S3.3) WK case study: Model 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/Windkessel".

Load libraries

Load libraries and functions to simulate data from the models

```
# uncomment to install
# install.packages("rstan")
# install.packages("ggplot2")
# install.packages("ggpubr")
# install.packages("reshape2")
library(rstan)
library(ggplot2)
library(ggpubr)
library(reshape2)
theme_set(theme_classic()) # set ggplot theme
rstan_options(auto_write = TRUE)
options(mc.cores = 3) # allocate 3 cores (for each model we run 3 chains in parallel)
# numerical simulators of the WK2 and WK3 models
source("functions/WK2and3_sim_fn.R")
# functions to create observed data (noisy WK2 or WK3 data) and to extract the stan output
source("functions/WK_exp_fn.R")
# load inflow and time data
d = readRDS("Data/Inflow_time.rds")
```

Section 3.3: Model discrepancy

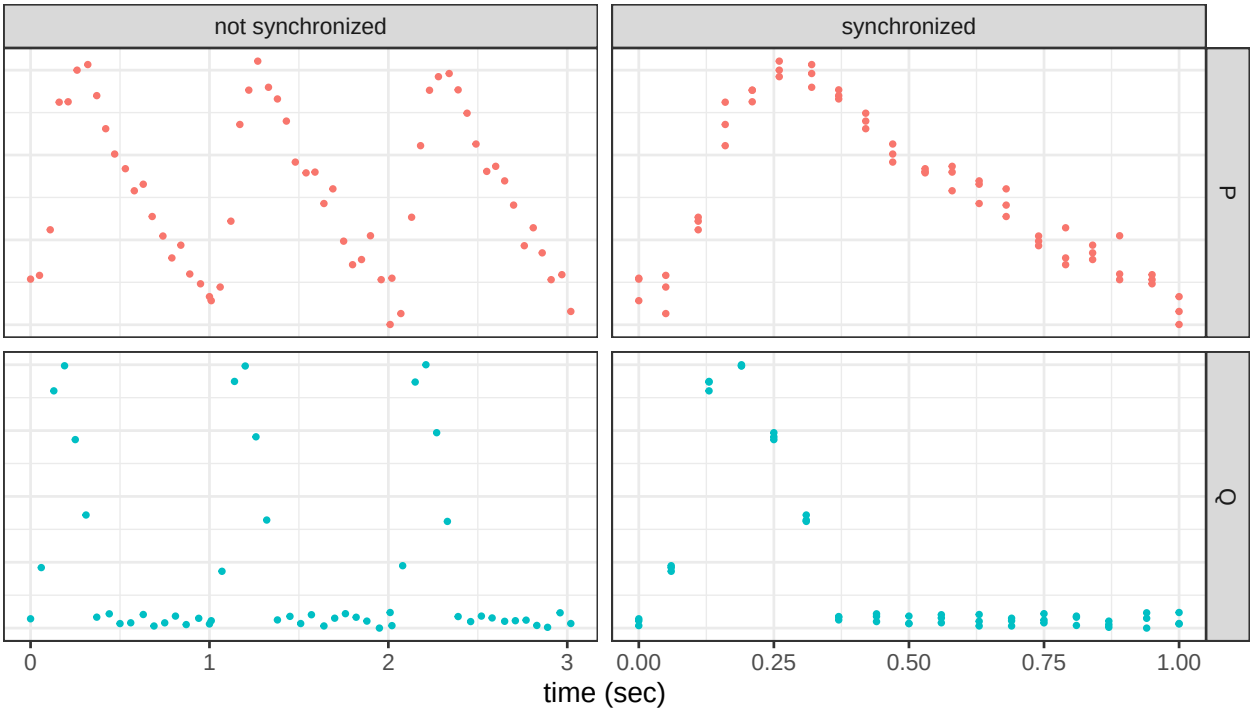
In this Section we simulate noisy data from the Windkessel three parameters model (WK3) and we fit the PI priors to the observed data, where the WK3 is the following differential equation

$$\frac{dP(t)}{dt} + \frac{P(t)}{R_2 C} = \frac{Q(t)}{C}\left(1 + \frac{R_1}{R_2}\right) + R_1 \frac{dQ(t)}{dt}. \tag{WK3}$$

More specifically, for a given inflow (see below) we simulate three blood pressure cycles and we add to both $P(t)$ and $Q(t)$, i.i.d. Gaussian noise, $\varepsilon_P \sim N(0, 4^2)$ and $\varepsilon_Q \sim N(0, 10^2)$. We also create replicates by synchronizing the three cycles in one (see Figure below).

```
# Set parameter values
# (Ztrue is the R_1 parameter and Rtrue is the R_2 parameter)
Rtrue= 1; Ctrue = 1.1; Ztrue = 0.05
flow = d$inflow*0.95
time = d$time
# For the observed flow and the corresponding observed time
# simulate pressure data from the WK2 model
```

```
# for the given Rtrue and Ctrue values
# return 3 cycles (nc=3) of pressure and flow
# with 20 pressure points at each cycle (nP=20)
# and 17 pressure points at each cycle (nI=17)
# for both pressure and inflow add N(0,4^2) and N(0, 10^2) i.i.d. noise
# Create also predictions on a grid of 50 time points (n_pred=50)
ddd = create_data(flow, time, Rtrue, Ctrue, Ztrue=Ztrue,
                  nc=3, nP=20, nI=17, n_pred=50, Pnoise=4, Inoise=10, seed = 0)
```



WK2 model (not accounting for model discrepancy)

First, we fit the model that does not account for model discrepancy. This is the model formulation in Section 2.1 and for the WK2 model we have that

$$\begin{aligned} y_P &= P^{\text{WK2}}(t_P) + \varepsilon_P \\ y_Q &= Q^{\text{WK2}}(t_Q) + \varepsilon_Q, \end{aligned} \tag{1}$$

where $P^{\text{WK2}}(t_P) \sim GP(\mu_P, K(t_P, t'_P)), \varepsilon_P \sim N(0, \sigma_P^2)$ and $\varepsilon_Q \sim N(0, \sigma_Q^2)$. This results in the following multi-output GP prior

$$p(\mathbf{y} \mid \boldsymbol{\theta}, \boldsymbol{\phi}, \sigma_P, \sigma_Q) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{2}$$

where $\mathbf{y} = \begin{bmatrix} \mathbf{y_P} \\ \mathbf{y_Q} \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_P \\ R^{-1}\boldsymbol{\mu}_P \end{bmatrix}$ and

$$\mathbf{K} = \begin{bmatrix} K_{PP}(t_P, t_P \mid \boldsymbol{\theta}) + \sigma_P^2 I_P & K_{PQ}(t_P, t_Q \mid \boldsymbol{\theta}, \boldsymbol{\phi}) \\ K_{QP}(t_Q, t_P \mid \boldsymbol{\theta}, \boldsymbol{\phi}) & K_{QQ}(t_Q, t_Q \mid \boldsymbol{\theta}, \boldsymbol{\phi}) + \sigma_Q^2 I_Q \end{bmatrix}.$$

WK2 model accounting for model discrepancy

Then, we fit the model that accounts for model discrepancy. This is the model formulation in Section 2.2 and for the WK2 model we have that

$$\begin{aligned} y_P &= P^{\text{WK2}}(t_P) + \delta(t_P) + \varepsilon_P \\ y_Q &= Q^{\text{WK2}}(t_Q) + \varepsilon_Q, \end{aligned} \tag{3}$$

where $P^{\text{WK2}}(t_P) \sim GP(\mu_P, K(t_P, t_P'))$, $\varepsilon_P \sim N(0, \sigma_P^2)$ and $\varepsilon_Q \sim N(0, \sigma_Q^2)$. This results in the following multi-output GP prior

$$p(\mathbf{y} \mid \boldsymbol{\theta}, \boldsymbol{\phi}, \sigma_P, \sigma_Q) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{4}$$

where $\mathbf{y} = \begin{bmatrix} \mathbf{y_P} \\ \mathbf{y_Q} \end{bmatrix}$, $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_P \\ R^{-1}\boldsymbol{\mu}_P \end{bmatrix}$ and

$$\mathbf{K} = \begin{bmatrix} K_{PP}(t_P, t_P \mid \boldsymbol{\theta}) + K_\delta(t_P, t_P \mid \boldsymbol{\theta}_\delta) + \sigma_P^2 I_P & K_{PQ}(t_P, t_Q \mid \boldsymbol{\theta}, \boldsymbol{\phi}) \\ K_{QP}(t_Q, t_P \mid \boldsymbol{\theta}, \boldsymbol{\phi}) & K_{QQ}(t_Q, t_Q \mid \boldsymbol{\theta}, \boldsymbol{\phi}) + \sigma_Q^2 I_Q \end{bmatrix}.$$

Squared exponential (SE) kernel model

We fit the two models using $K_{PP}(t, t') = \sigma^2 exp\left(-0.5\left(\frac{t-t'}{l}\right)^2\right)$.

WK2 model

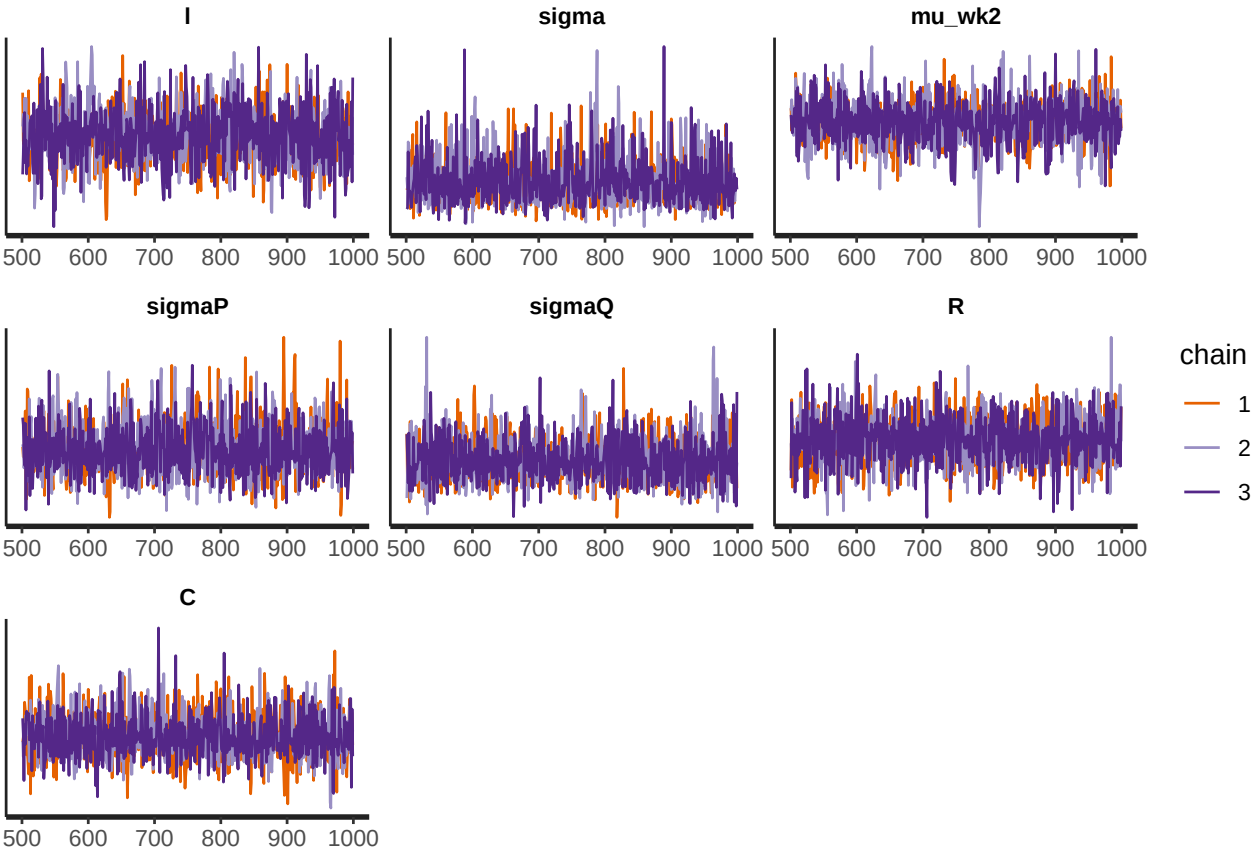
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/SE/WK2__SE_PI_prior.stan'))
```

```
# WK2 (no discrepancy)
fit_sq_exp = stan(file='STAN/Windkessel/SE/WK2__SE_PI_prior.stan',
                  data=ddd$data_noisy_pred,
                  chains=3,
                  iter=1000,
                  seed=123
)
```

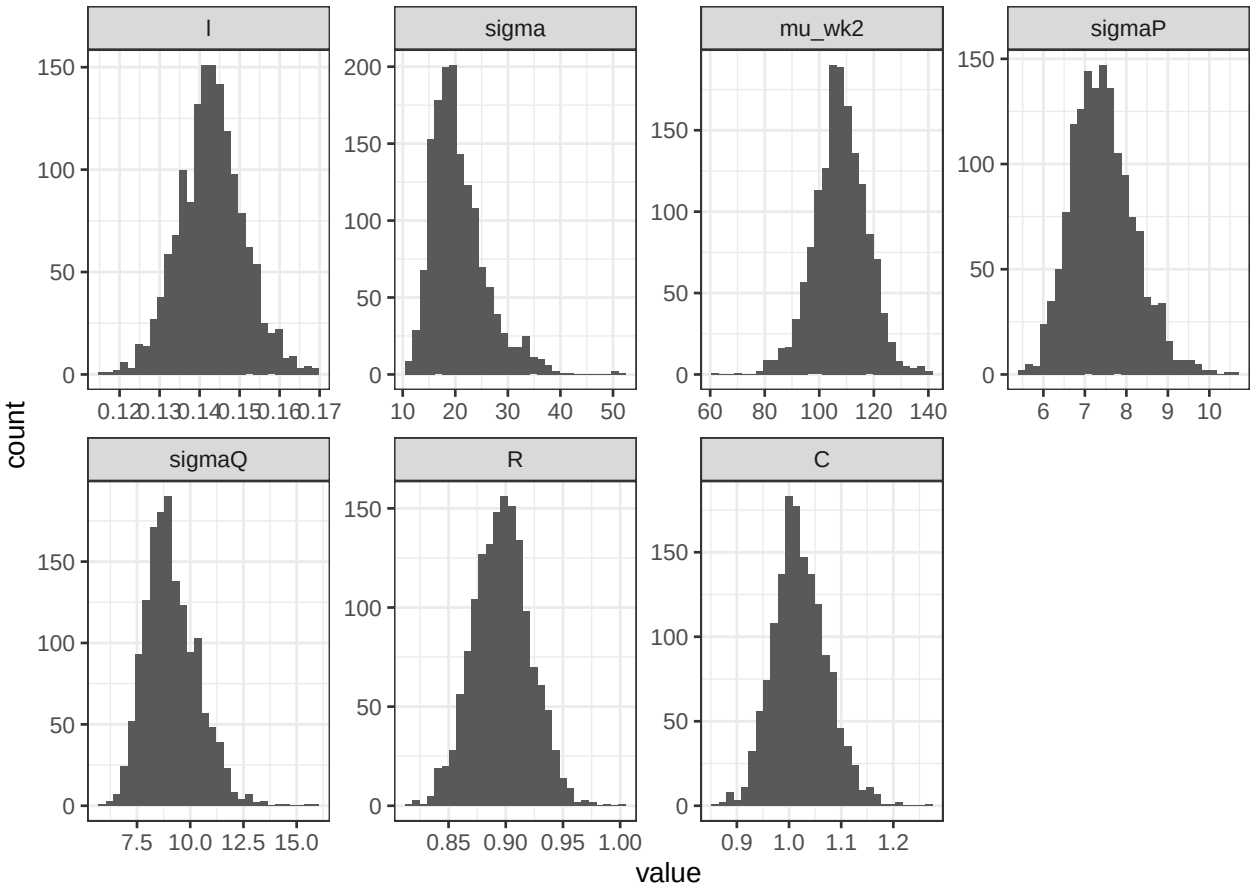
Trace plots

```
stan_trace(fit_sq_exp, pars=names(fit_sq_exp)[1:7])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Transforming the posterior in the original scale

```
y = ddd$y # the original observed data
pp_se= transform_post(y, fit_sq_exp)
pl_df=pp_se[,names(fit_sq_exp)[1:7]]
pl_df$sample=1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



WK2 + $\delta(t)$ model

For the $\delta(t)$ we use also a squared exponential kernel, $K_\delta(t,t') = \sigma_\delta^2 exp\left(-0.5\left(\frac{t-t'}{l_\delta}\right)^2\right)$.

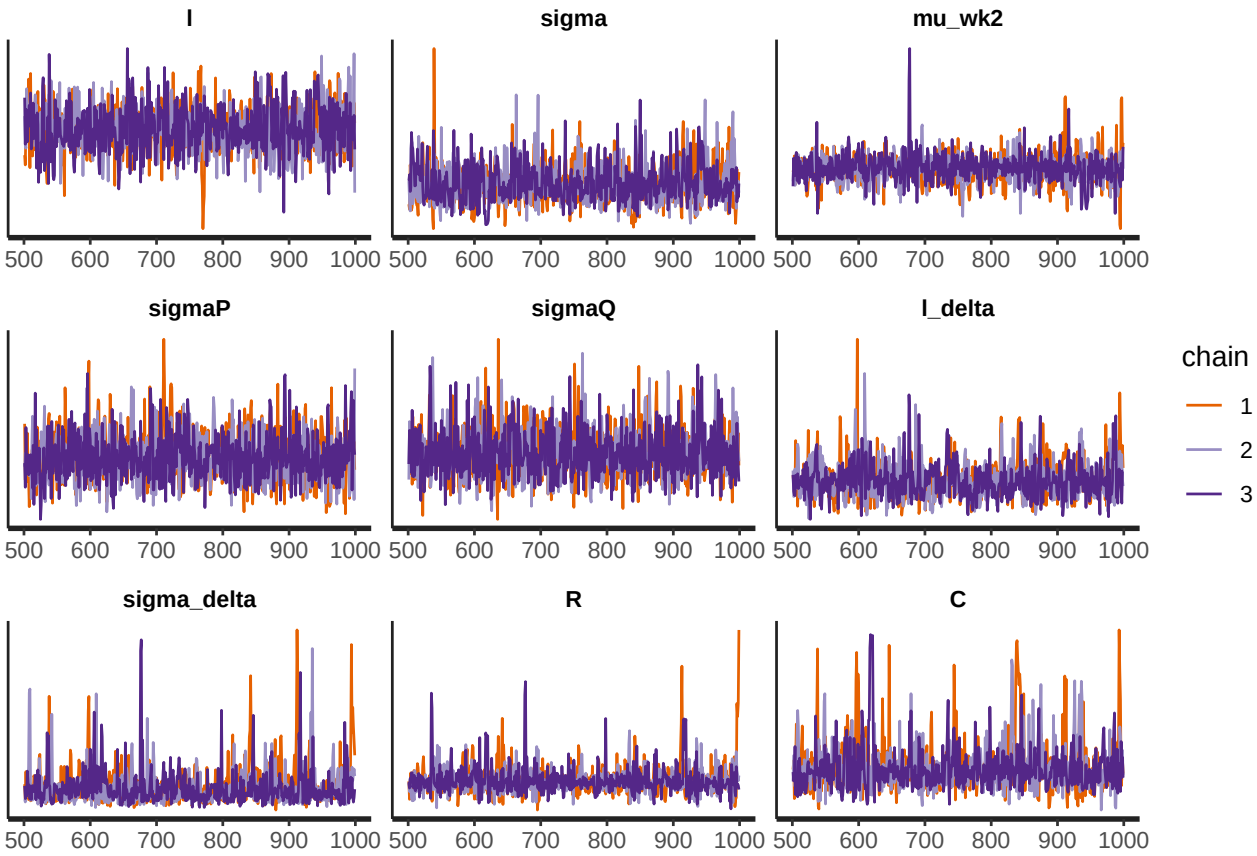
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/SE/WK2_Pbias__SE_PI_prior.stan'))
```

```
# WK2+delta(t)
fit_sq_exp_Bias = stan(file  = 'STAN/Windkessel/SE/WK2_Pbias__SE_PI_prior.stan',
                       data = ddd$data_noisy_pred,
                       chains = 3,
                       iter = 1000,
                       seed = 123
)
```

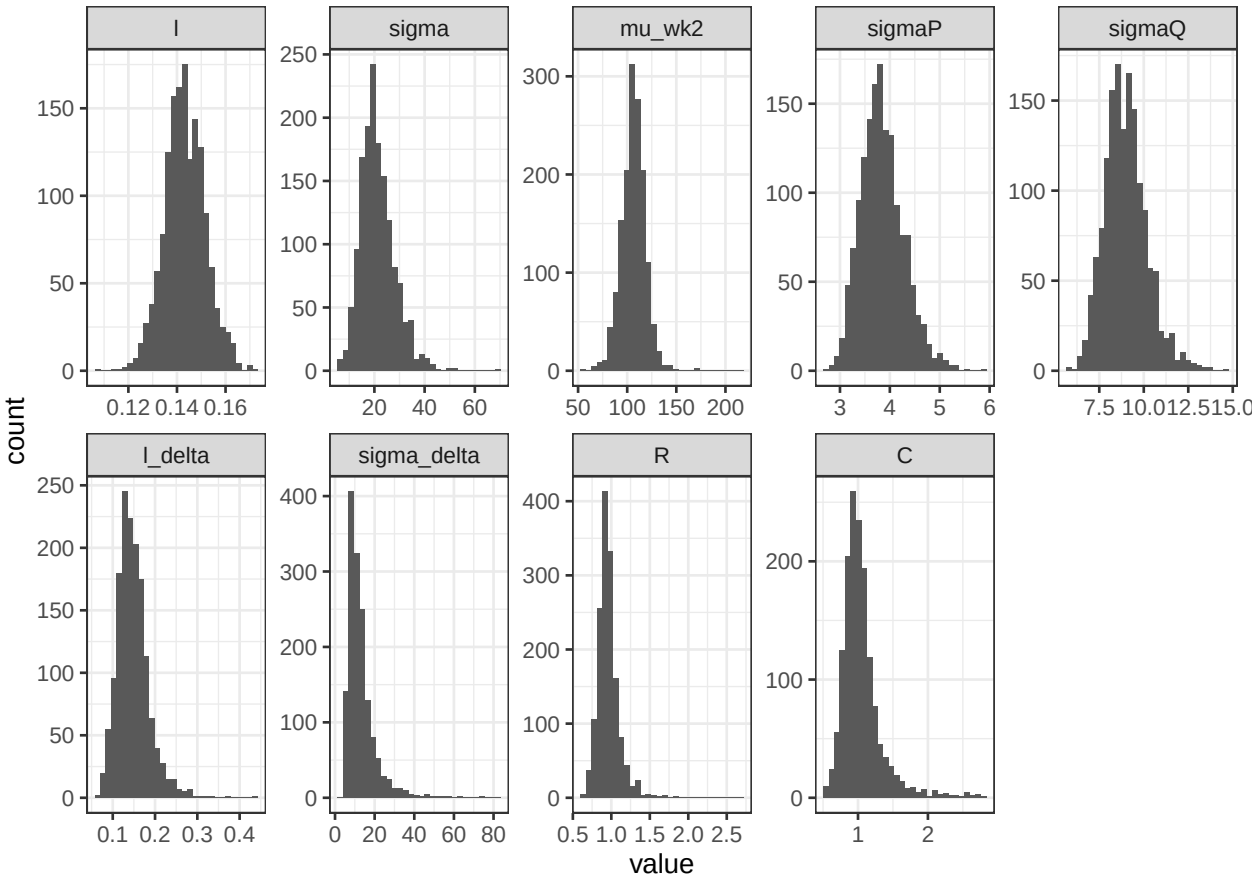
Trace plots

```
stan_trace(fit_sq_exp_Bias, pars=names(fit_sq_exp_Bias)[1:9])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Transforming the posterior in the original scale

```
y = ddd$y # the original observed data
pp_se_Bias = transform_post(y=y, fit=fit_sq_exp_Bias)
pl_df = pp_se_Bias[,names(fit_sq_exp_Bias)[1:9]]
pl_df$sample = 1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



Rational quadratic (RQ) kernel model

We fit the two models using $K_{\mathrm{PP}}(t,t') = \sigma^2 \left(1 + \frac{(t-t')^2}{2\alpha\ell^2}\right)^{-\alpha}$

WK2 model

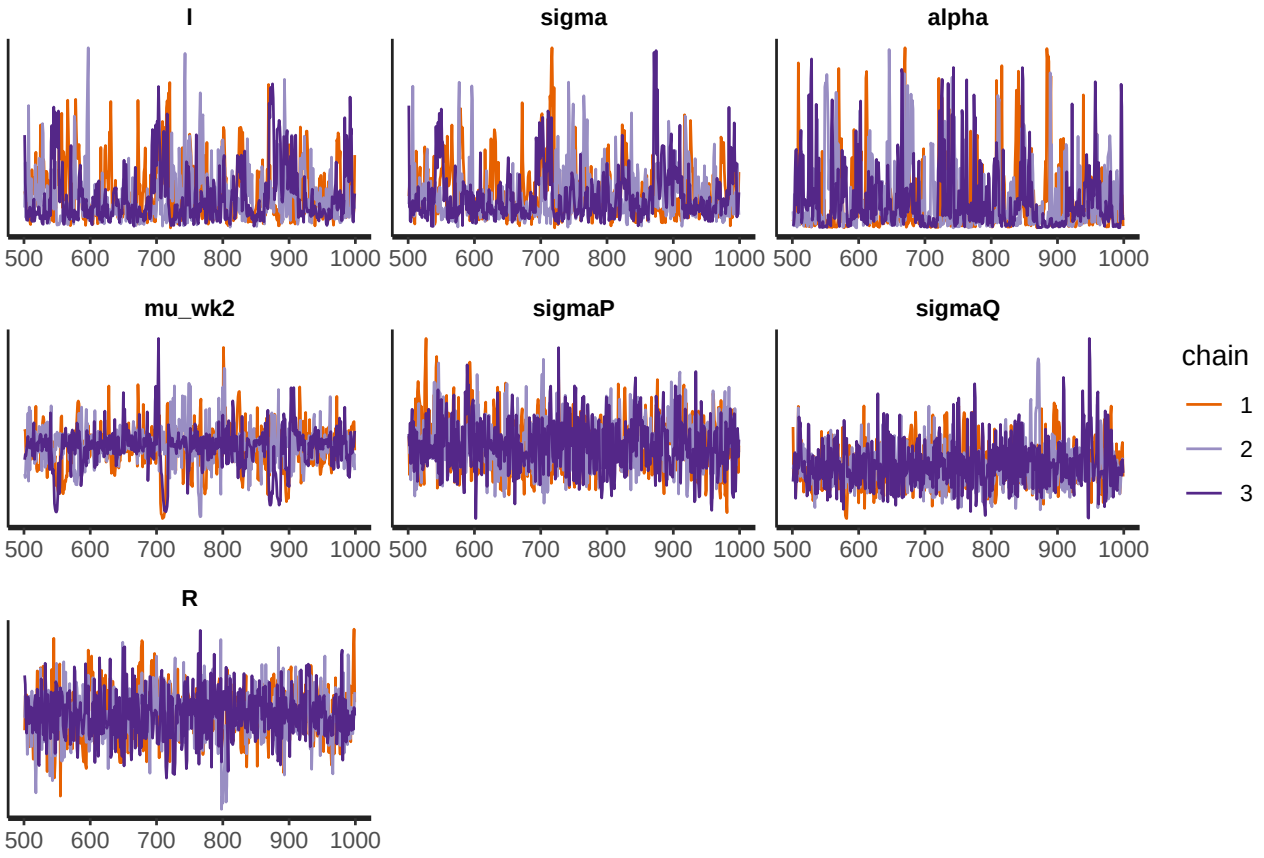
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/RQ/WK2__RQ_PI_prior.stan'))
```

```
# WK2 (no discrepancy)
fit_RQ = stan(file  = 'STAN/Windkessel/RQ/WK2__RQ_PI_prior.stan',
              data = ddd$data_noisy_pred,
              chains = 3,
              iter = 1000,
              seed = 0
)
```

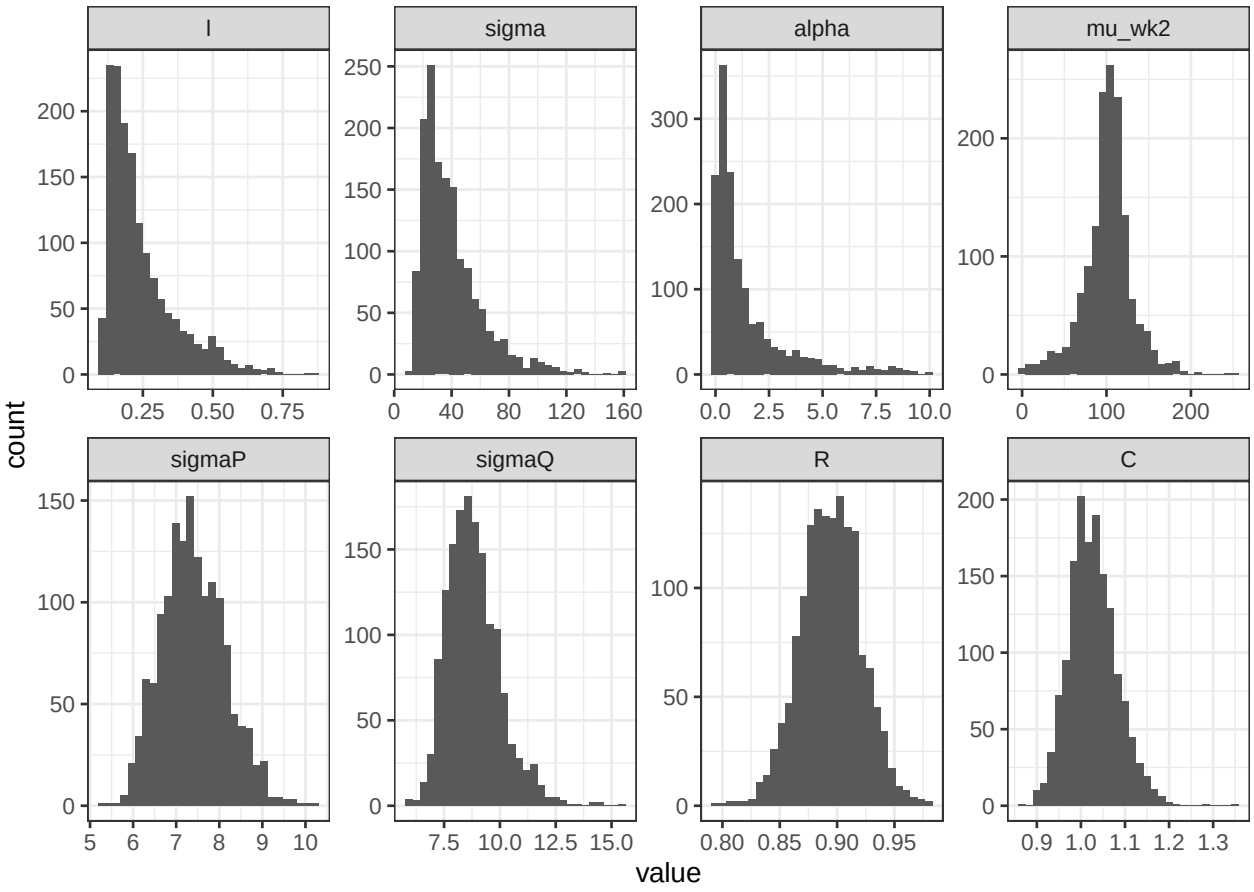
Trace plots

```
stan_trace(fit_RQ, pars=names(fit_RQ)[1:7])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Transforming the posterior in the original scale

```
y = ddd$y # the original observed data
pp_rq =transform_post(y=y, fit=fit_RQ)
pl_df=pp_rq[,names(fit_RQ)[1:8]]
pl_df$sample=1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



WK2 + $\delta(t)$ model

For the $\delta(t)$ we use also a squared exponential kernel, $K_\delta(t,t') = \sigma_\delta^2 exp\left(-0.5\left(\frac{t-t'}{l_\delta}\right)^2\right)$.

WK2 model

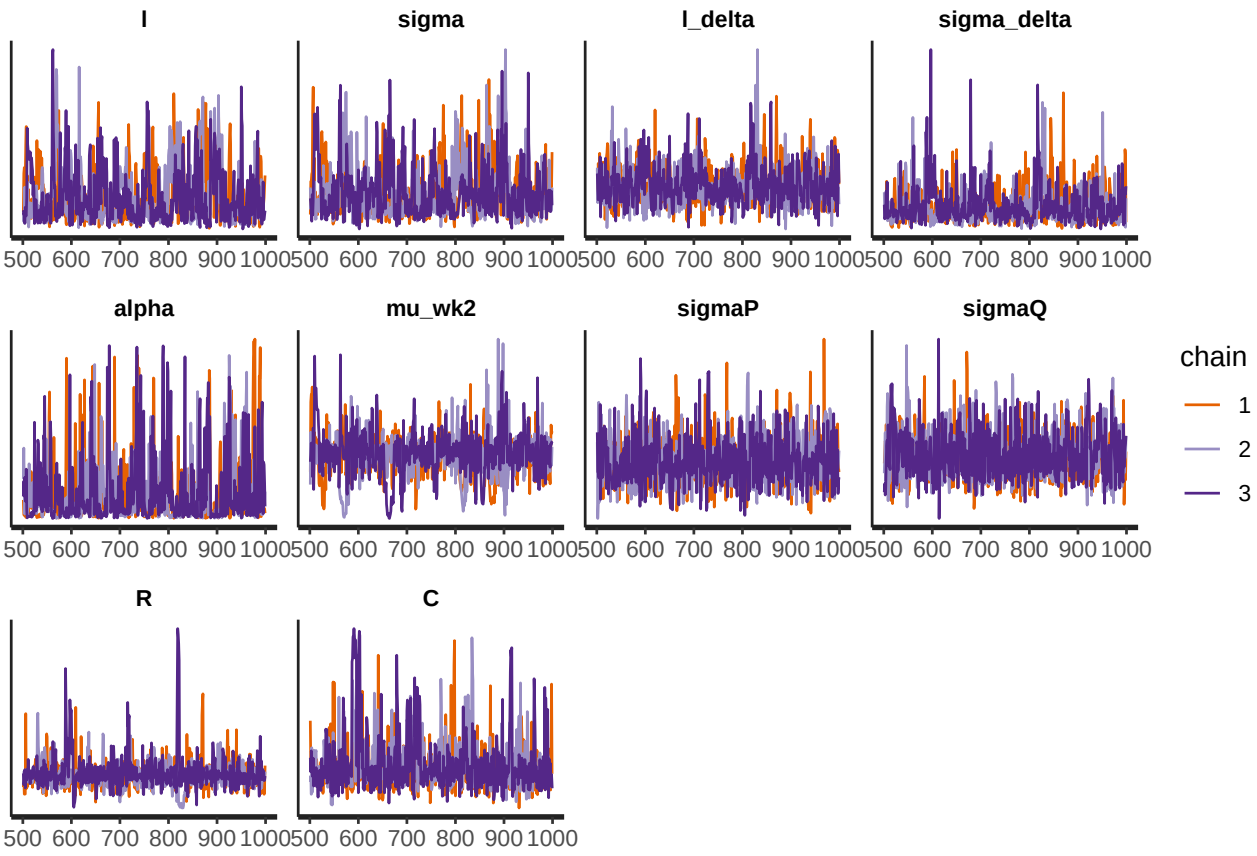
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/RQ/WK2_Pbias__RQ_PI_prior.stan'))
```

```
# WK2+delta(t)
fit_RQ_Bias = stan(file  = 'STAN/Windkessel/RQ/WK2_Pbias__RQ_PI_prior.stan',
                   data = ddd$data_noisy_pred,
                   chains = 3,
                   iter = 1000,
                   seed = 123
)
```

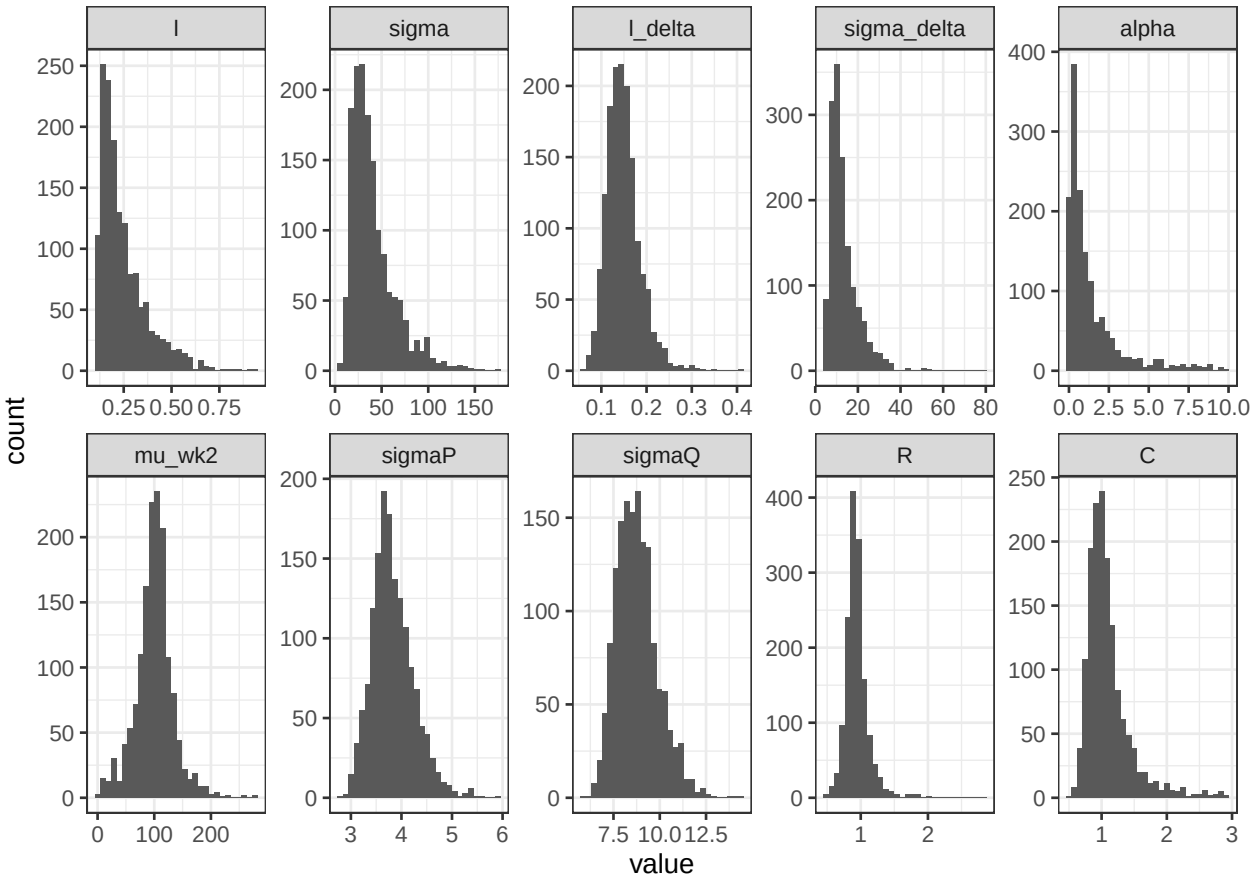
Trace plots

```
stan_trace(fit_RQ_Bias, pars=names(fit_RQ_Bias)[1:10])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Transforming the posterior in the original scale

```
y = ddd$y # the original observed data
pp_rq_Bias = transform_post(y=y, fit=fit_RQ_Bias)
pl_df = pp_rq_Bias[,names(fit_RQ_Bias)[1:10]]
pl_df$sample = 1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



Periodic (Per) kernel model

We fit the two models using $K_{\text{Per}}(t,t') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi(t-t')p)}{\ell^2}\right)$. Now we use the "raw" data (not synchronized in one cycle)

WK2 model

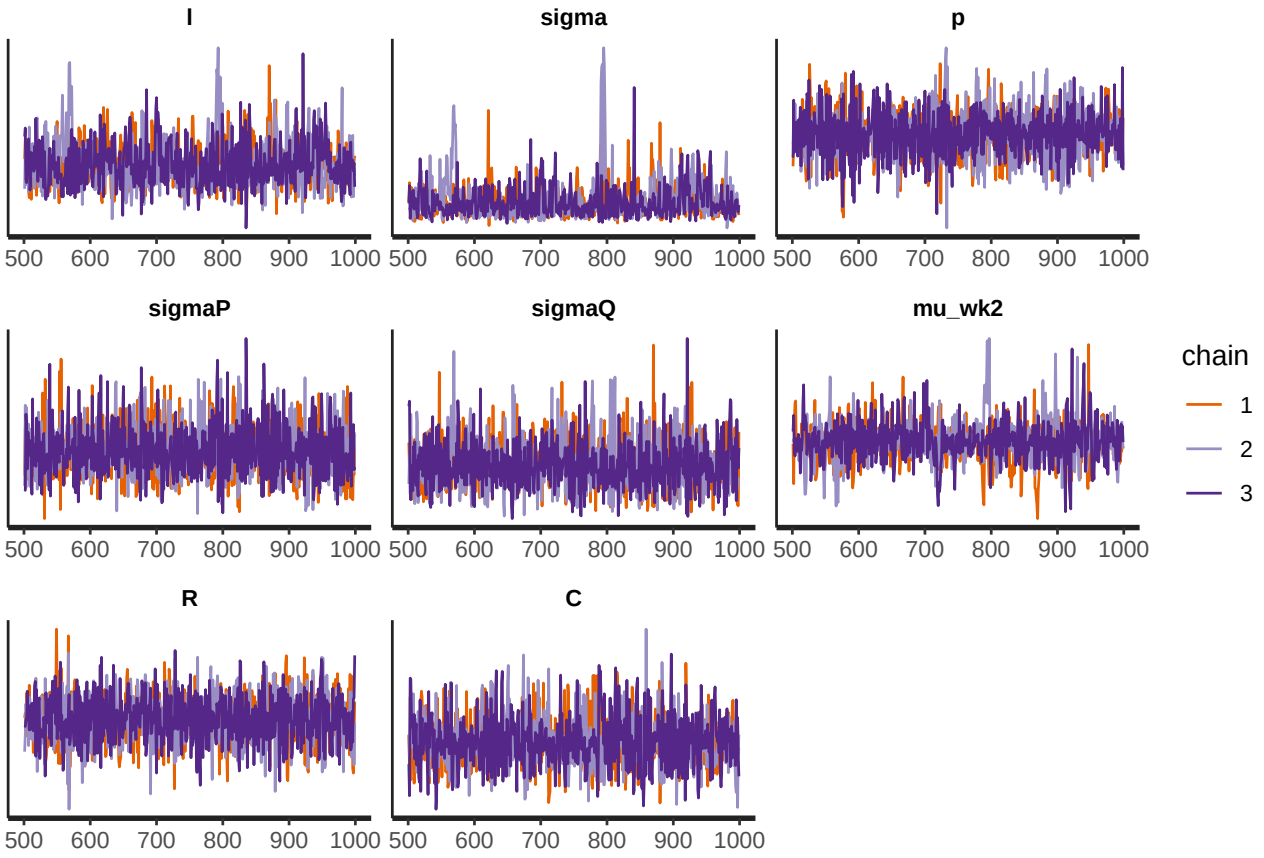
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/Per/WK2__Per_PI_prior.stan'))
```

```
# WK2 (no discrepancy)
fit_Per = stan(file  = 'STAN/Windkessel/Per/WK2__Per_PI_prior.stan',
               data = data_withPred,
               chains = 3,
               iter = 1000,
               seed = 123
)
```

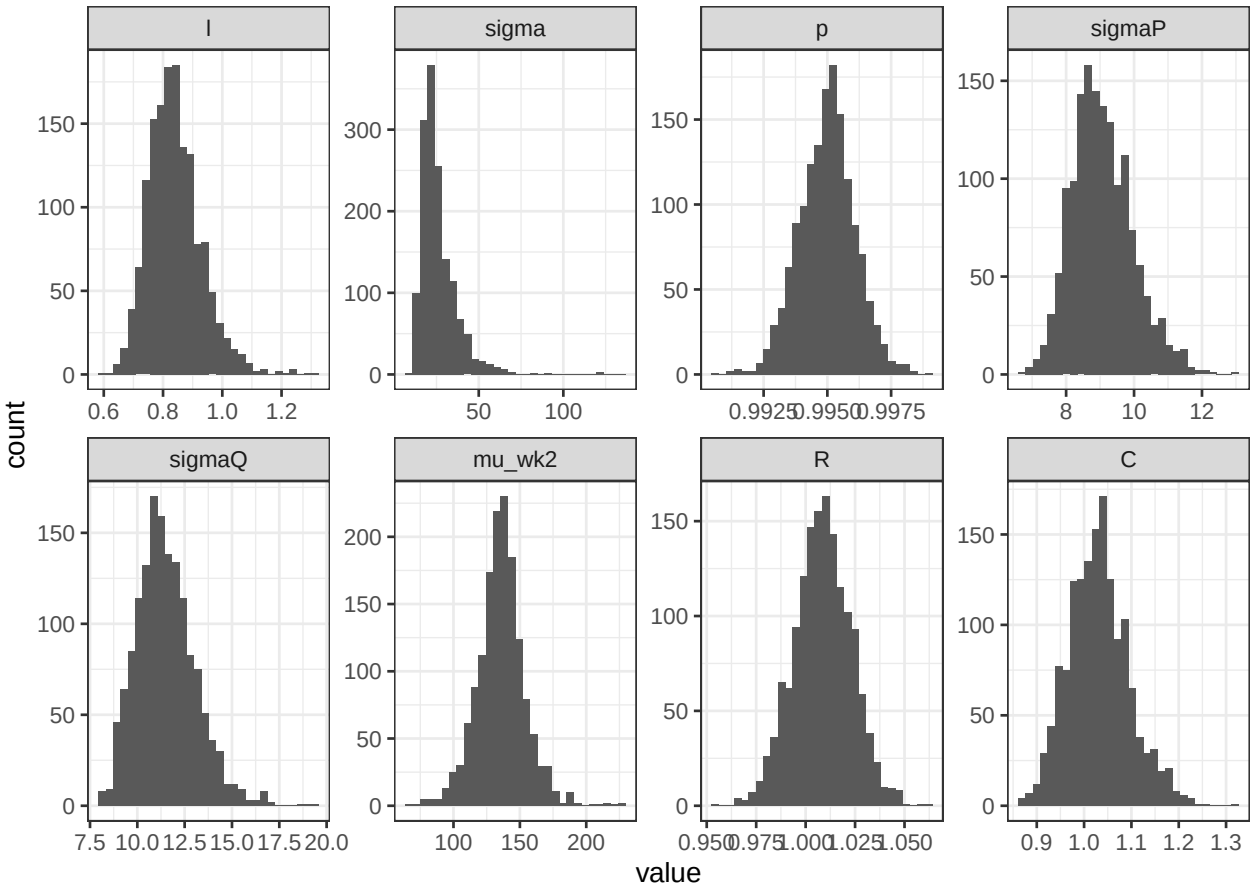
Trace plots

```
stan_trace(fit_Per, pars=names(fit_Per)[1:8])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Posterior densities at the original scales

```
pp_per = transform_post(y=c(yP, yI), fit=fit_Per)
pl_df=pp_per[,names(fit_Per)[1:8]]
pl_df$sample=1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



WK2 + $\delta(t)$ model

For the $\delta(t)$ we use also a squared exponential kernel, $K_\delta(t,t') = \sigma_\delta^2 \exp\left(-\frac{2\sin^2(\pi(t-t')p)}{\ell_\delta^2}\right)$.

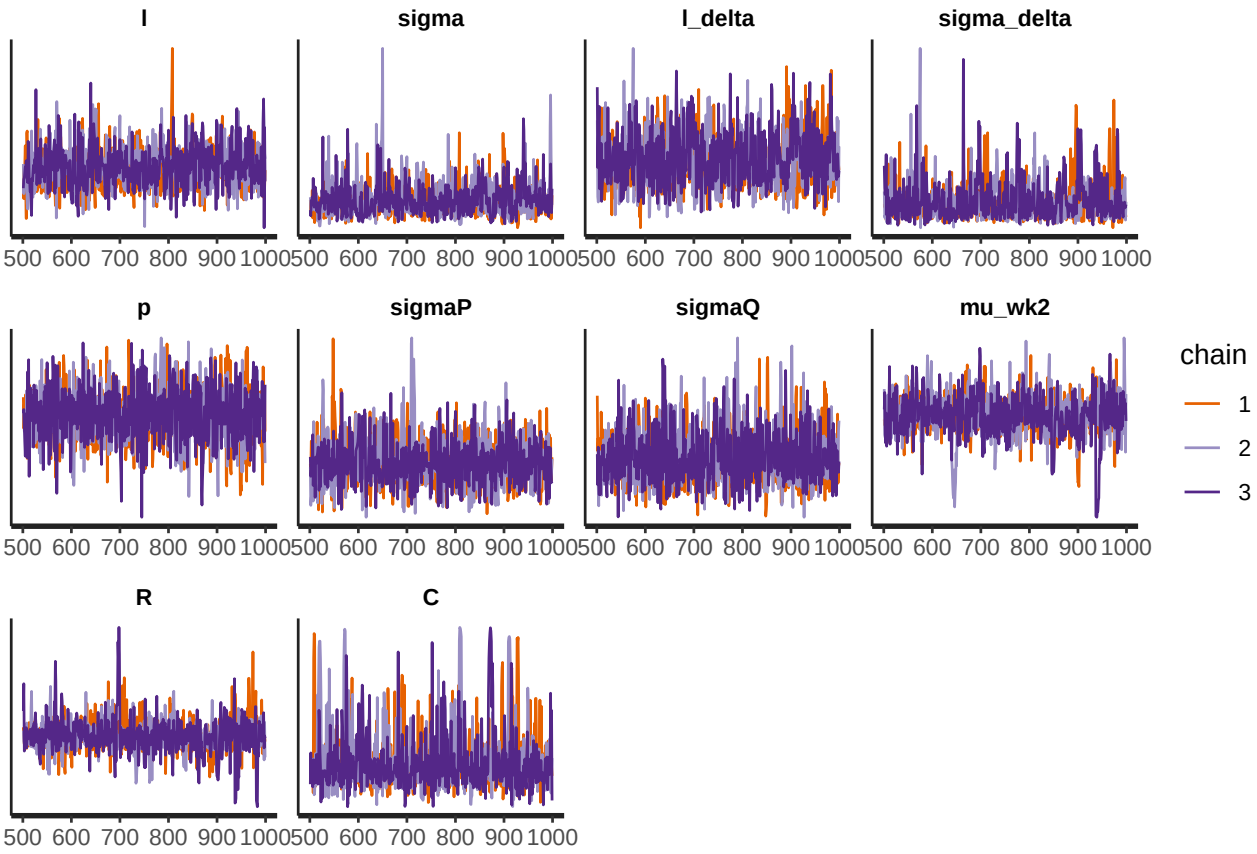
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/Per/WK2_PBias_Per__PI_prior.stan'))
```

```
# WK2 + delta(t)
fit_Per_Bias = stan(file  = 'STAN/Windkessel/Per/WK2_PBias_Per__PI_prior.stan',
                data = data_withPred,
                chains = 3,
                iter = 1000,
                seed = 123
)
```

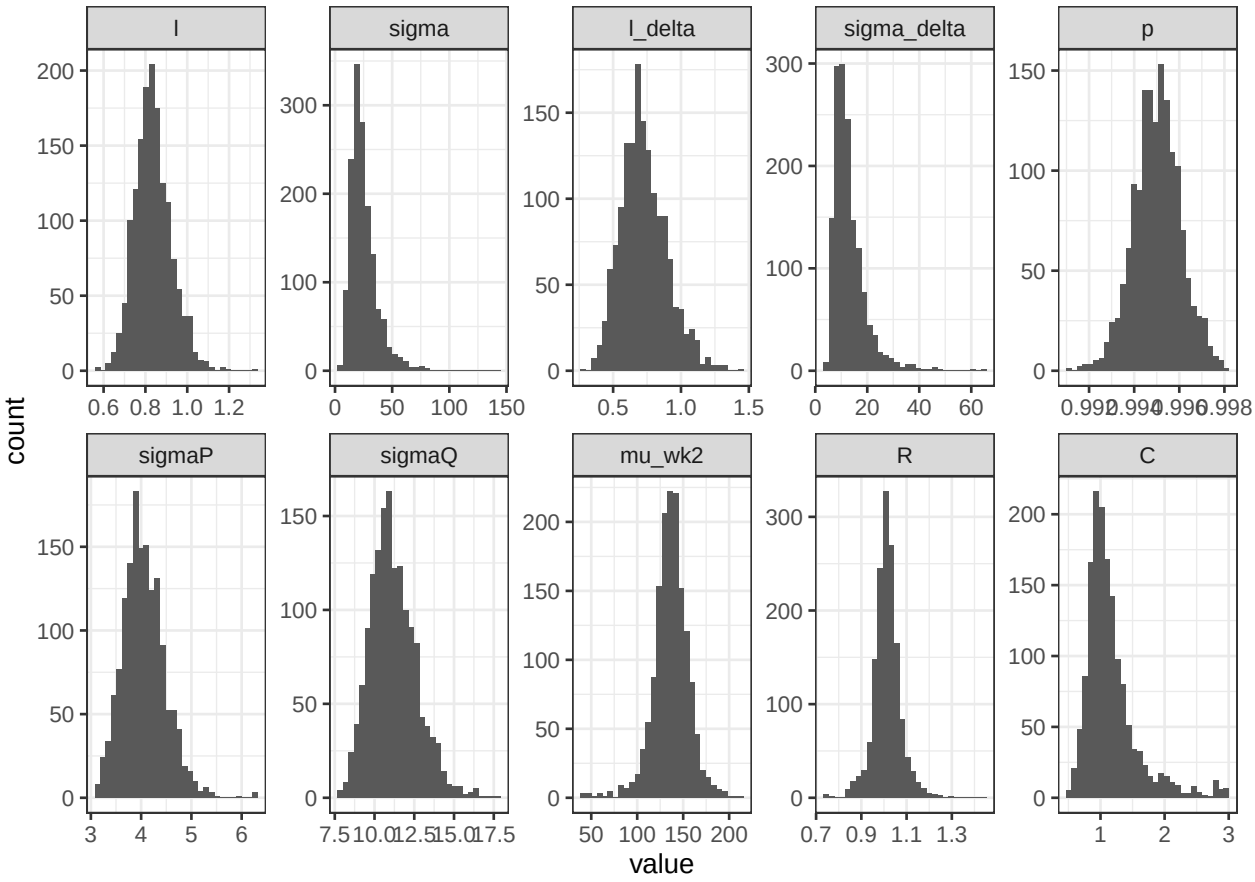
Trace plots

```
stan_trace(fit_Per_Bias, pars=names(fit_Per_Bias)[1:10])+
  theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Posterior densities at the original scales

```
pp_per_Bias = transform_post(y=c(yP, yI), fit=fit_Per_Bias)
pl_df=pp_per_Bias[,names(fit_Per_Bias)[1:10]]
pl_df$sample=1:nrow(pl_df)
m_pl_df = melt(pl_df, id="sample")
ggplot(data=m_pl_df)+
  geom_histogram(aes(x=value))+
  facet_wrap(~variable,nrow = 2, scales = "free")+theme_bw()
```



Plots in Section 3.3

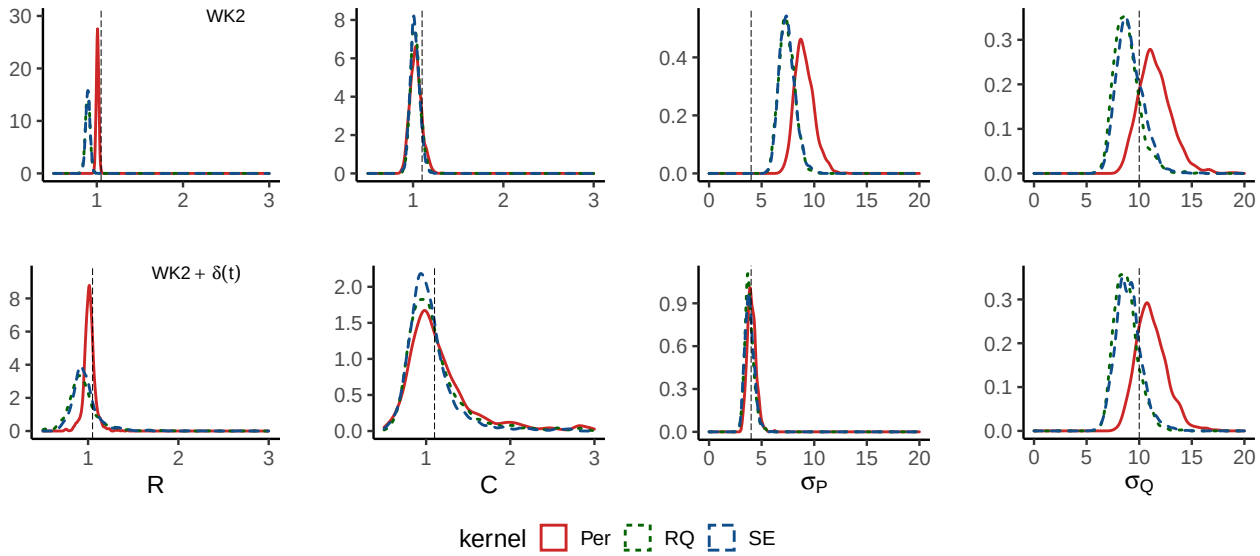
The posterior distributions and the predictions of Section 3.3 are presented.

Posterior distributions of R, C, σ_P and σ_Q

```
rd = list(
  data_list_SE_RQ = ddd$data_noisy_pred, data_list_Per = data_withPred
  , pp_se=pp_se, pp_se_Bias=pp_se_Bias
  , pp_rq=pp_rq, pp_rq_Bias=pp_rq_Bias
  , pp_per=pp_per, pp_per_Bias=pp_per_Bias
  , obsP_SE_RQ=data.frame(obs = ddd$y[1:ddd$data_noisy_pred$nP],
                          time = ddd$data_noisy_pred$tP)
  , obsP_Per=data.frame(obs=yP, time=tP)
  , obsI_SE_RQ=data.frame(obs = ddd$y[(ddd$data_noisy_pred$nP+1):length(ddd$y)],
                          time = ddd$data_noisy_pred$tI)
  , obsI_Per=data.frame(obs=yI, time=tI)
  , data_mod_true_SE_RQ = ddd$data_mod_true
  , data_mod_true_Per = data.frame(P=Pobs, I=flow, time=time)
)
cols_kernel = c("firebrick3", "darkgreen", "dodgerblue4")
Rtrue=1; Ctrue=1.1
attach(rd)
nsample = nrow(pp_se)
post_noBias = data.frame(kernel = c(rep("SE",nsample),
```

```
                                    rep("RQ", nsample),
                                    rep("Per", nsample)))

post_noBias$R = c(pp_se$R, pp_rq$R, pp_per$R)
post_noBias$C = c(pp_se$C, pp_rq$C, pp_per$C)
post_noBias$sigmaP = c(pp_se$sigmaP, pp_rq$sigmaP, pp_per$sigmaP)
post_noBias$sigmaQ = c(pp_se$sigmaQ, pp_rq$sigmaQ, pp_per$sigmaQ)

plR_noBias = ggplot()+
  geom_density(data = post_noBias, aes(x=R, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = Rtrue+Ztrue, linetype = "longdash",size=0.2)+
  xlim(0.5,3)+ xlab("")+
  theme(legend.position = "none") +ylab("")+
  annotate('text', x = 2.5, y = 30, fontface =2,
           label = "WK2",parse = TRUE,size=2.9)+
  scale_color_manual(values = cols_kernel)
plC_noBias =ggplot()+
  geom_density(data = post_noBias, aes(x=C, color=kernel, linetype=kernel),
               size=0.6)+
  geom_vline(xintercept = Ctrue, linetype = "longdash",size=0.2)+
  xlim(0.5,3)+xlab("")+ylab("")+ theme(legend.position = "none") +
  scale_color_manual(values = cols_kernel)

plPsig_noBias =ggplot()+
  geom_density(data = post_noBias, aes(x=sigmaP, color=kernel, linetype=kernel),
               size=0.6)+
  geom_vline(xintercept = 4, linetype = "longdash",size=0.2)+
  xlim(0,20)+xlab("")+ylab("")+
  theme(legend.position = "none") + scale_color_manual(values = cols_kernel)

plQsig_noBias =ggplot()+
  geom_density(data = post_noBias, aes(x=sigmaQ, color=kernel, linetype=kernel),
               size=0.6)+
  geom_vline(xintercept = 10, linetype = "longdash",size=0.2)+
  xlim(0,20)+xlab("")+ylab("")+
  theme(legend.position = "none") + scale_color_manual(values = cols_kernel)


pl_kernels_noBias = ggarrange(plR_noBias, plC_noBias, plPsig_noBias, plQsig_noBias,
                              nrow = 1)
#-------------------------------------------------------------------------------------------

post_Bias = data.frame(kernel = c(rep("SE",nsample), rep("RQ", nsample), rep("Per", nsample)))
post_Bias$R = c(pp_se_Bias$R, pp_rq_Bias$R, pp_per_Bias$R)
post_Bias$C = c(pp_se_Bias$C, pp_rq_Bias$C, pp_per_Bias$C)
post_Bias$sigmaP = c(pp_se_Bias$sigmaP, pp_rq_Bias$sigmaP, pp_per_Bias$sigmaP)
post_Bias$sigmaQ = c(pp_se_Bias$sigmaQ, pp_rq_Bias$sigmaQ, pp_per_Bias$sigmaQ)

plR_Bias = ggplot()+
  geom_density(data = post_Bias, aes(x=R, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = Rtrue+Ztrue, linetype = "longdash",size=0.2)+
  xlim(0.5,3)+ scale_y_continuous(breaks=c(0,2,4,6,8,10))+ylab("")+
  theme(legend.position = "none"
```

```
        #,legend.direction="horizontal"
        ,legend.key.size = unit(0.35, 'cm')
  ) +
  annotate('text', x = 2.2, y = 9.5, fontface =2,
           label = "WK2~+~delta(t)",parse = TRUE,size=2.9) +
  scale_color_manual(values = cols_kernel)

plC_Bias  =ggplot()+
  geom_density(data = post_Bias, aes(x=C, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = Ctrue, linetype = "longdash",size=0.2)+
  xlim(0.5,3)+ylab("")+ theme(legend.position = "none") +
  scale_color_manual(values = cols_kernel)

plPsig_Bias  =ggplot()+
  geom_density(data = post_Bias, aes(x=sigmaP, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = 4, linetype = "longdash",size=0.2)+
  xlim(0,20)+ylab("")+xlab(expression(sigma[P]))+
  theme(legend.position = "none") +
  scale_color_manual(values = cols_kernel)

plQsig_Bias  =ggplot()+
  geom_density(data = post_Bias, aes(x=sigmaQ, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = 10, linetype = "longdash",size=0.2)+
  xlim(0,20)+ylab("")+xlab(expression(sigma[Q]))+
  theme(legend.position = "none") + scale_color_manual(values = cols_kernel)


pl_kernels_Bias = ggarrange(plR_Bias , plC_Bias , plPsig_Bias , plQsig_Bias ,nrow = 1)

pl_legend = ggplot()+
  geom_density(data = post_noBias, aes(x=R, color=kernel, linetype=kernel), size=0.6)+
  geom_vline(xintercept = Rtrue+Ztrue, linetype = "longdash",size=0.2)+
  xlim(0.5,3)+
  theme(
    legend.direction="horizontal"
    ,legend.key.size = unit(0.45, 'cm')
  ) + scale_color_manual(values = cols_kernel)

pl_post_both = ggarrange(pl_kernels_noBias, pl_kernels_Bias, nrow = 2,
                         legend.grob = get_legend(pl_legend, position = "bottom"),
                         legend = "bottom")
pl_post_both
```




```
# ggsave("Figures/pl_post_both.pdf", plot = pl_post_both, width = 18, height = 9, units = "cm")
```

Blood pressure, P(t) and inflow, Q(t) predictions

The predictions can be obtained using the prediction equations in Sections 2.1 and 2.2.

```
t.size = 5
l.size = 7
size.line = 0.3+0.1

### Pressure predictions noBias
cn = colnames(pp_se)
yP_SE = pp_se[,grep("y_P", cn)]
P_mean_CIs = data.frame(
  mean = colMeans(yP_SE)
  , lower = apply(yP_SE,2,quantile,probs=0.05)
  , upper = apply(yP_SE,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_SE_RQ$tP_pred

pl_Ppred_SE = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"), colour = cols_kernel[3], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=P, x=time, linetype = "true"), colour = cols_kernel[3], siz
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(#legend.position = c(0.8, 0.8)
    legend.position = "none"
    , legend.title = element_blank()
    , axis.title.x = element_blank()
    , plot.title = element_text(size=10)
    , legend.spacing.y = unit(0.01, 'cm')
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
```

```
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("Pressure (mmHg)") +
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160) +annotate('text', x = 0.1, y = 155, fontface =2,label = "SE",parse = TRUE,size=3.5) + gg
# pl_Ppred_SE

cn = colnames(pp_rq)
yP_RQ = pp_rq[,grep("y_P", cn)]
P_mean_CIs = data.frame(
  mean = colMeans(yP_RQ)
  , lower = apply(yP_RQ,2,quantile,probs=0.05)
  , upper = apply(yP_RQ,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_SE_RQ$tP_pred#+1.00

pl_Ppred_RQ = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"), colour = cols_kernel[2],
            size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=P, x=time, linetype = "true"),
            colour = cols_kernel[2], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(#legend.position = c(0.8, 0.8)
    legend.position = "none"
    , legend.title = element_blank()
    , axis.title.x = element_blank()
    , legend.spacing.y = unit(0.01, 'cm')
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("") +
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160) +annotate('text', x = 0.1, y = 155, fontface =2,label = "RQ",parse = TRUE,size=3.5)+
  ggtitle("")
# pl_Ppred_RQ

cn = colnames(pp_per)
yP_Per = pp_per[,grep("y_P", cn)]
P_mean_CIs = data.frame(
  mean = colMeans(yP_Per)
  , lower = apply(yP_Per,2,quantile,probs=0.05)
  , upper = apply(yP_Per,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_Per$tP_pred
```

```
pl_Ppred_Per = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"), colour = cols_kernel[1],
            size=size.line)+
  geom_line(data=data_mod_true_Per, aes(y=P, x=time, linetype = "true"),
            colour = cols_kernel[1], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_Per, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(#legend.position = c(0.8, 0.8)
    legend.position = "none"
    , legend.title = element_blank()
    , axis.title.x = element_blank()
    , legend.spacing.y = unit(0.01, 'cm')
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("") +
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160) +
  annotate('text', x = 0.4, y = 155, fontface =2,label = "Per",parse = TRUE,size=3.5)+
  ggtitle("")
# pl_Ppred_Per
pl_Ppred_kernels_noBias = ggarrange(pl_Ppred_SE, pl_Ppred_RQ, pl_Ppred_Per, nrow = 1)
#------------
### Pressure predictions Bias
cn = colnames(pp_se_Bias)
yP_SE_Bias = pp_se_Bias[,grep("y_P", cn)]
P_mean_CIs = data.frame(
  mean = colMeans(yP_SE_Bias)
  , lower = apply(yP_SE_Bias,2,quantile,probs=0.05)
  , upper = apply(yP_SE_Bias,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_SE_RQ$tP_pred

pl_Ppred_SE = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[3], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=P, x=time, linetype = "true"),
            colour = cols_kernel[3], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(#legend.position = c(0.8, 0.8)
    legend.position = "none"
    , legend.title = element_blank()
    , plot.title = element_text(size=10)
    # , axis.title.x = element_blank()
    , legend.spacing.y = unit(0.01, 'cm')
```

```
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("Pressure (mmHg)") + xlab("time (sec)")+
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160) +
  ggtitle(expression(WK2+delta(t)))
# pl_Ppred_SE

cn = colnames(pp_rq_Bias)
yP_RQ_Bias = pp_rq_Bias[,grep("y_P", cn)]
P_mean_CIs = data.frame(
  mean = colMeans(yP_RQ_Bias)
  , lower = apply(yP_RQ_Bias,2,quantile,probs=0.05)
  , upper = apply(yP_RQ_Bias,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_SE_RQ$tP_pred#+1.00

pl_Ppred_RQ = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[2], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=P, x=time, linetype = "true"),
            colour = cols_kernel[2], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(#legend.position = c(0.8, 0.8)
    legend.position = "none"
    , legend.title = element_blank()
    # , axis.title.x = element_blank()
    , legend.spacing.y = unit(0.01, 'cm')
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("") + xlab("time (sec)")+
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160) +
  ggtitle("")
# pl_Ppred_RQ

cn = colnames(pp_per_Bias)
yP_Per_Bias = pp_per_Bias[,grep("y_P", cn)]
P_mean_CIs = data.frame(
```

```
  mean = colMeans(yP_Per_Bias)
  , lower = apply(yP_Per_Bias,2,quantile,probs=0.05)
  , upper = apply(yP_Per_Bias,2,quantile,probs=0.95)
)
P_mean_CIs$time = data_list_Per$tP_pred

pl_Ppred_Per = ggplot(P_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[1], size=size.line)+
  geom_line(data=data_mod_true_Per, aes(y=P, x=time, linetype = "true"),
            colour = cols_kernel[1], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsP_Per, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(
    legend.position = "none"
    , legend.title = element_blank()
    # , axis.title.x = element_blank()
    , legend.spacing.y = unit(0.01, 'cm')
    , legend.direction = "horizontal"
    , legend.background = element_rect(fill='transparent')
    , legend.key.size = unit(0.3, 'cm')
    , legend.key.height = unit(0.05, 'cm')
    , legend.spacing.x = unit(0.01, 'cm')
    , axis.text = element_text(size = t.size)
    , axis.title = element_text(size = l.size)
  )+
  ylab("") + xlab("time (sec)")+
  guides(colour = guide_legend(nrow = 1))+
  ylim(50,160)+
  ggtitle("")
# pl_Ppred_Per
pl_Ppred_kernels_Bias = ggarrange(pl_Ppred_SE, pl_Ppred_RQ, pl_Ppred_Per, nrow = 1)


### Inflow predictions noBias

### Inflow predictions
cn = colnames(pp_se)
yI_SE = pp_se[,grep("y_I", cn)]
I_mean_CIs = data.frame(
  mean = colMeans(yI_SE)
  , lower = apply(yI_SE,2,quantile,probs=0.05)
  , upper = apply(yI_SE,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_SE_RQ$tI_pred

pl_Ipred_SE = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[3], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[3], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
```

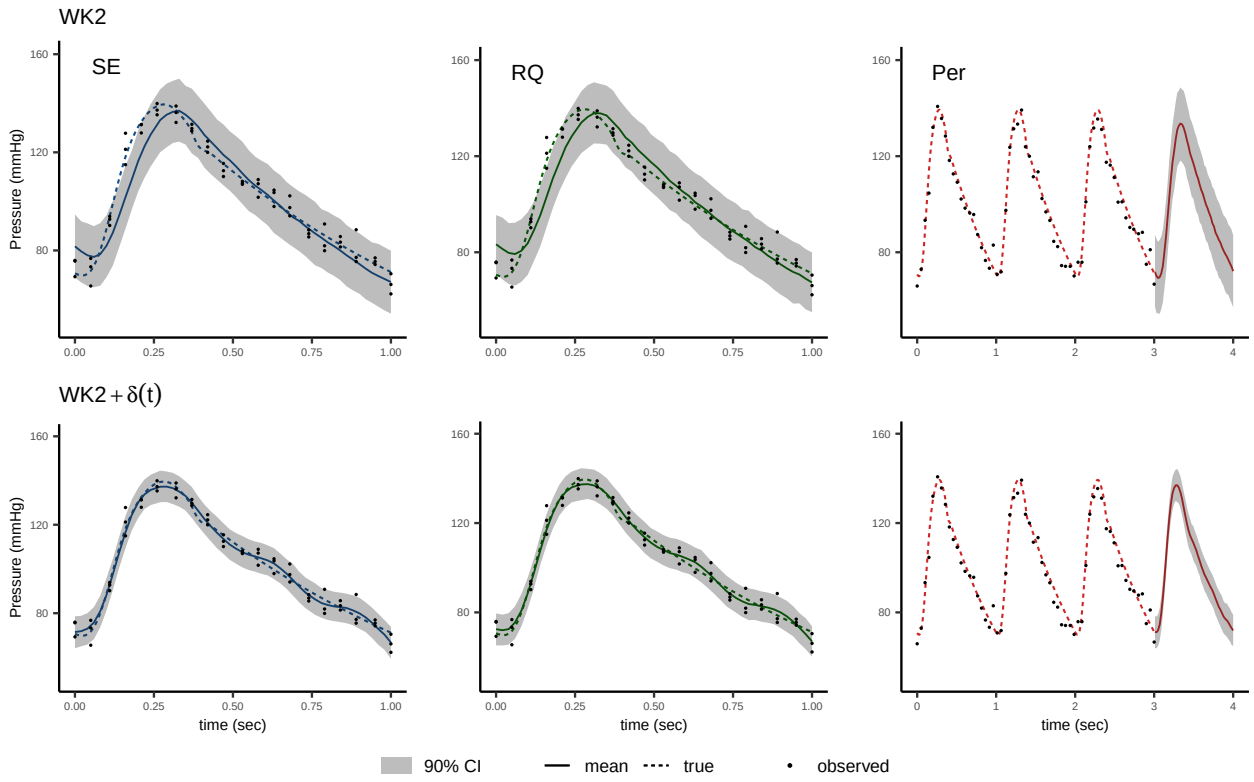
```
  geom_point(data= obsI_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        , axis.title.x = element_blank()
        , plot.title = element_text(size=10)
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  ylab("Inflow (ml/min)")+
  annotate('text', x = 0.8, y = 480, fontface =2,label = "SE",parse = TRUE,size=3.5)+
  ggtitle("WK2")
# pl_Ipred_SE

cn = colnames(pp_rq)
yI_RQ = pp_rq[,grep("y_I", cn)]
I_mean_CIs = data.frame(
  mean = colMeans(yI_RQ)
  , lower = apply(yI_RQ,2,quantile,probs=0.05)
  , upper = apply(yI_RQ,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_SE_RQ$tI_pred

pl_Ipred_RQ = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[2], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[2], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        , axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  ylab("")+
  annotate('text', x = 0.8, y = 480, fontface =2,label = "RQ",parse = TRUE,size=3.5)+
  ggtitle("")
# pl_Ipred_RQ
```

```
cn = colnames(pp_per)
yI_Per = pp_per[,grep("y_I", cn)]
I_mean_CIs = data.frame(
  mean = colMeans(yI_Per)
  , lower = apply(yI_Per,2,quantile,probs=0.05)
  , upper = apply(yI_Per,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_Per$tI_pred

pl_Ipred_Per = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[1], size=size.line)+
  geom_line(data=data_mod_true_Per, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[1], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_Per, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        , axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  ylab("")+
  annotate('text', x = 3.6, y = 480, fontface =2,label = "Per",parse = TRUE,size=3.5)+
  ggtitle("")
# pl_Ipred_Per

pl_Ipred_kernels_noBias = ggarrange(pl_Ipred_SE, pl_Ipred_RQ, pl_Ipred_Per, nrow = 1)

### Inflow predictions
cn = colnames(pp_se_Bias)
yI_SE = pp_se_Bias[,grep("y_I", cn)]
I_mean_CIs = data.frame(
  mean = colMeans(yI_SE)
  , lower = apply(yI_SE,2,quantile,probs=0.05)
  , upper = apply(yI_SE,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_SE_RQ$tI_pred

pl_Ipred_SE = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[3], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[3], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
```

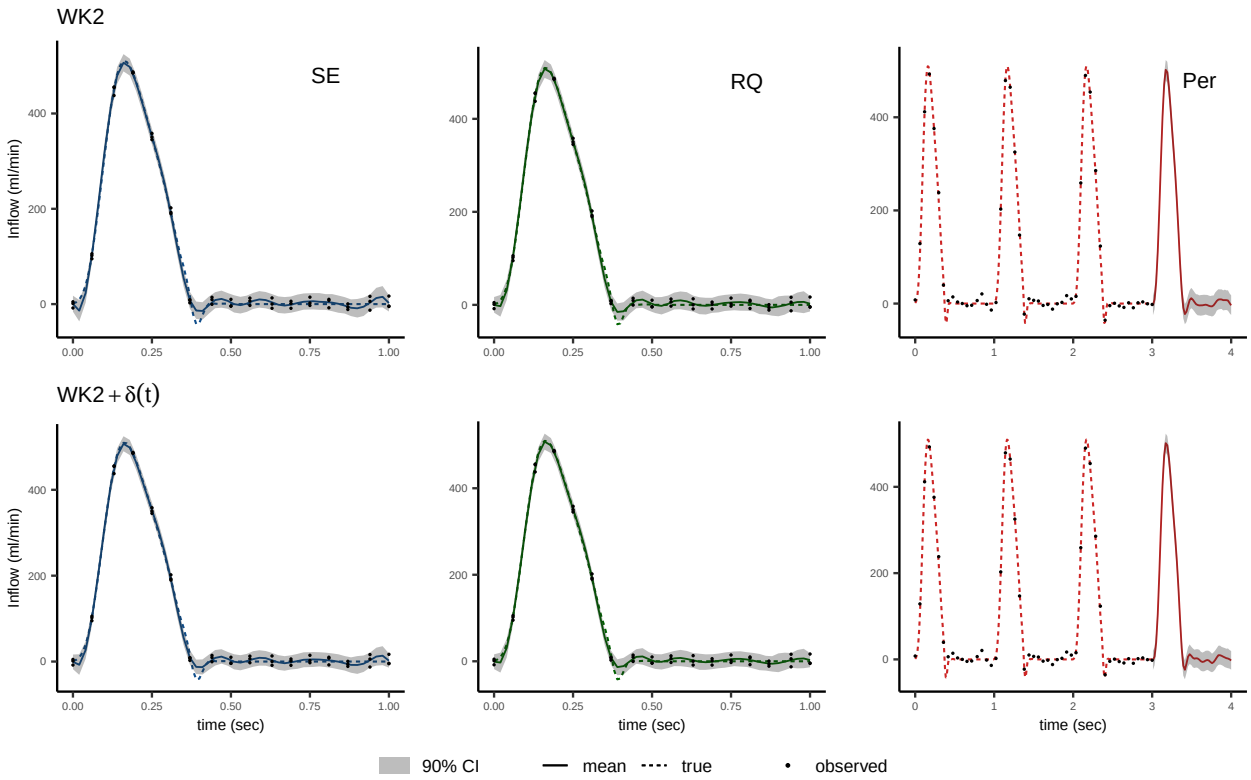
```
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        , plot.title = element_text(size=10)
        #, axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  xlab("time (sec)")+ylab("Inflow (ml/min)")+
  ggtitle(expression(WK2+delta(t)))
# pl_Ipred_SE

cn = colnames(pp_rq_Bias)
yI_RQ = pp_rq_Bias[,grep("y_I", cn)]
I_mean_CIs = data.frame(
  mean = colMeans(yI_RQ)
  , lower = apply(yI_RQ,2,quantile,probs=0.05)
  , upper = apply(yI_RQ,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_SE_RQ$tI_pred

pl_Ipred_RQ = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[2], size=size.line)+
  geom_line(data=data_mod_true_SE_RQ, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[2], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_SE_RQ, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        #, axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  xlab("time (sec)")+ylab("")+
  ggtitle("")
# pl_Ipred_RQ

cn = colnames(pp_per_Bias)
yI_Per = pp_per_Bias[,grep("y_I", cn)]
```

```r
I_mean_CIs = data.frame(
  mean = colMeans(yI_Per)
  , lower = apply(yI_Per,2,quantile,probs=0.05)
  , upper = apply(yI_Per,2,quantile,probs=0.95)
)
I_mean_CIs$time = data_list_Per$tI_pred

pl_Ipred_Per = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = cols_kernel[1], size=size.line)+
  geom_line(data=data_mod_true_Per, aes(y=I, x=time, linetype = "true"),
            colour = cols_kernel[1], size=size.line+0.05)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_Per, aes(x=time, y=obs, shape = "observed"), size=0.4)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = "none"
        , legend.title = element_blank()
        #, axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+
  xlab("time (sec)")+ylab("")+ggtitle("")
# pl_Ipred_Per

pl_Ipred_kernels_Bias = ggarrange(pl_Ipred_SE, pl_Ipred_RQ, pl_Ipred_Per, nrow = 1)


#---------
### create unique legend for all
pl_pred_legend = ggplot(I_mean_CIs) +
  geom_line(aes(y=mean, x=time, linetype = "mean"),
            colour = "black", size=size.line)+
  geom_line(data=data_mod_true_Per, aes(y=I, x=time, linetype = "true"),
            colour = "black", size=size.line)+
  geom_ribbon(aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"), alpha = 0.3)+
  geom_point(data= obsI_Per, aes(x=time, y=obs, shape = "observed"), size=0.7)+
  scale_fill_manual("",values=c("90% CI" = "grey12"))+
  theme(legend.position = c(0.7, 0.7)
        , legend.title = element_blank()
        , axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.5, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=8)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
```

```
        , axis.title = element_text(size = l.size)
  )+
  ylab("")
pl_Ipred = ggarrange(pl_Ppred_kernels_noBias, pl_Ipred_kernels_noBias
                     , nrow = 2, legend = "bottom", common.legend = TRUE
                     , legend.grob =
                       get_legend(pl_pred_legend, position = "bottom"))
# pl_PIpred_noBias

(pl_Ppred_both=ggarrange(pl_Ppred_kernels_noBias,pl_Ppred_kernels_Bias, ncol = 1
                         , legend = "bottom", common.legend = TRUE
                         , legend.grob =
                           get_legend(pl_pred_legend, position = "bottom")))
```




```
(pl_Ipred_both=ggarrange(pl_Ipred_kernels_noBias,pl_Ipred_kernels_Bias, ncol = 1
                         , legend = "bottom", common.legend = TRUE
                         , legend.grob =
                           get_legend(pl_pred_legend, position = "bottom")))
```




```
# ggsave("Figures/pl_Ppred_both.pdf"
#        , plot = pl_Ppred_both, width = 15, height = 8, units = "cm")
# ggsave("Figures/pl_Ipred_both.pdf"
#        , plot = pl_Ipred_both, width = 15, height = 8, units = "cm")
```

The total run time is

```
Time difference of 21.5205 mins
```

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] reshape2_1.4.4       ggpubr_0.4.0         rstan_2.21.3
[4] ggplot2_3.3.5        StanHeaders_2.21.0-7

loaded via a namespace (and not attached):
 [1] tidyselect_1.1.1   xfun_0.29          purrr_0.3.4        carData_3.0-5
 [5] colorspace_2.0-2   vctrs_0.3.8        generics_0.1.2     htmltools_0.5.2
 [9] stats4_4.0.3       loo_2.4.1          yaml_2.2.2         utf8_1.2.2
[13] rlang_1.0.0        pkgbuild_1.3.1     pillar_1.7.0       glue_1.6.1
[17] withr_2.4.3        DBI_1.1.2          plyr_1.8.6         matrixStats_0.61.0
[21] lifecycle_1.0.1    stringr_1.4.0      munsell_0.5.0      ggsignif_0.6.3
[25] gtable_0.3.0       codetools_0.2-18   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        broom_0.7.12
[37] Rcpp_1.0.8         backports_1.4.1    scales_1.1.1       RcppParallel_5.1.5
[41] abind_1.4-5        farver_2.1.0       gridExtra_2.3      digest_0.6.29
[45] stringi_1.7.6      rstatix_0.7.0      processx_3.5.2     dplyr_1.0.7
[49] cowplot_1.1.1      grid_4.0.3         cli_3.1.1          tools_4.0.3
[53] magrittr_2.0.2     tibble_3.1.6       car_3.0-12         tidyr_1.2.0
[57] crayon_1.4.2       pkgconfig_2.0.3    ellipsis_0.3.2     prettyunits_1.1.1
[61] assertthat_0.2.1   rmarkdown_2.11     rstudioapi_0.13    R6_2.5.1
[65] compiler_4.0.3
```