S7.2) Big data approximations without discrepancy

This notebook contains the code of the paper “Bayesian Calibration of Imperfect Computer Models using Physics-Informed Priors”. The models are fitted in rstan and the code is available in the folder “STAN/Approximations”.

Load packages

```
# uncomment to install
# install.packages("rstan")
# install.packages("ggplot2")
# install.packages("tidyverse")
library(rstan)
library(ggplot2)
library(tidyverse)
rstan_options(auto_write = TRUE)
options(mc.cores = 3) # allocate 3 cores (for each model we run 3 chains in parallel)
# Numerical simulator of the WK3 model
source("functions/WK2and3_sim_fn.R")
# Load flow data
d = readRDS("Data/Inflow_time.rds")
```

Function for extracting posteriors summaries

```
post_mu_CIs.fn = function(post_pred, ci = c(0.05,0.95), time = tP_pred){
  pp = rstan::extract(post_pred)
  dfP = pp$y_P[1,,]
  Psmr = data.frame(mean = colMeans(dfP),
                    lower = apply(dfP, 2, quantile, probs = ci[1]),
                    upper = apply(dfP, 2, quantile, probs = ci[2]),
                    time=time)
  dfI = pp$y_I[1,,]
  Ismr = data.frame(mean = colMeans(dfI),
                    lower = apply(dfI, 2, quantile, probs = ci[1]),
                    upper = apply(dfI, 2, quantile, probs = ci[2]),
                    time=time)
  return(list(Psmr=Psmr, Ismr=Ismr))
}
```

No discrepancy

```
Rtrue = 0.95; Ctrue = 1.1
flow = d$inflow*0.95
time = d$time
nP = 90 # number of pressure data
nI = 100# number of inflow data
nc = 1  # number of cardiac cycles
```

```r
nflow = length(flow)
# 1. simulate WK3 data (R=R_2, Z=R_1)
Psim = WK2_simulate(flow = flow, time = time, R = Rtrue, C = Ctrue) # simulate WK3 data for a given flo
P_true = Psim
# 2. choose pressure and inflow indices
indP = round(seq(1, nflow, length.out = nP)); indI = round(seq(1, nflow, length.out = nI))
yP_real = Psim[indP]; yI_real = flow[indI] # noise free fimulated pressure and flow
# 3. Add noise
# set.seed(0)
set.seed(123)
# Pnoise = rnorm(nP*nc, 0, 1) # sample pressure noise from N(0, 3^2)
# Inoise = rnorm(nI*nc, 0, 1) # sample flow noise from N(0,4^2)
Pnoise = rnorm(nP*nc, 0, 4) # sample pressure noise from N(0, 3^2)
Inoise = rnorm(nI*nc, 0, 10) # sample flow noise from N(0,4^2)
yP_real = rep(yP_real,nc)
yI_real = rep(yI_real,nc)
# 4. store individual data in the population matrices
yP = yP_real + Pnoise # add noise
yI = yI_real + Inoise # add noise
tP = time[indP] # corresponding time (synchronized for the two cycles)
tI = time[indI] # corresponding time (synchronized for the two cycles)
nP_pred = tP_pred = 40
ind_pred = seq(1, length(time) ,length.out = nP_pred)
tP_pred = tI_pred = time[ind_pred]
data_PI = list(nP=nc*nP, nI=nc*nI, tP=rep(tP,nc), tI=rep(tI,nc), yP=yP, yI=yI, mP=8, mI=10)
```

```r
WK2_VFE = stan_model('STAN/Approximations/VFE/WK2_VFE_nodelta.stan')

set.seed(123)
kp = kmeans(data.frame(x=data_PI$tP), centers = data_PI$mP)
ki = kmeans(data.frame(x=data_PI$tI), centers = data_PI$mI)
init = list("zP" = as.vector(kp$centers), "zI" = as.vector(ki$centers))

WK2_VFE_opt=optimizing(WK2_VFE, data=data_PI, hessian=FALSE, verbose=TRUE, init=init,seed=0)
```

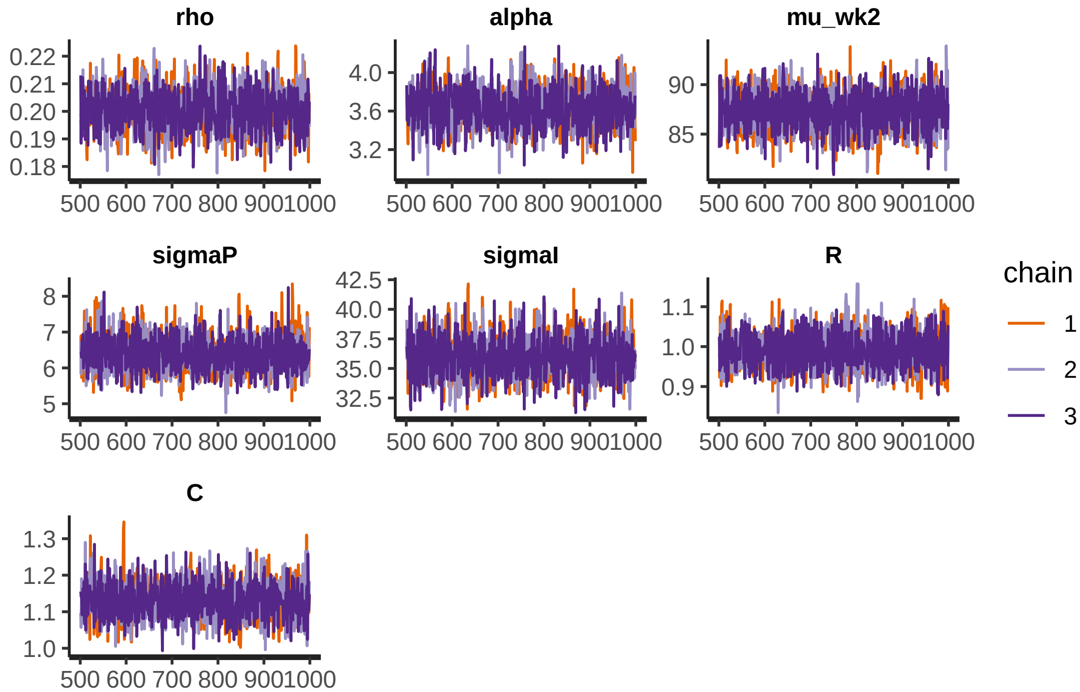
```
Chain 1: Initial log joint probability = -2957.98
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       19      -1020.19      0.222561       26.5483      0.9328      0.9328       29
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       39      -976.747    3.7037e-05     0.0372181      0.3484      0.3484       51
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       40      -976.747   0.000218586      0.019166           1           1       52
Chain 1: Optimization terminated normally:
Chain 1:   Convergence detected: relative gradient magnitude is below tolerance
```

```r
WK2_VFE_opt
```

```
$par
        rho       alpha      mu_wk2      sigmaP      sigmaI           R
 0.19974961  3.58704908 87.27913771  6.26041420 35.64791311  0.98166155
          C       zP[1]       zP[2]       zP[3]       zP[4]       zP[5]
 1.12985480  0.20277738  0.93386221  0.67454238  0.06616389  0.79727264
      zP[6]       zP[7]       zP[8]       zI[1]       zI[2]       zI[3]
 0.32549907  0.55706111  0.44303102  0.16012938  0.27486290  0.71024315
      zI[4]       zI[5]       zI[6]       zI[7]       zI[8]       zI[9]
```

```
 0.79994610  0.51120953  0.96501122  0.39502434  0.88494666  0.05004348
     zI[10]
 0.61453609

$value
[1] -976.7475

$return_code
[1] 0

$theta_tilde
           rho    alpha   mu_wk2   sigmaP   sigmaI         R        C     zP[1]
[1,] 0.1997496 3.587049 87.27914 6.260414 35.64791 0.9816615 1.129855 0.2027774
         zP[2]     zP[3]      zP[4]     zP[5]     zP[6]     zP[7]    zP[8]
[1,] 0.9338622 0.6745424 0.06616389 0.7972726 0.3254991 0.5570611 0.443031
         zI[1]     zI[2]     zI[3]     zI[4]     zI[5]     zI[6]     zI[7]
[1,] 0.1601294 0.2748629 0.7102432 0.7999461 0.5112095 0.9650112 0.3950243
         zI[8]      zI[9]    zI[10]
[1,] 0.8849467 0.05004348 0.6145361
```

```r
op_VFE = WK2_VFE_opt
zP_opt=op_VFE$par[grep("zP",names(op_VFE$par))]
zI_opt=op_VFE$par[grep("zI",names(op_VFE$par))]
data_VFE_Z = data=data_PI
data_VFE_Z$zP = zP_opt
data_VFE_Z$zI = zI_opt

fit_post_VFE=stan(file="STAN/Approximations/VFE/WK2_VFE_nodelta_fixed_Z.stan",
                  data=data_VFE_Z,
                  chains=3,
                  iter=1000,
                  seed=0
)

# stan_hist(fit_post_VFE)
stan_trace(fit_post_VFE)
```




```r
FITC = stan_model('STAN/Approximations/FITC/WK2_FITC_nodelta.stan')
init = list("zP" = seq(0.1,0.9,length.out = data_PI$mP), "zI" = seq(0.1,0.9,length.out = data_PI$mI))
# op_FITC =optimizing(FITC , data=data_PI, hessian=FALSE, verbose=TRUE, init=init, seed=112233, iter=2e
# op_FITC
op_FITC =optimizing(FITC , data=data_PI, hessian=FALSE, verbose=TRUE, init=init, seed=11111)
```

```
Chain 1: Initial log joint probability = -1339.73
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       19      -672.384      0.215037       33.0435           1           1       26
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       39      -654.778      0.344012       16.4423           1           1       55
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       59      -649.058      0.118151       35.7803           1           1       82
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       79      -648.163     0.0186175       10.5152           1           1      106
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       99      -647.727    0.00960389       43.3073           1           1      142
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      119      -647.544     0.0129817       33.7443           1           1      166
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      139      -647.366    0.00241861       20.0175           1           1      193
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      159      -647.253    0.00155685       25.3581           1           1      216
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      179       -647.21     0.0023381       17.2265      0.4972      0.4972      247
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      199      -647.189    0.00231947       7.45124       0.388           1      288
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      219      -647.181    0.00499317       89.8035           1           1      320
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:      239      -647.119   0.000811487       8.55701           1           1      352
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
```

```
Chain 1:     259      -647.102    0.0012386      48.1069          1          1     374
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     279      -647.064   0.00717283      112.369      0.3259          1     402
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     299      -647.044  0.000821833      4.77806          1          1     425
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     319       -647.04   0.00027721      37.0545      0.5786    0.05786     448
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     339      -647.019  0.000682562      28.6464          1          1     477
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     359      -647.015   0.00161268      22.8149          1          1     503
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     379      -647.001    0.0015854      128.119      0.1241          1     527
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     399      -646.986  0.000891651       29.531          1          1     552
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     419      -646.976   0.00114069      45.6557      0.1595          1     580
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     439      -646.974  0.000143099      5.55645      0.1839          1     605
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     459      -646.963  0.000702334      2.34357          1          1     638
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     479      -646.961  0.000267971      2.08652          1          1     663
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     499      -646.953   0.00238661      29.9942      0.1598          1     691
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     519      -646.917  0.000710367      81.3889      0.1484     0.1484     721
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     539      -646.903   0.00116695      8.98642          1          1     746
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     559      -646.883  0.000482885      32.5151      0.4749          1     771
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     579      -646.877   0.00326764      4.08842          1          1     793
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     599      -646.841   0.00102145      6.93452      0.4638    0.04638     820
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     619      -646.824   0.00395195        4.353          1          1     848
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     639        -646.8   0.00472934      6.90889      0.4498          1     878
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     659      -646.791    0.0018022      3.56325          1          1     904
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     679      -646.779   0.00332201      6.79038          1          1     928
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     699      -646.776   0.00319086      4.69106          1          1     951
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     719      -646.768    0.0017926      10.6979     0.07961          1     980
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     739      -646.758   0.00259716      2.84809      0.1139          1    1005
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     759      -646.727   0.00603233       6.9592       0.333      0.333    1037
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:     779      -646.695   0.00933387      114.678          1          1    1062
Chain 1:    Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
```

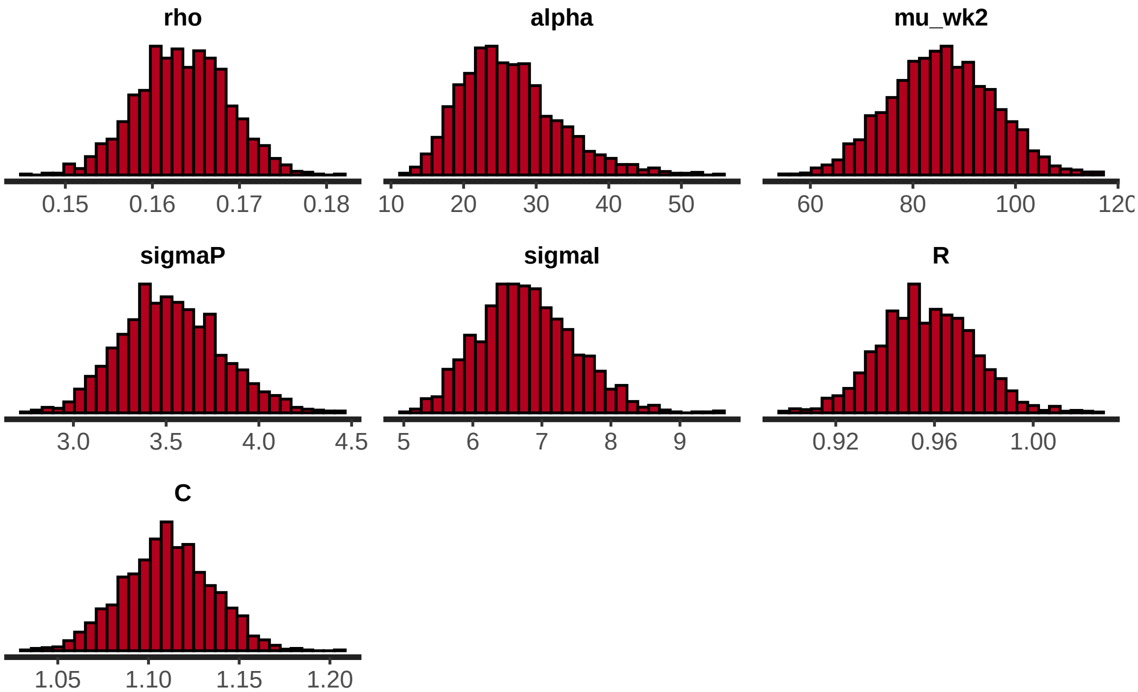
```
Chain 1:      799      -646.671     0.0186519       17.0705           1            1      1092
Chain 1:     Iter      log prob          ||dx||      ||grad||        alpha       alpha0  # evals  Notes
Chain 1:      819      -646.642     0.0180576       39.7255       0.3876            1      1115
Chain 1:     Iter      log prob          ||dx||      ||grad||        alpha       alpha0  # evals  Notes
Chain 1:      839      -646.621    0.00396611       7.71047        1.389      0.01389      1146
Chain 1:     Iter      log prob          ||dx||      ||grad||        alpha       alpha0  # evals  Notes
Chain 1:      859      -646.609    0.00189821       1.76691            1            1      1172
Chain 1:     Iter      log prob          ||dx||      ||grad||        alpha       alpha0  # evals  Notes
Chain 1:      879      -646.606   4.85335e-05       6.30003       0.2145      0.02145      1195
Chain 1:     Iter      log prob          ||dx||      ||grad||        alpha       alpha0  # evals  Notes
Chain 1:      880      -646.606   0.000499357       1.01461            1            1      1196
Chain 1: Optimization terminated normally:
Chain 1:   Convergence detected: relative gradient magnitude is below tolerance
```

```r
op_FITC
```

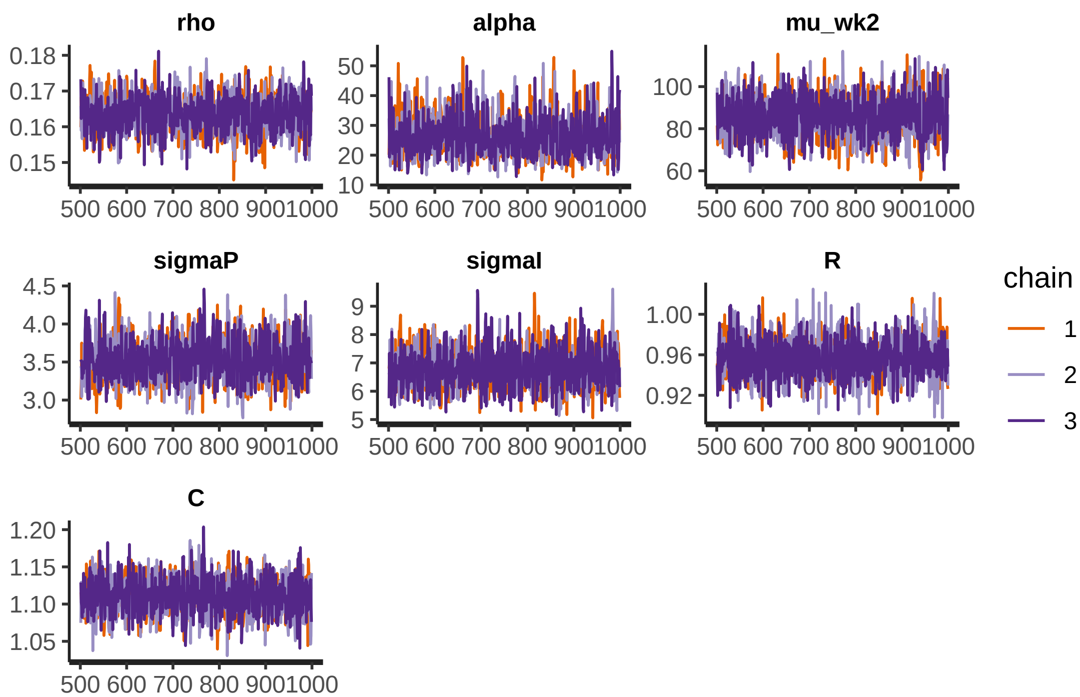
```
$par
         rho        alpha       mu_wk2       sigmaP       sigmaI            R
 0.161386210 22.827667320 85.638495122  3.454034118  6.639721597  0.955208902
           C        zP[1]        zP[2]        zP[3]        zP[4]        zP[5]
 1.109251747  0.118244344  0.208169076  0.262524648  0.266092569  0.622105364
       zP[6]        zP[7]        zP[8]        zI[1]        zI[2]        zI[3]
 0.624357280  0.909090184  0.923152986  0.006130810  0.145378361  0.006825127
       zI[4]        zI[5]        zI[6]        zI[7]        zI[8]        zI[9]
 0.282689728  0.283033361  0.263203782  0.677502222  0.977324125  0.937231790
      zI[10]
 0.988805653

$value
[1] -646.6057

$return_code
[1] 0

$theta_tilde
           rho    alpha  mu_wk2   sigmaP   sigmaI         R        C     zP[1]
[1,] 0.1613862 22.82767 85.6385 3.454034 6.639722 0.9552089 1.109252 0.1182443
         zP[2]     zP[3]     zP[4]     zP[5]     zP[6]     zP[7]    zP[8]
[1,] 0.2081691 0.2625246 0.2660926 0.6221054 0.6243573 0.9090902 0.923153
          zI[1]     zI[2]       zI[3]     zI[4]     zI[5]     zI[6]     zI[7]
[1,] 0.00613081 0.1453784 0.006825127 0.2826897 0.2830334 0.2632038 0.6775022
         zI[8]     zI[9]    zI[10]
[1,] 0.9773241 0.9372318 0.9888057
```

```r
zP_opt_FITC = op_FITC$par[grep("zP",names(op_FITC$par))]
zI_opt_FITC = op_FITC$par[grep("zI",names(op_FITC$par))]
data_FITC_Z = data_PI
data_FITC_Z$zP = zP_opt_FITC
data_FITC_Z$zI = zI_opt_FITC

fit_post_FITC=stan(file="STAN/Approximations/FITC/WK2_FITC_nodelta_fixed_Z.stan",
                  data=data_FITC_Z,
                  chains=3,
                  iter=1000,
                  seed=0
```

```
)
stan_hist(fit_post_FITC)
```




```
stan_trace(fit_post_FITC)
```




```
post_VFE = data.frame(rstan::extract(fit_post_VFE))
N_samples = nrow(post_VFE)
data_post_VFE = list(alpha=post_VFE$alpha, rho=post_VFE$rho,
                     sigmaP=post_VFE$sigmaP, sigmaI=post_VFE$sigmaI
                      , R=post_VFE$R, C=post_VFE$C, N_samples=N_samples
```

```r
)


data_pred_VFE = c(data_VFE_Z, data_post_VFE)
data_pred_VFE$nP_pred = nP_pred
data_pred_VFE$tP_pred = tP_pred
data_pred_VFE$nI_pred = nP_pred
data_pred_VFE$tI_pred = tP_pred
```

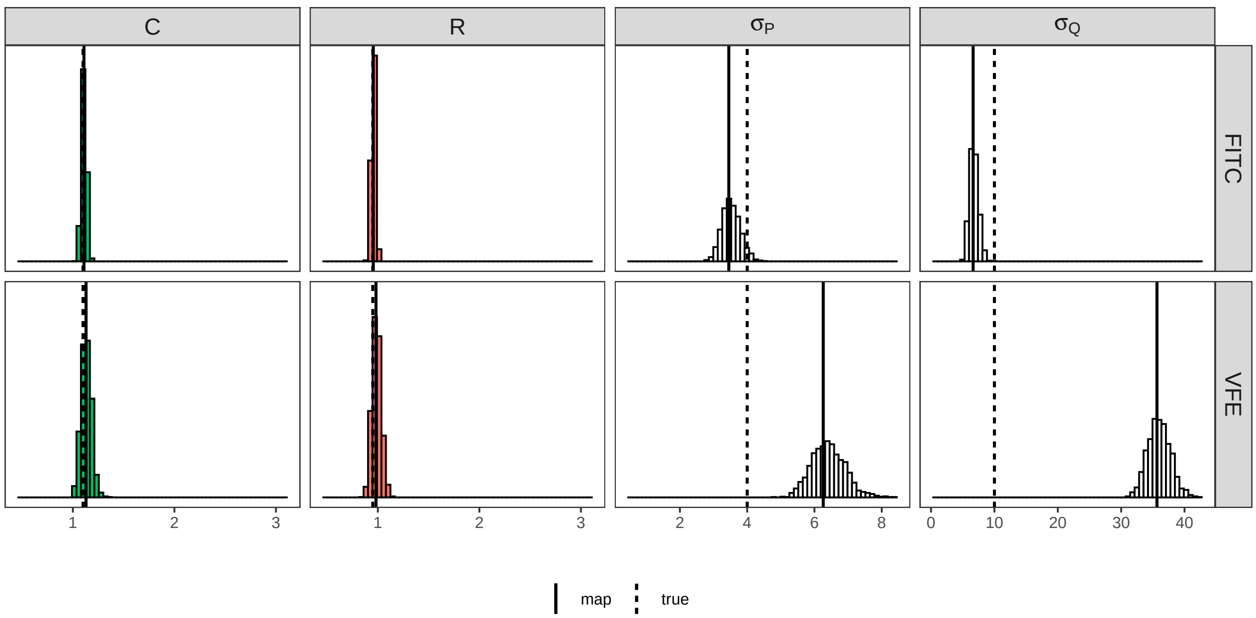
```r
pred_VFE = stan(file = 'STAN/Approximations/VFE/WK2_nodelta_VFE_predictions.stan',
                 data = data_pred_VFE,
                 chains = 1, iter = 1, seed=123,
                 algorithm = "Fixed_param")
```

```
SAMPLING FOR MODEL 'WK2_nodelta_VFE_predictions' NOW (CHAIN 1).
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                1.91131 seconds (Sampling)
Chain 1:                1.91131 seconds (Total)
Chain 1:
```

```r
post_VFE = data.frame(rstan::extract(fit_post_VFE))
post_FITC = data.frame(rstan::extract(fit_post_FITC))

pr = c("R", "C", "sigmaP", "sigmaI")
pVFE = as.vector(as.matrix(post_VFE[,pr]))
pFITC = as.vector(as.matrix(post_FITC[,pr]))
Ns = nrow(post_VFE)

df_plot_post = data.frame(post = c(pVFE, pFITC), par = rep(rep(pr, each = Ns),2), model = c(rep("VFE",le
```

```r
mod_dat = df_plot_post%>%
  mutate(par = recode(par,
    "R" = "R",
    "C" = "C",
    "sigmaP" = "sigma[P]",
    "sigmaI" = "sigma[Q]"
  ))
df_true_par = data.frame(post=c(Rtrue, Ctrue, 4, 10),par = c("R", "C", "sigma[P]", "sigma[Q]"))
set_lim = data.frame(x=c(0.5,3.1),y=c(600, 600))
df_point_est = data.frame(val=c(op_VFE$par[pr],op_FITC$par[pr]),par = rep(df_true_par$par,2), model = re
pl_post = ggplot()+
  geom_histogram(data = mod_dat, aes(x=post, fill = par), color="black",bins = 60)+
  facet_grid(model~par, scales = "free", labeller = labeller(par = label_parsed))+
  geom_point(data = set_lim, aes(x=x,y=y), color = "white",alpha=0.000001, size=0.000001)+ # set limits
  geom_vline(aes(xintercept = post, linetype = "true"), data=df_true_par, size=0.8)+
  geom_vline(aes(xintercept = val, linetype = "map"), data=df_point_est, size=0.8)+
  theme_bw()+
  theme(#legend.position = "none",
        legend.title = element_blank(),
        legend.position="bottom",
        axis.title.y=element_blank(),
        axis.text.y=element_blank(),
```

```
        axis.ticks.y=element_blank(),
        strip.text.x = element_text(size = 13),
        strip.text.y = element_text(size = 13))+
  xlab("") + ylab("")+
  scale_fill_manual(
  breaks=c("R", "C", "sigma[P]", "sigma[Q]"),
  values=c("#F8766D","#00BE67","white", "white"),guide = "none")
(pl_post=pl_post + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()))
```




```
ggsave("figures/Appr_nodelta_post.pdf", plot = pl_post, width = 20, height = 12, units = "cm")
```

FITC predictions

```
post_FITC = data.frame(rstan::extract(fit_post_FITC))
N_samples = nrow(post_FITC)
post_FITC= list(alpha=post_FITC$alpha, rho=post_FITC$rho,
                     sigmaP=post_FITC$sigmaP, sigmaI=post_FITC$sigmaI
                      , R=post_FITC$R, C=post_FITC$C, N_samples=N_samples
)


data_pred_FITC = c(data_FITC_Z, post_FITC)
data_pred_FITC$nP_pred = nP_pred
data_pred_FITC$tP_pred = tP_pred
data_pred_FITC$nI_pred = nP_pred
data_pred_FITC$tI_pred = tP_pred
```

```
pred_FITC = stan(file = 'STAN/Approximations/FITC/WK2_nodelta_FITC_predictions.stan',
                 data = data_pred_FITC,
                 chains = 1, iter = 1, seed=123,
                 algorithm = "Fixed_param")
```

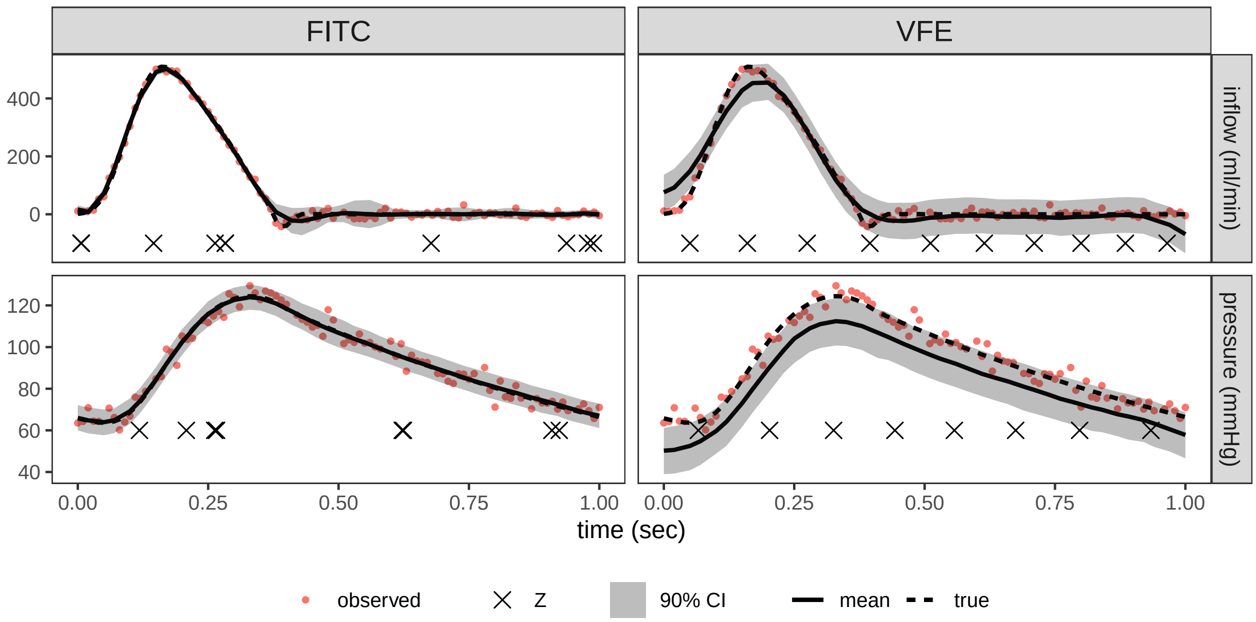
```
SAMPLING FOR MODEL 'WK2_nodelta_FITC_predictions' NOW (CHAIN 1).
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                2.46482 seconds (Sampling)
Chain 1:                2.46482 seconds (Total)
Chain 1:
```

```
pp_VFE = rbind(post_mu_CIs.fn(post_pred=pred_VFE)$Psmr, post_mu_CIs.fn(post_pred=pred_VFE)$Ismr)
pp_VFE$output = c(rep("pressure (mmHg)", nP_pred),rep("inflow (ml/min)", nP_pred))
pp_VFE$model = "VFE"
pp_FITC = rbind(post_mu_CIs.fn(post_pred=pred_FITC)$Psmr,post_mu_CIs.fn(post_pred=pred_FITC)$Ismr)
pp_FITC$output = c(rep("pressure (mmHg)", nP_pred),rep("inflow (ml/min)", nP_pred))
pp_FITC$model = "FITC"
pred_df = rbind(pp_VFE, pp_FITC)
head(pred_df)
```

```
      mean    lower    upper time          output model
1 50.23263 38.93225 61.15048 0.00 pressure (mmHg)   VFE
2 50.61417 39.26796 62.06065 0.02 pressure (mmHg)   VFE
3 52.52035 40.86959 63.54943 0.05 pressure (mmHg)   VFE
4 54.78157 43.35888 65.65682 0.07 pressure (mmHg)   VFE
5 59.60443 48.75642 70.71439 0.10 pressure (mmHg)   VFE
6 64.13454 52.56674 75.27698 0.12 pressure (mmHg)   VFE
```

```
df_zP_VFE = data.frame(z=data_VFE_Z$zP, y = rep(60, data_VFE_Z$mP), model = "VFE", output = "pressure (m
df_zI_VFE = data.frame(z=data_VFE_Z$zI, y = rep(-100, data_VFE_Z$mI), model = "VFE", output = "inflow (m
df_zP_FITC = data.frame(z=data_FITC_Z$zP, y = rep(60, data_FITC_Z$mP), model = "FITC", output = "pressu
df_zI_FITC = data.frame(z=data_FITC_Z$zI, y = rep(-100, data_FITC_Z$mI), model = "FITC", output = "infl
df_z = rbind(df_zP_VFE, df_zI_VFE, df_zP_FITC, df_zI_FITC)
P_true = data.frame(val=Psim, time=time)
P_true$output = "pressure (mmHg)"
I_true = data.frame(val=flow, time=time)
I_true$output = "inflow (ml/min)"
true_out = rbind(P_true, I_true)

obsP = data.frame(value=data_PI$yP, time = data_PI$tP, output = "pressure (mmHg)")
obsI = data.frame(value=data_PI$yI, time = data_PI$tI, output = "inflow (ml/min)")
obs = rbind(obsP,obsI)

pl_pred=ggplot()+
  geom_point(data = obs, aes(y=value, x=time, colour = "observed"), shape = 20)+
  geom_line(data = pred_df, aes(y=mean, x=time, linetype = "mean"), size=0.9)+
  geom_line(data = true_out, aes(y=val, x=time, linetype="true"), size=0.9)+
  geom_ribbon(data = pred_df,aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  facet_grid(output~model,scales = "free")+
  geom_point(data = df_z, aes(x=z, y=y, shape="Z"), size=3)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme_bw()+xlab("time (sec)")+ylab("")+
  scale_shape_manual("", values = c("Z" = 4))+
  theme(#legend.position = "none",
        legend.title = element_blank(),
        legend.position="bottom",
        strip.text.x = element_text(size = 13),
        strip.text.y = element_text(size = 10))
```

```
(pl_pred=pl_pred + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()))
```




```
ggsave("figures/Appr_nodelta_pred.pdf", plot = pl_pred, width = 16, height = 10, units = "cm")
```

Plug in noise estimates

We observe that the VFE model can severely overestimate the noise and therefore result in underfitting. A remedy to this problem is to fix the noise parameter of the functions $P(t)$ and $Q(t)$, (σ_P and σ_I). A possible solution for obtaining estimates for the noise parameters is to fit a standard GP model for each dataset (y_P, t_P) and y_Q, t_Q indepentently and obtain MLE estimates via maximizing the marginal log-likelihood.

```
nsP = 25
indP = seq(1,data_PI$nP,length.out = nsP)
data_sample_P = list(N=nsP, x = data_PI$tP[indP], y = data_PI$yP[indP])
data_sample_I = list(N=nsP, x = data_PI$tI[indP], y = data_PI$yI[indP])

GP = stan_model('STAN/Approximations/GP_full/GP.stan')

GP_MLE_P=optimizing(GP, data=data_sample_P, hessian=FALSE, verbose=TRUE,seed=0)
```

```
Chain 1: Initial log joint probability = -82.8231
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1: Exception: cholesky_decompose: Matrix m is not positive definite  (in 'modelab6b5f88c2dc_GP' a


Chain 1: Exception: cholesky_decompose: Matrix m is not positive definite  (in 'modelab6b5f88c2dc_GP' a


Chain 1:       19      -67.0609     0.0147192     0.0116441      0.6397      0.6397       44
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       22      -67.0609    8.8084e-05   0.000200878      0.2972      0.6239       48
Chain 1: Optimization terminated normally:
Chain 1:   Convergence detected: relative gradient magnitude is below tolerance
```

```r
GP_MLE_P
```

```
$par
       rho      alpha      sigma
 0.2277805 73.7318552  3.8529303

$value
[1] -67.06089

$return_code
[1] 0

$theta_tilde
           rho    alpha   sigma
[1,] 0.2277805 73.73186 3.85293
```

```r
sigma_P_MLE = GP_MLE_P$par["sigma"]

GP_MLE_I=optimizing(GP, data=data_sample_I, hessian=FALSE, verbose=TRUE,seed=0)
```

```
Chain 1: Initial log joint probability = -627.531
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       19      -108.239      0.927345    0.00263734           1           1       42
Chain 1:     Iter      log prob        ||dx||      ||grad||       alpha      alpha0  # evals  Notes
Chain 1:       33      -108.239      0.315648    0.00203595      0.4346           1       79
Chain 1: Optimization terminated normally:
Chain 1:   Convergence detected: relative gradient magnitude is below tolerance
```

```r
GP_MLE_I
```

```
$par
         rho         alpha         sigma
  0.07795791  99.99999850   8.30736756

$value
[1] -108.2386

$return_code
[1] 0

$theta_tilde
            rho alpha    sigma
[1,] 0.07795791   100 8.307368
```

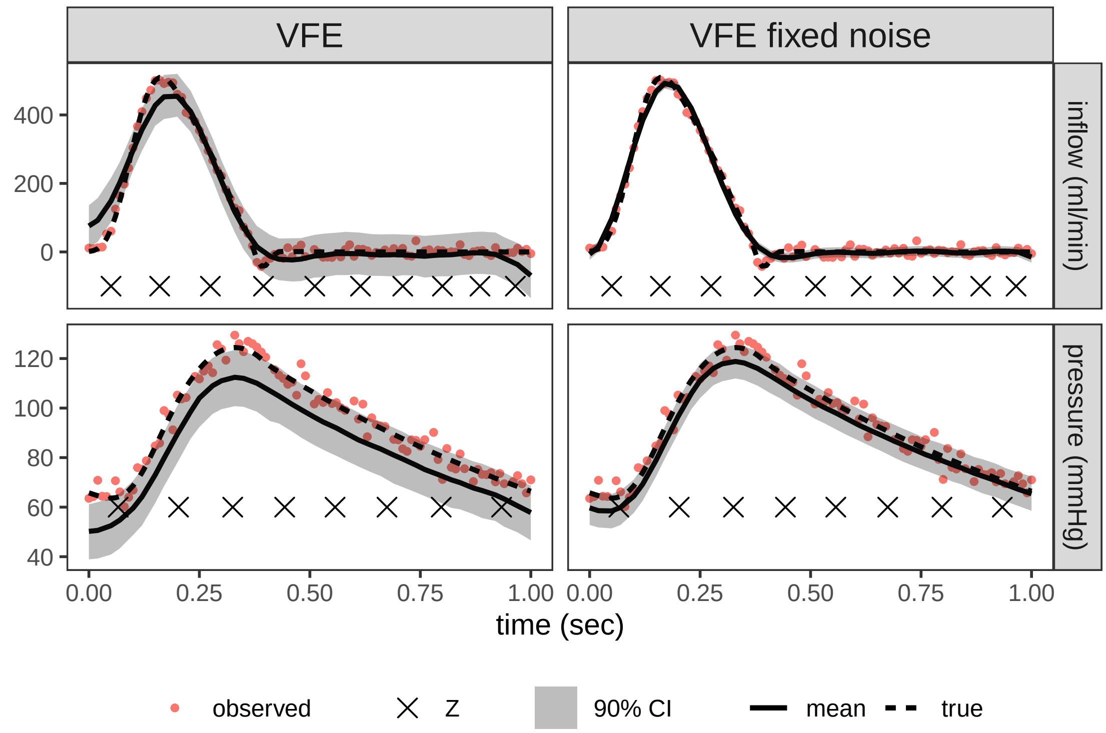
```r
sigma_I_MLE = GP_MLE_I$par["sigma"]
```

```r
data_pred_VFE_MLE = data_pred_VFE
data_pred_VFE_MLE$sigmaP = sigma_P_MLE
data_pred_VFE_MLE$sigmaI = sigma_I_MLE
pred_VFE_MLE = stan(file = 'STAN/Approximations/VFE/MLE_sigma/WK2_nodelta_VFE_predictions.stan',
                 data = data_pred_VFE_MLE ,
                 chains = 1, iter = 1, seed=123,
                 algorithm = "Fixed_param")
```

```
SAMPLING FOR MODEL 'WK2_nodelta_VFE_predictions' NOW (CHAIN 1).
```

```
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                1.89004 seconds (Sampling)
Chain 1:                1.89004 seconds (Total)
Chain 1:
```

```
pp_VFE_MLE = rbind(post_mu_CIs.fn(post_pred=pred_VFE_MLE)$Psmr, post_mu_CIs.fn(post_pred=pred_VFE_MLE)$I
pp_VFE_MLE$output = c(rep("pressure (mmHg)", nP_pred),rep("inflow (ml/min)", nP_pred))
pp_VFE_MLE$model = "VFE fixed noise"
pp_VFE = rbind(post_mu_CIs.fn(post_pred=pred_VFE)$Psmr, post_mu_CIs.fn(post_pred=pred_VFE)$Ismr)
pp_VFE$output = c(rep("pressure (mmHg)", nP_pred),rep("inflow (ml/min)", nP_pred))
pp_VFE$model = "VFE"
pred_df = rbind(pp_VFE_MLE,pp_VFE)
df_zP_VFE = data.frame(z=data_VFE_Z$zP, y = rep(60, data_VFE_Z$mP), model = "VFE", output = "pressure (m
df_zI_VFE = data.frame(z=data_VFE_Z$zI, y = rep(-100, data_VFE_Z$mI), model = "VFE", output = "inflow (m
df_zP_VFE_MLE = data.frame(z=data_pred_VFE_MLE$zP, y = rep(60, data_pred_VFE_MLE$mP), model = "VFE fixed
df_zI_VFE_MLE = data.frame(z=data_pred_VFE_MLE$zI, y = rep(-100, data_pred_VFE_MLE$mI), model = "VFE fi
df_z = rbind(df_zP_VFE, df_zI_VFE,df_zP_VFE_MLE,df_zI_VFE_MLE)
P_true = data.frame(val=Psim, time=time)
P_true$output = "pressure (mmHg)"
I_true = data.frame(val=flow, time=time)
I_true$output = "inflow (ml/min)"
true_out = rbind(P_true, I_true)

obsP = data.frame(value=data_PI$yP, time = data_PI$tP, output = "pressure (mmHg)")
obsI = data.frame(value=data_PI$yI, time = data_PI$tI, output = "inflow (ml/min)")
obs = rbind(obsP,obsI)

pl_pred=ggplot()+
  geom_point(data = obs, aes(y=value, x=time, colour = "observed"), shape = 20)+
  geom_line(data = pred_df, aes(y=mean, x=time, linetype = "mean"), size=0.9)+
  geom_line(data = true_out, aes(y=val, x=time, linetype="true"), size=0.9)+
  geom_ribbon(data = pred_df,aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  facet_grid(output~model,scales = "free")+
  geom_point(data = df_z, aes(x=z, y=y, shape="Z"), size=3)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme_bw()+xlab("time (sec)")+ylab("")+
  scale_shape_manual("", values = c("Z" = 4))+
  theme(#legend.position = "none",
        legend.title = element_blank(),
        legend.position="bottom",
        strip.text.x = element_text(size = 13),
        strip.text.y = element_text(size = 10))
(pl_pred=pl_pred + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()))
```




```r
ggsave("figures/Appr_nodelta_pred_noise.pdf", plot = pl_pred, width = 16, height = 10, units = "cm")
```

```r
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] forcats_0.5.1       stringr_1.4.0       dplyr_1.0.7
 [4] purrr_0.3.4         readr_2.1.2         tidyr_1.2.0
 [7] tibble_3.1.6        tidyverse_1.3.1     rstan_2.21.3
[10] ggplot2_3.3.5       StanHeaders_2.21.0-7

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.8         lubridate_1.8.0    prettyunits_1.1.1  ps_1.6.0
 [5] assertthat_0.2.1   digest_0.6.29      utf8_1.2.2         cellranger_1.1.0
 [9] R6_2.5.1           backports_1.4.1    reprex_2.0.1       stats4_4.0.3
[13] evaluate_0.14      httr_1.4.2         pillar_1.7.0       rlang_1.0.0
[17] readxl_1.3.1       rstudioapi_0.13    callr_3.7.0        rmarkdown_2.11
[21] labeling_0.4.2     loo_2.4.1          munsell_0.5.0      broom_0.7.12
```

```
[25] compiler_4.0.3     modelr_0.1.8       xfun_0.29          pkgconfig_2.0.3
[29] pkgbuild_1.3.1     htmltools_0.5.2    tidyselect_1.1.1   gridExtra_2.3
[33] codetools_0.2-18   matrixStats_0.61.0 fansi_1.0.2        crayon_1.4.2
[37] tzdb_0.2.0         dbplyr_2.1.1       withr_2.4.3        grid_4.0.3
[41] jsonlite_1.7.3     gtable_0.3.0       lifecycle_1.0.1    DBI_1.1.2
[45] magrittr_2.0.2     scales_1.1.1       RcppParallel_5.1.5 cli_3.1.1
[49] stringi_1.7.6      farver_2.1.0       fs_1.5.2           xml2_1.3.3
[53] ellipsis_0.3.2     generics_0.1.2     vctrs_0.3.8        tools_4.0.3
[57] glue_1.6.1         hms_1.1.1          processx_3.5.2     parallel_4.0.3
[61] fastmap_1.1.0      yaml_2.2.2         inline_0.3.19      colorspace_2.0-2
[65] rvest_1.0.2        knitr_1.37         haven_2.4.3
```