Toy example in Appendix D

This notebook contains the code of the paper "Learning Physics between Digital Twins with Low-Fidelity Models and Physics-Informed Gaussian Processes". The models are fitted in rstan and the code is available in the folder "STAN/toy".

Load packages

```
# uncomment to install
# install.packages("rstan")
# install.packages("ggplot2")
# install.packages("SAVE")
library(rstan)
library(ggplot2)
library(SAVE) # package with the data
rstan_options(auto_write = TRUE)
options(mc.cores = 3) # allocate 3 cores (for each model we run 3 chains in parallel)
```

Reality and modelling choice

Note that in the paper there is a mistake in the presentation. The coefficients of the true model $\mathcal{R}$ and model M are swapped and the correct models are the following

$$y^{\mathcal{R}}(x) = 3.5 \cdot \exp(-u \cdot x) + b + \varepsilon \quad \text{(the model we use to simulate data)}$$
$$\eta(x, u) = 5 \cdot \exp(-u \cdot x) \quad \text{(the misspesified model we use to fit the data)}$$

```
R = function(u,x,b) 3.5*exp(-u*x)+b
sd_noise = 0.3
```

```
data("synthfield") # data from the rpackage SAVE
X_loc = unique(synthfield$x)

# simulate data for different u val and add iid noise
u_val = seq(0.8,1.7,by=0.1)
dl=list()
set.seed(123)
offsets=runif(length(u_val),0.5,5) # sample offsets in [0.5,5]
# input locations with 3 relpicates
xobs = c(unique(synthfield$x),unique(synthfield$x),unique(synthfield$x)); N=length(xobs);
X_mat = matrix(NA, nrow = length(u_val), ncol = length(xobs))
set.seed(0)
# sample random input locations
dev=c(runif(length(u_val), 0.1,0.25))
for(i in 1:nrow(X_mat)){
  X_mat[i,] = xobs+dev[i]
}
```

```r
# Predictions at 20 locations x_pred (both interpolation and extrapolation)
Ns = length(u_val) # total number of individuals
id = seq_along(u_val) # individual ids
N_pred=20
x_pred_vec=seq(0.1,5,length.out = N_pred);
x_pred = matrix(rep(x_pred_vec,Ns), nrow = Ns, ncol = N_pred, byrow = TRUE)
# add i.i.d. N(0,0.3^2) noise
for(i in seq_along(u_val)){
  set.seed(0)
  y=R(u_val[i], xobs, offsets[i])+rnorm(N,0,sd=sd_noise);
  dl[[i]] = list(x= X_mat[i,], y = y, N = N, x_pred=x_pred_vec, N_pred=N_pred)
}
y = matrix(NA, nrow = length(u_val), ncol = length(xobs))
for(i in 1:nrow(y)) y[i,] = dl[[i]]$y


# population data
data_population = list(x= X_mat, y = y, N = N, Ns=Ns, id=id, N_pred = N_pred, x_pred=x_pred)
```

Model 1 (no-without delta in paper, Figure 3)

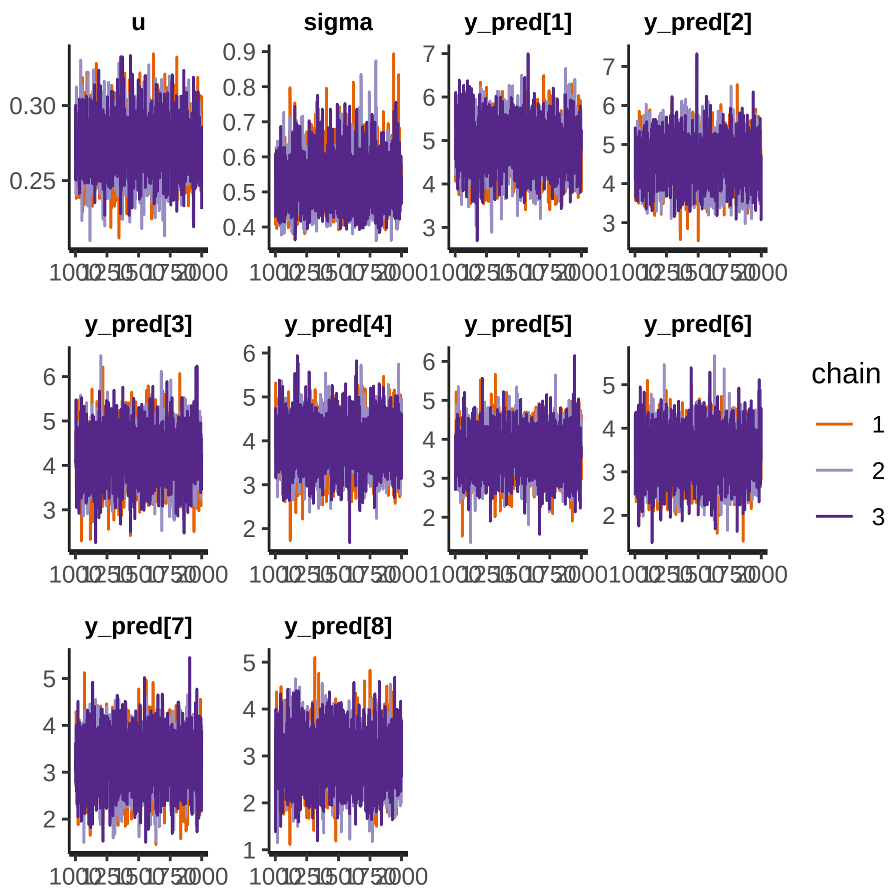
Predictions for the model that does not account for discrepancy

Stan code:

```r
writeLines(readLines('STAN/toy/toy_nodelta_pred.stan'))
```

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;

  int<lower=0>N_pred;
  vector[N_pred] x_pred;
}
parameters {
  real<lower=0,upper=5> u;
  real<lower=0, upper=2> sigma;
}

model {
  // priors
  u ~ normal(1,2);
  // likelihood
  y ~ normal(5*exp(-u*x), sigma);
}
generated quantities {
  vector[N_pred] y_pred;
  for(n in 1:N_pred){
    y_pred[n] = normal_rng(5*exp(-u*x_pred[n]), sigma);
  }
}
```

```r
#-----------------------------------------------------------------------------------------------
# Fit without accounting for discrepancy for each invidual separately (no-without delta))
lu_no=lpred_no=list()
for(i in seq_along(u_val)){
  fit_no_without_delta = stan(
    file='STAN/toy/toy_nodelta_pred.stan', # without delta
    data=dl[[i]],
    chains=3,
    iter=2*1000,
    seed=123
  )
  lu_no[[i]] = extract(fit_no_without_delta)$u
  smr_nnd=summary(fit_no_without_delta)$summary
  ind=grep("y_pred",rownames(smr_nnd))
  lpred_no[[i]]=smr_nnd[ind,c("mean", "2.5%", "97.5%")]
}
stan_trace(fit_no_without_delta)
```




```r
pred_nnd=data.frame(do.call(rbind, lpred_no))
pred_nnd$sharing = "no-without delta"
```

Model 2 (no-with delta in paper, Figure 3)

Now we account for model discrepancy $\delta_m(x_m) \sim GP(0, K_\delta(x_m, x'_m))$, where we use the squared exponential kernel $K_\delta(x_m, x'_m) = \alpha_m^2 \exp\left(-\frac{(x_m - x'_m)^2}{2\rho_m^2}\right)$ and we have the following formulation

$$y^R(\mathbf{x}) = \eta(\mathbf{x}, \boldsymbol{\phi}) + \delta(\mathbf{x}) + \varepsilon, \text{ where } \varepsilon \sim N(0, \sigma^2).$$

This is equivalent to

$$y^{\mathcal{R}} \sim GP(5 \cdot \exp(-u_m \cdot \mathbf{X}_m), K_\delta(\mathbf{X}_m, \mathbf{X}_m \mid \boldsymbol{\omega}_m) + \sigma^2 I).$$

Stan code:

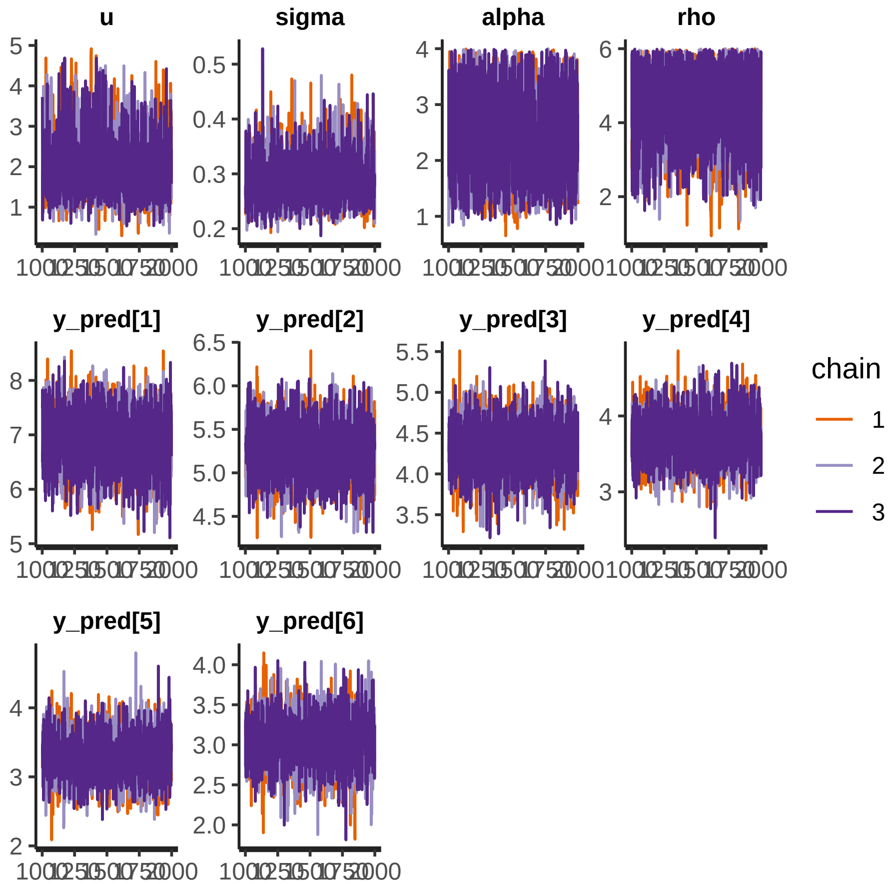
```r
writeLines(readLines('STAN/toy/toy_delta_pred.stan'))
```

```
functions{ // squared exponential kernel
  matrix cov_exp(vector x,
                 real alpha,
                 real rho){
    int n = rows(x);
    matrix[n, n] K;
    // KP
    for (i in 1:(n)){
      K[i,i] = pow(alpha, 0.2e1);
      for (j in (i+1):n){
        K[i,j] = exp(-pow(x[i] - x[j], 0.2e1) * pow(rho, -0.2e1));
        K[i,j] = pow(alpha, 0.2e1) * K[i,j];
        K[j,i] = K[i,j];
      }
      K[n,n] = pow(alpha, 0.2e1);
    }
    return(K);
  }
}

data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;

  int<lower=0> N_pred;
  vector[N_pred] x_pred;
}
transformed data{
  int<lower=1> N_tot;
  N_tot=N+N_pred;
}
parameters {
  real<lower=0,upper=5> u; // physical parameter
  real<lower=0, upper=2> sigma; // noise parameter
  real<lower=0, upper=4> alpha; // marginal sd (delta)
  real<lower=0, upper=6> rho; // length scale (delta)
  // predictions
  vector[N_pred] y_pred;
}
```

```
model {
  vector[N_tot] x_tot;
  vector[N_tot] z;
  matrix[N_tot, N_tot] cov;
  matrix[N_tot, N_tot] L_cov;

  x_tot = append_row(x,x_pred);
  z = append_row(y,y_pred);
  cov = cov_exp(x_tot, alpha, rho)+diag_matrix(rep_vector(sigma^2, N_tot));
  L_cov = cholesky_decompose(cov);

  // priors
  u ~ normal(1,2);
  z ~ multi_normal_cholesky(5*exp(-u*x_tot), L_cov);
}
```

We fit this model to each individual data set separately

```r
#------------------------------------------------------------------------------------------
# Accounting for discrepancy for each invidual separately (no-with delta)
lu=lpred=list()
for(i in seq_along(u_val)){
  fit_no_with_delta = stan(
    file='STAN/toy/toy_delta_pred.stan', # with delta
    data=dl[[i]],
    chains=3,
    iter=2*1000,
    seed=123
  )
  lu[[i]] = extract(fit_no_with_delta)$u
  smr_nwd=summary(fit_no_with_delta)$summary
  ind=grep("y_pred",rownames(smr_nwd))
  lpred[[i]]=smr_nwd[ind,c("mean", "2.5%", "97.5%")]
}
stan_trace(fit_no_with_delta)
```




```
pred_nwd=data.frame(do.call(rbind, lpred))
pred_nwd$sharing = "no-with delta"
```

Model 3 (yes/common delta, Figure 3)

We allow individuals to share information about the physical parameters $u_m, m = 1, 2, \ldots, 10$ through a global level parameter as described in Section 3.2. The model assumes same discrepancy parameters for all individuals.

Stan code:

```
writeLines(readLines('STAN/toy/toy_common_delta_pred.stan'))
```

```
functions{ // squared exponential kernel
  matrix cov_exp(vector x,
                 real alpha,
                 real rho){
    int n = rows(x);
    matrix[n, n] K;
    // KP
    for (i in 1:(n)){
      K[i,i] = pow(alpha, 0.2e1);
      for (j in (i+1):n){
        K[i,j] = exp(-pow(x[i] - x[j], 0.2e1) * pow(rho, -0.2e1));
        K[i,j] = pow(alpha, 0.2e1) * K[i,j];
        K[j,i] = K[i,j];
```

```
      }
      K[n,n] = pow(alpha, 0.2e1);
    }
    return(K);
  }
}
data {
  int<lower=1> N; //number of observations per individual
  int<lower=1> Ns;
  int<lower=1,upper=Ns> id[Ns]; //number of individuals
  matrix[Ns, N] x; //different across individuals (e.g. time)
  matrix[Ns, N] y; //input data
  // predictions
  int<lower=0> N_pred;
  matrix[Ns, N_pred] x_pred;
}
transformed data{
  int<lower=1> N_tot;
  N_tot=N+N_pred;
}
parameters {
  real<lower=0, upper=3.0> u_tilde[Ns]; // non-centered parameterization

  real<lower=0, upper=4> alpha; // marginal sd (delta)
  real<lower=0, upper=6> rho;  // length scale (delta)

  real<lower=0> sigma; // noise sd
  real<lower=0.5, upper=1.8> mu; // Global mean for u
  real<lower=1, upper=2> tau; // Global sd for u
  // predictions
  matrix[Ns,N_pred] y_pred;
}
transformed parameters {
  real<lower=0> u[Ns];   // individual u
  // Non-centered parameterization
  for (s in 1:Ns) {
    u[s] = mu + tau * u_tilde[s];
  }
}
model {
  matrix[Ns,N_tot] x_tot;
  matrix[N_tot, N_tot] cov[Ns];
  matrix[N_tot, N_tot] L_cov[Ns];
  matrix[Ns,N_tot] z;

  x_tot=append_col(x,x_pred);
  z= append_col(y,y_pred);
  for (s in 1:Ns) {
    cov[s] = cov_exp(to_vector(x_tot[s]), alpha, rho)+diag_matrix(rep_vector(sigma^2, N_tot));
    L_cov[s] = cholesky_decompose(cov[s]);
  }

  u_tilde ~ normal(0, 1);  // non-centered
  // likelihood
```

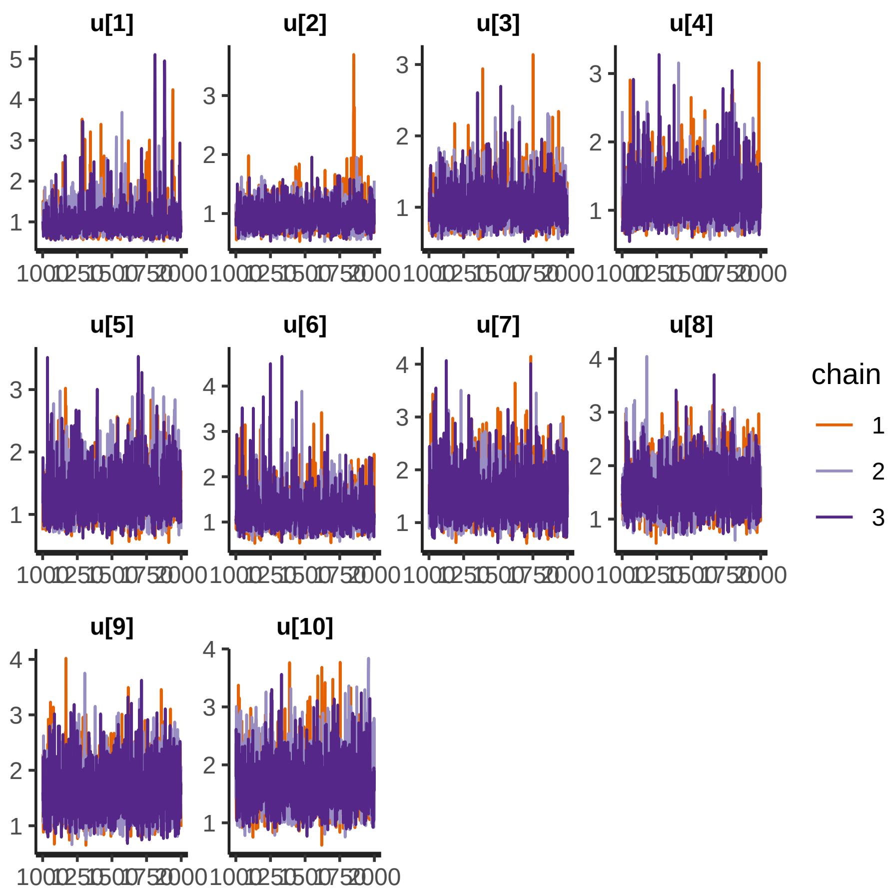
```
  for (i in 1:Ns){
        z[i] ~ multi_normal_cholesky(5*exp(-u[id[i]]*x_tot[i]), L_cov[id[i]]);

  }
}
```

```
#------------------------------------------------------------------------------------------------
# shared u common delta model
fit_yes_common_delta = stan(file='STAN/toy/toy_common_delta_pred.stan',
                            data=data_population,
                            chains=3,
                            iter=2*1000,
                            seed=123
)
names(fit_yes_common_delta)
```

```
  [1] "u_tilde[1]"    "u_tilde[2]"    "u_tilde[3]"    "u_tilde[4]"
  [5] "u_tilde[5]"    "u_tilde[6]"    "u_tilde[7]"    "u_tilde[8]"
  [9] "u_tilde[9]"    "u_tilde[10]"   "alpha"         "rho"
 [13] "sigma"         "mu"            "tau"           "y_pred[1,1]"
 [17] "y_pred[2,1]"   "y_pred[3,1]"   "y_pred[4,1]"   "y_pred[5,1]"
 [21] "y_pred[6,1]"   "y_pred[7,1]"   "y_pred[8,1]"   "y_pred[9,1]"
 [25] "y_pred[10,1]"  "y_pred[1,2]"   "y_pred[2,2]"   "y_pred[3,2]"
 [29] "y_pred[4,2]"   "y_pred[5,2]"   "y_pred[6,2]"   "y_pred[7,2]"
 [33] "y_pred[8,2]"   "y_pred[9,2]"   "y_pred[10,2]"  "y_pred[1,3]"
 [37] "y_pred[2,3]"   "y_pred[3,3]"   "y_pred[4,3]"   "y_pred[5,3]"
 [41] "y_pred[6,3]"   "y_pred[7,3]"   "y_pred[8,3]"   "y_pred[9,3]"
 [45] "y_pred[10,3]"  "y_pred[1,4]"   "y_pred[2,4]"   "y_pred[3,4]"
 [49] "y_pred[4,4]"   "y_pred[5,4]"   "y_pred[6,4]"   "y_pred[7,4]"
 [53] "y_pred[8,4]"   "y_pred[9,4]"   "y_pred[10,4]"  "y_pred[1,5]"
 [57] "y_pred[2,5]"   "y_pred[3,5]"   "y_pred[4,5]"   "y_pred[5,5]"
 [61] "y_pred[6,5]"   "y_pred[7,5]"   "y_pred[8,5]"   "y_pred[9,5]"
 [65] "y_pred[10,5]"  "y_pred[1,6]"   "y_pred[2,6]"   "y_pred[3,6]"
 [69] "y_pred[4,6]"   "y_pred[5,6]"   "y_pred[6,6]"   "y_pred[7,6]"
 [73] "y_pred[8,6]"   "y_pred[9,6]"   "y_pred[10,6]"  "y_pred[1,7]"
 [77] "y_pred[2,7]"   "y_pred[3,7]"   "y_pred[4,7]"   "y_pred[5,7]"
 [81] "y_pred[6,7]"   "y_pred[7,7]"   "y_pred[8,7]"   "y_pred[9,7]"
 [85] "y_pred[10,7]"  "y_pred[1,8]"   "y_pred[2,8]"   "y_pred[3,8]"
 [89] "y_pred[4,8]"   "y_pred[5,8]"   "y_pred[6,8]"   "y_pred[7,8]"
 [93] "y_pred[8,8]"   "y_pred[9,8]"   "y_pred[10,8]"  "y_pred[1,9]"
 [97] "y_pred[2,9]"   "y_pred[3,9]"   "y_pred[4,9]"   "y_pred[5,9]"
[101] "y_pred[6,9]"   "y_pred[7,9]"   "y_pred[8,9]"   "y_pred[9,9]"
[105] "y_pred[10,9]"  "y_pred[1,10]"  "y_pred[2,10]"  "y_pred[3,10]"
[109] "y_pred[4,10]"  "y_pred[5,10]"  "y_pred[6,10]"  "y_pred[7,10]"
[113] "y_pred[8,10]"  "y_pred[9,10]"  "y_pred[10,10]" "y_pred[1,11]"
[117] "y_pred[2,11]"  "y_pred[3,11]"  "y_pred[4,11]"  "y_pred[5,11]"
[121] "y_pred[6,11]"  "y_pred[7,11]"  "y_pred[8,11]"  "y_pred[9,11]"
[125] "y_pred[10,11]" "y_pred[1,12]"  "y_pred[2,12]"  "y_pred[3,12]"
[129] "y_pred[4,12]"  "y_pred[5,12]"  "y_pred[6,12]"  "y_pred[7,12]"
[133] "y_pred[8,12]"  "y_pred[9,12]"  "y_pred[10,12]" "y_pred[1,13]"
[137] "y_pred[2,13]"  "y_pred[3,13]"  "y_pred[4,13]"  "y_pred[5,13]"
[141] "y_pred[6,13]"  "y_pred[7,13]"  "y_pred[8,13]"  "y_pred[9,13]"
[145] "y_pred[10,13]" "y_pred[1,14]"  "y_pred[2,14]"  "y_pred[3,14]"
[149] "y_pred[4,14]"  "y_pred[5,14]"  "y_pred[6,14]"  "y_pred[7,14]"
[153] "y_pred[8,14]"  "y_pred[9,14]"  "y_pred[10,14]" "y_pred[1,15]"
```

```
[157] "y_pred[2,15]"  "y_pred[3,15]"  "y_pred[4,15]"  "y_pred[5,15]"
[161] "y_pred[6,15]"  "y_pred[7,15]"  "y_pred[8,15]"  "y_pred[9,15]"
[165] "y_pred[10,15]" "y_pred[1,16]"  "y_pred[2,16]"  "y_pred[3,16]"
[169] "y_pred[4,16]"  "y_pred[5,16]"  "y_pred[6,16]"  "y_pred[7,16]"
[173] "y_pred[8,16]"  "y_pred[9,16]"  "y_pred[10,16]" "y_pred[1,17]"
[177] "y_pred[2,17]"  "y_pred[3,17]"  "y_pred[4,17]"  "y_pred[5,17]"
[181] "y_pred[6,17]"  "y_pred[7,17]"  "y_pred[8,17]"  "y_pred[9,17]"
[185] "y_pred[10,17]" "y_pred[1,18]"  "y_pred[2,18]"  "y_pred[3,18]"
[189] "y_pred[4,18]"  "y_pred[5,18]"  "y_pred[6,18]"  "y_pred[7,18]"
[193] "y_pred[8,18]"  "y_pred[9,18]"  "y_pred[10,18]" "y_pred[1,19]"
[197] "y_pred[2,19]"  "y_pred[3,19]"  "y_pred[4,19]"  "y_pred[5,19]"
[201] "y_pred[6,19]"  "y_pred[7,19]"  "y_pred[8,19]"  "y_pred[9,19]"
[205] "y_pred[10,19]" "y_pred[1,20]"  "y_pred[2,20]"  "y_pred[3,20]"
[209] "y_pred[4,20]"  "y_pred[5,20]"  "y_pred[6,20]"  "y_pred[7,20]"
[213] "y_pred[8,20]"  "y_pred[9,20]"  "y_pred[10,20]" "u[1]"
[217] "u[2]"          "u[3]"          "u[4]"          "u[5]"
[221] "u[6]"          "u[7]"          "u[8]"          "u[9]"
[225] "u[10]"         "lp__"
```

```
stan_trace(fit_yes_common_delta, pars = "u")
```




```
ex_ycd=extract(fit_yes_common_delta)
```

Model 4 (yes/shared delta, Figure 3)

We allow individuals to share information about both the physical parameters $u_m, m = 1, 2, \ldots, 10$ and the discrepancy through a global level parameters for bothas described in Section 3.1. The model assumes same discrepancy parameters for all individuals.

Stan code:

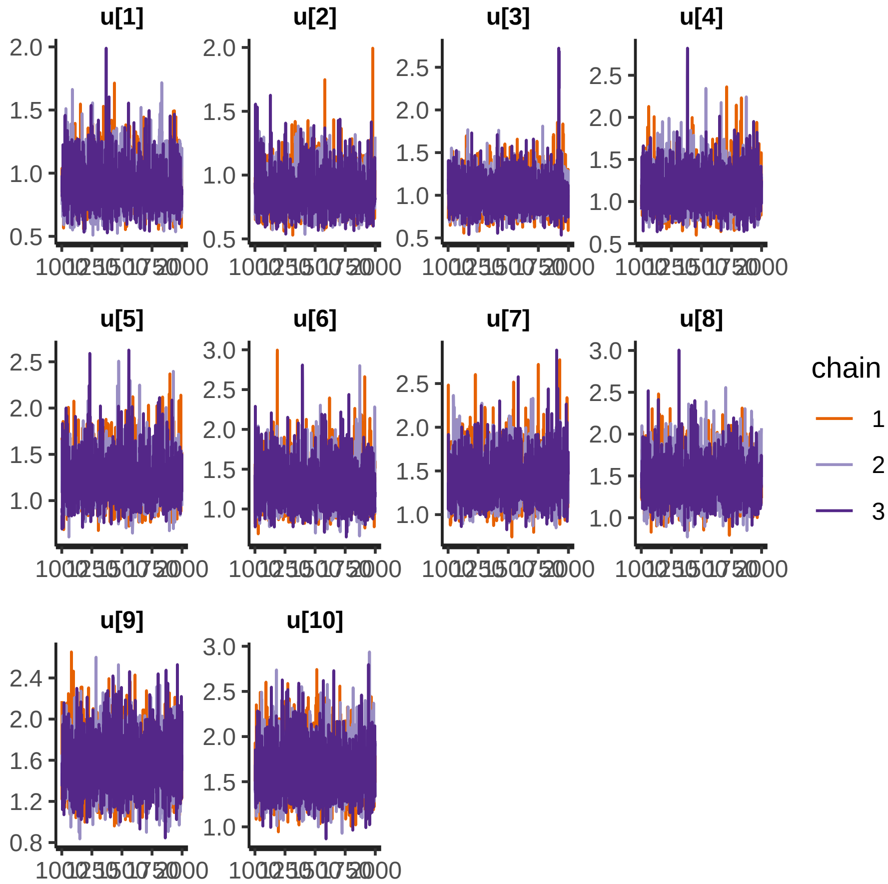
```
writeLines(readLines('STAN/toy/toy_shared_delta_pred.stan'))
```

```
functions{
  matrix cov_exp(vector x,
                 real alpha,
                 real rho){
    int n = rows(x);
    matrix[n, n] K;
    // KP
    for (i in 1:(n)){
      K[i,i] = pow(alpha, 0.2e1);
      for (j in (i+1):n){
        K[i,j] = exp(-pow(x[i] - x[j], 0.2e1) * pow(rho, -0.2e1));
        K[i,j] = pow(alpha, 0.2e1) * K[i,j];
        K[j,i] = K[i,j];
      }
      K[n,n] = pow(alpha, 0.2e1);
    }
    return(K);
  }
}
data {
  int<lower=1> N; // number of observations per individual
  int<lower=1> Ns; // number of individuals
  int<lower=1,upper=Ns> id[Ns]; // individual id
  matrix[Ns, N] x; // individual input vector
  matrix[Ns, N] y; // matrix of all individual outputs
  // predictions
  int<lower=0> N_pred;
  matrix[Ns, N_pred] x_pred;
}
transformed data{
  int<lower=1> N_tot;
  N_tot=N+N_pred;
}
parameters {
  // non-centered parameterization parameters
  real<lower=0,upper=3.0> u_tilde[Ns];
  real rho_tilde[Ns];   // non-centered sd of rho (delta process)
  real alpha_tilde[Ns]; // non-centered sd of alpha (delta precess)
  real<lower=0> sigma;  // same noise across individuals

  // Global-level parameters for delta
  real<lower=0> rho_m;   // median of individual log-normal
  real<lower=0> rho_s;   //sd of of individual log-normal
  real<lower=0> alpha_m; // median of alpha log-normal
  real<lower=0> alpha_s; //sd of alpha log-normal
```

```
  // Global-level parameters for u
  real<lower=0.5, upper=1.8> mu;
  real<lower=1, upper=2> tau;
  // predictions
  matrix[Ns,N_pred] y_pred;
}
transformed parameters {
  real<lower=0> u[Ns];   // physical parameters
  real<lower=0> rho[Ns];   // length scale
  real<lower=0> alpha[Ns];  // marginal standard deviation
  // Non-centered parameterization of individual parameters
  for (s in 1:Ns) {
    rho[s] = exp(log(rho_m) + rho_s * rho_tilde[s]);
    alpha[s] = exp(log(alpha_m) + alpha_s * alpha_tilde[s]);
    u[s] = mu + tau * u_tilde[s];
  }
}
model {
  matrix[Ns, N_tot] x_tot;
  matrix[N_tot, N_tot] cov[Ns];
  matrix[N_tot, N_tot] L_cov[Ns];
  matrix[Ns, N_tot] z;

  x_tot=append_col(x,x_pred);
  z= append_col(y,y_pred);
  for (s in 1:Ns) {
    cov[s] = cov_exp(to_vector(x_tot[s]), alpha[s], rho[s])+diag_matrix(rep_vector(sigma^2, N_tot));
    L_cov[s] = cholesky_decompose(cov[s]);
  }

  // priors
  // Global parameters
  rho_m ~ inv_gamma(2, 0.5);
  alpha_m ~ normal(0,2);
  rho_s ~ normal(0, 0.5);
  alpha_s ~ normal(0, 0.5);

  // non-centered parameterization of individual parameters
  rho_tilde ~ normal(0, 1);
  alpha_tilde ~ normal(0, 1);
  u_tilde ~ normal(0, 1);

  // likelihood
  for (i in 1:Ns){
    z[i] ~ multi_normal_cholesky(5*exp(-u[id[i]]*x_tot[i]), L_cov[id[i]]);
  }
}
```

```r
#-----------------------------------------------------------------------------------------
# shared u and delta model
fit_yes_shared_delta = stan(file='STAN/toy/toy_shared_delta_pred.stan',
                            data=data_population,
                            chains=3,
```

```
                              iter=2*1000,
                              seed=123
)
smr_ysd= summary(fit_yes_shared_delta)
stan_trace(fit_yes_shared_delta, pars = "u")
```

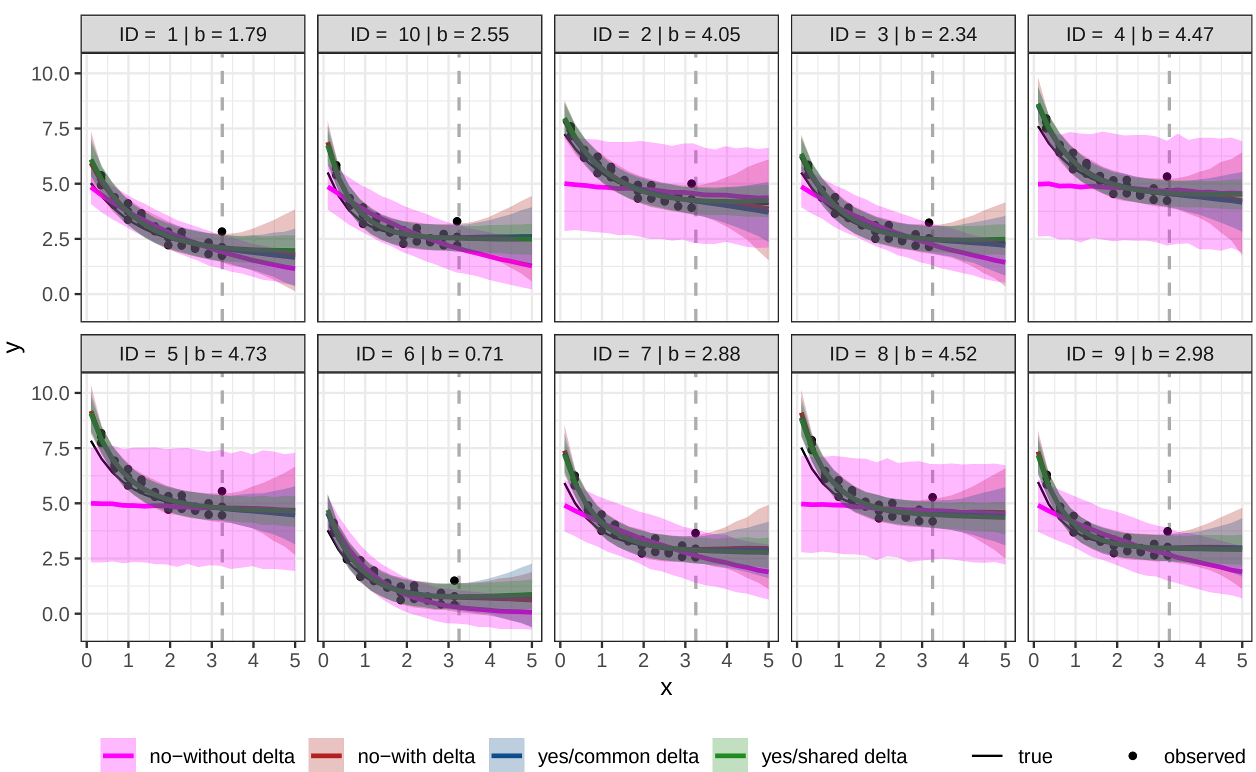



```
ex_ycd=extract(fit_yes_shared_delta)
```

Plot predictions for all methods (Figure 2 in the Appendix)

```
# Extract means and quantiles
indy=grep("y_", rownames(smr_ysd$summary))
pred_ysd=data.frame(smr_ysd$summary[indy,c("mean", "2.5%", "97.5%")])
pred_ysd$sharing = "yes/shared delta"
smr_ycd= summary(fit_yes_common_delta)
indy=grep("y_", rownames(smr_ycd$summary))
pred_ycd=data.frame(smr_ycd$summary[indy,c("mean", "2.5%", "97.5%")])
pred_ycd$sharing = "yes/common delta"

df_pred = rbind( pred_nwd, pred_ysd, pred_nnd,pred_ycd)

df_pred$x = rep(as.vector(t(x_pred)),4)
colnames(df_pred)[2:3] = c("lower", "upper")

ID = paste("ID = ", id)
```

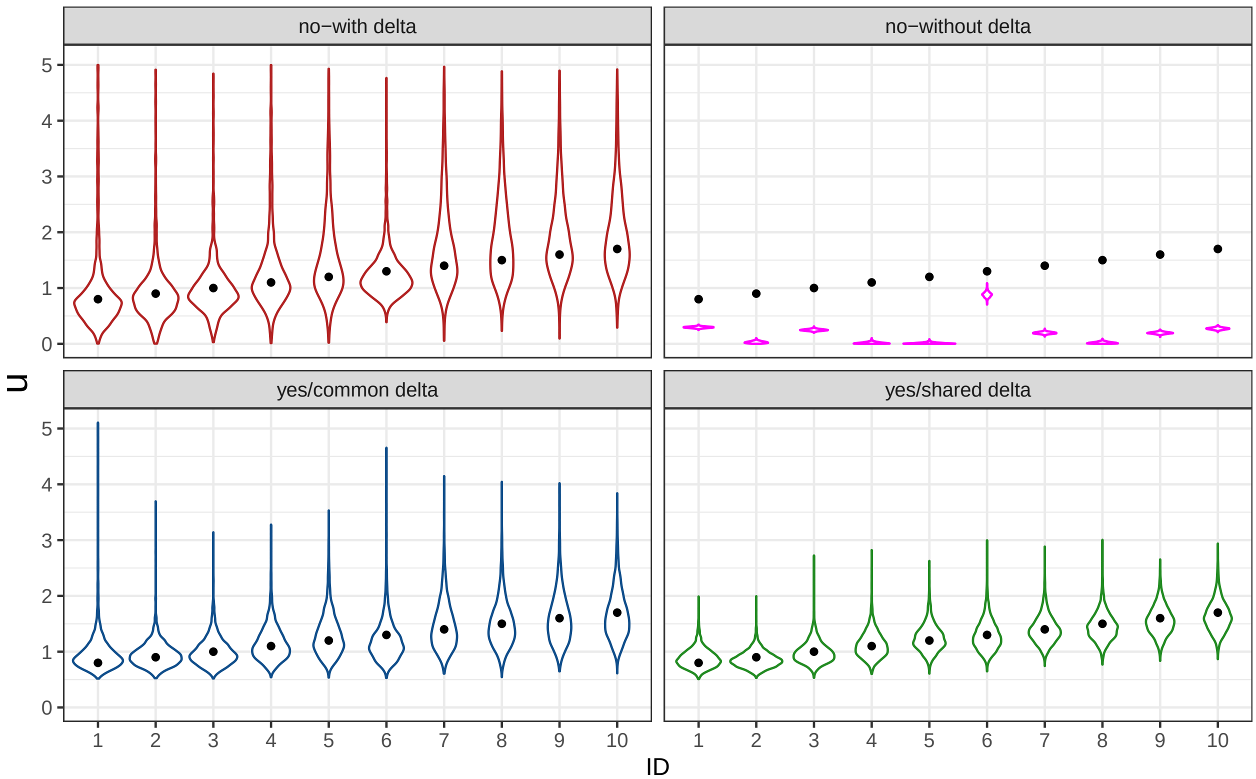
```r
bs = paste("| b =", round(offsets,2))
ID=paste(ID, bs)
df_obs = data.frame(x=as.vector(data_population$x), y=as.vector(data_population$y), id=rep(ID,N))
df_pred$id = rep(rep(ID,each=N_pred),4)

y_true=matrix(NA,nrow = nrow(y), ncol=N_pred)
for (i in 1:Ns) {
  y_true[i,]=R(u_val[i], sort(x_pred[i,]), offsets[i])
}
df_true = data.frame(x=as.vector(t(x_pred)), y=as.vector(t(y_true)), id=rep(ID, each=N_pred))
data_fig_toy_pred = list(df_true=df_true, df_obs=df_obs, df_pred=df_pred, data_population=data_populati
# saveRDS(data_fig_toy_pred, file = "Data/data_fig_toy_pred.rds")
pl_toy_pred=ggplot()+
  geom_vline(xintercept = max(X_mat), linetype=2, colour="grey68", size=0.7)+
  geom_line(data=df_true, aes(x=x, y=y, linetype="true"))+
  geom_point(data=df_obs, aes(x=x, y=y, shape="observed"))+
  geom_line(data=df_pred, aes(x=x, y=mean, color= sharing), size=1)+
  geom_ribbon(data = df_pred, aes(x=x,ymin=lower,ymax=upper, fill=sharing),alpha=0.28)+
  facet_wrap(~id, nrow = 2)+
  theme_bw()+
  theme(legend.position = "bottom", legend.title = element_blank())+
  scale_color_manual(
    breaks=c('no-without delta','no-with delta', "yes/common delta", "yes/shared delta"),
    values=c("magenta","firebrick","dodgerblue4", "forestgreen"))+
  scale_fill_manual(
    breaks=c('no-without delta','no-with delta', "yes/common delta", "yes/shared delta"),
    values=c("magenta","firebrick","dodgerblue4", "forestgreen"))
pl_toy_pred
```



```
# ggsave("Figs/toy_pred.pdf", plot = pl_toy_pred, width = 20, height = 12, units = "cm")
```

Plot individual posteriors for u (Figure 1 in the Appendix)

```
### create posterior densities plot
ex_ycd=extract(fit_yes_common_delta)
ex_ysd=extract(fit_yes_shared_delta)

nnd_u = nwd_u = c()
for(i in 1:Ns){
  nwd_u = c(nwd_u,lu[[i]])
  nnd_u = c(nnd_u,lu_no[[i]])
}

n_sam = nrow(ex_ycd$u)
post_nnd = data.frame(u = nnd_u, id = rep(id,each=n_sam), sharing='no-without delta')
post_nwd = data.frame(u = nwd_u, id = rep(id,each=n_sam), sharing='no-with delta')
post_ycd = data.frame(u = as.vector(ex_ycd$u), id = rep(id,each=n_sam), sharing='yes/common delta')
post_ysd = data.frame(u = as.vector(ex_ysd$u), id = rep(id, each = n_sam), sharing='yes/shared delta')
post_u_df = rbind( post_nnd,post_nwd, post_ycd, post_ysd)
u_true_df= data.frame(u = u_val, id = as.factor(id))
data_post_toy = list(post_u_df = post_u_df, u_true_df = u_true_df)
# saveRDS(data_post_toy, file = "Data/data_post_toy.rds")
(post_toy=ggplot() +
  geom_violin(data = post_u_df, aes(x = as.factor(id), y=u, color = sharing))+
  theme_bw()+
  theme(axis.title.y = element_text(size = rel(1.5)), legend.position = "none")+
  geom_point(data = u_true_df, aes(x = id, y = u, shape = "true"))+
  facet_wrap(sharing~.)+
  xlab("ID")+
  scale_color_manual(
    breaks=c('no-without delta', 'no-with delta', "yes/common delta", "yes/shared delta"),
    values=c("magenta","firebrick","dodgerblue4", "forestgreen")))
```




```
# ggsave("Figs/post_toy.pdf", plot = post_toy, width = 18, height = 12, units = "cm")
```

Session information

```
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] SAVE_1.0             rstan_2.21.3         ggplot2_3.3.5
[4] StanHeaders_2.21.0-7

loaded via a namespace (and not attached):
 [1] tidyselect_1.1.1   xfun_0.29          DiceKriging_1.6.0  purrr_0.3.4
 [5] lattice_0.20-45    colorspace_2.0-2   vctrs_0.3.8        generics_0.1.2
 [9] htmltools_0.5.2    stats4_4.0.3       loo_2.4.1          yaml_2.2.2
```

```
[13] utf8_1.2.2         rlang_1.0.0         pkgbuild_1.3.1     pillar_1.7.0
[17] glue_1.6.1         withr_2.4.3         DBI_1.1.2          matrixStats_0.61.0
[21] lifecycle_1.0.1    stringr_1.4.0       munsell_0.5.0      gtable_0.3.0
[25] codetools_0.2-18   coda_0.19-4         evaluate_0.14      labeling_0.4.2
[29] inline_0.3.19      knitr_1.37          callr_3.7.0        fastmap_1.1.0
[33] ps_1.6.0           parallel_4.0.3      fansi_1.0.2        Rcpp_1.0.8
[37] scales_1.1.1       RcppParallel_5.1.5  farver_2.1.0       gridExtra_2.3
[41] digest_0.6.29      stringi_1.7.6       processx_3.5.2     dplyr_1.0.7
[45] grid_4.0.3         cli_3.1.1           tools_4.0.3        magrittr_2.0.2
[49] tibble_3.1.6       crayon_1.4.2        pkgconfig_2.0.3    ellipsis_0.3.2
[53] prettyunits_1.1.1  assertthat_0.2.1    rmarkdown_2.11     rstudioapi_0.13
[57] R6_2.5.1           compiler_4.0.3
```