data to standard exponentials using the known parameter values and doing an exponential qqplot. This is just to make sure the spacings transformation is performing as expected.

```
check_v <- v * init_rates
check_v |> exp() |> qqplotfunc(k = length(check_v), ylabel = "Standardised and ordered log
```



Now we simulate from the posterior

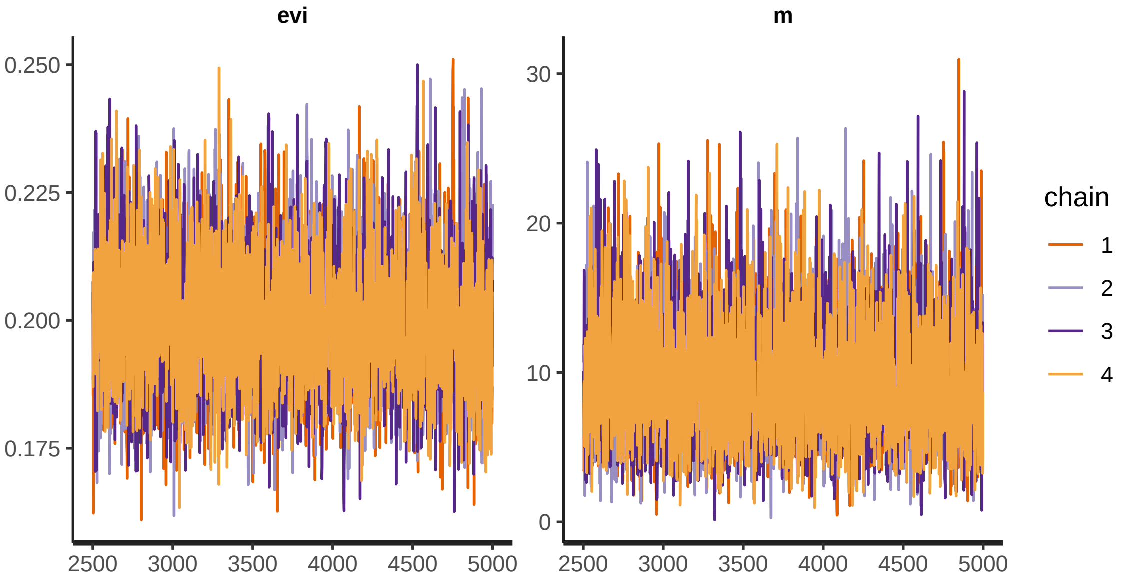
```
standardspacings |>
  sampling(data = list(y = v, k = length(v), lambda = 0.1, a = 1),
           pars = pars_of_interest,
           chains = mycores,
           iter = 5000,
           init = \(core) {list(
             m = m_example,
             evi = evi_guess,
             rate = init_rates
           )}) -> post_fit
standardspacings2 |>
  sampling(data = list(y = v, k = length(v), lambda = 1, a = 1),
           pars = pars_of_interest,
           chains = mycores,
           iter = 5000,
           init = \(core) {list(
             m = m_example,
             evi = evi_guess,
             rate = init_rates
           )}) -> post_fit2
```

And examine the results

```
(post_fit |> summary(pars = pars_of_interest))$summary |> kable(digits = 3)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
evi	0.200	0.000	0.012	0.178	0.191	0.200	0.208	0.226	3147.228	1.001
m	9.504	0.065	3.864	3.042	6.722	9.149	11.959	17.870	3532.769	1.001

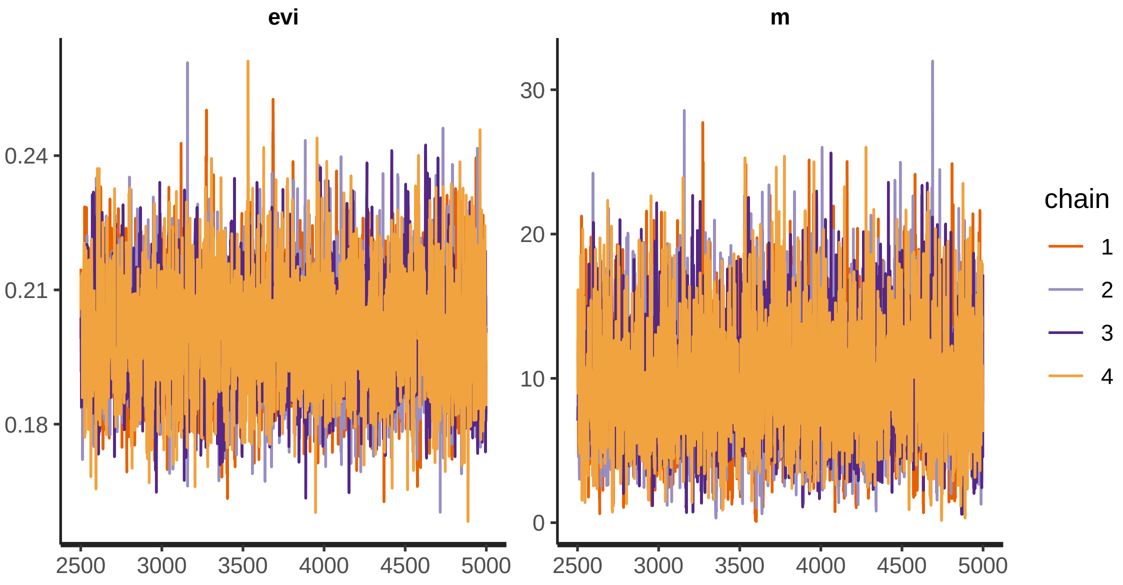
```
post_fit |> traceplot(pars = pars_of_interest)
```



```
post_fit |> rstan::extract(pars = pars_of_interest) -> post_sims
(post_fit2 |> summary(pars = pars_of_interest))$summary |> kable(digits = 3)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
evi	0.200	0.000	0.013	0.177	0.192	0.200	0.209	0.227	3313.774	1
m	9.641	0.068	4.147	2.572	6.686	9.247	12.234	18.797	3761.568	1

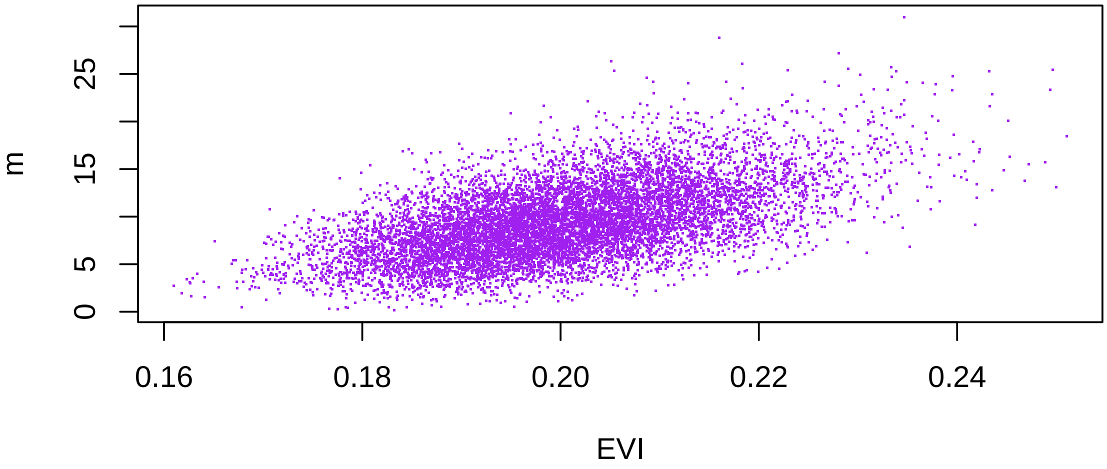
```
post_fit2 |> traceplot(pars = pars_of_interest)
```



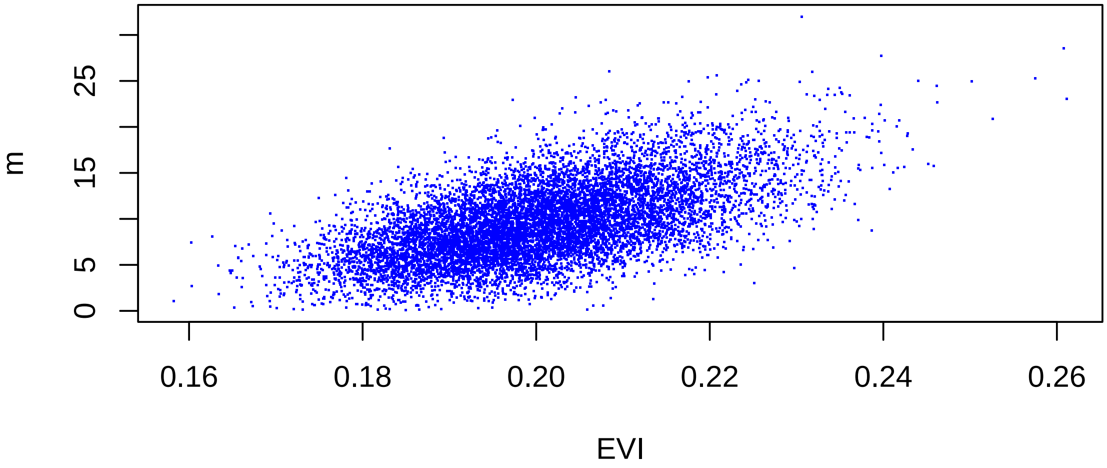
```
post_fit2 |> rstan::extract(pars = pars_of_interest) -> post_sims2
```

Raw posterior simulations:

```
plot(post_sims$evi, post_sims$m, pch = '.', col = 'purple', xlab = "EVI", ylab = "m")
```



```
plot(post_sims2$evi, post_sims2$m, pch = '.', col = 'blue', xlab = "EVI", ylab = "m")
```



While EVI estimation remains a very difficult problem with a lot of uncertainty; we see a good fit and accurate estimation in this carefully selected example.

Simulation study

Now we formally investigate the efficacy of the chosen priors. The details of the simulation study and the calculated statistics of interest are all saved to an Excel file for evaluation and distribution.

Scenarios

Since there are many aspects to this problem that may vary, we implement a generic and flexible structure for testing that consists of constructing a table of scenarios first.

Here we define k to be the actual number of log spacings used in the fitting, beyond the missing values. We start with N observations, exclude the m largest, and then model using the next $k+1$ largest to get k log spacings beyond the assumed threshold of the $(k+1)^{th}$ largest observation.

```
reps_per_scenario <- 1000 # Set to say 1000 for main run

base_scenarios <- rbind(
  expand.grid(
    Distribution = c("Pareto", "Frechet"),
    EVI_true = c(0.2),
    m_true = c(5, 10),
    k = c(100, 200, 300),
    N = 500,
    lambda = c(0.4, 0.3, 0.2, 0.1, 0.05, 0.02, 0.01),
    model = "Exponential decay",
    a = c(1),
    spacings = c("Standard") # vs Generalised
  ),
  expand.grid(
    Distribution = c("Pareto", "Frechet"),
    EVI_true = c(0.2),
    m_true = c(5, 10),
    k = c(100, 200, 300),
    N = 500,
    lambda = c(0.05, 0.01, 0.005, 0.001),
    model = "Exponential decay",
    a = c(2),
    spacings = c("Standard") # vs Generalised
  ),
  expand.grid(
    Distribution = c("Pareto", "Frechet"),
    EVI_true = c(0.2),
    m_true = c(5, 10),
    k = c(100, 200, 300),
    N = 500,
    lambda = c(0.1, 0.5, 1, 2, 3),
    model = "Inverse decay",
    a = c(0.5, 1),
    spacings = c("Standard")
  )
)

n_scenarios <- nrow(base_scenarios)
base_scenarios$Scenario <- seq_len(n_scenarios)
scenarios <- base_scenarios[rep(seq_len(n_scenarios), each = reps_per_scenario), ]
scenarios$rep <- rep(seq_len(reps_per_scenario), times = n_scenarios)
scenarios$ID <- seq_len(nrow(scenarios))
```

```
add_to_wb <- function(data_frame_to_add, wb, sheet_name) {
  wb |> addWorksheet(sheet_name)
  wb |> writeDataTable(sheet_name, data_frame_to_add, tableName = sheet_name)
  wb |> freezePane(sheet_name, firstRow = TRUE)
  data_frame_to_add |>
    sapply(\(v) {max(nchar(as.character(v)))}) |>
    pmax(nchar(names(data_frame_to_add))) -> col_widths
  wb |> setColWidths(sheet_name, seq_len(ncol(data_frame_to_add)),
               col_widths + 3)
  wb
}
```

```
wb <- createWorkbook("SimstudyBasic")
base_scenarios |> add_to_wb(wb, "Scenarios") -> wb
scenarios |> add_to_wb(wb, "Settings") -> wb
```

For each scenario table row we will simulate a sample, fit the model, and return a set of statistics.

Fitting function

We want to define a function that runs through a single scenario table row.

One thing that we do differently here than in the example is that we will **not** use knowledge of the scenario to initialise the model fitting, instead remaining objective by relying on Stan's default initial value algorithm.

Another thing is that we will use only a single chain within each repetition, relying on the repetitions to account for the between chain variability, and using a fixed chain length throughout. For a real sample chain length should be adaptive.

First we define the functions that calculate the relevant statistics. This function only seems long because we return a long list of statistics (you never know what might come in handy later).

```r
pars_of_interest <- c('evi', 'm')
shortestinterval <- function(postsims, width=0.95) { # Coded by Sean van der Merwe, UFS
  postsims |> sort() -> sorted.postsims
  round(length(postsims)*width) -> gap
  sorted.postsims |> diff(gap) |> which.min() -> pos
  sorted.postsims[c(pos, pos + gap)] }
get_stats <- function(postsims, true_value) {
  true_value <- true_value |> unname()
  dens <- density(postsims)
  interval1 <- shortestinterval(postsims)
  interval2 <- quantile(postsims, c(0.025, 0.5, 0.975)) |> unname()
  c(Mode = dens$x[which.max(dens$y)],
    Median = interval2[2],
    Mean = mean(postsims),
    L = interval1[1],
    U = interval1[2],
    L0.025 = interval2[1],
    U0.975 = interval2[3],
    Mode_error = dens$x[which.max(dens$y)] - true_value,
    Median_error = interval2[2] - true_value,
    Mean_error = mean(postsims) - true_value,
    Coverage = (interval1[1] < true_value) && (interval1[2] > true_value),
    Coverage_symmetric = (interval2[1] < true_value) && (interval2[3] > true_value),
    Width = interval1[2] - interval1[1],
    Width_symmetric = interval2[2] - interval2[1]
  )
}
```

```r
run_scenario <- function(scenario) {
  # Generate sample
  if (scenario$Distribution == "Pareto") {
    actuar::rpareto1(scenario$N, 1/scenario$EVI_true, 1) |>
      sort(decreasing = TRUE) -> y_full
  } else {
    actuar::rinvweibull(scenario$N, 1/scenario$EVI_true, 1) |>
      sort(decreasing = TRUE) -> y_full
  }
  # Remove top values
  y_full[-(1:scenario$m_true)] -> y_reduced
  # Calculate spacings
  v <- -diff(log(y_reduced[seq_len(scenario$k+1)]))
  # Select model
  if (scenario$model == "Exponential decay") {
    compiled_model <- standardspacings
  } else {
    compiled_model <- standardspacings2
  }
  # Do simulations
  compiled_model |>
      sampling(data = list(y = v,
                           k = length(v),
                           lambda = scenario$lambda,
                           a = scenario$a
                           ),
               pars = pars_of_interest,
               chains = 1,
               iter = 5000
      ) |> rstan::extract(pars = pars_of_interest) -> post_sims
  # Return statistics
  list(evi_stats = get_stats(post_sims$evi, scenario$EVI_true),
       m_stats = get_stats(post_sims$m, scenario$m_true)
  )
}
```

Simulation study run

First we initialise a parallel cluster, and then apply our function to each scenario table row in parallel and save the results.

```
library(parallel)
cl <- makeCluster(mycores)
cl |> clusterEvalQ({
  library(rstan)
  standardspacings <- readRDS("standardspacingsmodel1.rds")
  standardspacings2 <- readRDS("standardspacingsmodel2.rds")
  options(mc.cores = 1)
}) |> invisible()
cl |> clusterExport(c(
  'shortestinterval', 'get_stats', 'run_scenario', 'pars_of_interest'
))

scenariolist <- scenarios |> nrow() |> seq_len() |> lapply(\(i) {
  scenarios[i,]
})
system.time({
  parLapplyLB(cl, scenariolist, run_scenario) -> all_results_raw
})

cl |> stopCluster()

all_results_raw |> saveRDS("standardspacingsresults.rds")
```

Present results

The next step is to combine the results, and then summarise over repetitions.

```
evi_results_raw <- all_results_raw |> sapply(\(results) {results$evi_stats})
m_results_raw <- all_results_raw |> sapply(\(results) {results$m_stats})
stat_names <- row.names(m_results_raw)
num_stats <- length(stat_names)
all_results <- scenarios[rep(seq_len(nrow(scenarios)), each = num_stats), ]
all_results$Stat_name <- rep(stat_names, times = nrow(scenarios))
all_results$EVI <- c(evi_results_raw)
all_results$m <- c(m_results_raw)
# all_results |>
#   pivot_wider(values_from = c(EVI, m), names_from = Stat_name) |>
#   add_to_wb(wb, "All_Results") -> wb
all_results <- all_results |>
  pivot_longer(c(EVI, m), names_to = 'Parameter', values_to = 'Statistic')
```

```
rm(all_results_raw, evi_results_raw, m_results_raw)
```

However, there are hundreds of ways to summarise the results. Let us consider a selection of statistics at first and then expand as needed.

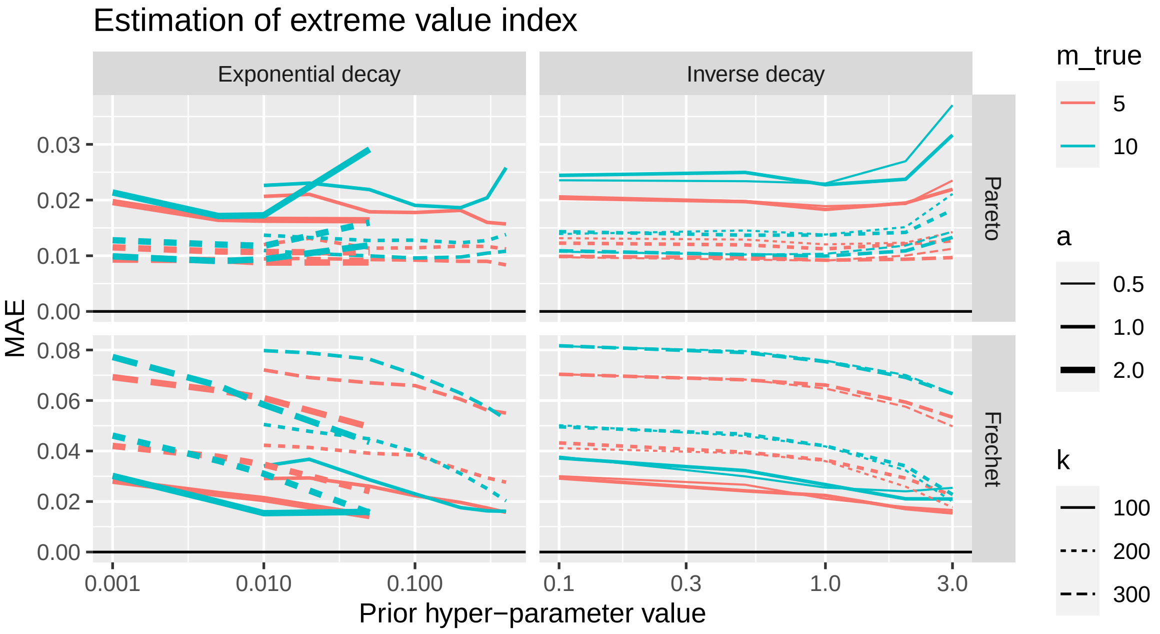
Posterior Median, Mode, and Mean Estimators

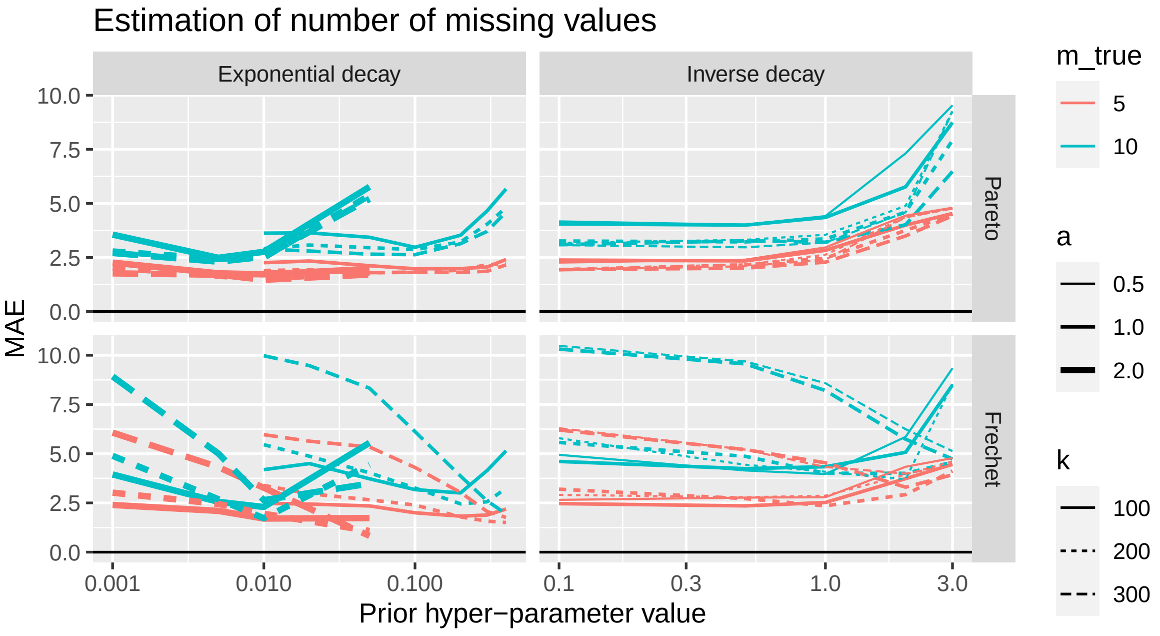
In a simulation study it is usually of primary interest to establish when and where we can accurately estimate the parameters of interest. We consider three popular estimators based on the posterior distributions.

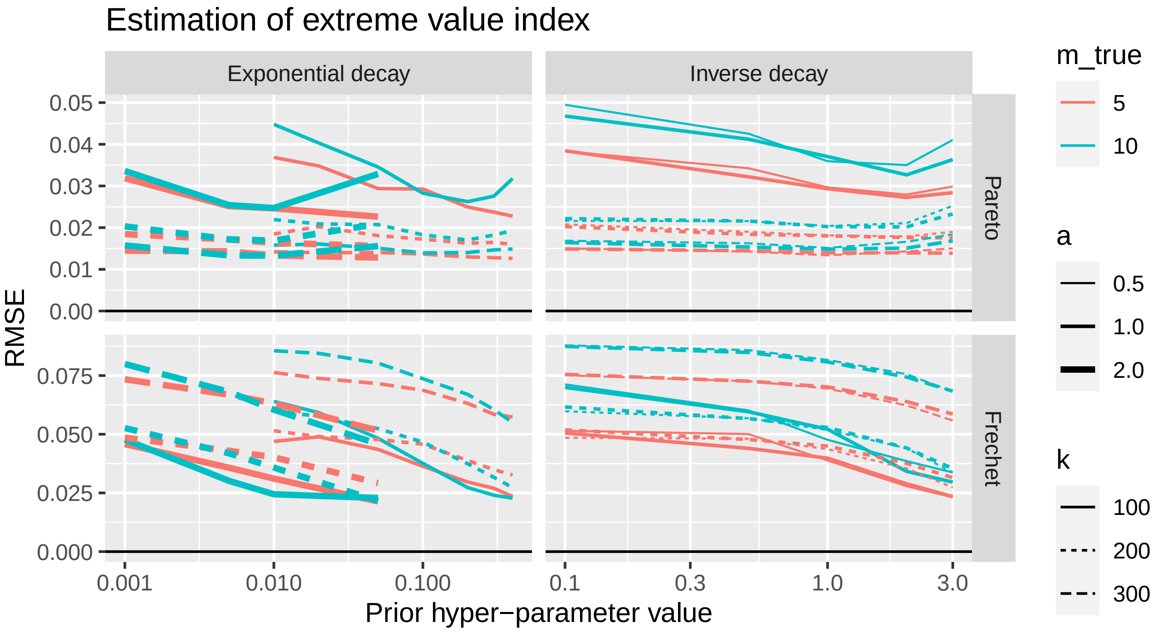
It is common practice to draw the summary statistics of an estimator relative to the true value for easier analysis; although both absolute and relative statistics are saved to Excel (*they are not tabulated here as the tables do not neatly fit on a page as is*).

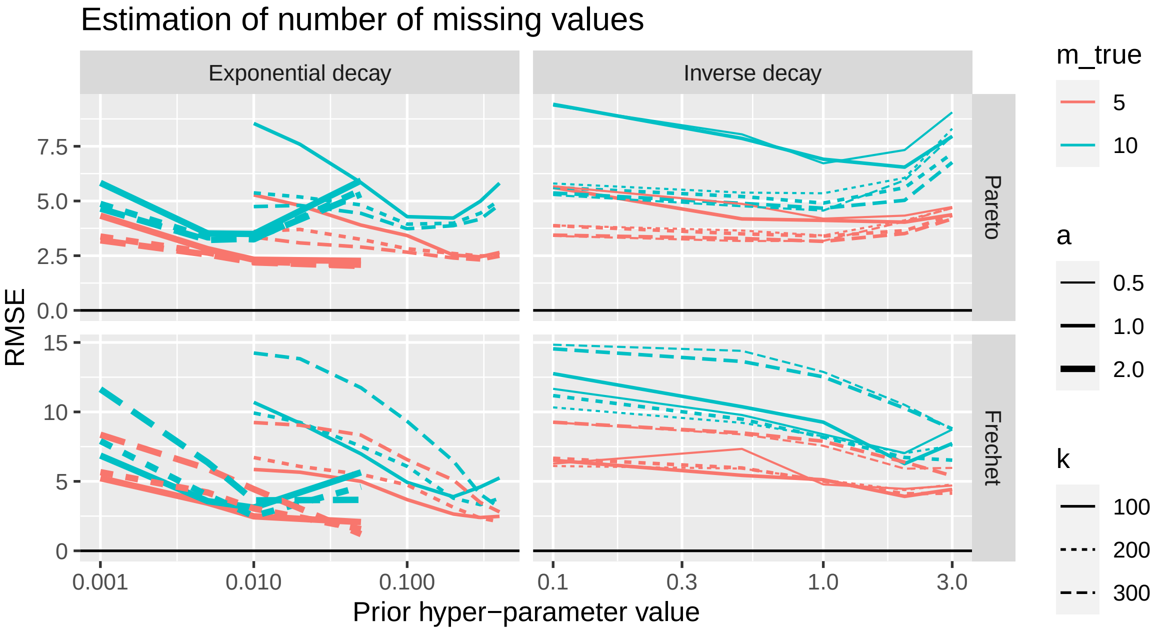
```
stats_of_interest <- c("Mode_error", "Median_error", "Mean_error")
nice_titles <- c("Posterior mode - true value", "Posterior median - true value", "Posterio
for (current_stat in seq_along(stats_of_interest)) {
  cat('\n\n##', nice_titles[current_stat], '\n\n')
  all_results |>
    filter(Stat_name == stats_of_interest[current_stat]) |>
    group_by(Parameter, Scenario) |>
    summarise(
      Bias = mean(Statistic),
      RMSE = sqrt(mean(Statistic^2)),
      MAE = median(abs(Statistic)),
      .groups = "keep"
    ) |> left_join(base_scenarios, by = join_by(Scenario)) |>
    mutate(Group = as.factor(paste0('m = ', m_true, '; k = ', k, '; a = ', a)),
           Bias_Squared = Bias^2,
           k = as.factor(k),
           m_true = as.factor(m_true)
          ) -> summary_table

  # summary_table |> kable(digits = 3) |> print()
  tryCatch({
    summary_table |> add_to_wb(wb, stats_of_interest[current_stat]) -> wb
  }, error = \(e) {print(e); cat('\n\n')})
  a_scale_options <- unique(summary_table$a) |> sort()

  for (stat in c('MAE', 'RMSE', 'Bias_Squared', 'Bias')) {
    (summary_table |>
      filter(Parameter == 'EVI') |>
      ggplot(aes(x = lambda, y = !!sym(stat),
                 group = Group, colour = m_true,
                 linetype = k, linewidth = a)) +
      geom_line() +
      scale_x_log10() +
      scale_linewidth(range = c(0.4, 1.2),
                      breaks = a_scale_options) +
      ggtitle('Estimation of extreme value index') +
      facet_grid(cols = vars(model),
                 rows = vars(Distribution),
                 scales = 'free') +
      xlab('Prior hyper-parameter value') +
      ylab(stat) + geom_hline(yintercept = 0)
    ) |> print()
    cat('\n\n')
    (summary_table |>
      filter(Parameter == 'm') |>
      ggplot(aes(x = lambda, y = !!sym(stat),
                 group = Group, colour = m_true,
                 linetype = k, linewidth = a)) +
      geom_line() +
      scale_x_log10() +
      scale_linewidth(range = c(0.4, 1.2),
                      breaks = a_scale_options) +
      ggtitle('Estimation of number of missing values') +
      facet_grid(cols = vars(model),
                 rows = vars(Distribution),
                 scales = 'free') +
      xlab('Prior hyper-parameter value') +
      ylab(stat) + geom_hline(yintercept = 0)
    ) |> print()
    cat('\n\n')
  }
  cat('---\n\n')
}
```

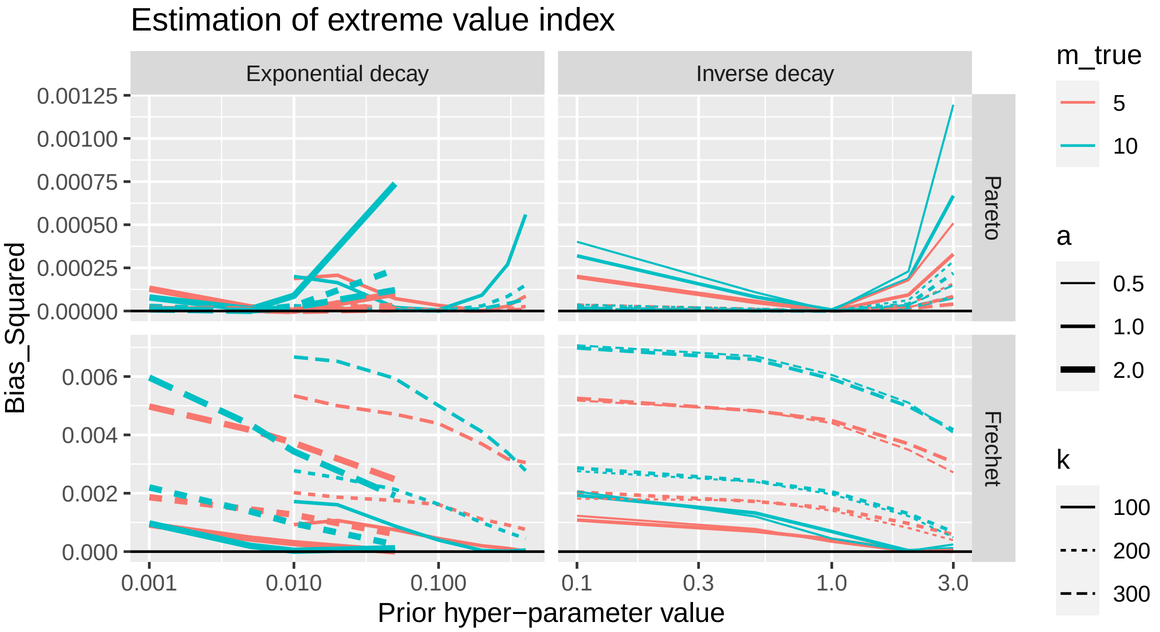
Posterior mode - true value

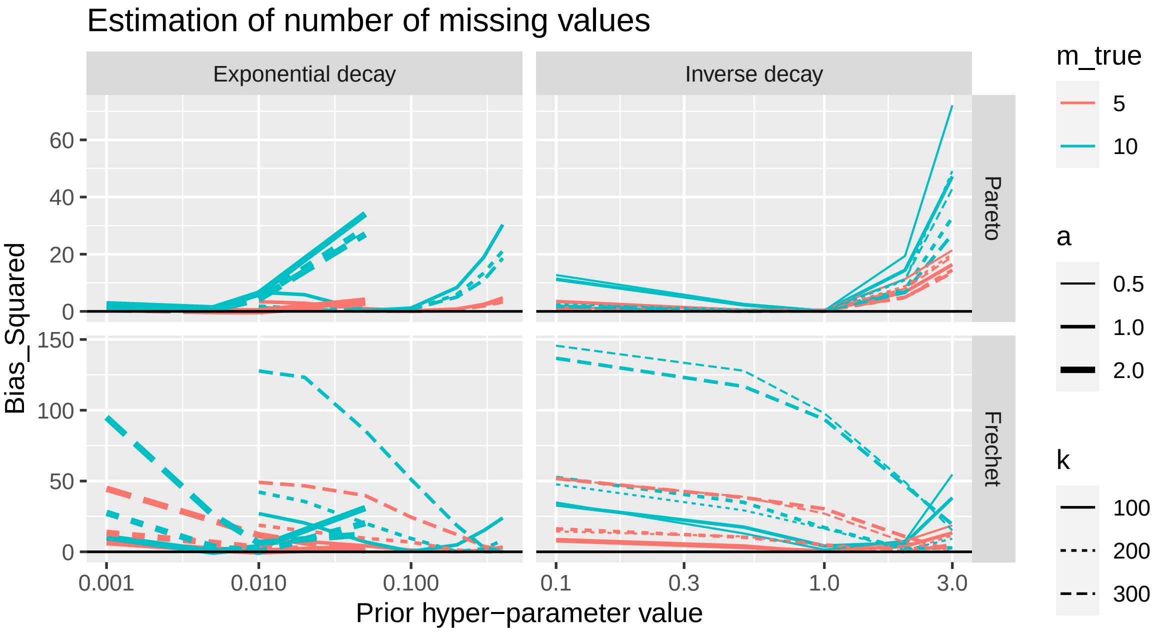


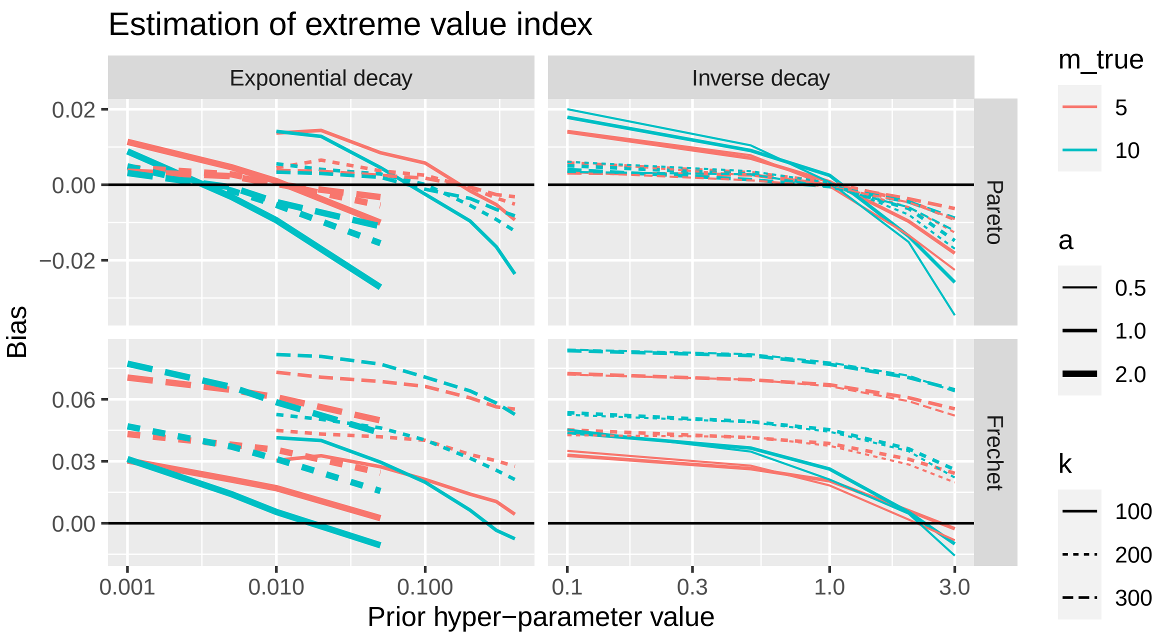


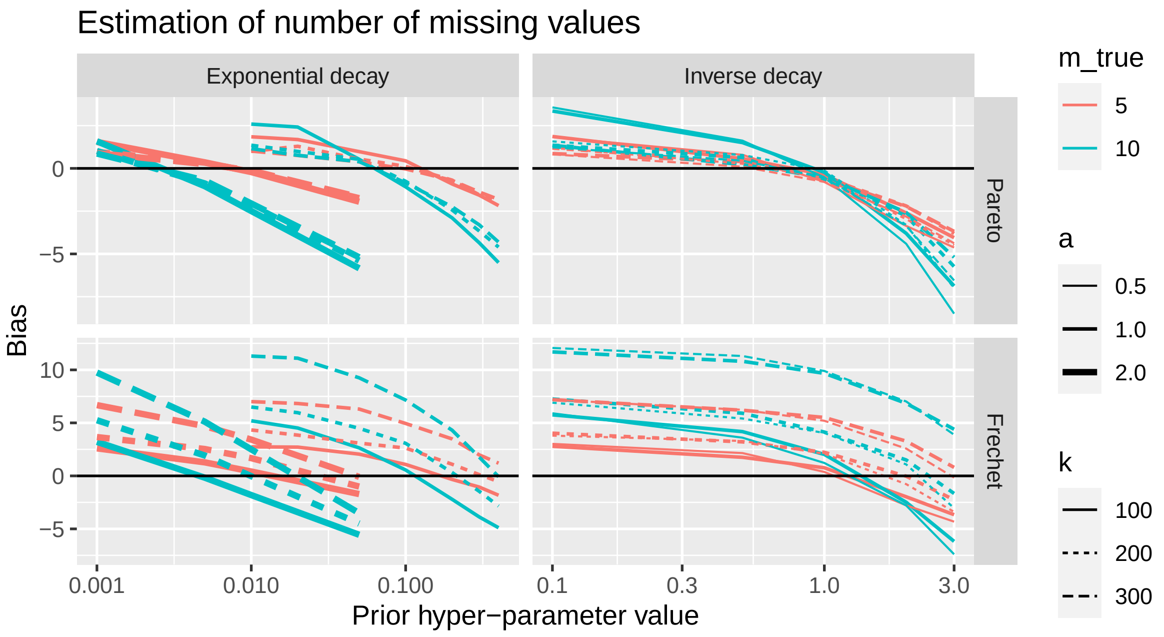

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
0.05
0.04
0.03
0.02
0.01
0.00
0.075
0.050
0.025
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
7.5
5.0
2.5
0.0
15
10
5
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

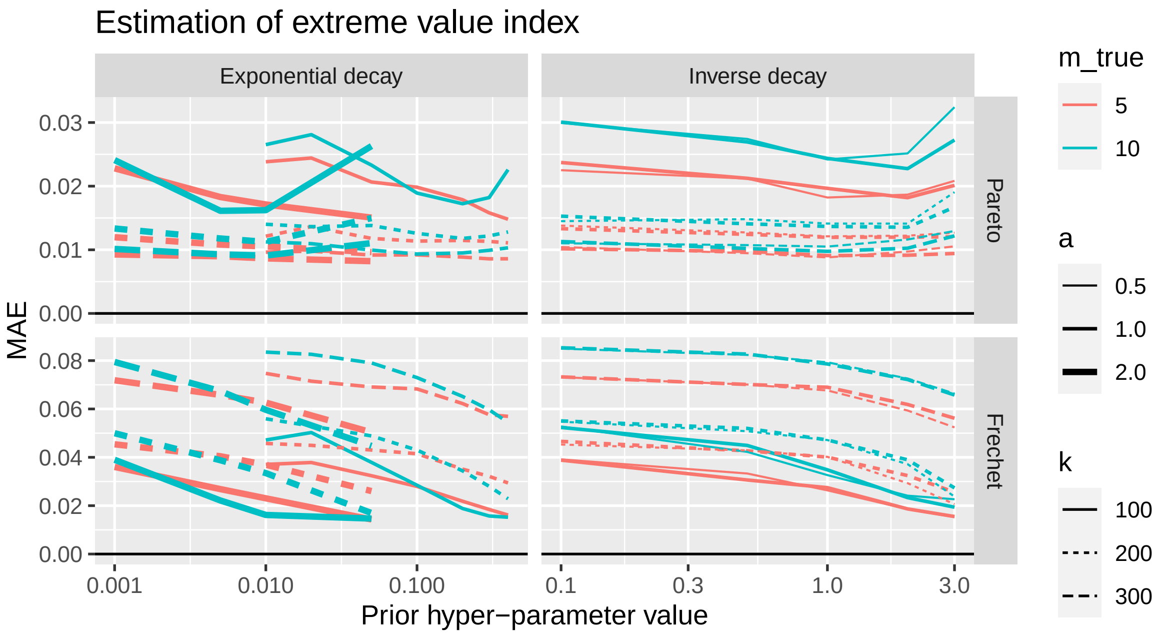

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0.006
0.004
0.002
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

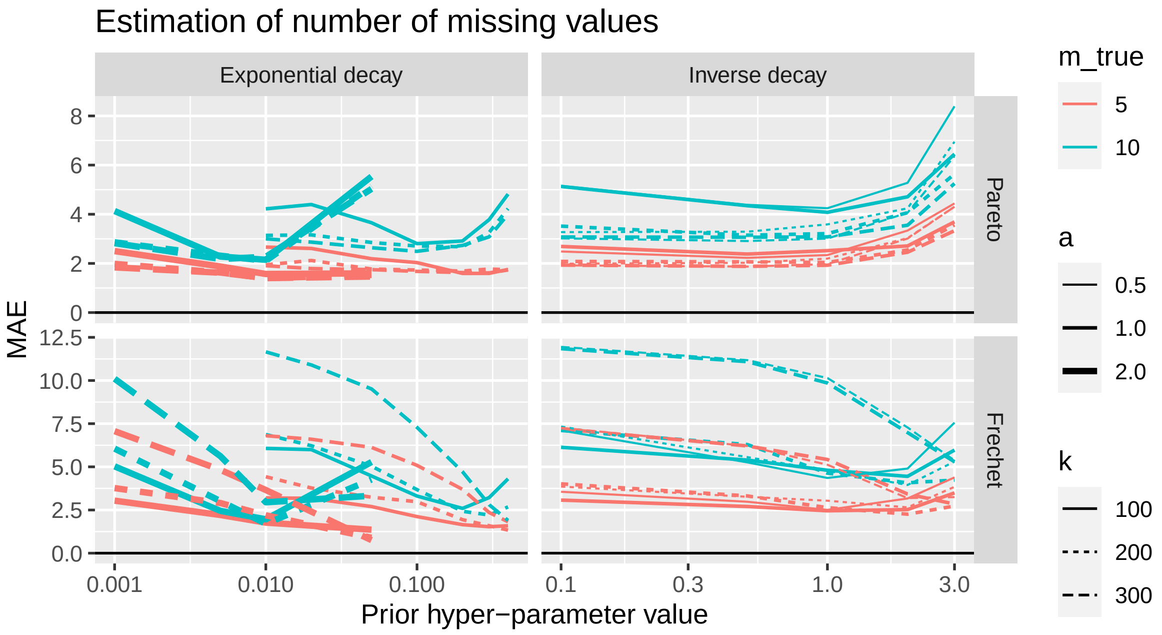

Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
60
40
20
0
150
100
50
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

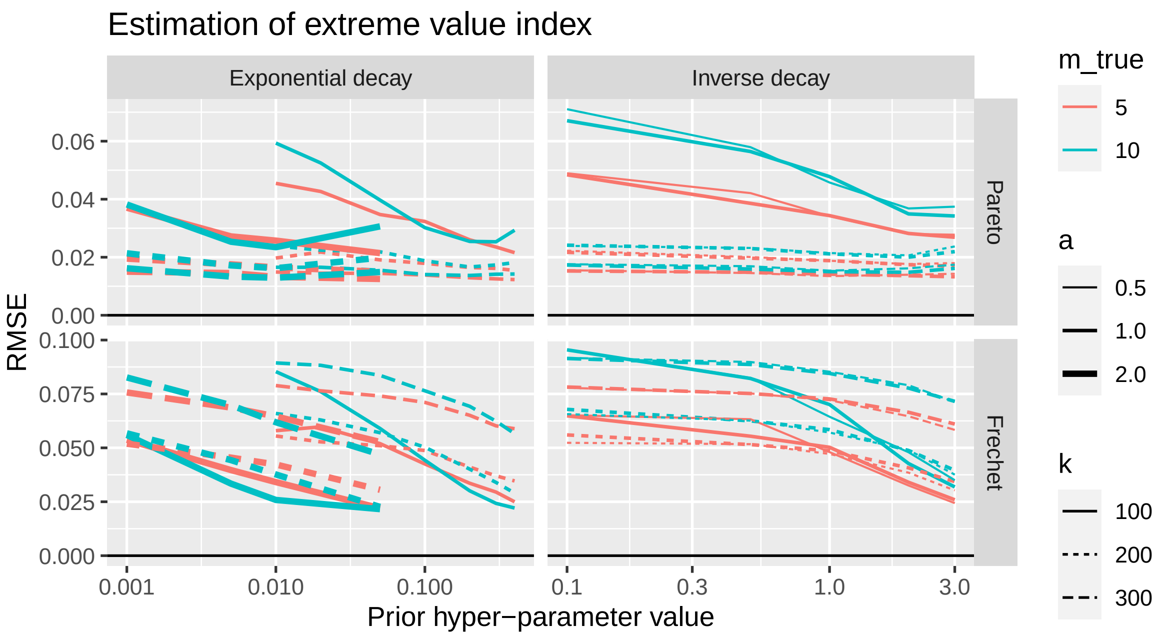

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias
0.02
0.00
−0.02
0.06
0.03
0.00
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

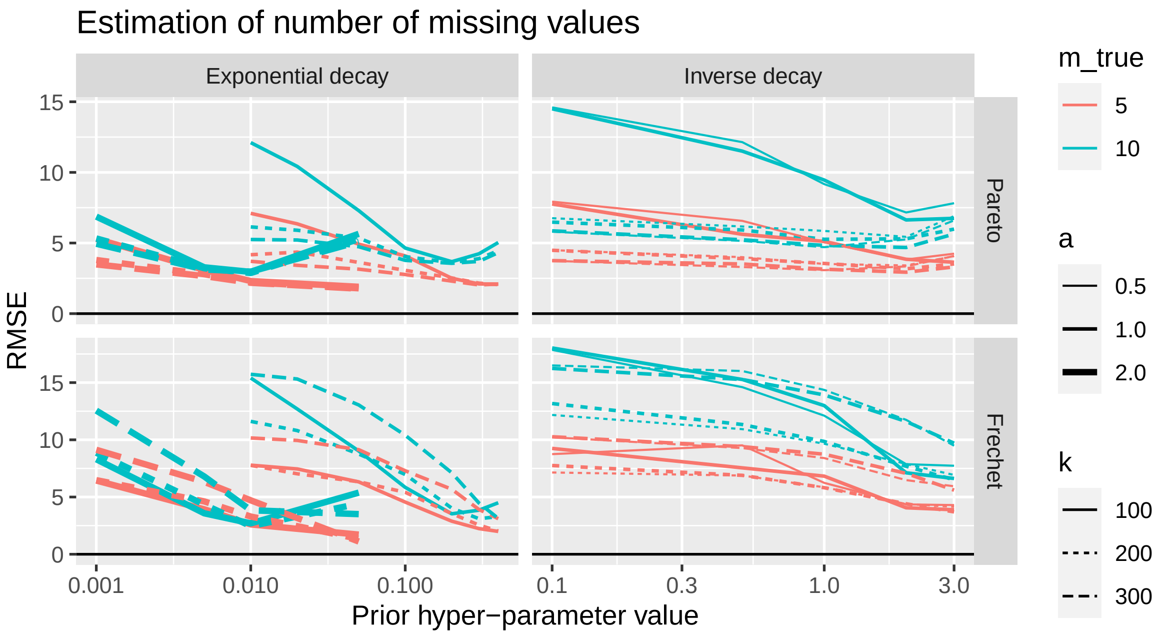

Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias
0
−5
10
5
0
−5
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

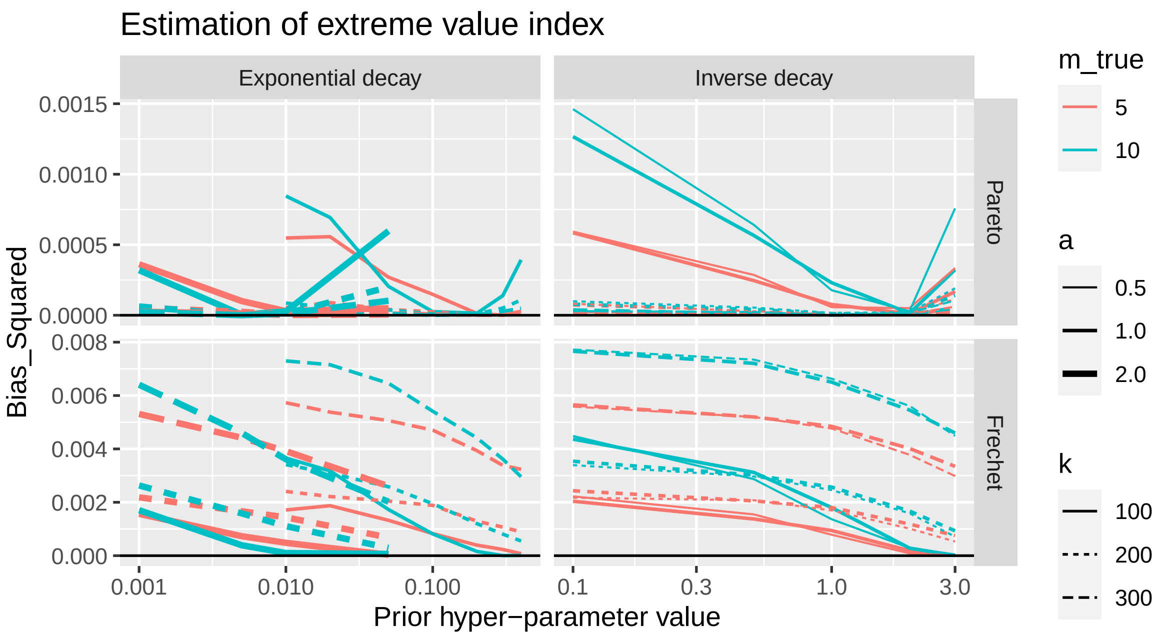
Posterior median - true value

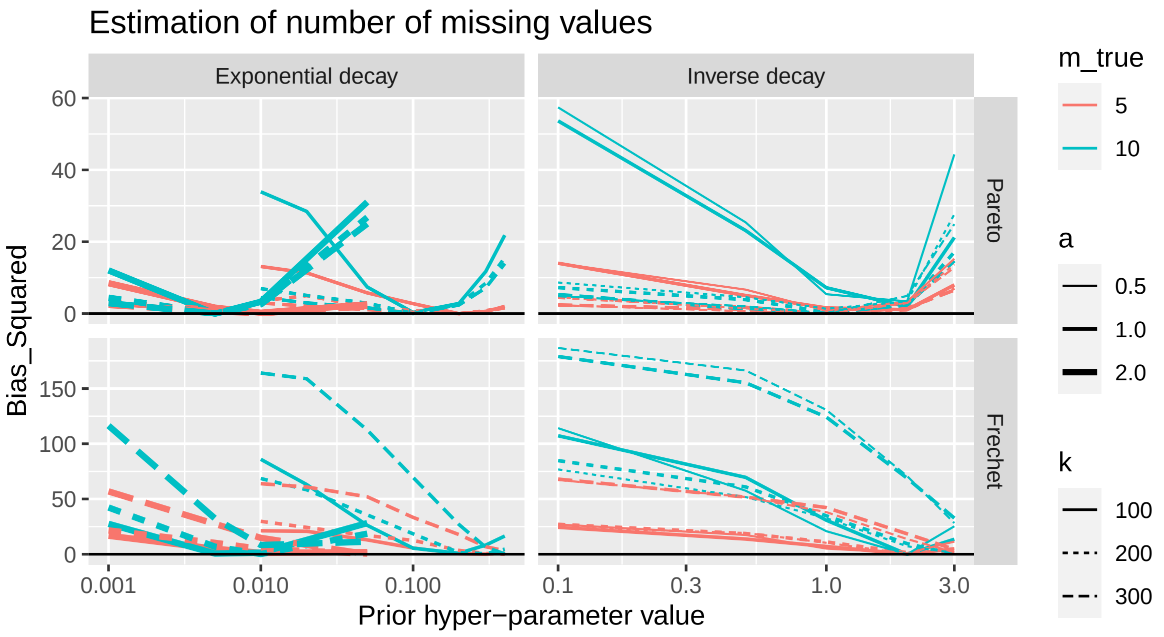


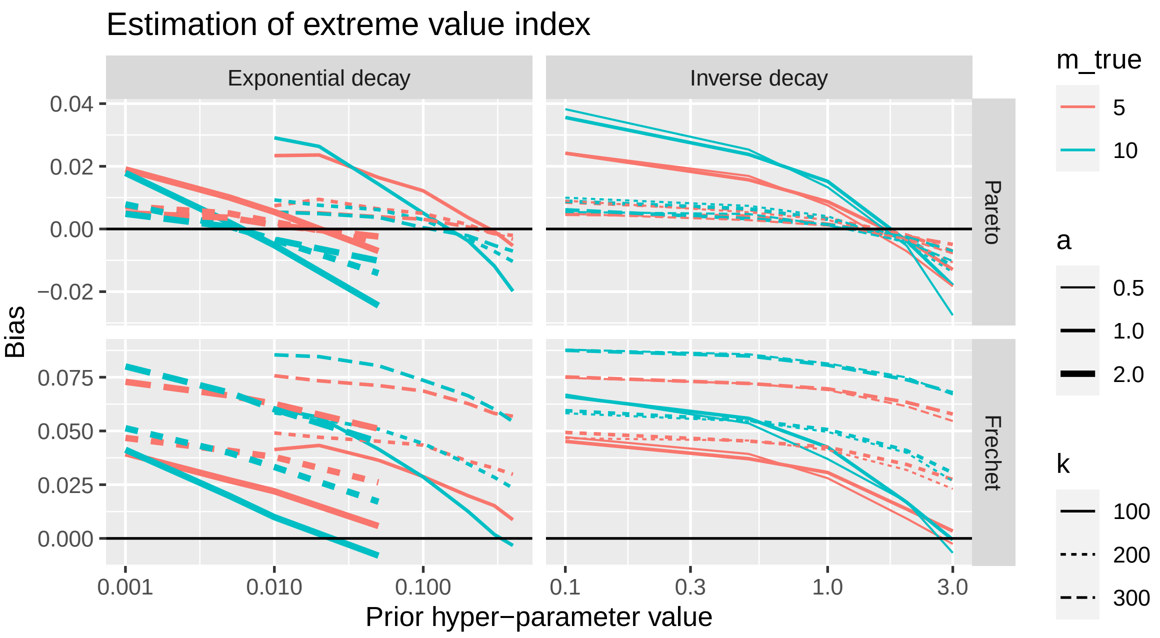


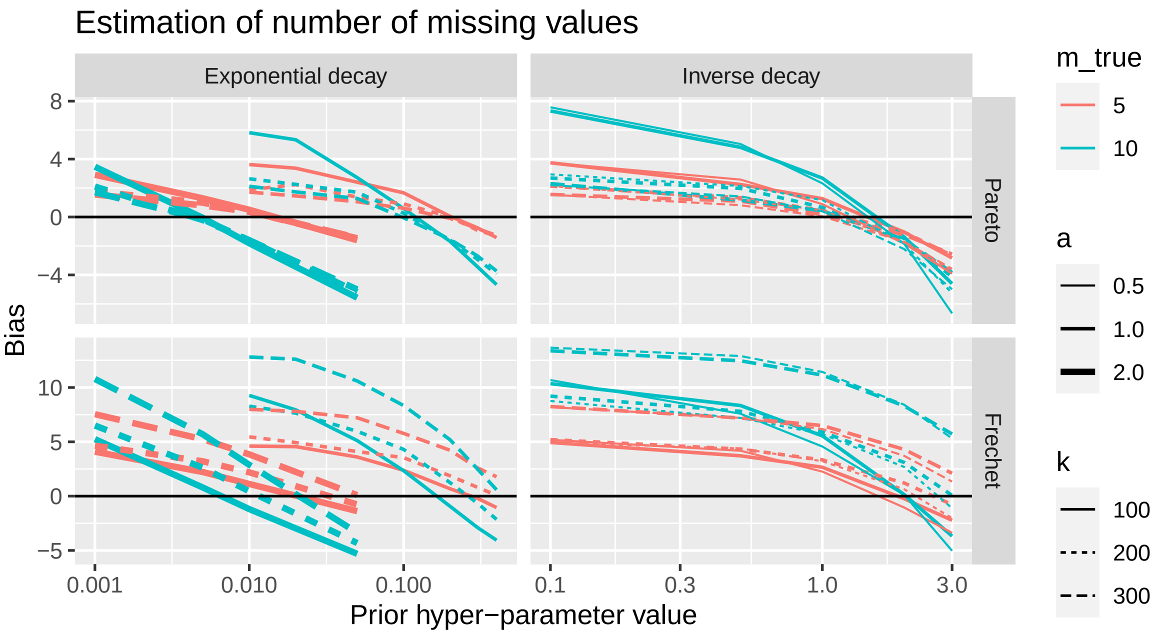

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
0.06
0.04
0.02
0.00
0.100
0.075
0.050
0.025
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
15
10
5
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

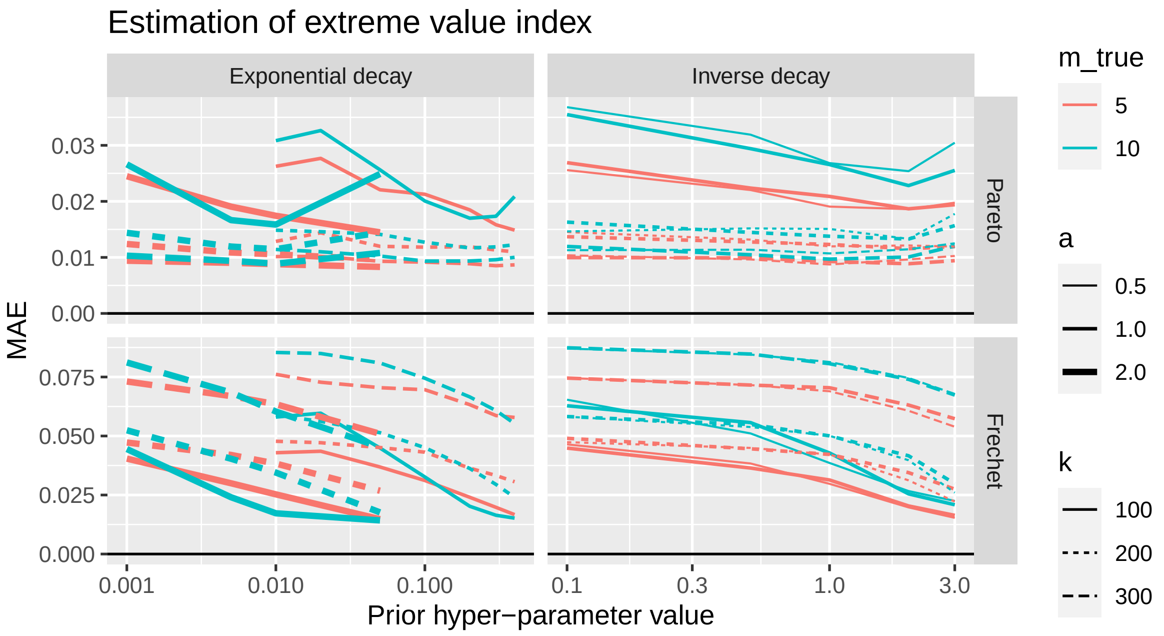

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
0.0015
0.0010
0.0005
0.0000
0.008
0.006
0.004
0.002
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

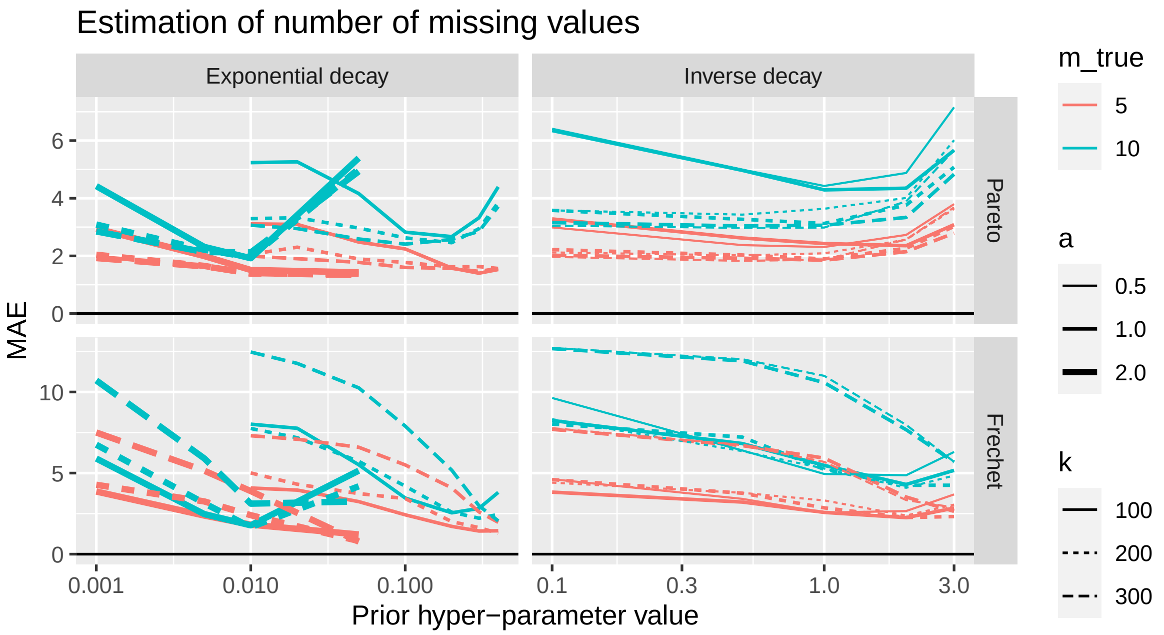

Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
60
40
20
0
150
100
50
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

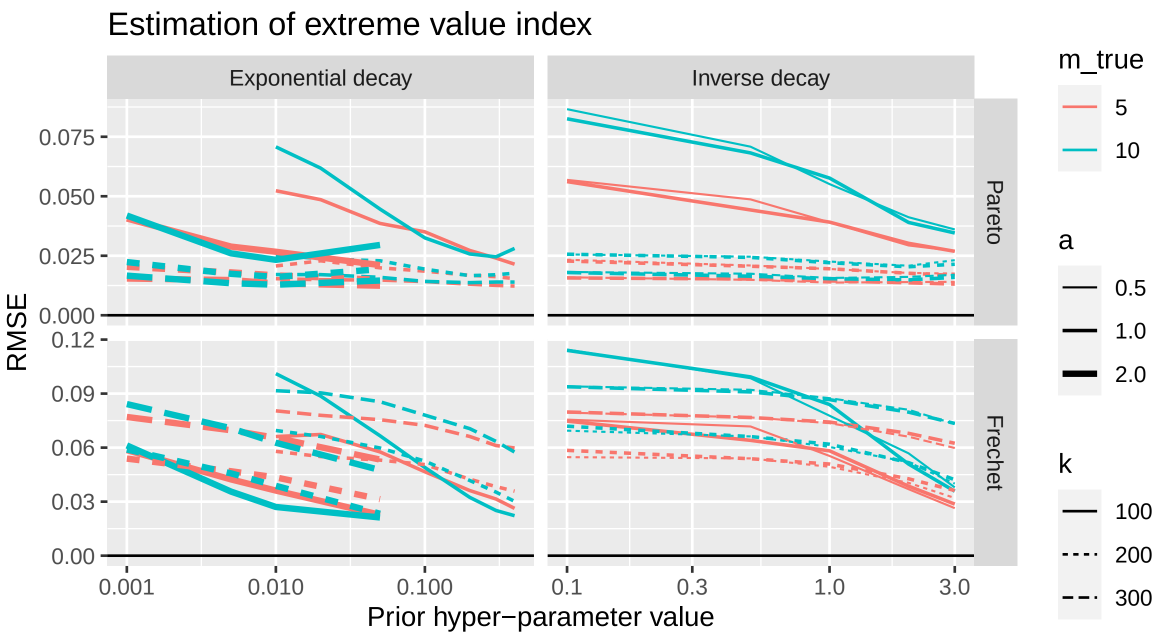

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias
0.04
0.02
0.00
−0.02
0.075
0.050
0.025
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

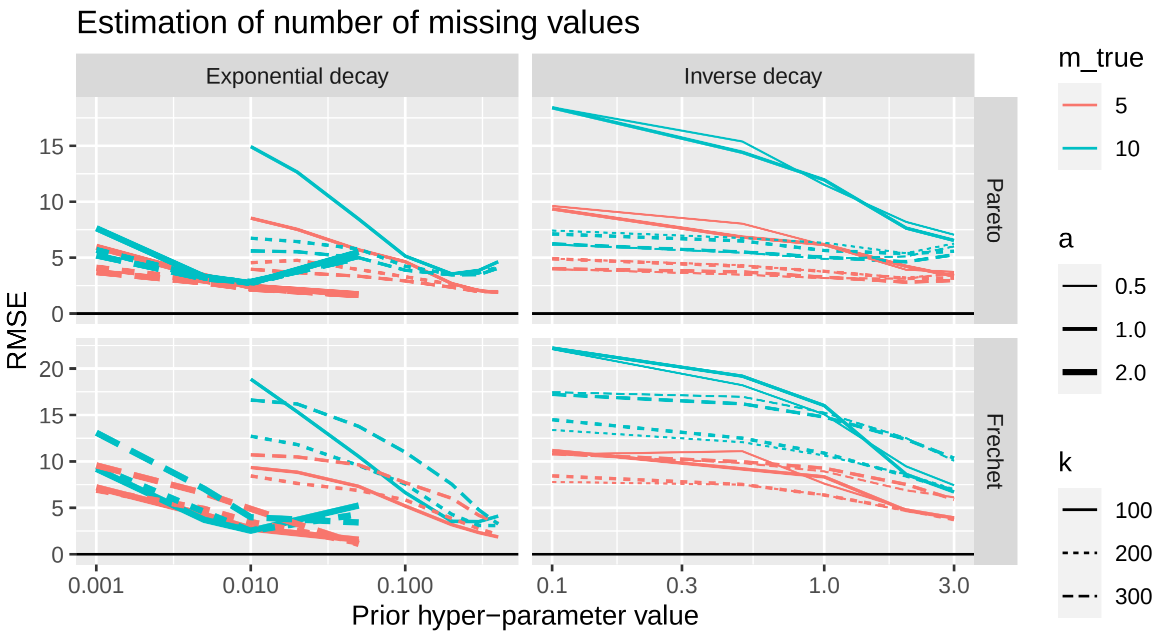

Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias
8
4
0
−4
10
5
0
−5
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

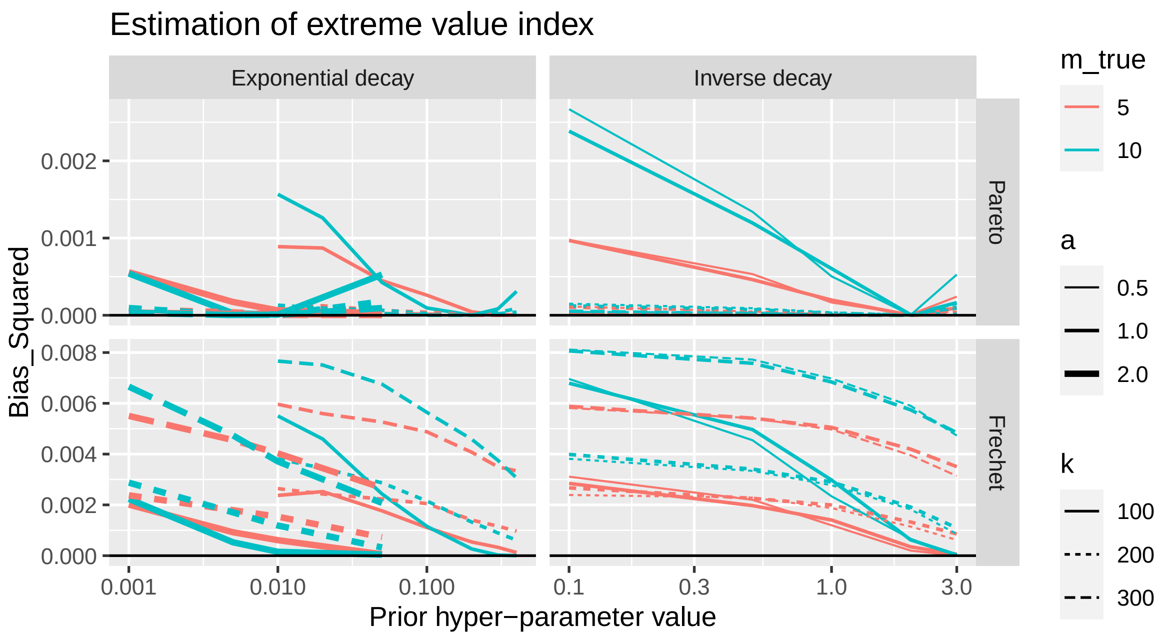
Posterior mean - true value

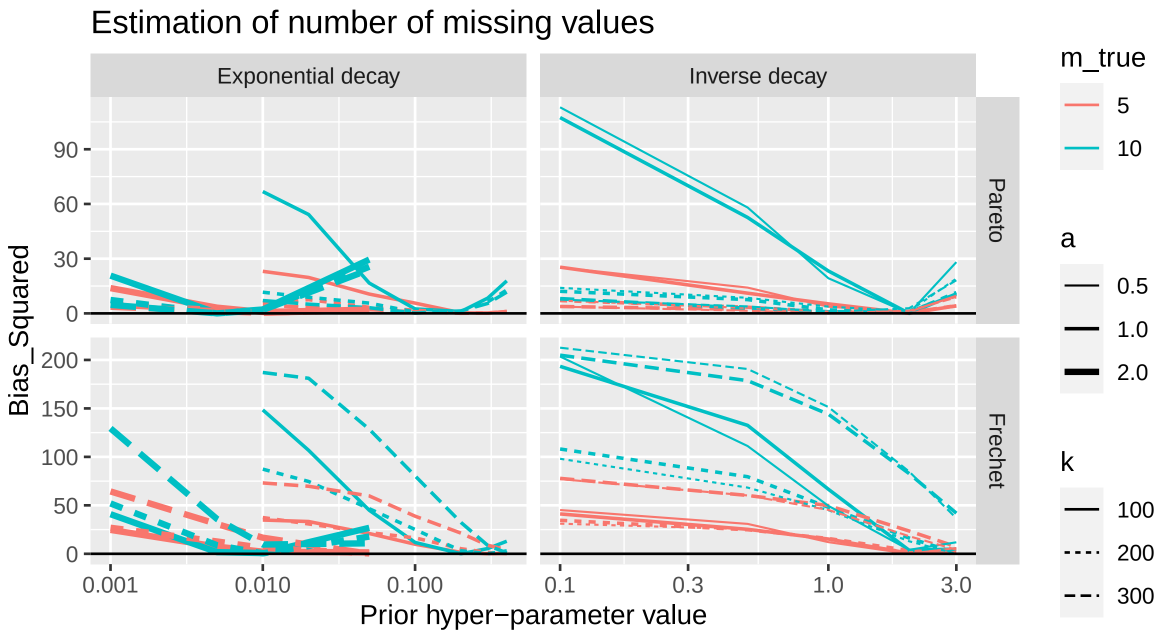


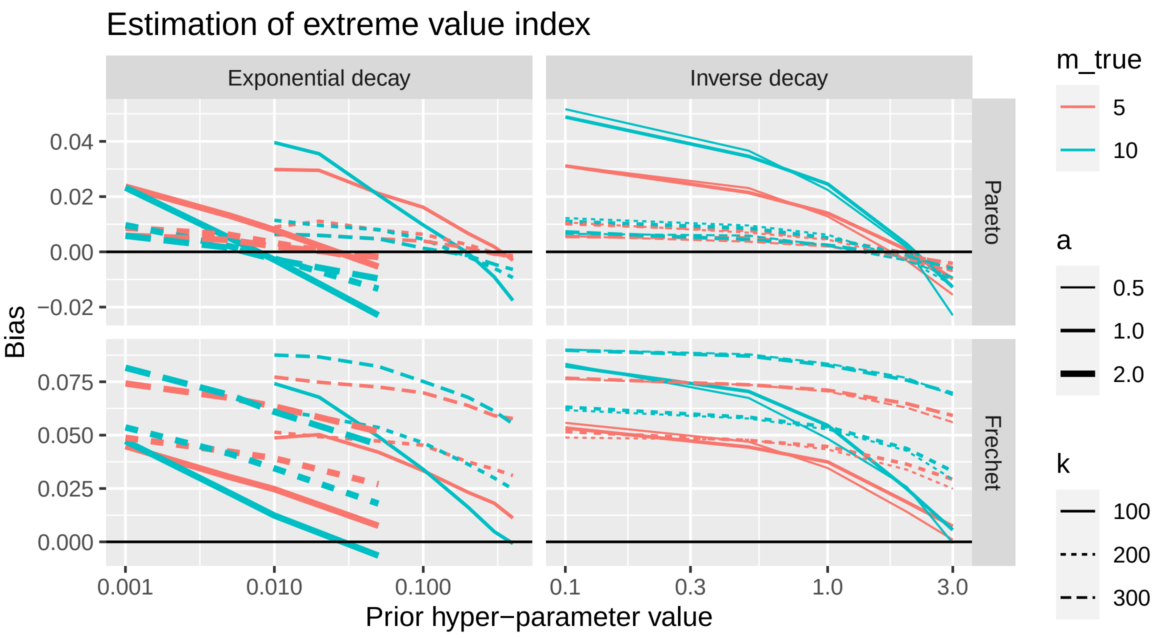


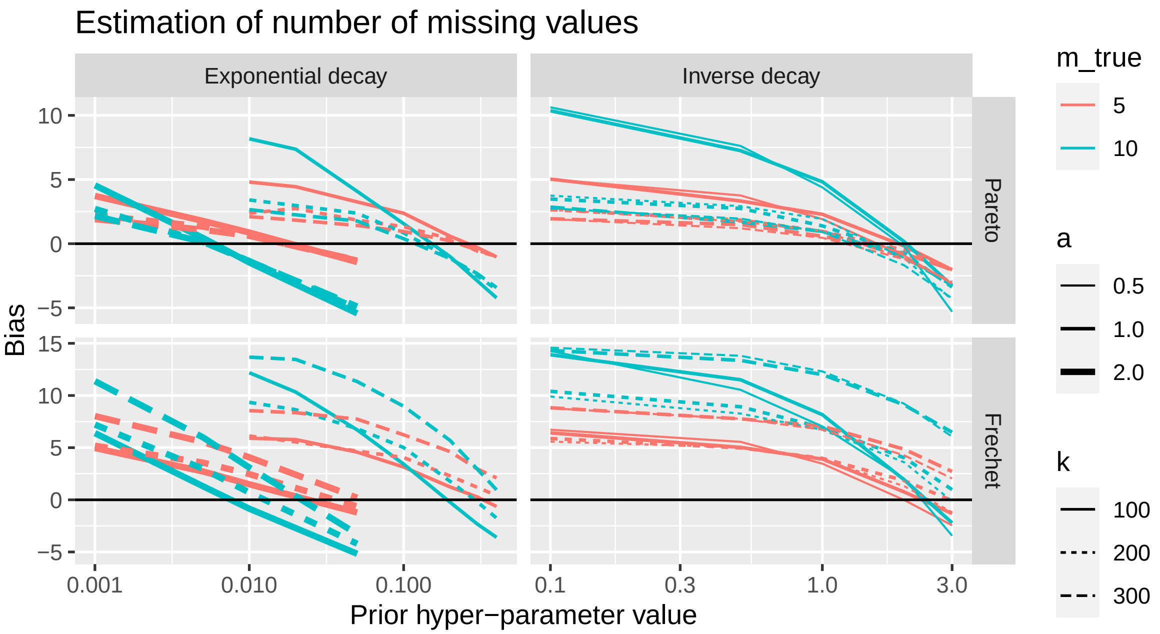

Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
0.075
0.050
0.025
0.000
0.12
0.09
0.06
0.03
0.00
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
15
10
5
0
20
15
10
5
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
0.002
0.001
0.000
0.008
0.006
0.004
0.002
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
90
60
30
0
200
150
100
50
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Bias
0.04
0.02
0.00
−0.02
0.075
0.050
0.025
0.000
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias
10
5
0
−5
15
10
5
0
−5
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

Coverage and interval width

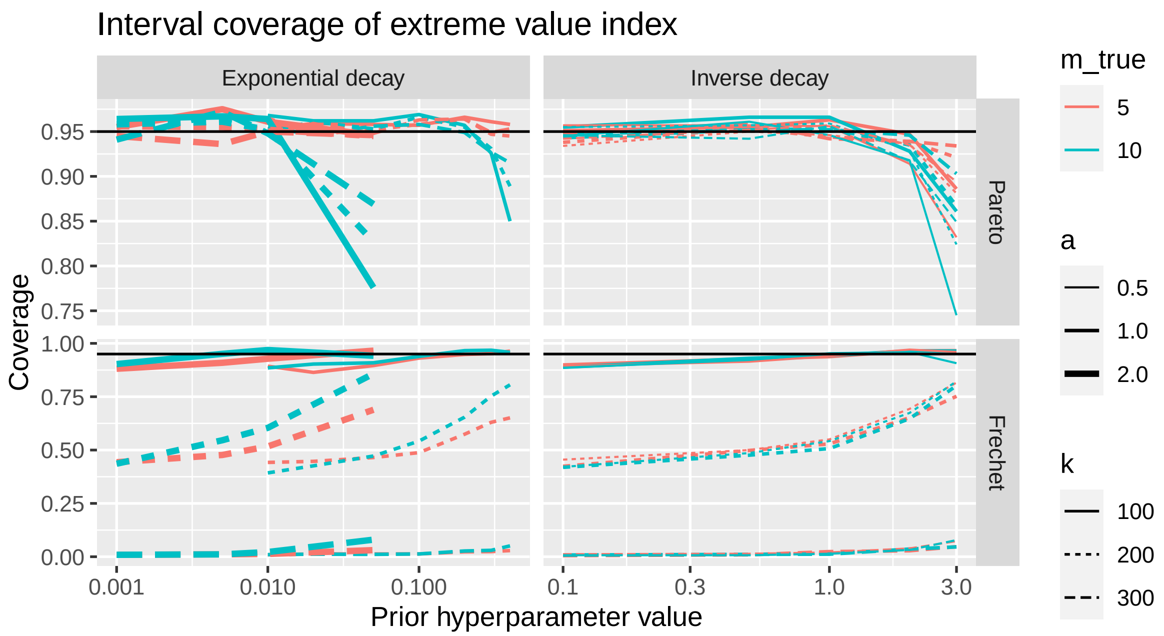
For every posterior fit we calculated two 95% intervals for each parameter of interest:

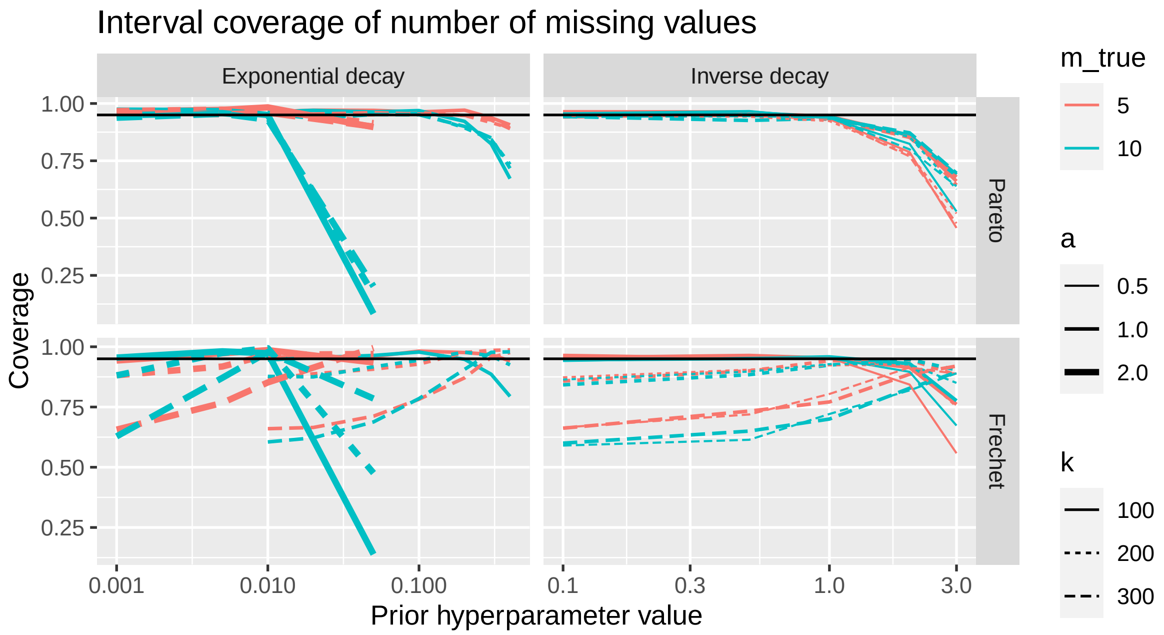
1. The first interval is the shortest interval, also known as the highest density interval (HDI) or highest posterior density interval (HPD interval).
2. The second interval is the symmetric interval, symmetric in probability, so 2.5% probability outside the interval on the left and 2.5% probability outside the interval on the right.

For each interval and parameter we are interested in the coverage (proportion of intervals that cover the true value) as well as the interval width (either absolute or relative to the true value).

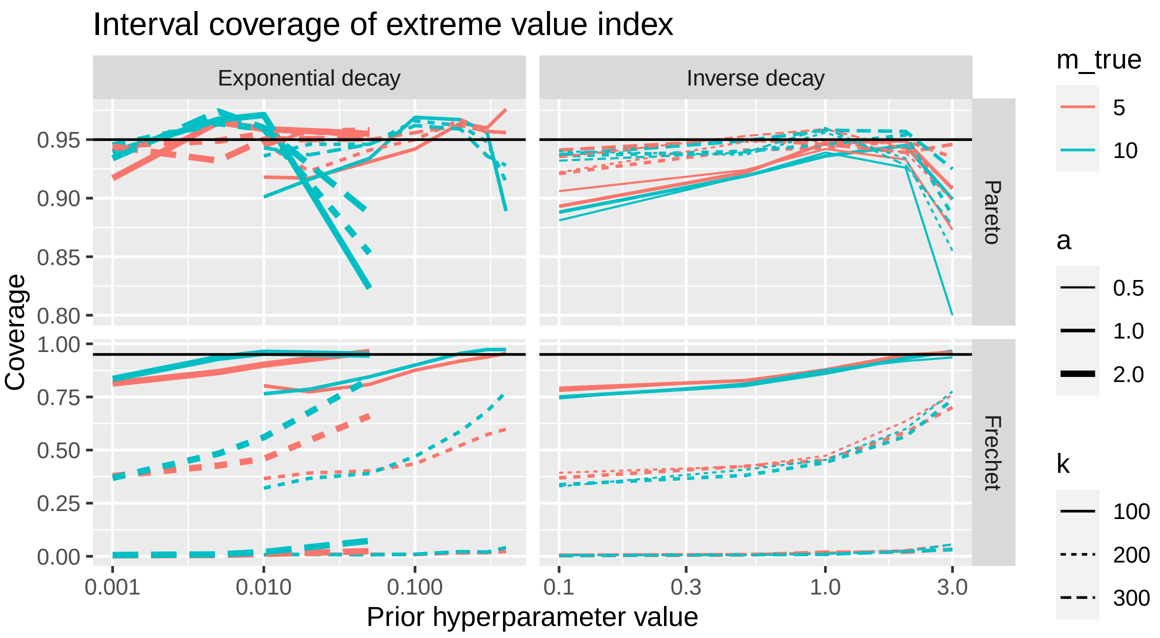
```r
stats_of_interest <- c("Coverage", "Coverage_symmetric")
nice_titles <- c("Highest posterior density (shortest) 95% interval",
                 "Symmetric 95% interval")
for (current_stat in seq_along(stats_of_interest)) {
  cat('\n\n####', nice_titles[current_stat], '\n\n')
  all_results |>
    filter(Stat_name == stats_of_interest[current_stat]) |>
    group_by(Parameter, Scenario) |>
    summarise(
      Coverage_95 = mean(Statistic),
      .groups = "keep"
    ) |> left_join(base_scenarios, by = join_by(Scenario)) |>
    mutate(Group = as.factor(paste0('m = ', m_true, '; k = ', k, '; a = ', a)),
           k = as.factor(k),
           m_true = as.factor(m_true)
          ) -> summary_table

  # summary_table |> kable(digits = 3) |> print()
  tryCatch({
    summary_table |> add_to_wb(wb, stats_of_interest[current_stat]) -> wb
  }, error = \(e) {print(e); cat('\n\n')})
  a_scale_options <- unique(summary_table$a) |> sort()

  (summary_table |>
    filter(Parameter == 'EVI') |>
    ggplot(aes(x = lambda, y = Coverage_95,
               group = Group, colour = m_true,
               linetype = k, linewidth = a)) +
    geom_line() +
    scale_x_log10() +
    scale_linewidth(range = c(0.4, 1.2),
                      breaks = a_scale_options) +
    ggtitle('Interval coverage of extreme value index') +
    facet_grid(cols = vars(model),
               rows = vars(Distribution),
               scales = 'free') +
    xlab('Prior hyperparameter value') +
    ylab('Coverage') + geom_hline(yintercept = 0.95)
    ) |> print()
  cat('\n\n')
  (summary_table |>
    filter(Parameter == 'm') |>
    ggplot(aes(x = lambda, y = Coverage_95,
               group = Group, colour = m_true,
               linetype = k, linewidth = a)) +
    geom_line() +
    scale_x_log10() +
    scale_linewidth(range = c(0.4, 1.2),
                      breaks = a_scale_options) +
    ggtitle('Interval coverage of number of missing values') +
    facet_grid(cols = vars(model),
               rows = vars(Distribution),
               scales = 'free') +
    xlab('Prior hyperparameter value') +
    ylab('Coverage') + geom_hline(yintercept = 0.95)
    ) |> print()
  cat('\n\n---\n\n')
}
```

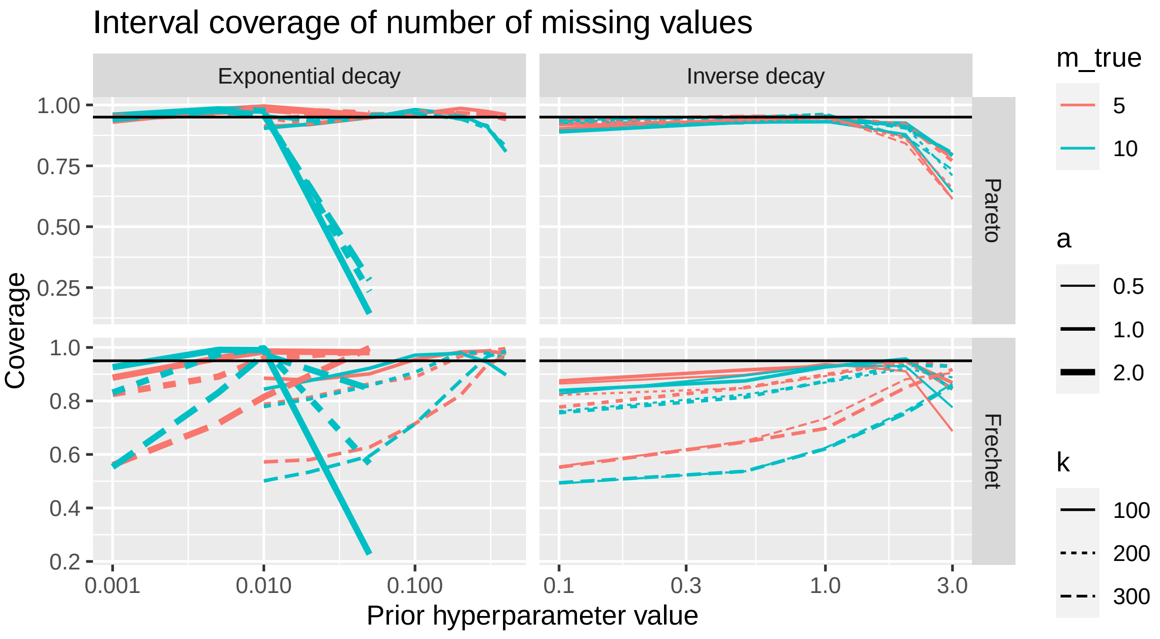
Highest posterior density (shortest) 95% interval





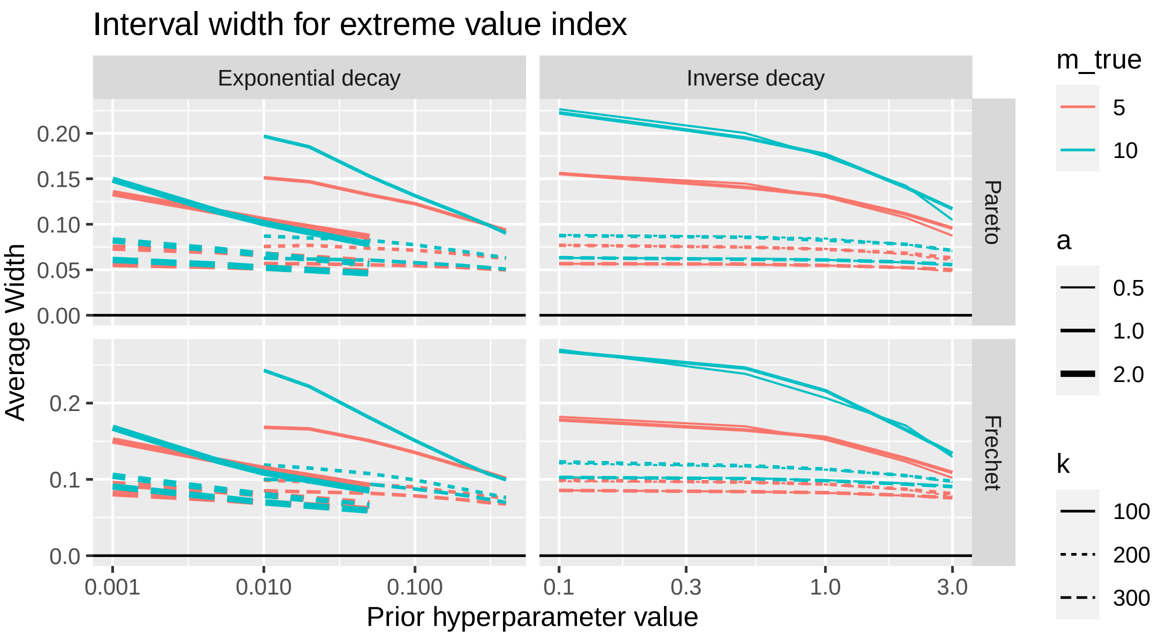
Symmetric 95% interval

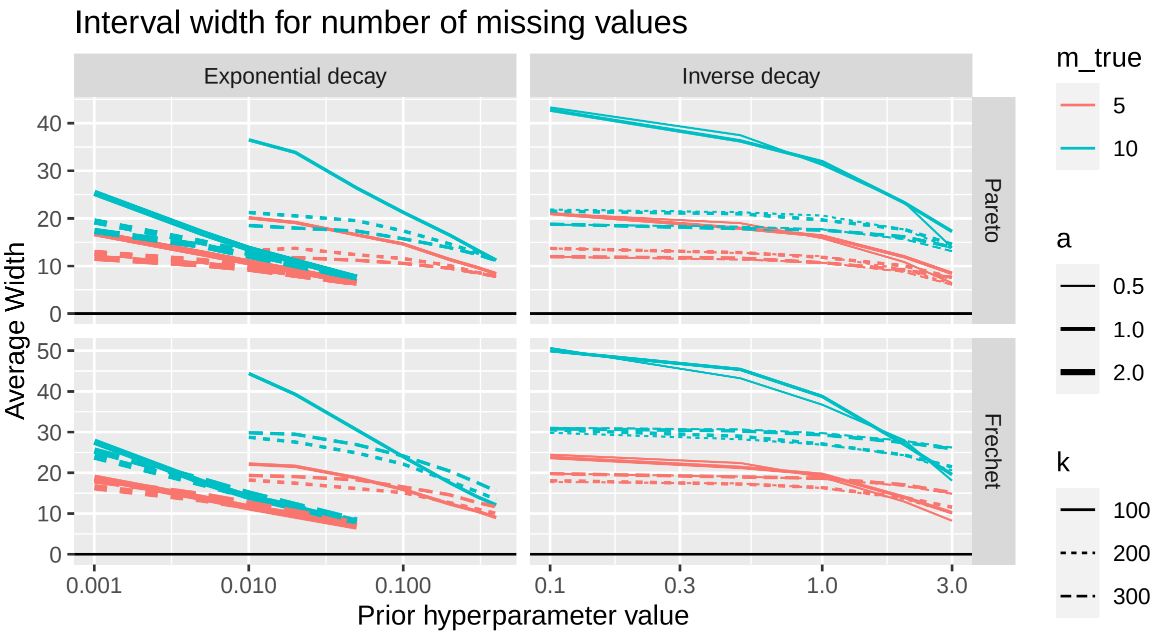

Interval coverage of extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Coverage
0.95
0.90
0.85
0.80
1.00
0.75
0.50
0.25
0.00
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

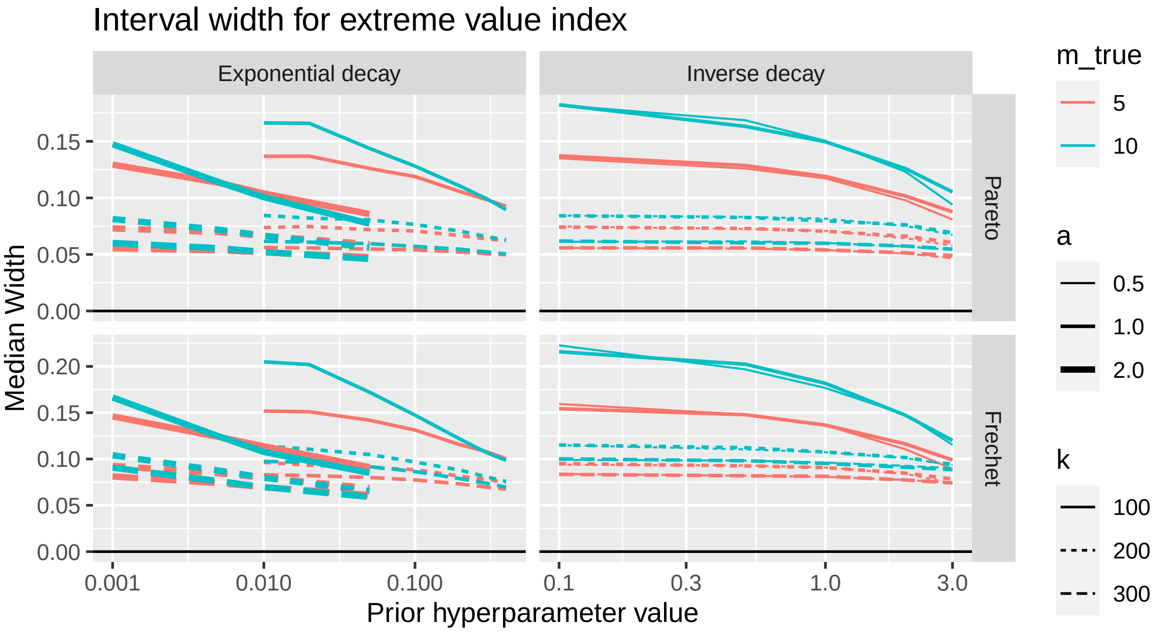

Interval coverage of number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Coverage
1.00
0.75
0.50
0.25
1.0
0.8
0.6
0.4
0.2
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

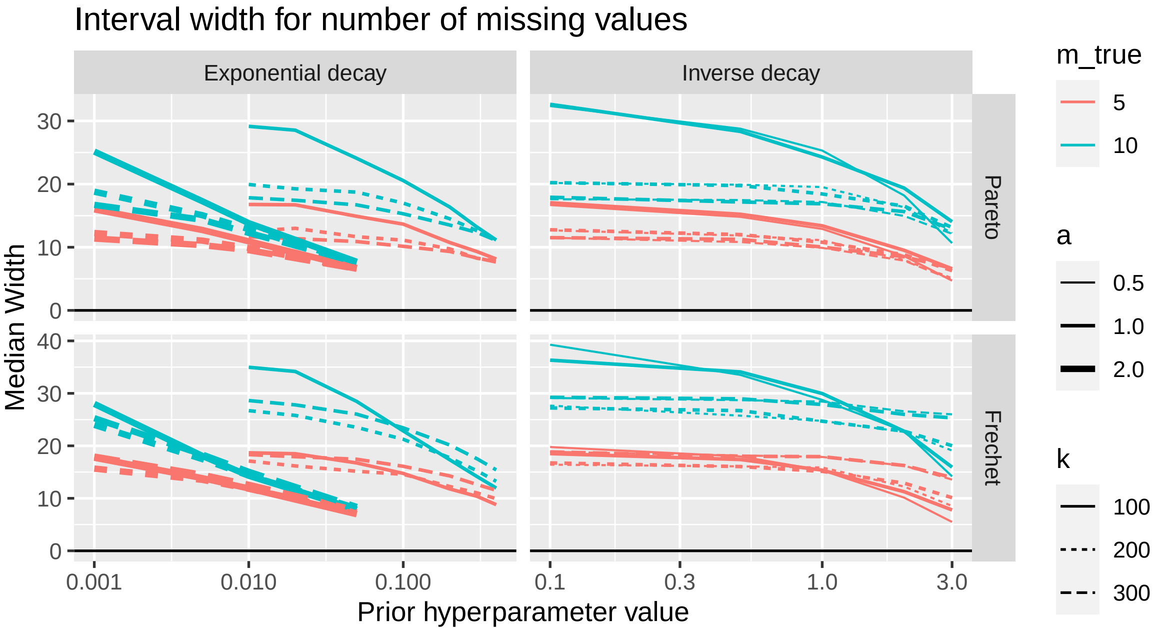
```
stats_of_interest <- c("Width", "Width_symmetric")
nice_titles <- c("Highest posterior density (shortest) 95% interval",
                 "Symmetric 95% interval")
for (current_stat in seq_along(stats_of_interest)) {
  cat('\n\n####', nice_titles[current_stat], '\n\n')
  all_results |>
    filter(Stat_name == stats_of_interest[current_stat]) |>
    group_by(Parameter, Scenario) |>
    summarise(
      Average_Width = mean(Statistic),
      Median_Width = median(Statistic),
      .groups = "keep"
    ) |> left_join(base_scenarios, by = join_by(Scenario)) |>
    mutate(Group = as.factor(paste0('m = ', m_true, '; k = ', k, '; a = ', a)),
           Relative_Average_Width = Average_Width/ifelse(Parameter == 'EVI', EVI_true, m_t
           Relative_Median_Width = Median_Width/ifelse(Parameter == 'EVI', EVI_true, m_tru
           k = as.factor(k),
           m_true = as.factor(m_true)
          ) -> summary_table

  # summary_table |> kable(digits = 3) |> print()
  tryCatch({
    summary_table |> add_to_wb(wb, stats_of_interest[current_stat]) -> wb
  }, error = \(e) {print(e); cat('\n\n')})
  a_scale_options <- unique(summary_table$a) |> sort()

  for (stat in c('Average_Width', 'Median_Width', 'Relative_Average_Width', 'Relative_Medi
    (summary_table |>
      filter(Parameter == 'EVI') |>
      ggplot(aes(x = lambda, y = !!sym(stat),
                 group = Group, colour = m_true,
                 linetype = k, linewidth = a)) +
      geom_line() +
      scale_x_log10() +
      scale_linewidth(range = c(0.4, 1.2),
                        breaks = a_scale_options) +
      ggtitle('Interval width for extreme value index') +
      facet_grid(cols = vars(model),
                 rows = vars(Distribution),
                 scales = 'free') +
      xlab('Prior hyperparameter value') +
      ylab(gsub("_", " ", stat)) + geom_hline(yintercept = 0)
    ) |> print()
    cat('\n\n')
    (summary_table |>
      filter(Parameter == 'm') |>
      ggplot(aes(x = lambda, y = !!sym(stat),
                 group = Group, colour = m_true,
                 linetype = k, linewidth = a)) +
      geom_line() +
      scale_x_log10() +
      scale_linewidth(range = c(0.4, 1.2),
                        breaks = a_scale_options) +
      ggtitle('Interval width for number of missing values') +
      facet_grid(cols = vars(model),
                 rows = vars(Distribution),
                 scales = 'free') +
      xlab('Prior hyperparameter value') +
      ylab(gsub("_", " ", stat)) + geom_hline(yintercept = 0)
    ) |> print()
    cat('\n\n')
  }
  cat('---\n\n')
}
```

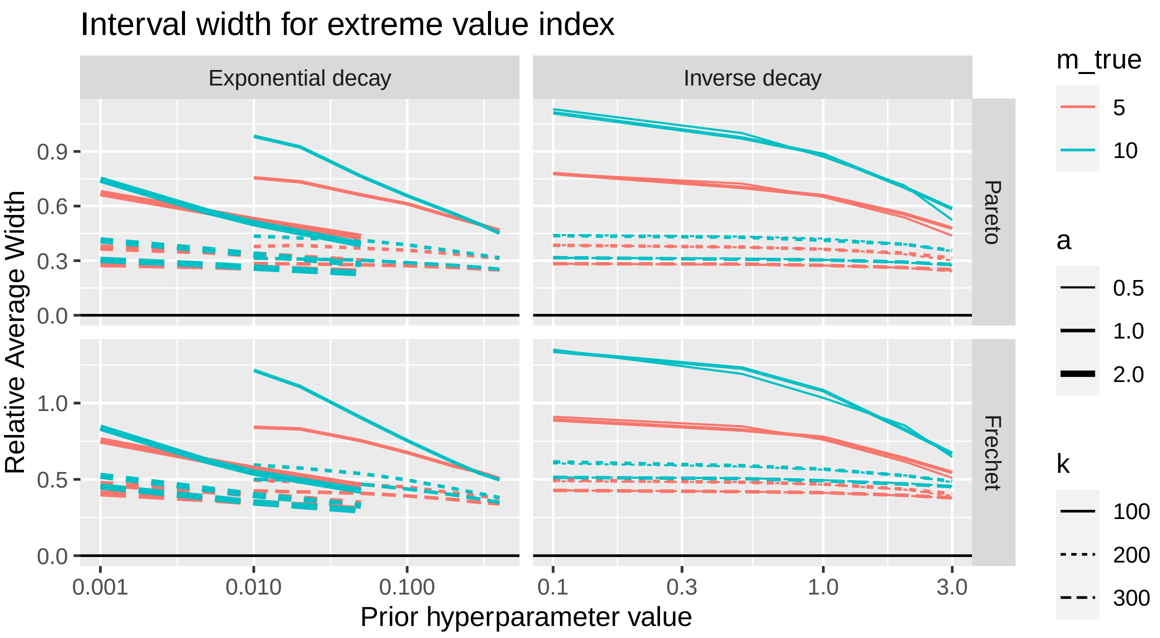
Highest posterior density (shortest) 95% interval

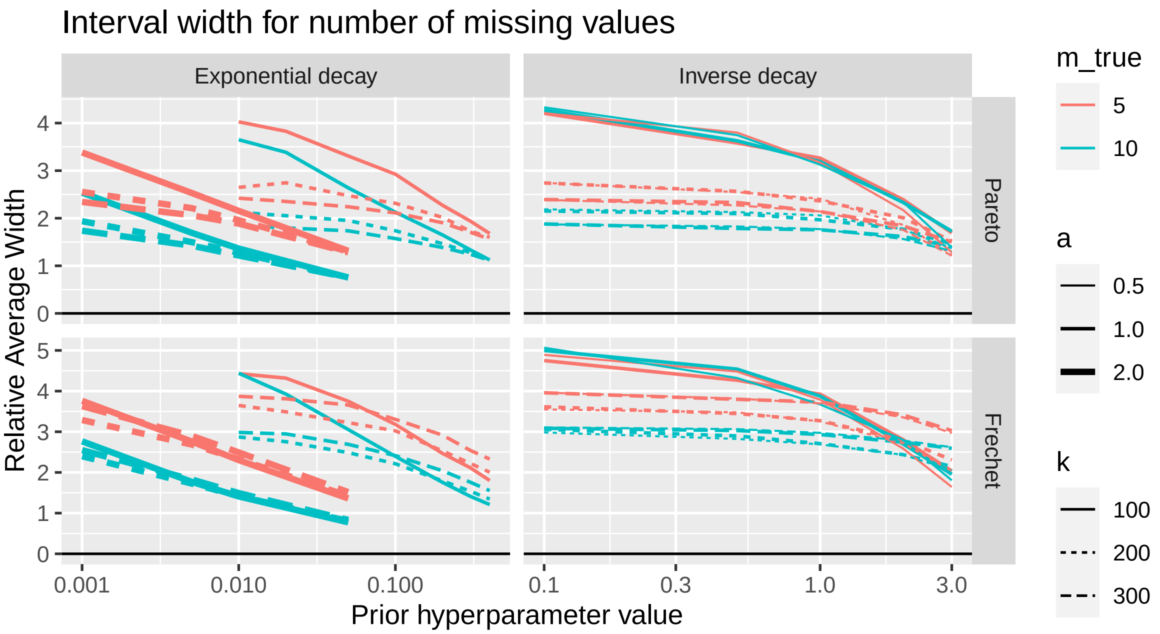

Interval width for extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Average Width
0.20
0.15
0.10
0.05
0.00
0.2
0.1
0.0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

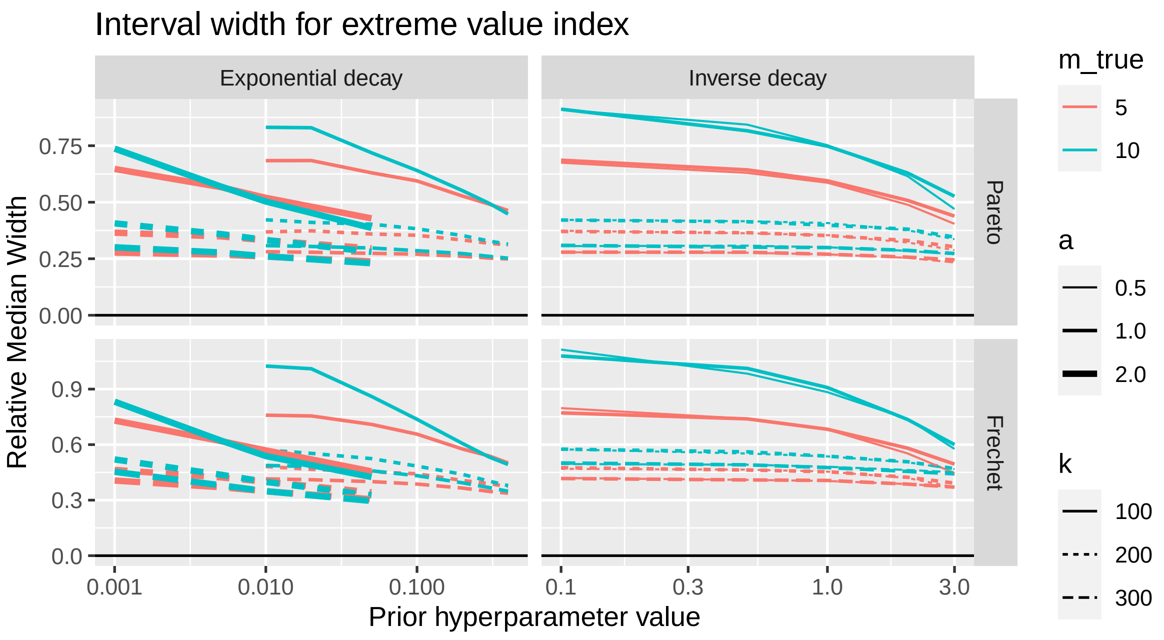

Interval width for number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Average Width
40
30
20
10
0
50
40
30
20
10
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

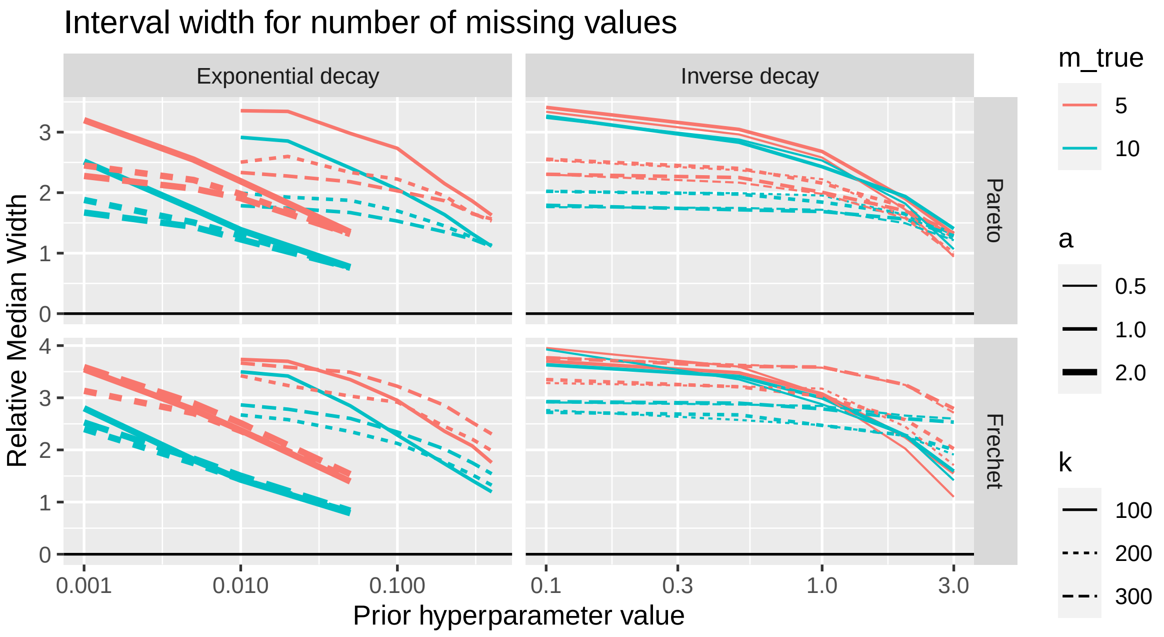

Interval width for extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Median Width
0.15
0.10
0.05
0.00
0.20
0.15
0.10
0.05
0.00
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Interval width for number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Median Width
30
20
10
0
40
30
20
10
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

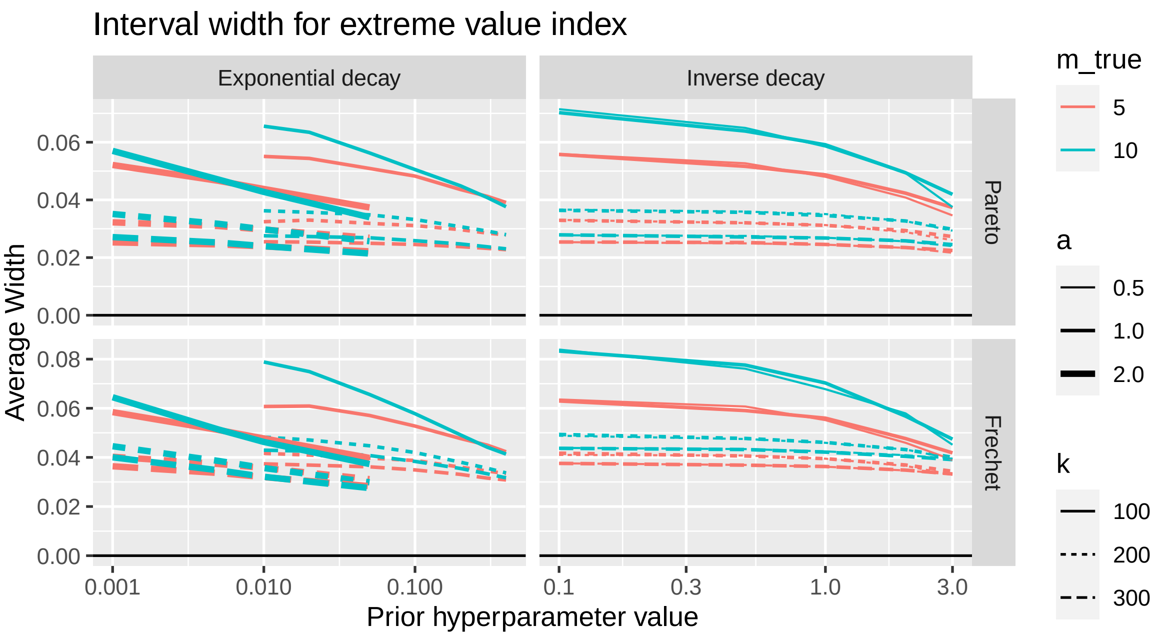

Interval width for extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Relative Average Width
0.0
0.3
0.6
0.9
0.5
1.0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

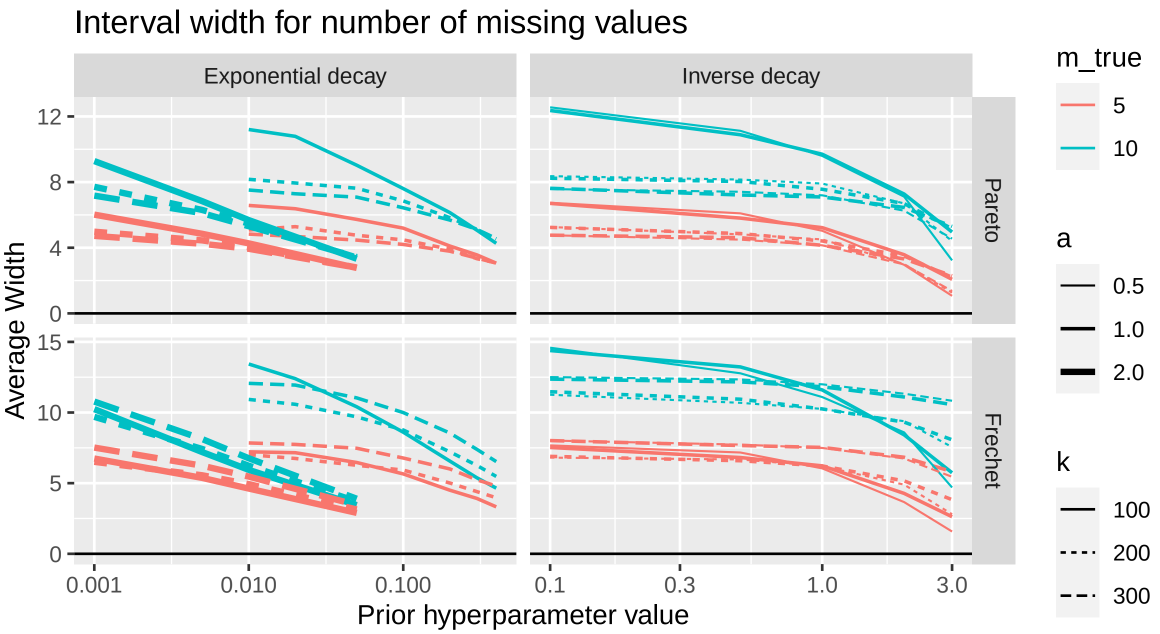

Interval width for number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Relative Average Width
0
1
2
3
4
5
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300



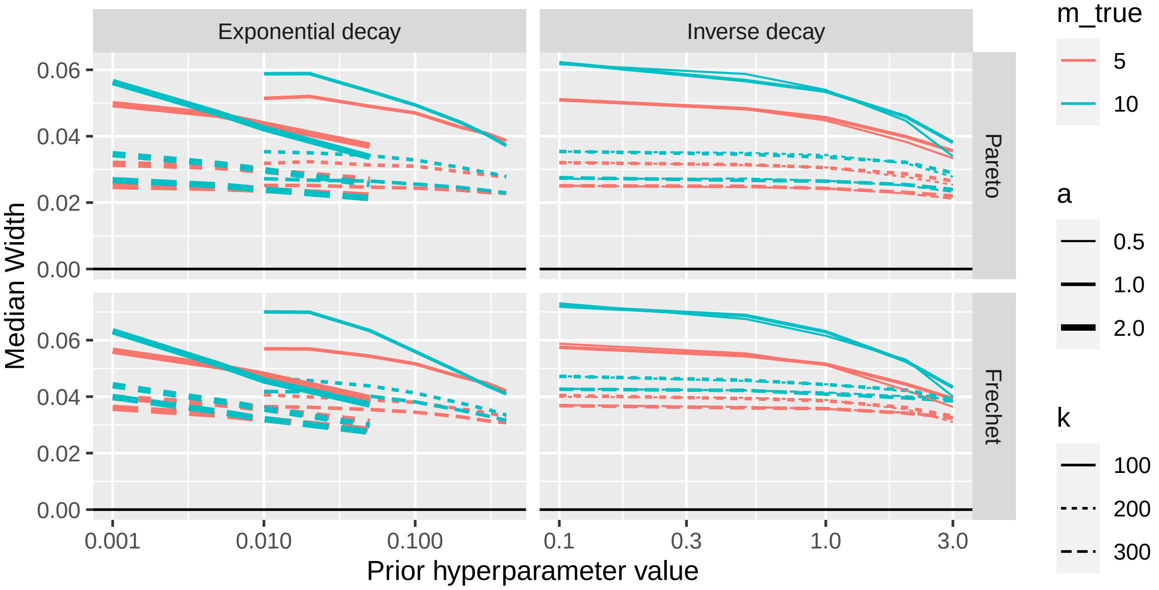


Symmetric 95% interval

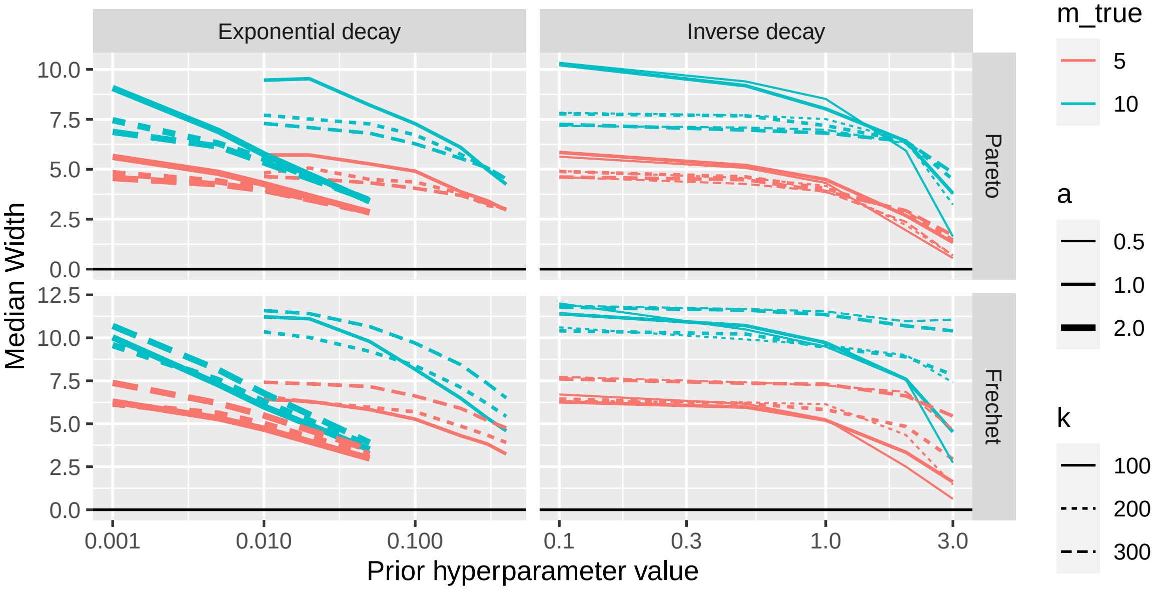

Interval width for extreme value index
Exponential decay
Inverse decay
Pareto
Frechet
Average Width
0.06
0.04
0.02
0.00
0.08
0.06
0.04
0.02
0.00
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

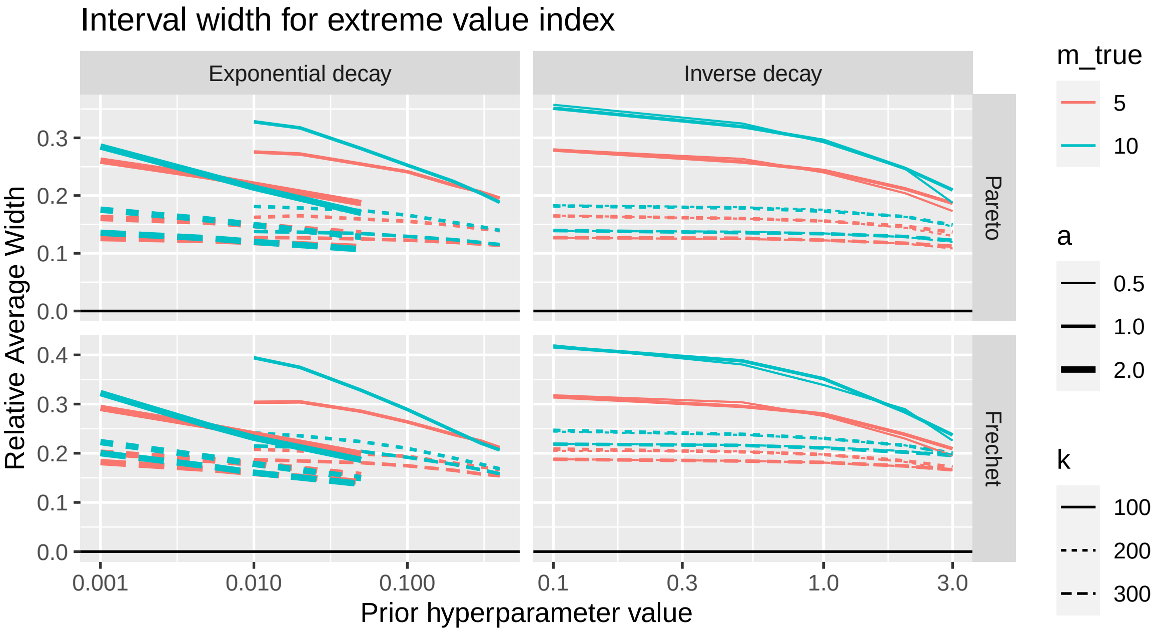

Interval width for number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Average Width
12
8
4
0
15
10
5
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyperparameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300

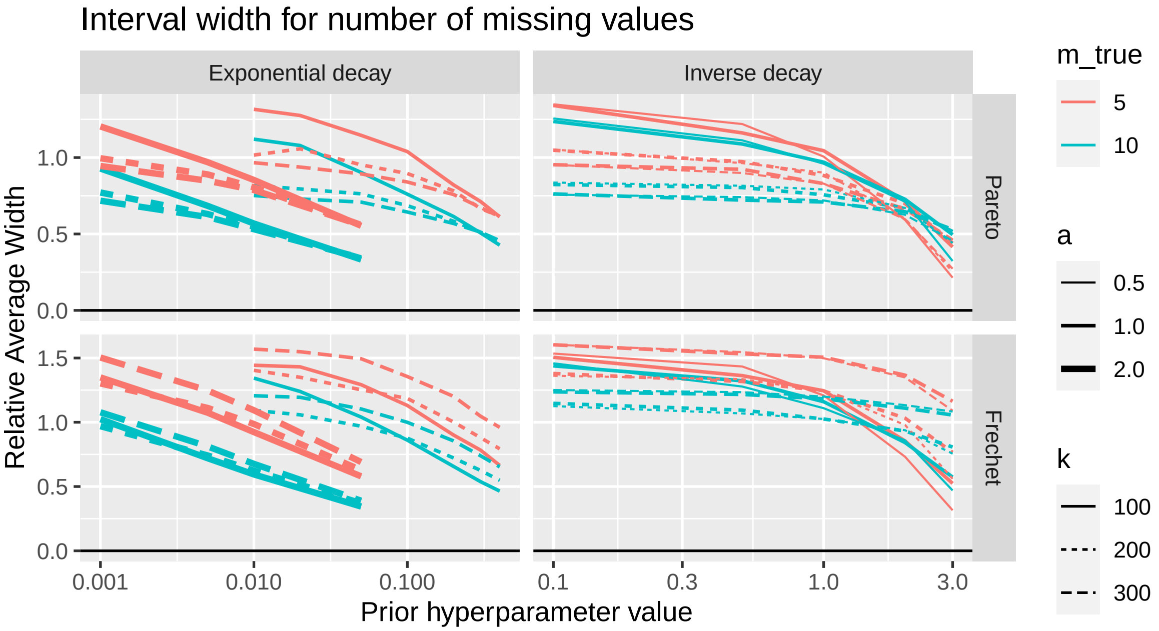
Interval width for extreme value index

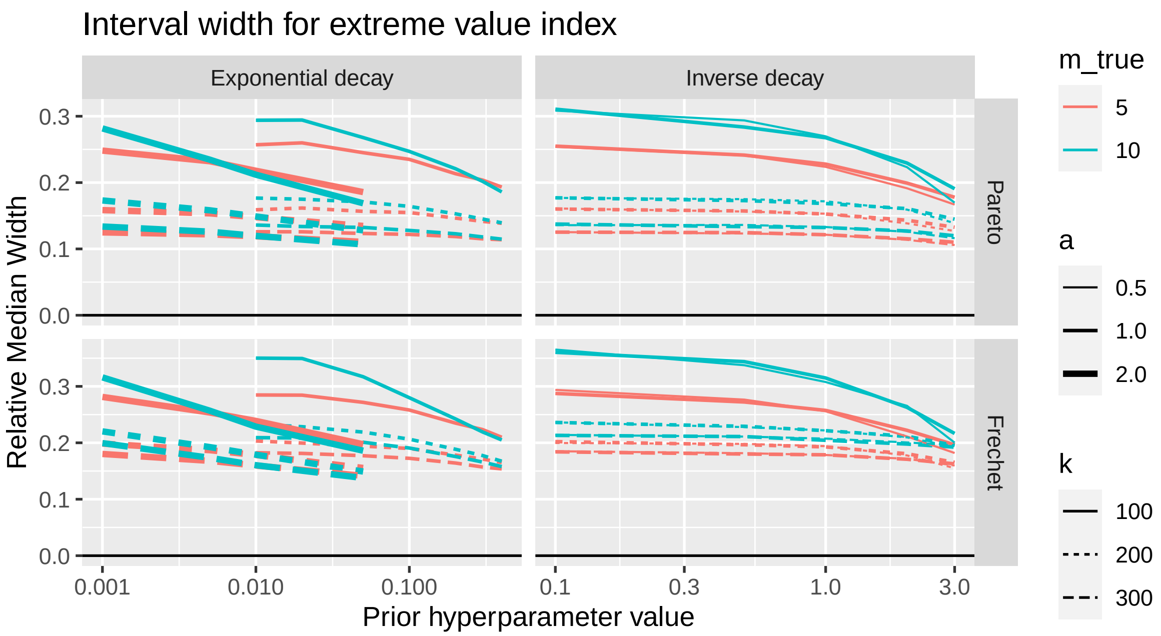


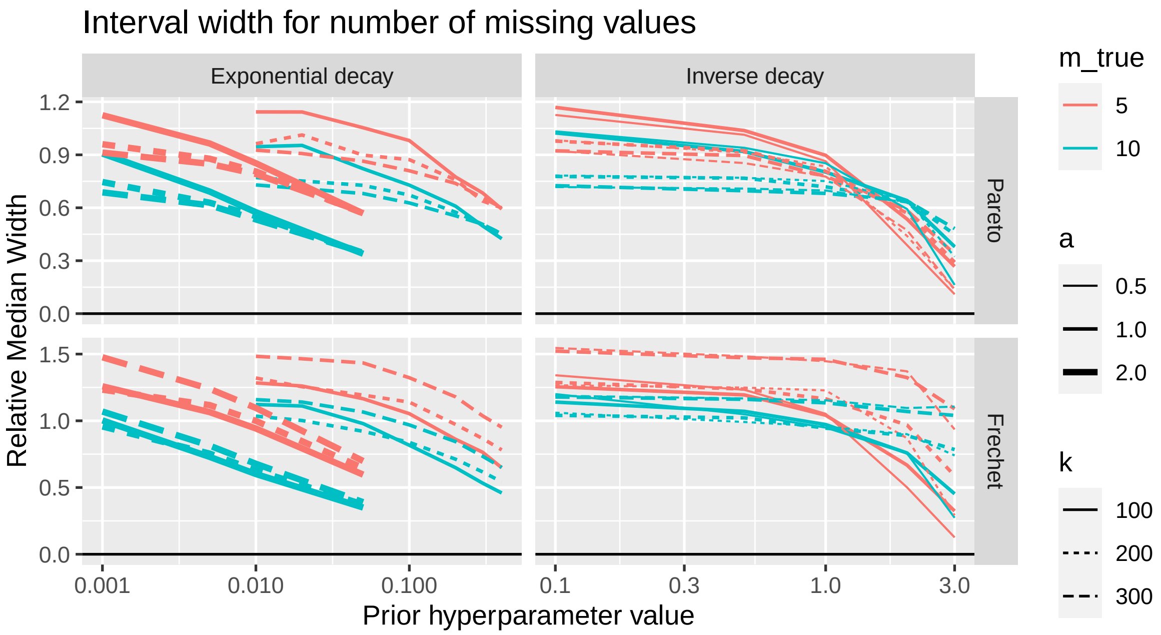
Interval width for number of missing values




Interval width for extreme value index


Interval width for number of missing values


Interval width for extreme value index


Interval width for number of missing values

Save Excel file

Finally we save the Excel file with the results.

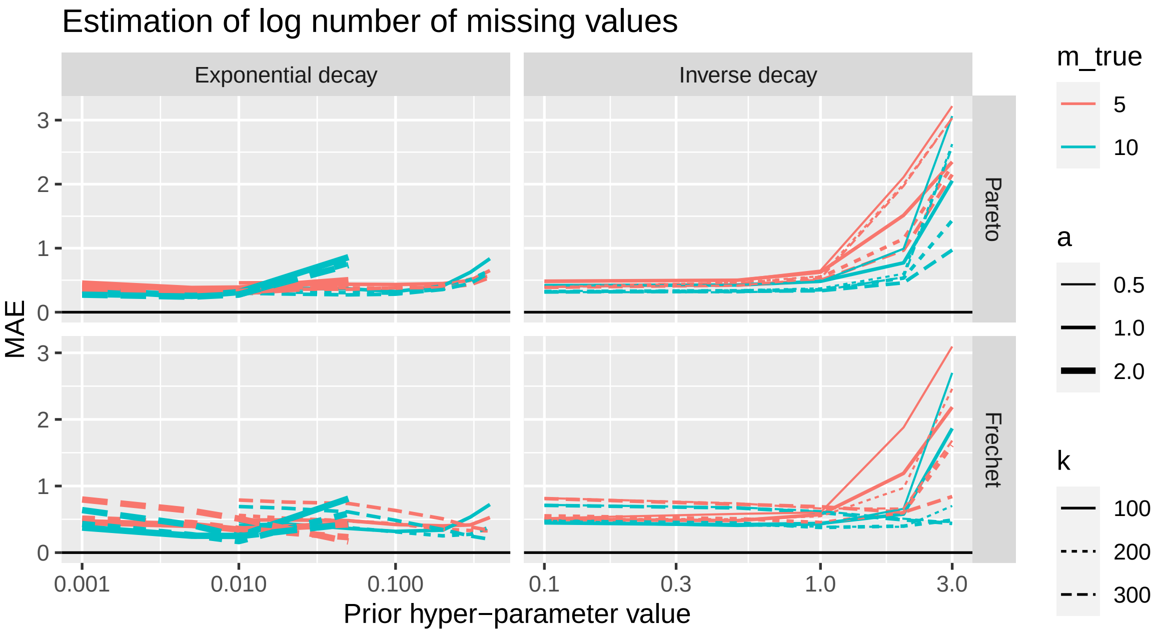
```
wb |> saveWorkbook("SimstudyStdResults.xlsx", overwrite = TRUE)
```

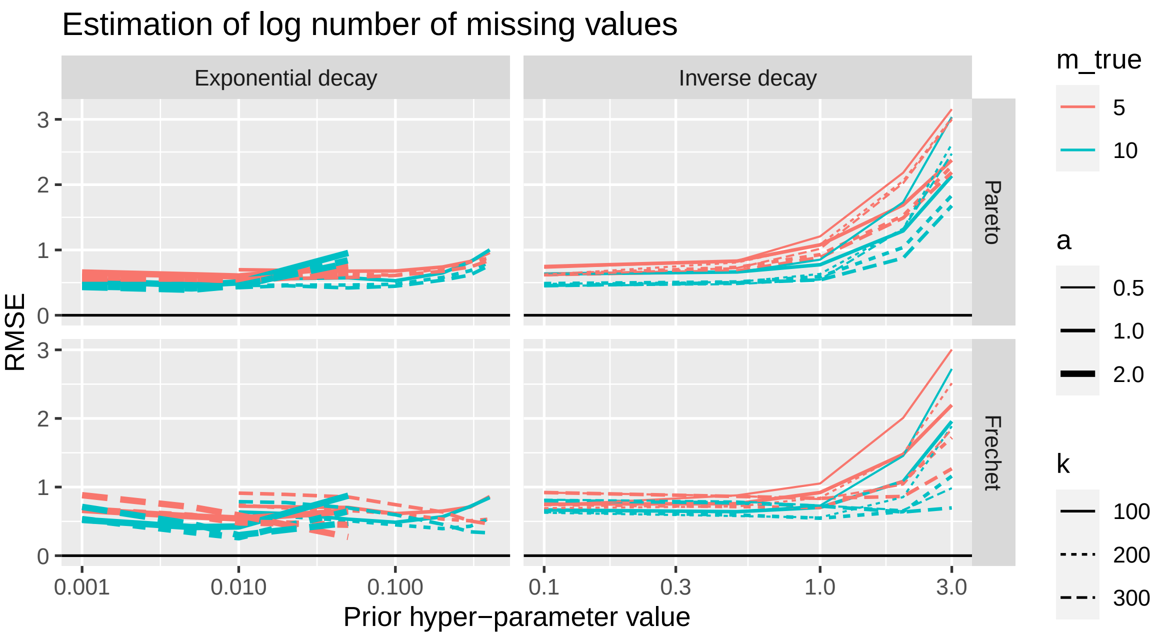
Error of the mode on log scale

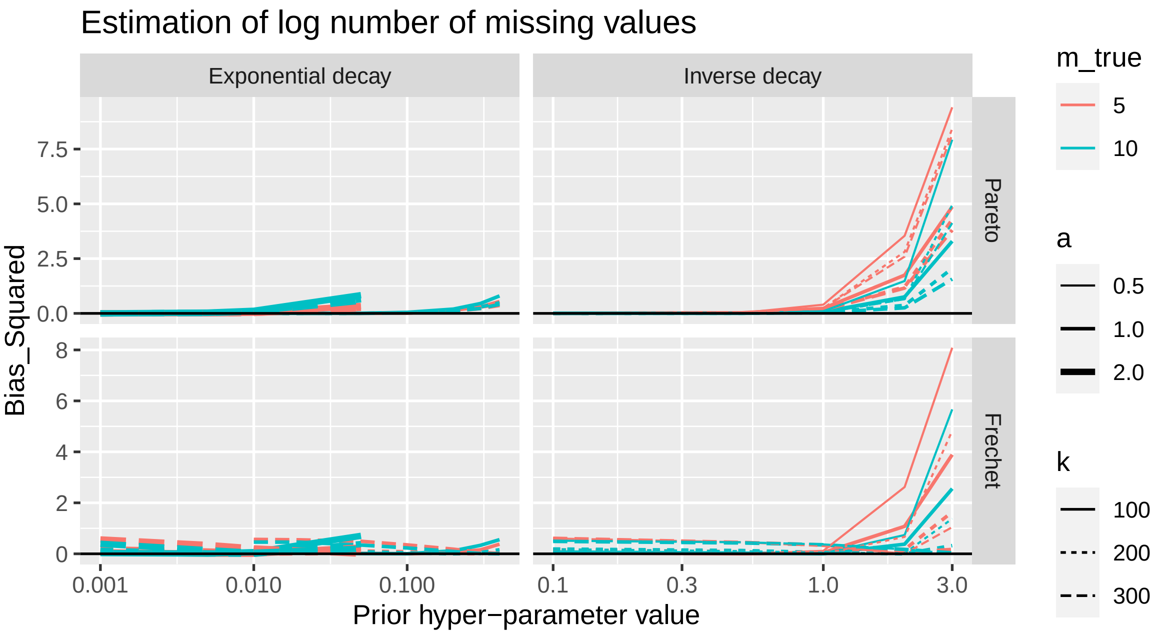
The measuring of the error in the estimation of the number of missing values on the raw scale may be fundamentally out of line with what is actually important. For a small number of missing values we would like to see better accuracy in absolute terms. So relative accuracy is of greater interest that absolute accuracy. Simply dividing the accuracy estimates by the true values can be seen as a relative accuracy but does not address the key issue that the statistics are calculated on the wrong scale and distorted by outlying cases.

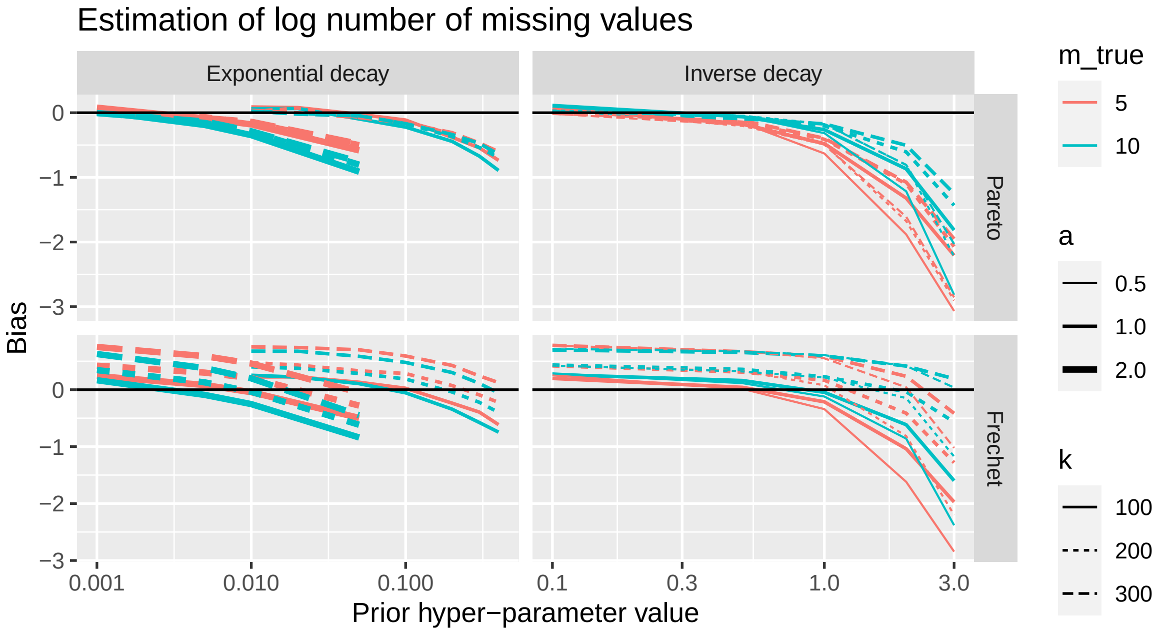
To address this the posterior mode errors for the number of missing values are recalculated on the log scale and reported in detail here.

```
all_results |>
  filter(Stat_name == "Mode") |>
  filter(Parameter == 'm') |>
  group_by(Parameter, Scenario) |>
  summarise(
    Bias = mean(log(Statistic) - log(m_true)),
    RMSE = sqrt(mean((log(Statistic) - log(m_true))^2)),
    MAE = median(abs(log(Statistic) - log(m_true))),
    .groups = "keep"
  ) |> left_join(base_scenarios, by = join_by(Scenario)) |>
  mutate(Group = as.factor(paste0('m = ', m_true, '; k = ', k, '; a = ', a)),
         Bias_Squared = Bias^2,
         k = as.factor(k),
         m_true = as.factor(m_true)
        ) -> summary_table

a_scale_options <- unique(summary_table$a) |> sort()

for (stat in c('MAE', 'RMSE', 'Bias_Squared', 'Bias')) {
  (summary_table |>
    ggplot(aes(x = lambda, y = !!sym(stat),
               group = Group, colour = m_true,
               linetype = k, linewidth = a)) +
    geom_line() +
    scale_x_log10() +
    scale_linewidth(range = c(0.4, 1.2),
                    breaks = a_scale_options) +
    ggtitle('Estimation of log number of missing values') +
    facet_grid(cols = vars(model),
               rows = vars(Distribution),
               scales = 'free') +
    xlab('Prior hyper-parameter value') +
    ylab(stat) + geom_hline(yintercept = 0)
  ) |> print()
}
```




Estimation of log number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
RMSE
3
2
1
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300


Estimation of log number of missing values
Exponential decay
Inverse decay
Pareto
Frechet
Bias_Squared
7.5
5.0
2.5
0.0
8
6
4
2
0
0.001
0.010
0.100
0.1
0.3
1.0
3.0
Prior hyper−parameter value
m_true
5
10
a
0.5
1.0
2.0
k
100
200
300



Conclusions

A few things are quite clear from these results:

1. The exponential decay prior appears superior to the alternative shape proposed.
2. The ideal value of the hyper-parameter for the exponential prior appears to be roughly $\lambda = 1/5 = 0.2$, although it is not yet clear whether this is universal.
3. If the expected value of the prior is reduced, so that $\lambda > 1/4$ then we end up with overly narrow intervals and poor coverage, in addition to bias in one direction.
4. If the expected value of the prior is too high, say $\lambda < 1/10$ then we end up with overly wide intervals and biases going the other way.
5. The posterior distributions of the parameters is fairly symmetric for the sample size considered. For large enough samples it does not seem to matter whether the parameters are estimated with a posterior mean, median, or mode. This may not continue to hold for smaller samples, where a mode might differ from a mean in accuracy, with the median usually a good balance in between.
6. While the shift of the hyperbolic decay prior is necessary, the extent of the shift appears to have little effect.
7. In the case where the entire sample is strict Pareto the more observations we include the more accurate the estimation and the tighter the intervals, as expected.
8. In the case where the sample is Frechet/Inverse Weibull, when we include additional observations to the point where we go out of the tail region then we lose accuracy fast

($k \geq 300$). Threshold selection continues to be critical here, as expected.