Power Analysis: Effect of Number of Measurements on Inference

Introduction

In this document, we focus on the century model, and analyze the effects of the number of measurements on the inference. We use a hierarchical Century model to generate simulated data and for the fit.

R function to generate simulated data (hierarchical Century)

```
simulate_data_century <- function(t_meas, t_cap, gamma_global, num_rep) {
# INPUTS:
#   t_meas:  measurement times
#   t_cap:   cap times
#   gamma_global:  initial pool contents
#   num_rep: number of replications
library(deSolve)
library(gtools)
set.seed(1234)
genDerivs <-function(t, Ct, params) {
  # General diff eq model: dC_dt = I(t) + A(t)*C(t)
  # INPUTS:
  #   t: time
  #   Ct: the value of the vector C at time t, C(t)
  #   params: it has two fields, params$I and params$A
  dC_dt = params$I + params$A %*% Ct;
  return(list(dC_dt));
}
m <- 3; # number of pools
turnover <- c(1.5, 25, 1000);
K <- 1/turnover;
I <- rep(0, m); # no input flux for century
N_t <- length(t_meas);
CO2_flux_mat <- matrix(NA, nrow = N_t, ncol = num_rep);
Alpha_rep <- array(0, c(m, m, num_rep));
gamma_rep <- array(0, c(m, num_rep));

Alpha <- matrix(0, m, m);
# Setting global transfer rates with expert-tuned values:
Alpha[2, 1] = 0.5;
Alpha[3, 1] = 0.004;
Alpha[1, 2] = 0.42;
Alpha[3, 2] = 0.03;
Alpha[1, 3] = 0.45;
Alpha[1, 1] = 1 - Alpha[2, 1] - Alpha[3, 1];
Alpha[2, 2] = 1 - Alpha[1, 2] - Alpha[3, 2];
Alpha[3, 3] = 1 - Alpha[1, 3] - Alpha[2, 3];
for (this_rep in 1:num_rep) {
  # Hierarchical modeling of initializations for replications:
  kappa_inits <- 10;
```

```
  gamma_rep[,this_rep] <- rdirichlet(1, gamma_global * kappa_inits);
  C_t0 <- gamma_rep[,this_rep];

  # Hierarchical modeling of transfer rates for replications:
  kappa_rates <- 10;
  Alpha_rep[,1, this_rep] <-  rdirichlet(1, Alpha[,1] * kappa_rates);
  Alpha_rep[,2, this_rep] <-  rdirichlet(1, Alpha[,2] * kappa_rates);
  Alpha_rep[,3, this_rep] <-  rdirichlet(1, Alpha[,3] * kappa_rates);
  Alpha_rep[1, 1, this_rep] <- 0;
  Alpha_rep[2, 2, this_rep] <- 0;
  Alpha_rep[3, 3, this_rep] <- 0;
  A <- Alpha_rep[,, this_rep] * matrix(rep(K, m), nrow = m, byrow = TRUE) - diag(K);
  params <- list(I=I, A=A);
  t0 <- 0;
  # Solving the ODE system for given parameters:
  meas_data<-ode(y = C_t0, func = genDerivs,
          times = c(t0,t_meas), parms = params);
  cap_data<-ode(y = C_t0, func = genDerivs,
                times = c(t0,t_cap), parms = params);
  # Calculating CO2 flux
  totalC_t0 = sum(meas_data[1,2:(m+1)]);
  CO2_t_meas <- totalC_t0 - rowSums(meas_data[2:nrow(meas_data), 2:(m+1)]);
  CO2_t_cap <- totalC_t0 - rowSums(cap_data[2:nrow(cap_data),2:(m+1)]);
  CO2_flux <- (CO2_t_meas - CO2_t_cap)/(t_meas-t_cap);
  # Adding log-normal noise:
  CO2_flux_mat[, this_rep] <- exp(log(CO2_flux) + rnorm(length(CO2_flux),0,.05));
}
simulated_data <- list(N_t = N_t, t_meas = t_meas, t_cap = t_cap,
                       num_rep = num_rep, totalC_t0 = totalC_t0,
                       t0=t0, CO2_flux=CO2_flux_mat, Alpha_rep=Alpha_rep, gamma_rep = gamma_rep);
return(simulated_data);
}
```

Measurement and cap times

We use a measurement scheme that is more frequent in the beginning (when we have a high flux) and is less frequent as time passes. The exact R code is shown below. Initially, we assume we have many measurements. Then we use subsets of these measurements and examine the effect of that subsampling on the inference. For the largest measurement set, we have two samples per day in the first week, one sample per day in the first month, one sample per two days in the next month, and weekly sample afterwards.

```
t_meas_all <- c(seq(from=1/360, to=14/360, by=0.5/360), seq(from=15/360, to=30/360, by=1/360),
                seq(from=32/360, to=60/360, by=2/360), seq(from=67/360, to=360/360, by=7/360))
```

Then, we downsaple this large set of measurement times as follows:

```
downsample_set <- c(1,2,4,5,10,20,50)
for (i in 1:length(downsample_set)) {
    n_downsample <- downsample_set[i]
    t_meas <- t_meas_all[seq(1,length(t_meas_all), n_downsample)]
    # Fit the model here ...
    # ...
}
```

Essentially, we keep all, every 2, every 4, every 5, etc. of the original set of measurement times. The original set has 100 measurements.

We choose cap times to be shortly before the measurement times ($\sim$ 1 hour in the beginning and $\sim$ 1 day towards the end). This is done as follows:

```
t_cap_all <-  head(t_meas_all, -1) + 0.85 * (tail(t_meas_all, -1) - head(t_meas_all, -1))
t_cap_all <- c(t_meas_all[1] - (t_meas_all[2] - t_cap_all[1]), t_cap_all)
downsample_set <- c(1,2,4,5,10,20,50)
for (i in 1:length(downsample_set)) {
    n_downsample <- downsample_set[i]
    t_cap <- t_cap_all[seq(1,length(t_cap_all), n_downsample)]
    # Fit the model here ...
    # ...
}
```

Stan code for fitting fully hierarchical Century model

```
// Fitting the century model with partial pooling
functions {
  /**
  * ODE system for the Century model with no input fluxes.
  * @param t time at which derivatives are evaluated.
  * @param C system state at which derivatives are evaluated.
  * @param theta parameters for system.
  * @param x_r real constants for system (empty).
  * @param x_i integer constants for system (empty).
  */
    real[] century_model(real t, real[] C, real[] theta,
                         real[] x_r, int[] x_i) {
      real k[3];
      real a21;
      real a31;
      real a12;
      real a32;
      real a13;
      real dC_dt[3];
      k = theta[1:3];
      a21 = theta[4];
      a31 = theta[5];
      a12 = theta[6];
      a32 = theta[7];
      a13 = theta[8];
      dC_dt[1] = -k[1] * C[1] + a12 * k[2] * C[2] + a13 * k[3] * C[3];
      dC_dt[2] = -k[2] * C[2] + a21 * k[1] * C[1];
      dC_dt[3] = -k[3] * C[3] + a31 * k[1] * C[1] + a32 * k[2] * C[2];
      return dC_dt;
    }

  /**
  * Compute evolved CO2 from the system given the specified
  * parameters and times. This is done by simulating the system
  * defined by the ODE function century_model and then
  * calculating the rate CO2 is emmited.
```

```
  *
  * @param N_t number of times
  * @param t0 initial time
  * @param ts times
  * @param gamma partitioning coefficient
  * @param k decomposition rates
  * @param ajk transfer rates
  * @param x_r real data (empty)
  * @param x_i integer data (empty)
  * @return evolved CO2 for times ts
  */
    vector evolved_CO2(int N_t, real t0, vector ts,
                       vector gamma, real totalC_t0,
                       vector k, real a21, real a31, real a12,
                       real a32, real a13, real[] x_r, int[] x_i) {

      real C_t0[3];         // initial state
      real theta[8];        // ODE parameters
      real C_t[N_t,3];      // predicted pool content
      vector[N_t] CO2_t;    // evolved CO2 at times ts

      C_t0 = to_array_1d(gamma*totalC_t0);
      theta[1:3] = to_array_1d(k);
      theta[4] = a21;
      theta[5] = a31;
      theta[6] = a12;
      theta[7] = a32;
      theta[8] = a13;
      C_t = integrate_ode_rk45(century_model,
                               C_t0, t0, to_array_1d(ts), theta, x_r, x_i);
      for (t in 1:N_t)
        CO2_t[t] = totalC_t0 - sum(C_t[t]);
      return CO2_t;
    }
}
data {
  real<lower=0> totalC_t0;      // initial total carbon
  real t0;                      // initial time
  int<lower=0> N_t;             // number of measurement times
  int<lower=0> num_rep;         // number of replicates
  vector<lower=t0>[N_t] t_meas; // measurement times
  vector<lower=t0>[N_t] t_cap;  // cap times
  matrix<lower=0>[N_t, num_rep] CO2_flux; // measured carbon fluxes
}
transformed data {
  real x_r[0];  // no real data for ODE system
  int x_i[0];   // no integer data for ODE system
}
parameters {
  vector<lower=0>[3] turnover;  // turnover rates
  simplex[3] gamma[num_rep];    // local partitioning coefficients (a simplex)
  simplex[3] gamma_g;           // global partiotioning coefficient
  vector<lower=0>[3] sigma;     // turnover standard deviation
  real<lower=0> sigma_obs;      // observation standard deviation
```

```
  simplex[3] A1[num_rep];       // output rates from pool 1
  simplex[3] A2[num_rep];       // output rates from pool 2
  simplex[3] A3[num_rep];       // output rates from pool 3
  simplex[3] A1_g;              // global values for rates
  simplex[3] A2_g;              // global values for rates
  simplex[3] A3_g;              // global values for rates
  real<lower=1> kappa_A;
  real<lower=1> kappa_gamma;
}
transformed parameters {
  vector<lower=0>[3] k;             // decomposition rates (1/turnover)
  matrix[N_t, num_rep] CO2_meas;    // evolved CO2 at measurement times
  matrix[N_t, num_rep] CO2_cap;     // evolved CO2 at cap times
  matrix[N_t, num_rep] CO2_flux_hat;// CO2 flux (average evolved CO2 between t_cap & t_meas)
  real<lower=0, upper=1> a21[num_rep];  // transfer rates
  real<lower=0, upper=1> a31[num_rep];
  real<lower=0, upper=1> a12[num_rep];
  real<lower=0, upper=1> a32[num_rep];
  real<lower=0, upper=1> a13[num_rep];
  k = 1 ./ turnover;
  // transfer rates are different for each replication:
  for (i in 1:num_rep) {
    a21[i] = A1[i, 2];
    a31[i] = A1[i, 3];
    a12[i] = A2[i, 1];
    a32[i] = A2[i, 3];
    a13[i] = A3[i, 1];
  }
  for (i in 1:num_rep) {
    CO2_meas[,i] = evolved_CO2(N_t, t0, t_meas, gamma[i], totalC_t0,
                         k, a21[i], a31[i], a12[i], a32[i], a13[i],
                         x_r, x_i);
    CO2_cap[,i] = evolved_CO2(N_t, t0, t_cap, gamma[i], totalC_t0,
                        k, a21[i], a31[i], a12[i], a32[i], a13[i],
                        x_r, x_i);
    CO2_flux_hat[,i] = (CO2_meas[,i] - CO2_cap[,i])./(t_meas - t_cap);
  }
}
model {
  // priors
  turnover[1] ~ normal(1.5, 0.15 * sigma[1]);
  turnover[2] ~ normal(25, 2.5 * sigma[2]);
  turnover[3] ~ normal(1000, 100 * sigma[3]);
  sigma ~ cauchy(0,1);
  sigma_obs ~ cauchy(0,.1);
  kappa_gamma ~ normal(10,5);
  kappa_A ~ normal(10,5);
  for (i in 1:num_rep) {
    A1[i] ~ dirichlet(kappa_A*A1_g);
    A2[i] ~ dirichlet(kappa_A*A2_g);
    A3[i] ~ dirichlet(kappa_A*A3_g);
    gamma[i] ~ dirichlet(kappa_gamma*gamma_g);
  }
```

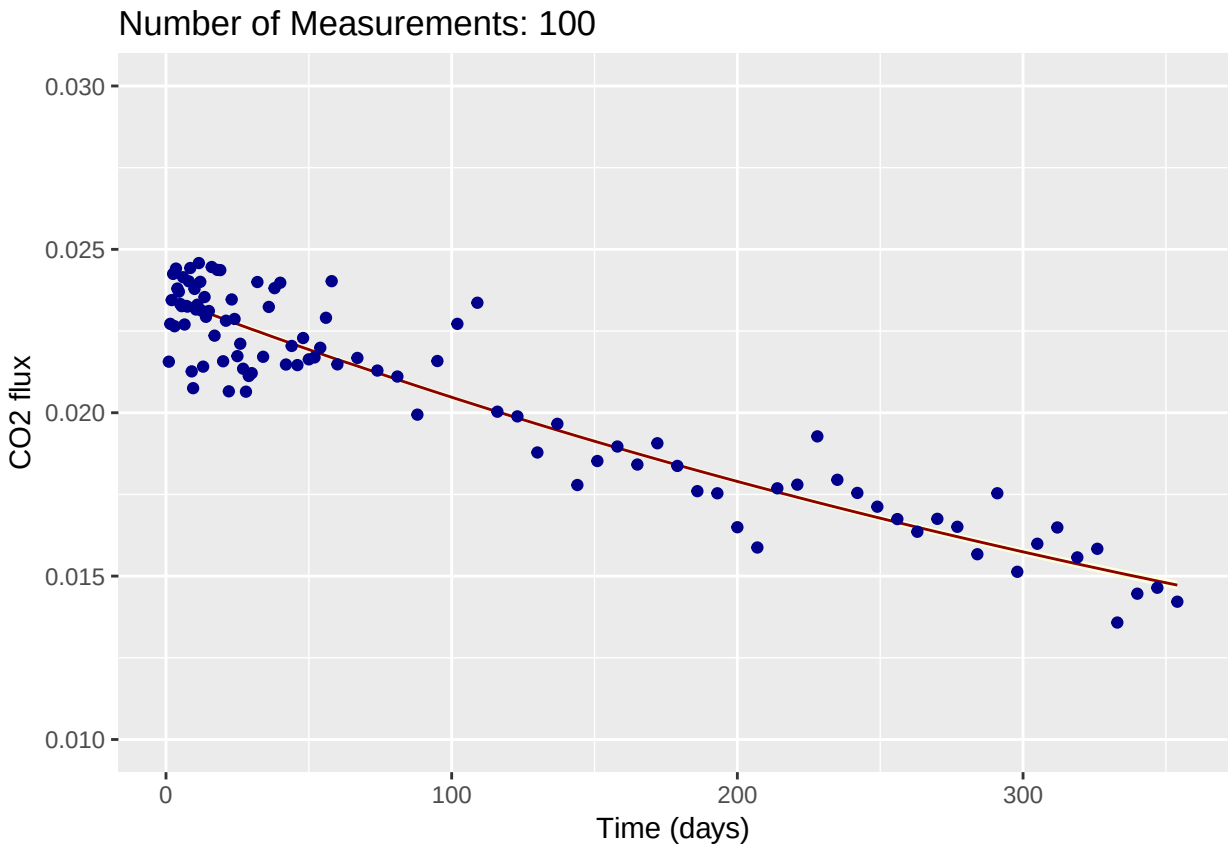
```
  // likelihood
  to_vector(CO2_flux) ~ lognormal(to_vector(log(CO2_flux_hat)), sigma_obs);
}
```

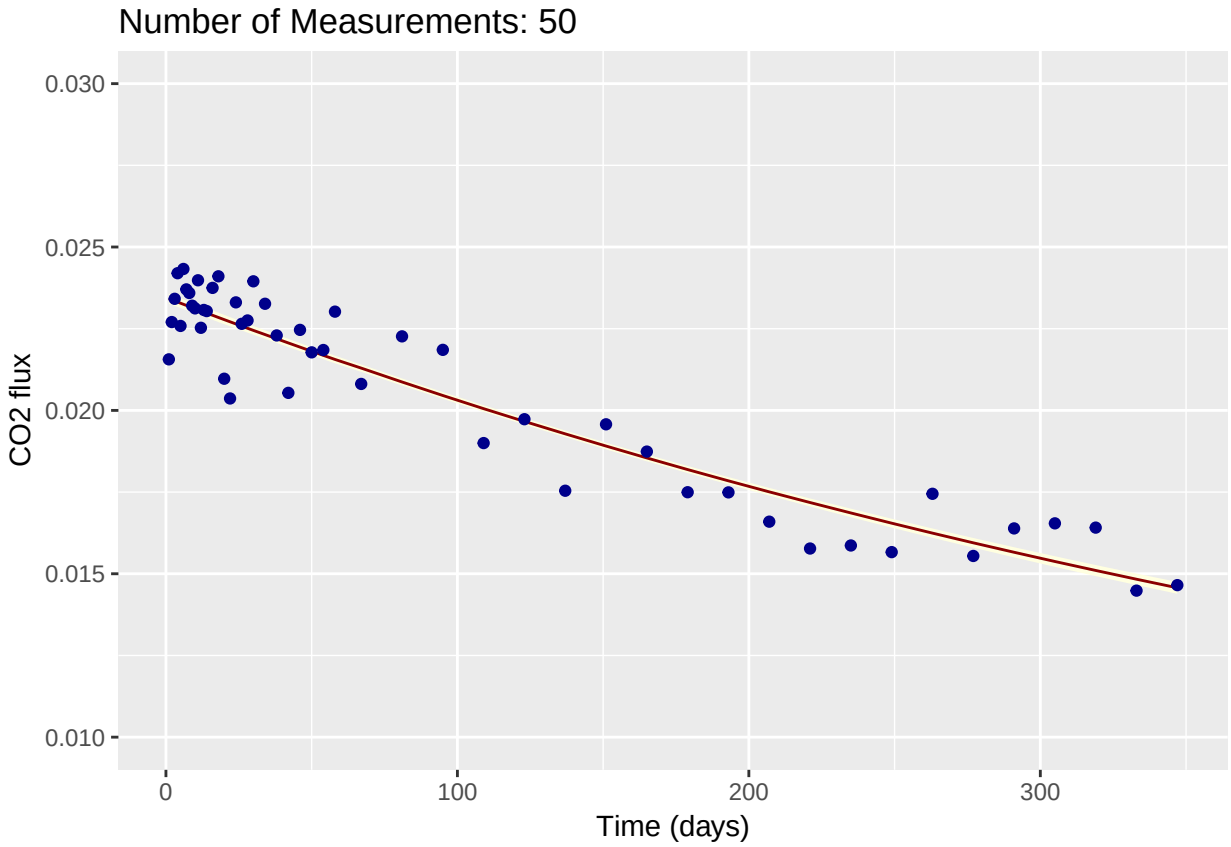
R code to simulate the data and fit the Stan model

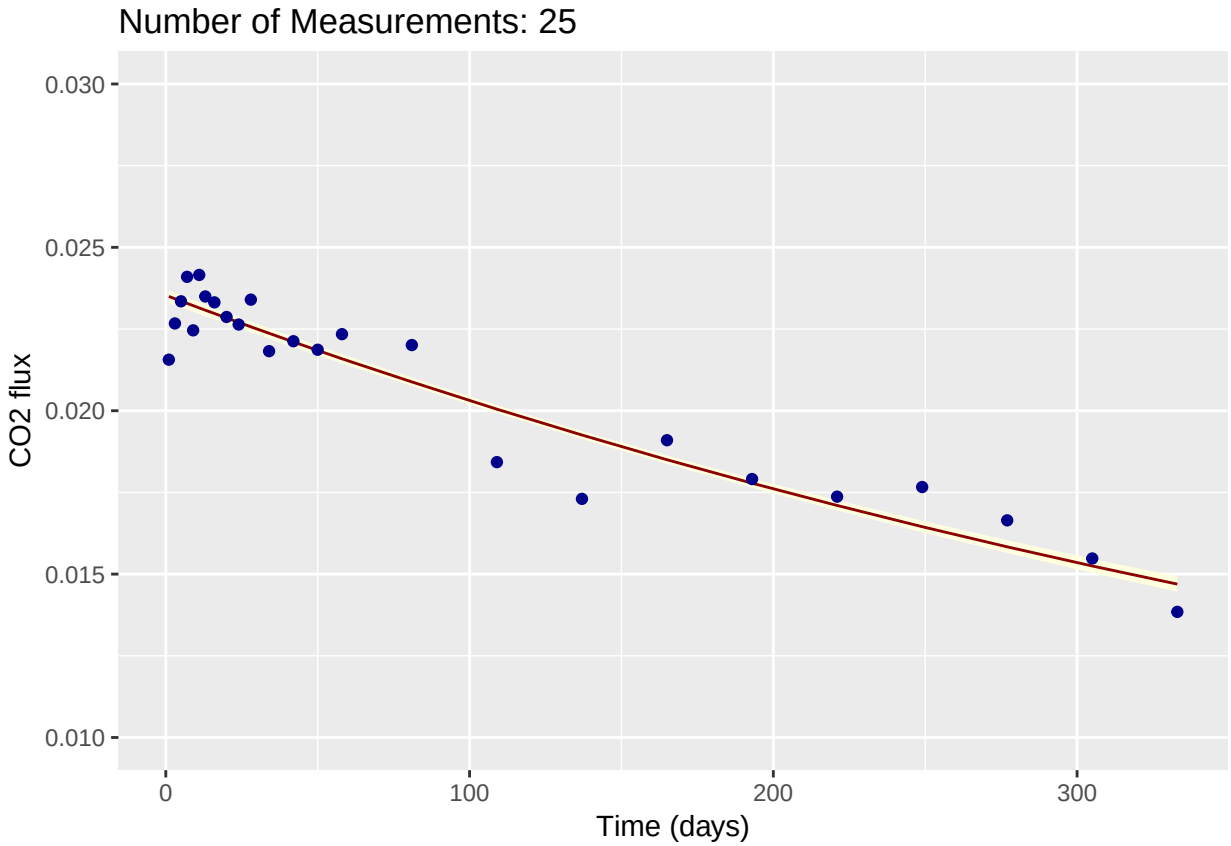
```
library(rstan)
num_rep <- 5
t_meas_all <- c(seq(from=1/360, to=14/360, by=0.5/360), seq(from=15/360, to=30/360, by=1/360),
            seq(from=32/360, to=60/360, by=2/360), seq(from=67/360, to=360/360, by=7/360))
t_cap_all <-  head(t_meas_all, -1) + 0.85 * (tail(t_meas_all, -1) - head(t_meas_all, -1))
t_cap_all <- c(t_meas_all[1] - (t_meas_all[2] - t_cap_all[1]), t_cap_all)
sm <- stan_model("century_hier_nonstiff.stan")
downsample_set <- c(1,2,4,5,10,20,50)
for (rep in 1:3) {
  for (i in 1:length(downsample_set)) {
    i
    n_downsample <- downsample_set[i]
    t_meas <- t_meas_all[seq(1,length(t_meas_all), n_downsample)]
    t_cap <- t_cap_all[seq(1,length(t_cap_all), n_downsample)]
    gamma_global <- 100*c(.1, .1, .8); # global initializations of carbons in the three pools
    source("simulate_data_century.R")
    data <- simulate_data_century(t_meas, t_cap, gamma_global, num_rep);
    rstan_options(auto_write = TRUE);
    options(mc.cores = parallel::detectCores());
    fit <- sampling(sm, data=data, iter=3000, seed=1234, control = list(adapt_delta=0.95));
    saveRDS(c(data=data, fit=fit), file=paste("FITS/fit_",i,"_",rep,".rds",sep=""))
  }
}
```

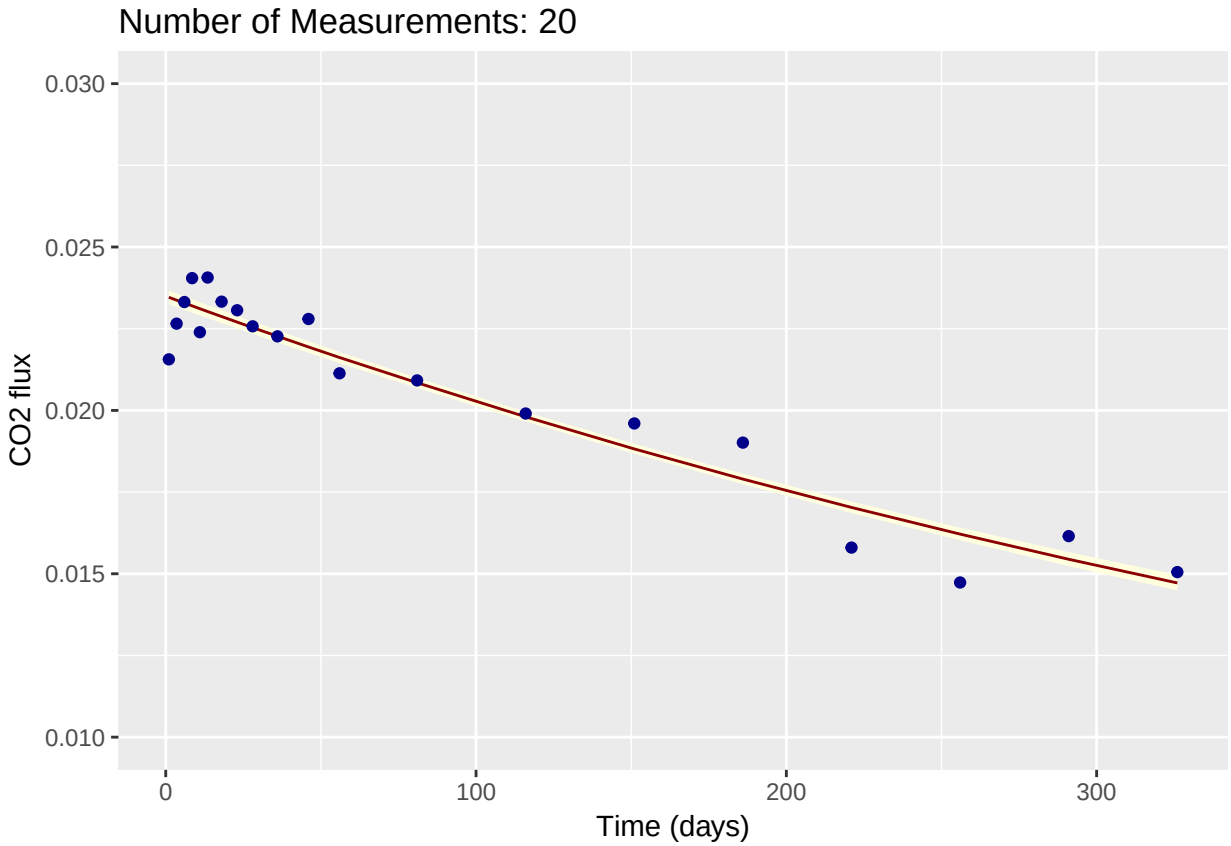
Results

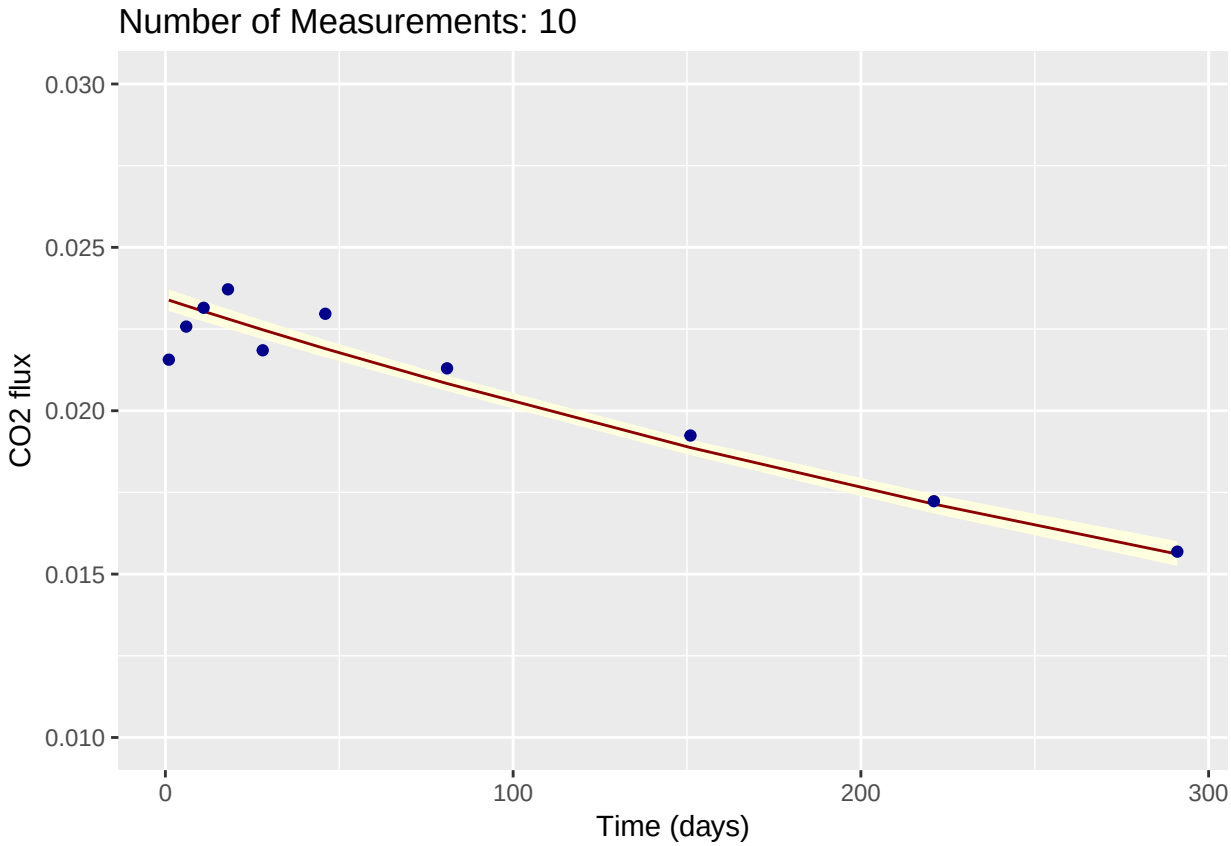
First, we look at the fits. In the figure below we look at the original data, and the fits using different number of measurements. (We have five replicates per experiment, but show only one of them here; results for others are similar.) The blue dots are the actual (noisy) data points. We plot the fitted curve in red and the 50% uncertainty interval in red. As we expect, the uncertainty intervals becomes wider as the number of measurements decreases.

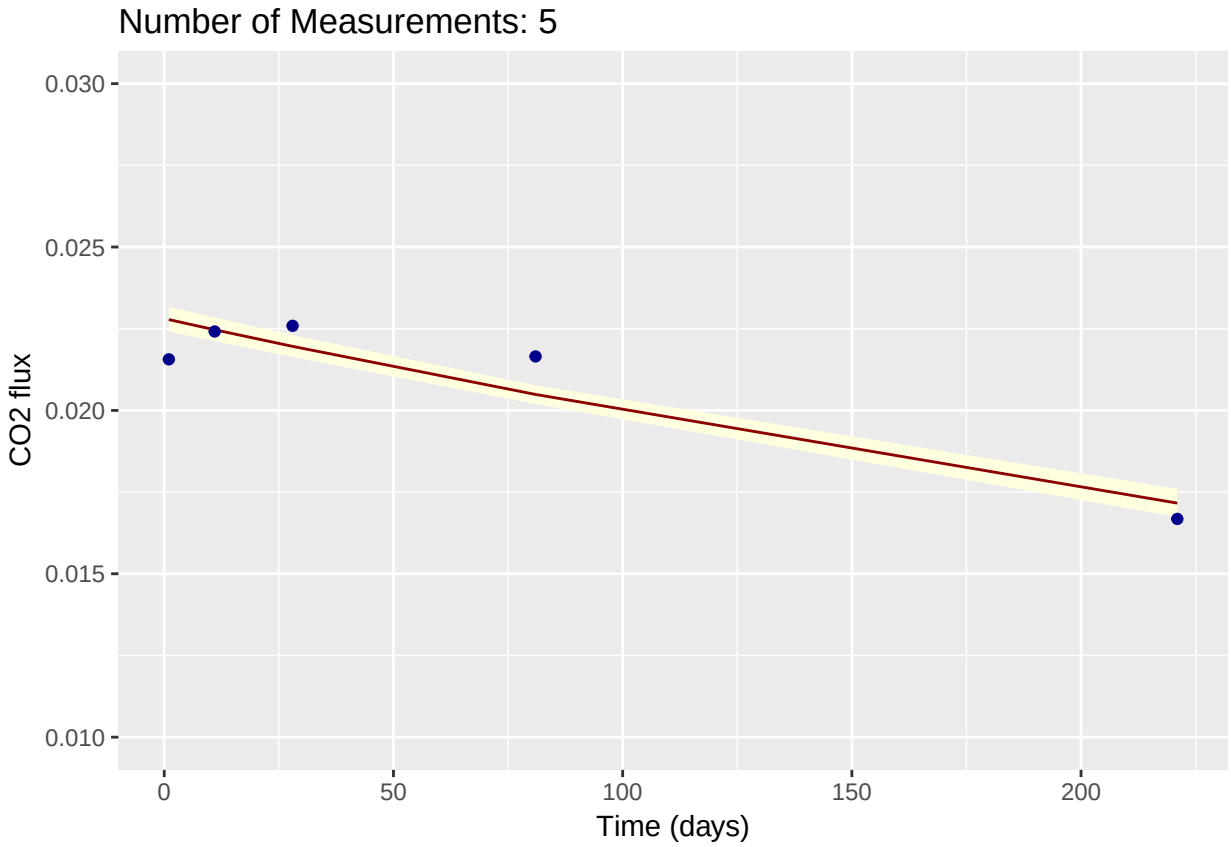

Number of Measurements: 100
0.030
0.025
0.020
0.015
0.010
CO2 flux
0
100
200
300
Time (days)

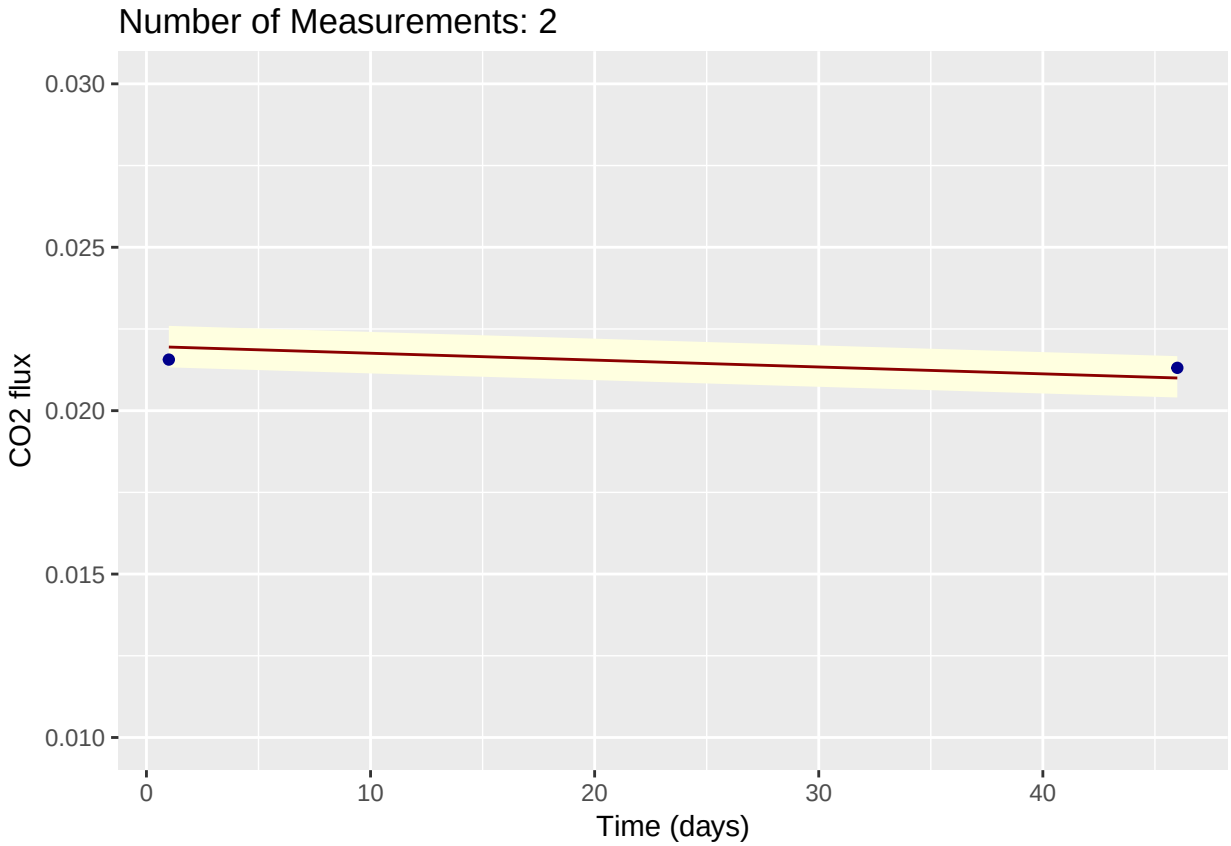

Number of Measurements: 50
0.030
0.025
0.020
0.015
0.010
CO2 flux
0
100
200
300
Time (days)


Number of Measurements: 25
CO2 flux
0.030
0.025
0.020
0.015
0.010
0
100
200
300
Time (days)


Number of Measurements: 20
CO2 flux
0.030
0.025
0.020
0.015
0.010
0
100
200
300
Time (days)

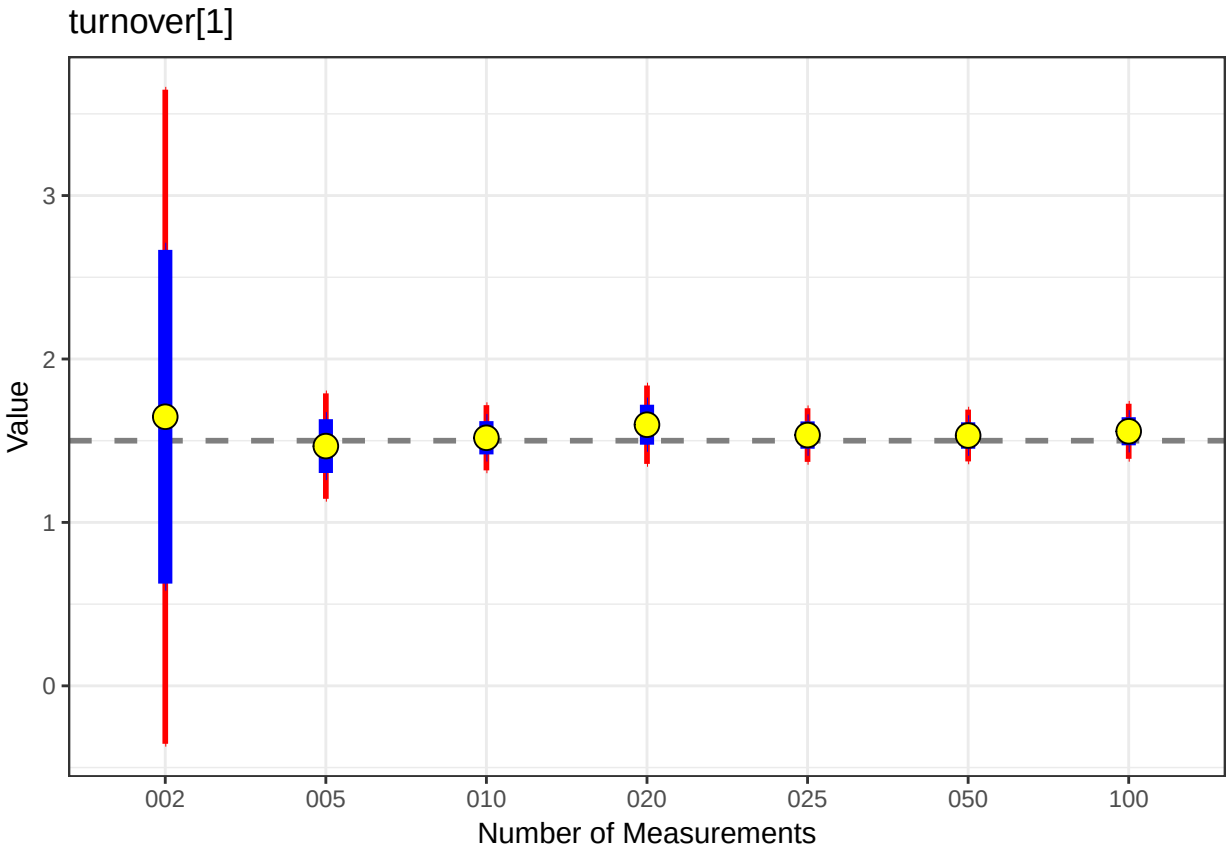

Number of Measurements: 10
CO2 flux
0.030
0.025
0.020
0.015
0.010
0
100
200
300
Time (days)

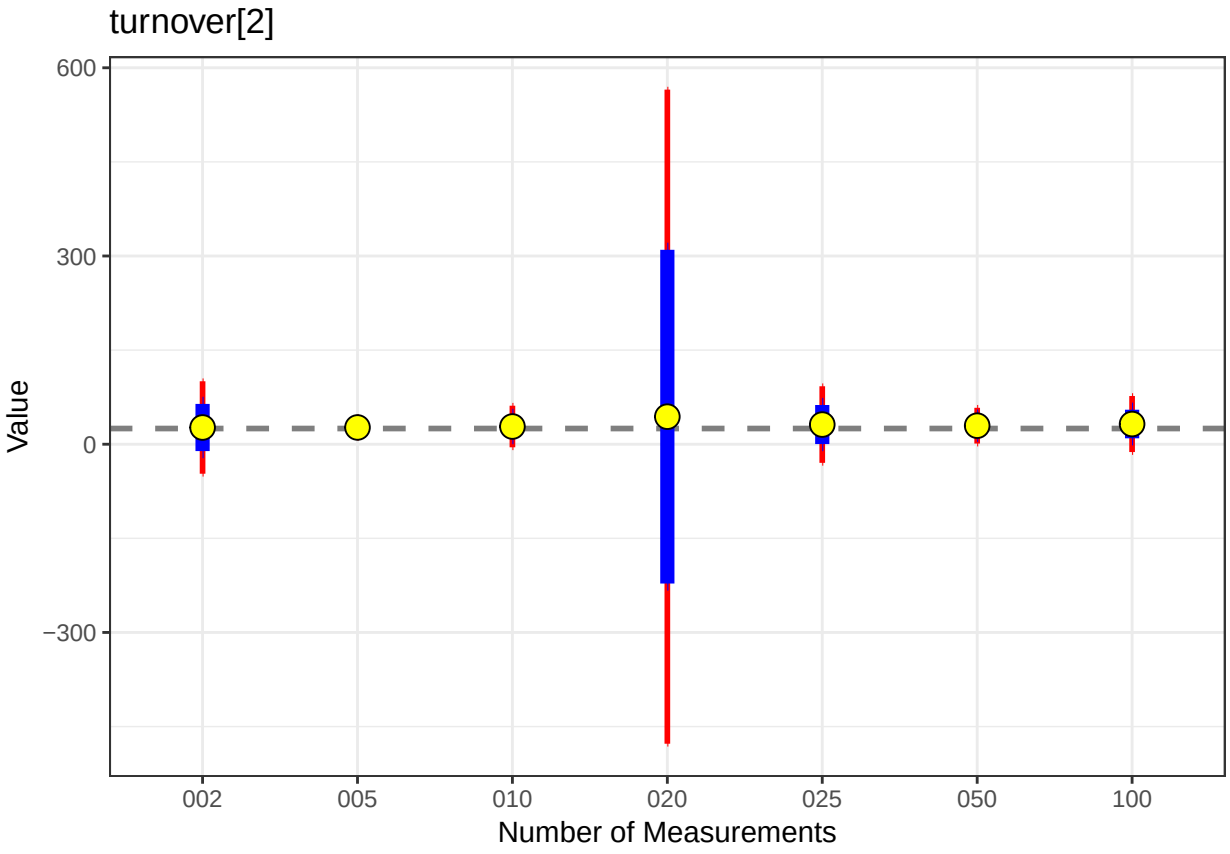

Number of Measurements: 5
CO2 flux
0.030
0.025
0.020
0.015
0.010
0
50
100
150
200
Time (days)

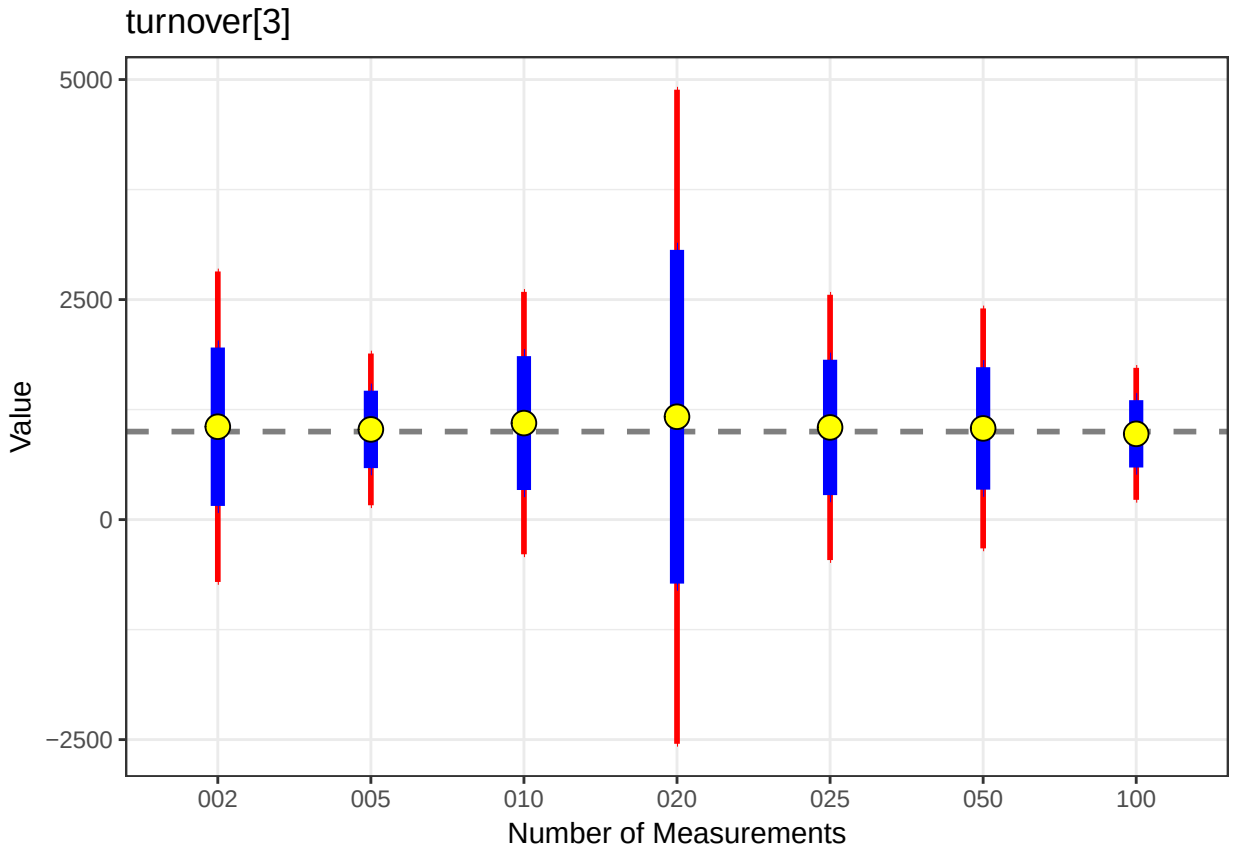


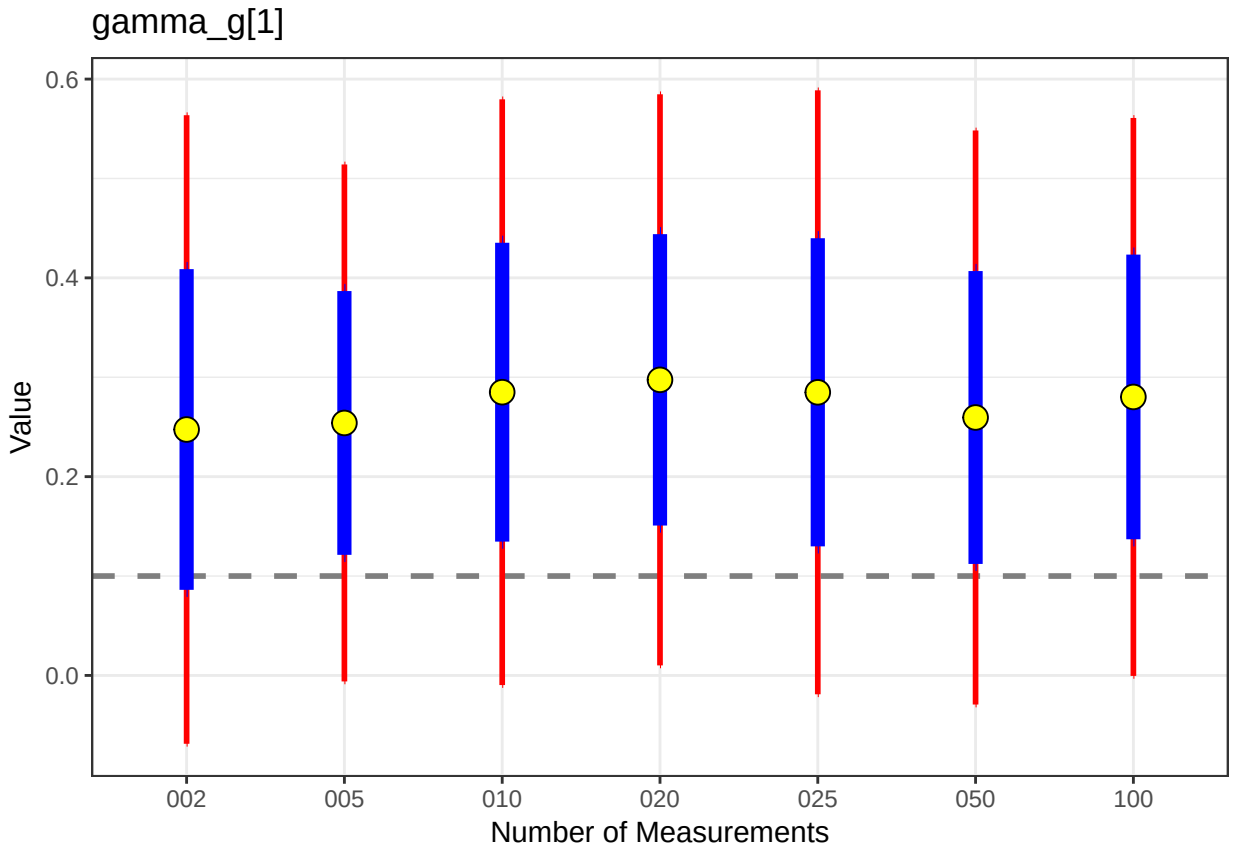
We can also examine the posterior distributions of parameters. In the figures below, we present the estimates (with +/- 1 se and +/- 1.96 se) along with the actual values.

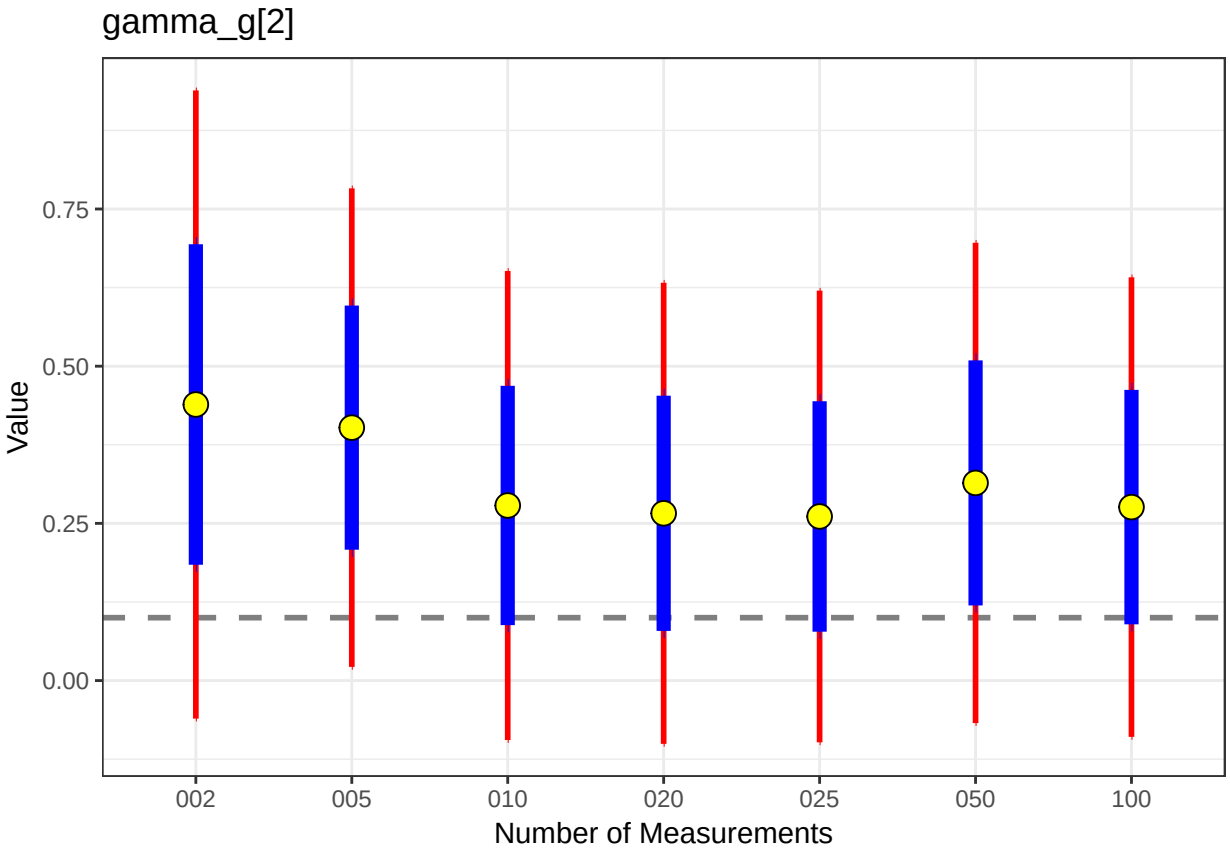
```
## Loading required package: MASS
## Loading required package: Matrix
## Loading required package: lme4
##
## arm (Version 1.9-3, built: 2016-11-21)
## Working directory is /Users/milad/Research/Carbon/decomPower/synthetic_data/Server_runs
##
## Attaching package: 'arm'
## The following object is masked from 'package:rstan':
##
##     traceplot
```

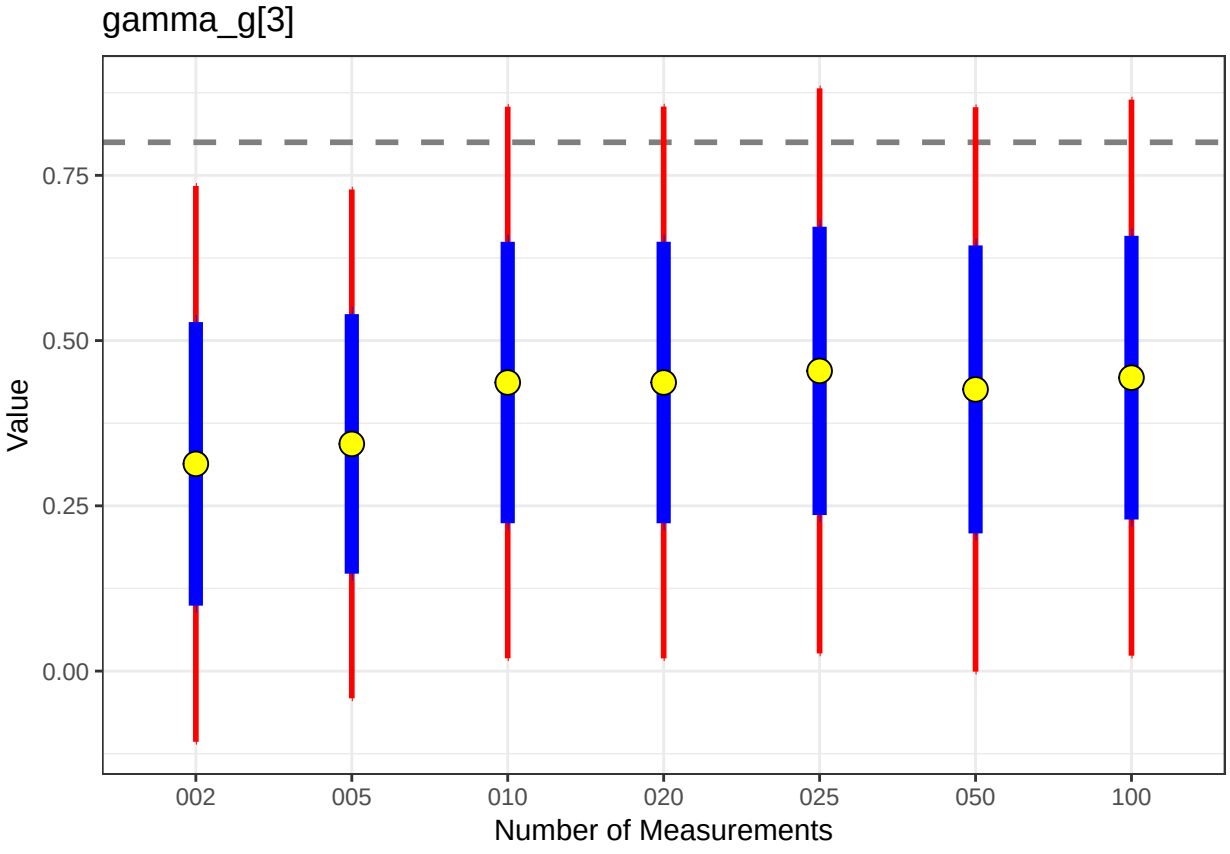

turnover[1]
Value
3
2
1
0
002
005
010
020
025
050
100
Number of Measurements

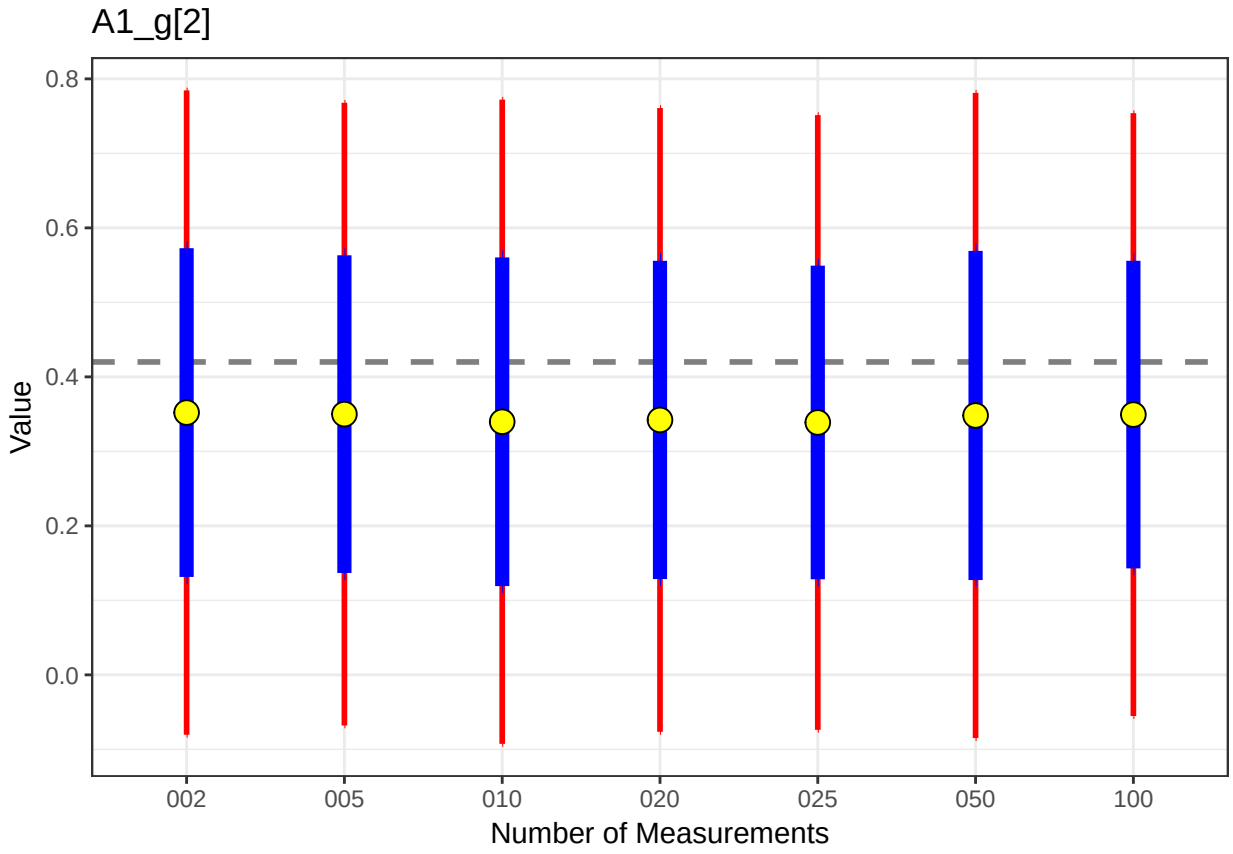

turnover[2]
Value
600
300
0
−300
002
005
010
020
025
050
100
Number of Measurements

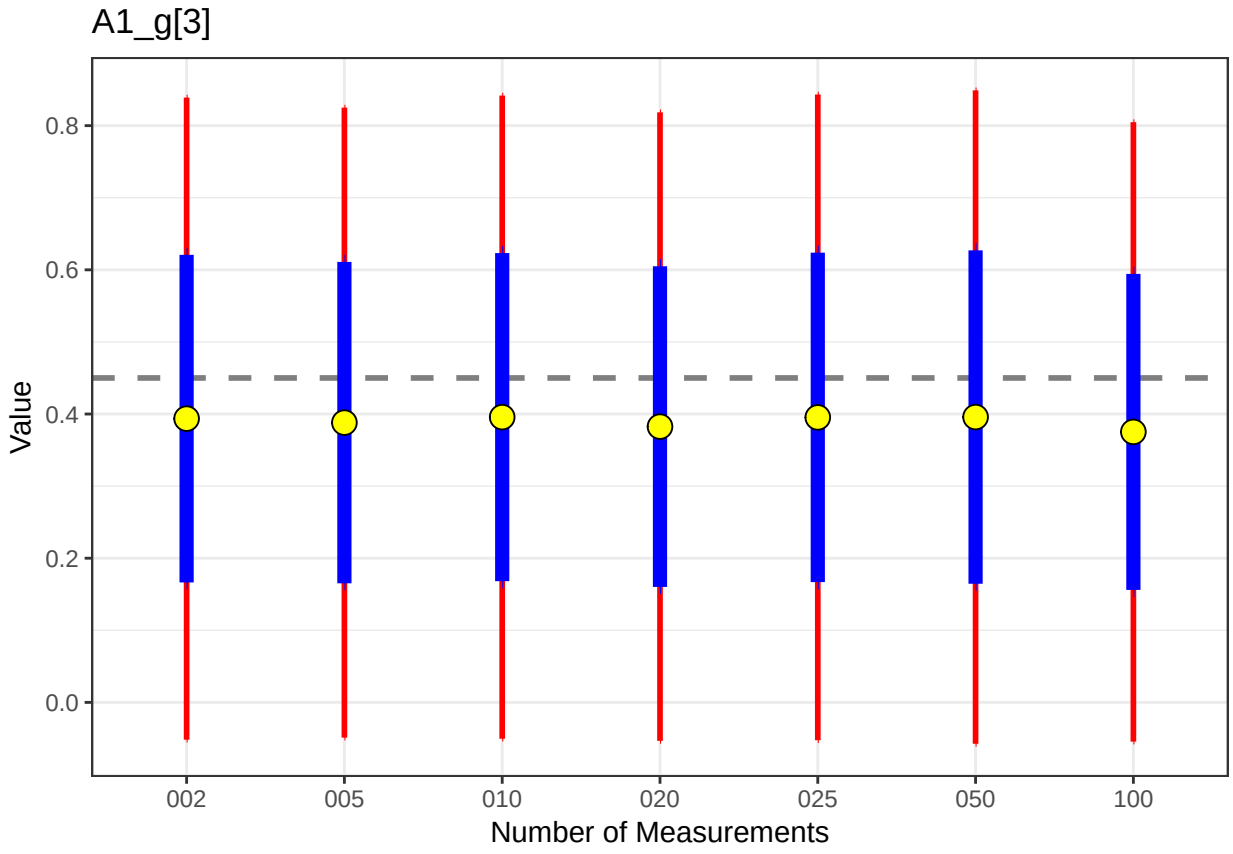

turnover[3]
5000
2500
0
−2500
Value
002
005
010
020
025
050
100
Number of Measurements

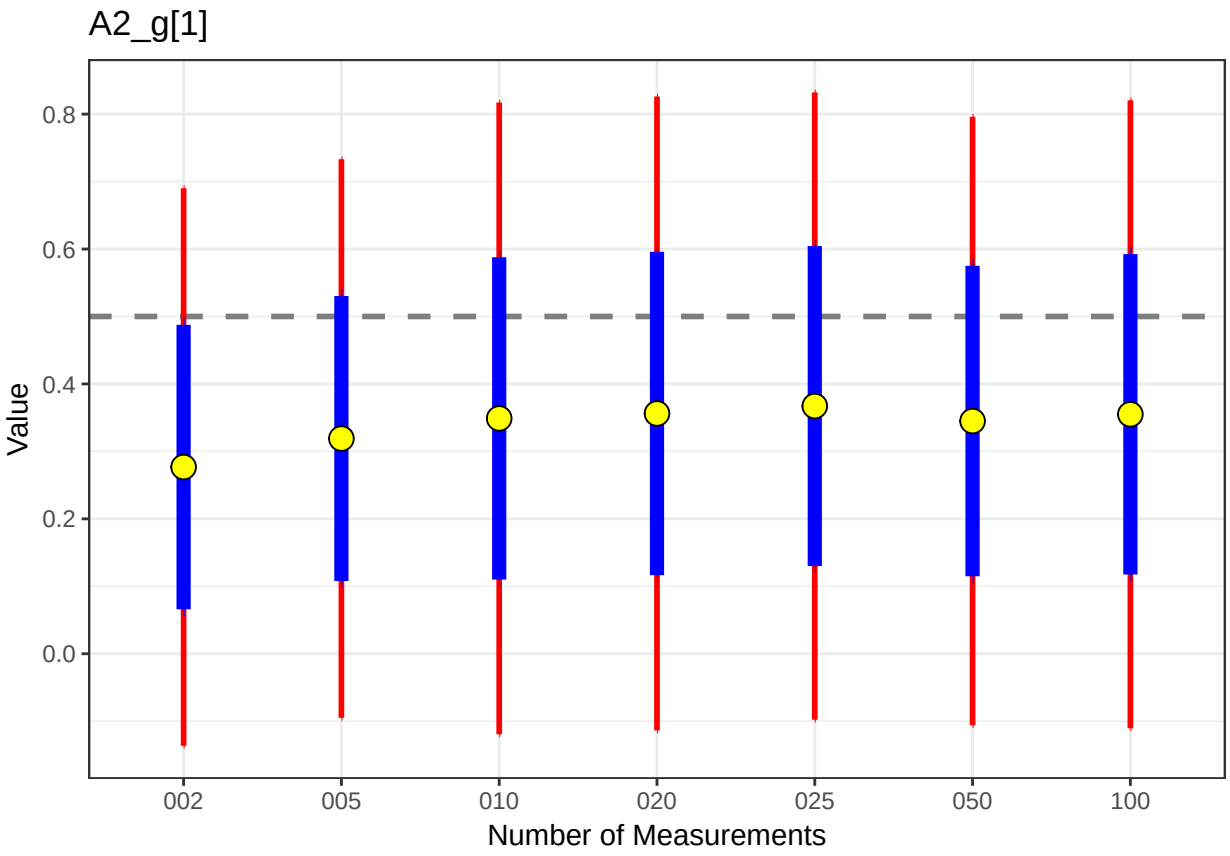

gamma_g[1]
0.6
0.4
0.2
0.0
Value
002
005
010
020
025
050
100
Number of Measurements

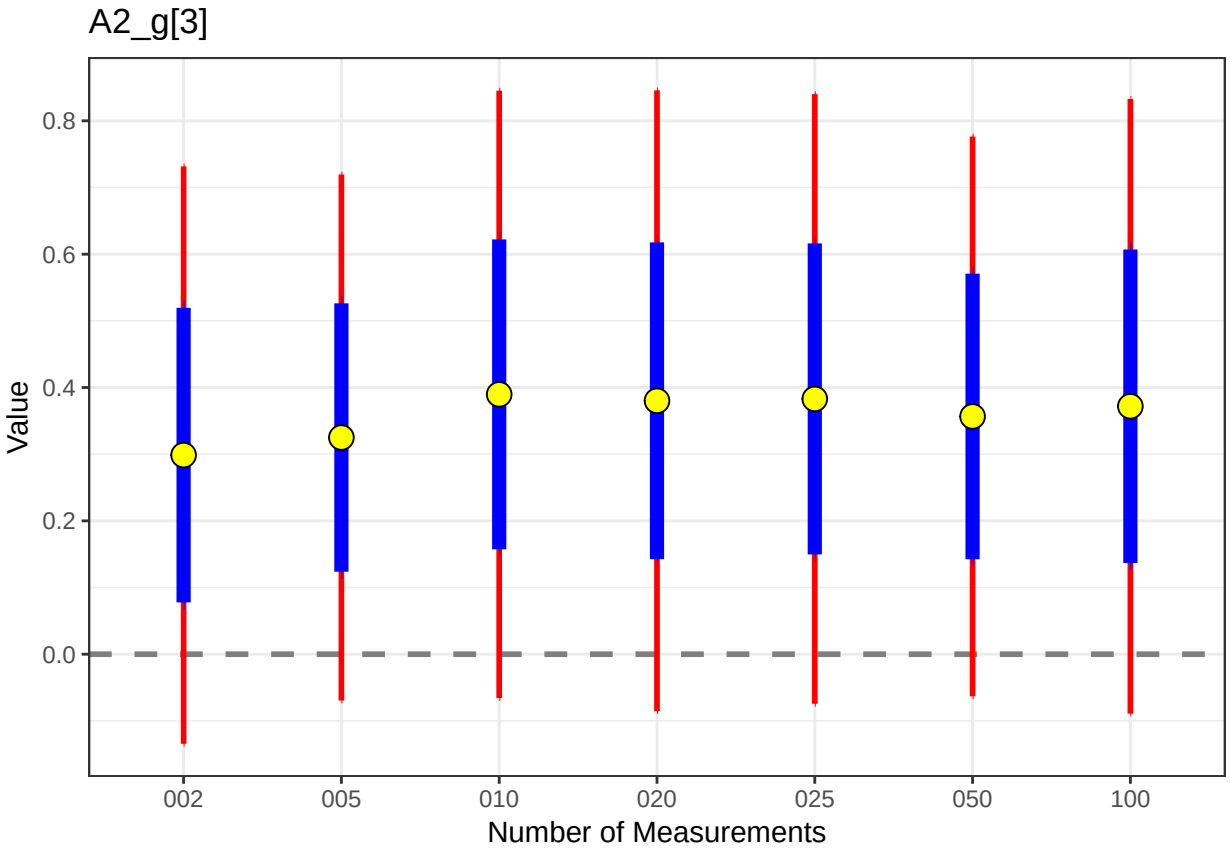

gamma_g[2]
Value
0.75
0.50
0.25
0.00
002
005
010
020
025
050
100
Number of Measurements

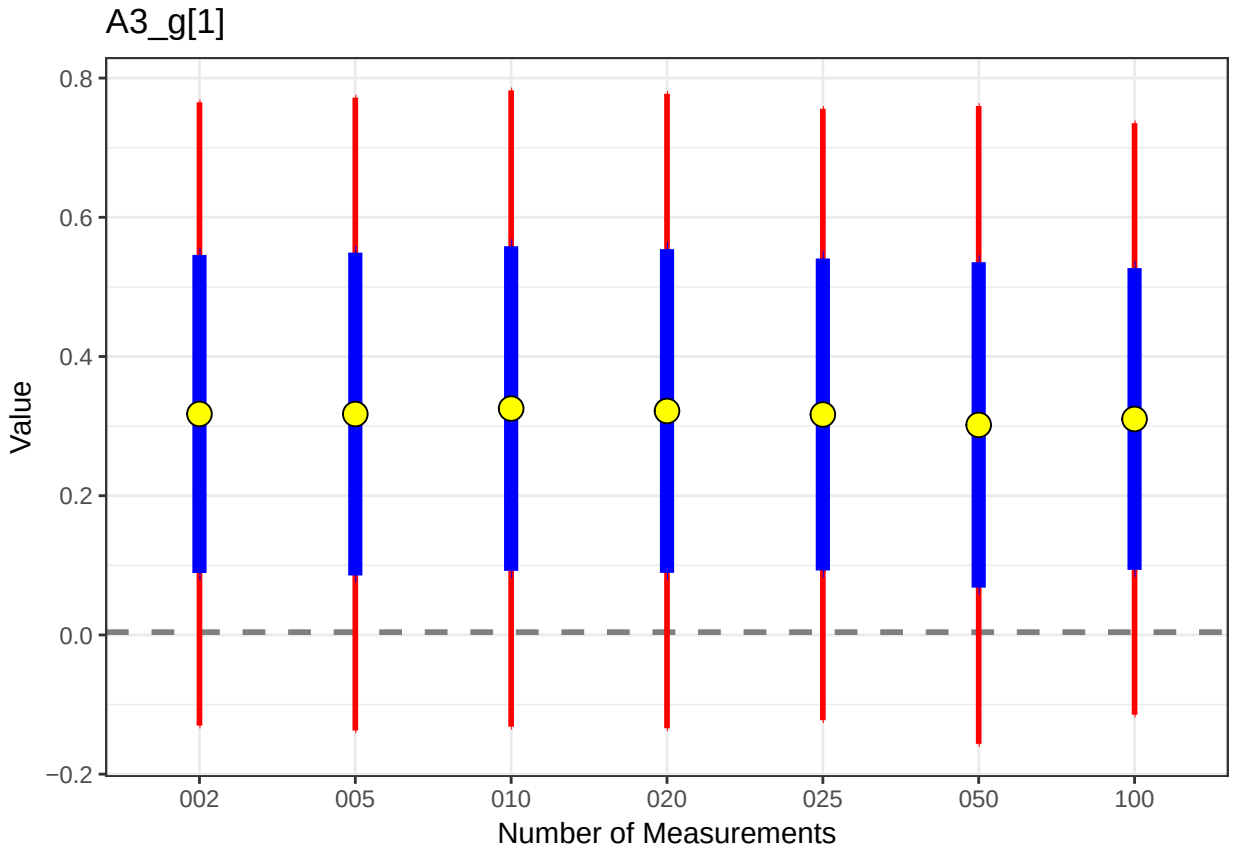

gamma_g[3]
Value
0.75
0.50
0.25
0.00
002
005
010
020
025
050
100
Number of Measurements

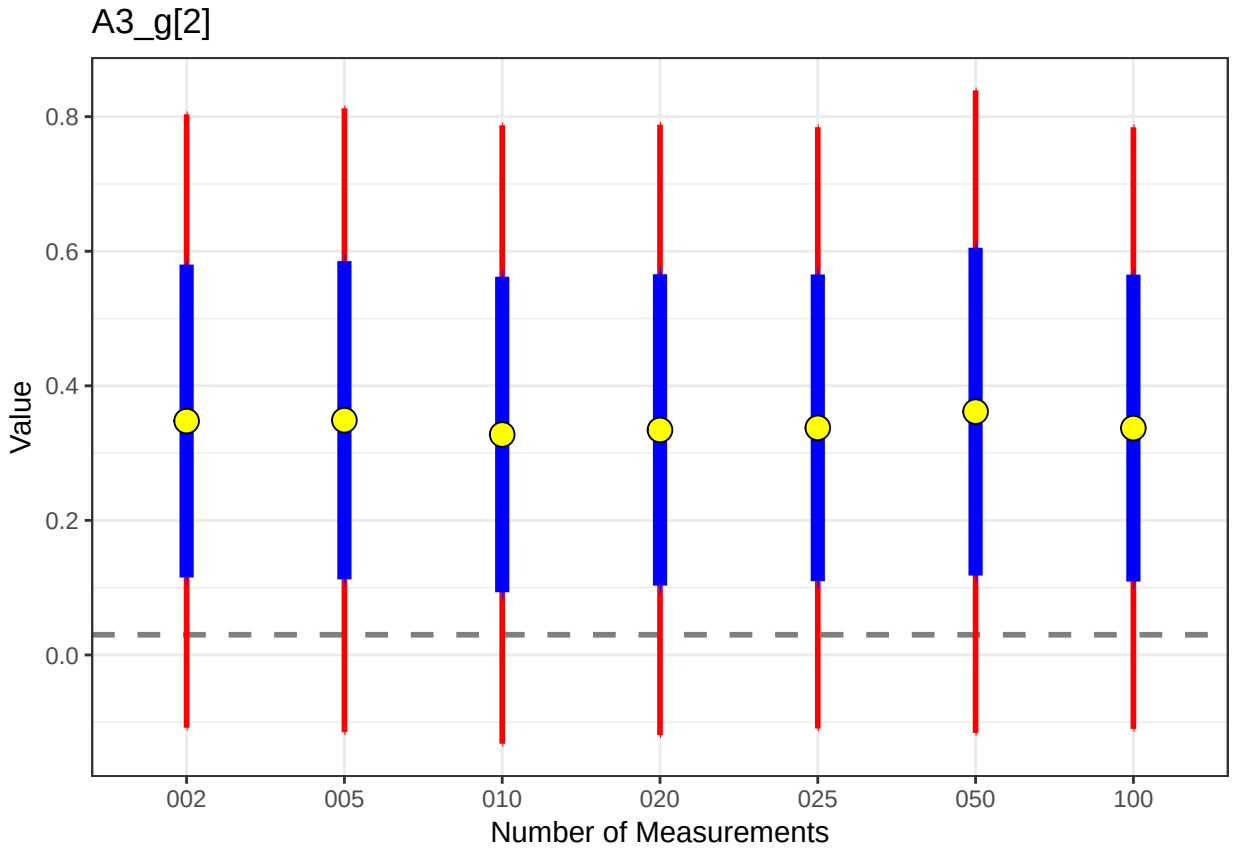

A1_g[2]
0.8
0.6
0.4
0.2
0.0
Value
002
005
010
020
025
050
100
Number of Measurements


A1_g[3]
0.8
0.6
0.4
0.2
0.0
Value
002
005
010
020
025
050
100
Number of Measurements


A2_g[1]
Value
0.8
0.6
0.4
0.2
0.0
002
005
010
020
025
050
100
Number of Measurements


A2_g[3]
Value
0.8
0.6
0.4
0.2
0.0
002
005
010
020
025
050
100
Number of Measurements


A3_g[1]
0.8
0.6
0.4
0.2
0.0
−0.2
Value
002
005
010
020
025
050
100
Number of Measurements


A3_g[2]
0.8
0.6
0.4
0.2
0.0
Value
002
005
010
020
025
050
100
Number of Measurements