S5) Real data–WK models

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)
# function to extract the stan output
source("functions/WK_exp_fn.R")
```

Section 5: Real data–WK models

This case study is based in observations of blood flow and blood pressure from one individual that took part in a randomized controlled trial. The model we use is the WK2 model and is given by the following differential equation

$$Q(t) = \frac{1}{R}P(t) + C\frac{dP(t)}{dt}. \tag{WK2}$$

We built two physics-informed GP priors for the model. The first does not account for model disrcepancy and the second accounts for model discrepancy. For both models we use the periodic kernel, since the blood pressure repeats itself at each cardiac cycle (heartbeat).

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

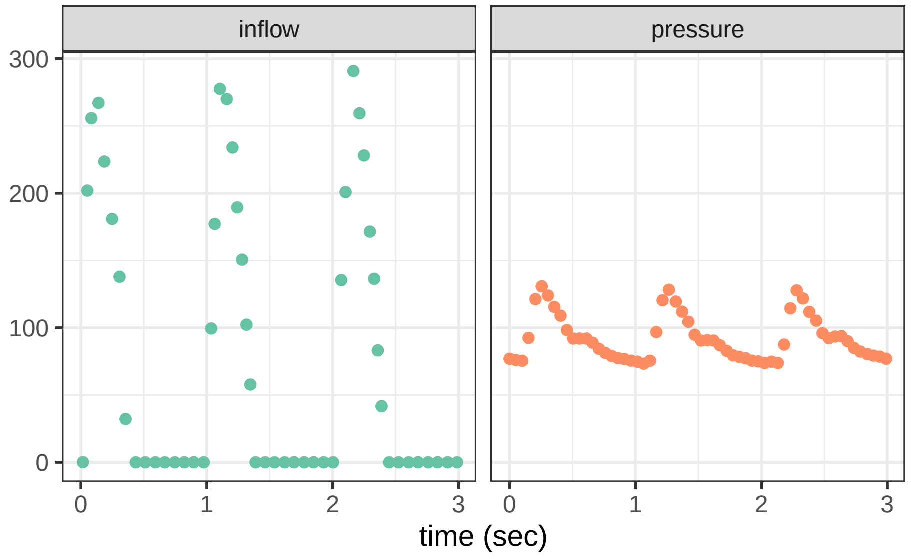
$$p(\mathbf{y} \mid \boldsymbol{\theta}, \boldsymbol{\phi}, \sigma_P, \sigma_Q) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{4}$$

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

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

Load data

```r
data_withPred=readRDS("Data/Real_data.rds")
Pdf = data.frame(time=data_withPred$tP, val=data_withPred$yP_original)
Pdf$var = "pressure"

Qdf = data.frame(time=data_withPred$tI, val=data_withPred$yI_original)
Qdf$var = "inflow"
dfPQ = rbind(Pdf, Qdf)
ggplot()+
  geom_point(data = dfPQ, aes(x=time, y=val, colour = var))+
  theme_bw()+
  xlab("time (sec)")+ylab("")+
  scale_color_brewer(palette="Set2")+
   theme(legend.position = "none")+
  facet_wrap(var~.)
```



- Observe that the inflow is 0 for approximately the 2/3 of each for the three cardiac cycles. This is during diastole that the heart fills with blood received by the veins.

Therefore, we set the inflow noise, ε_Q to be 0 during the diastole as follows

$$\sigma_Q = \begin{cases} s_Q, & \text{if } t = t_{\text{sys}} \\ 0, & \text{if } t = t_{\text{dia}} \end{cases},$$

where t_{sys} (systole) is the remaining cardiac cycle time.

WK2 PI prior-not accounting for model discrepancy

First, we fit the PI prior model that does not account for model discrepancy

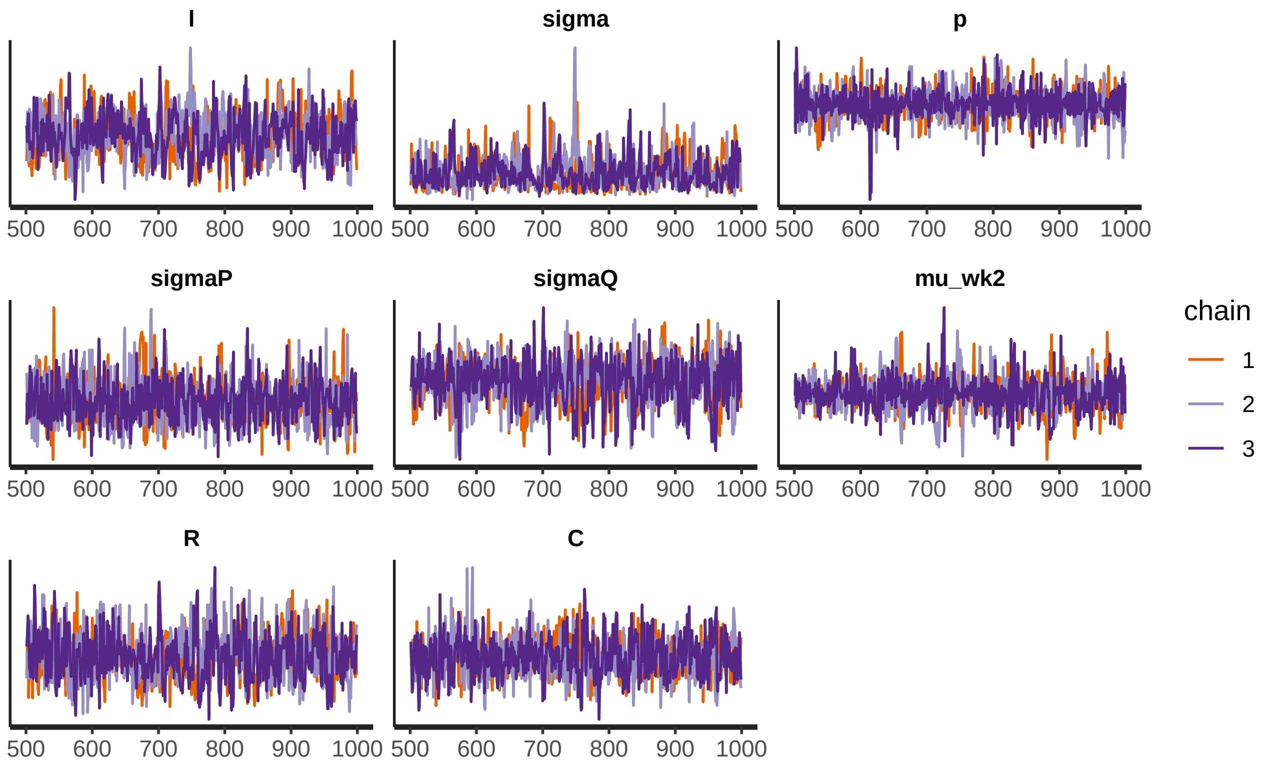
Stan code (select `eval=TRUE` in the code chunk to see the stan code):

```
writeLines(readLines('STAN/Windkessel/Per/WK2__Per_dia0_PI_prior.stan'))
```

```
data_withPred$yDia=0
fit_Per = stan(file  = 'STAN/Windkessel/Per/WK2__Per_dia0_PI_prior.stan',
               data = data_withPred,
               chains = 3,
               iter = 1000,
               seed = 123
)
```

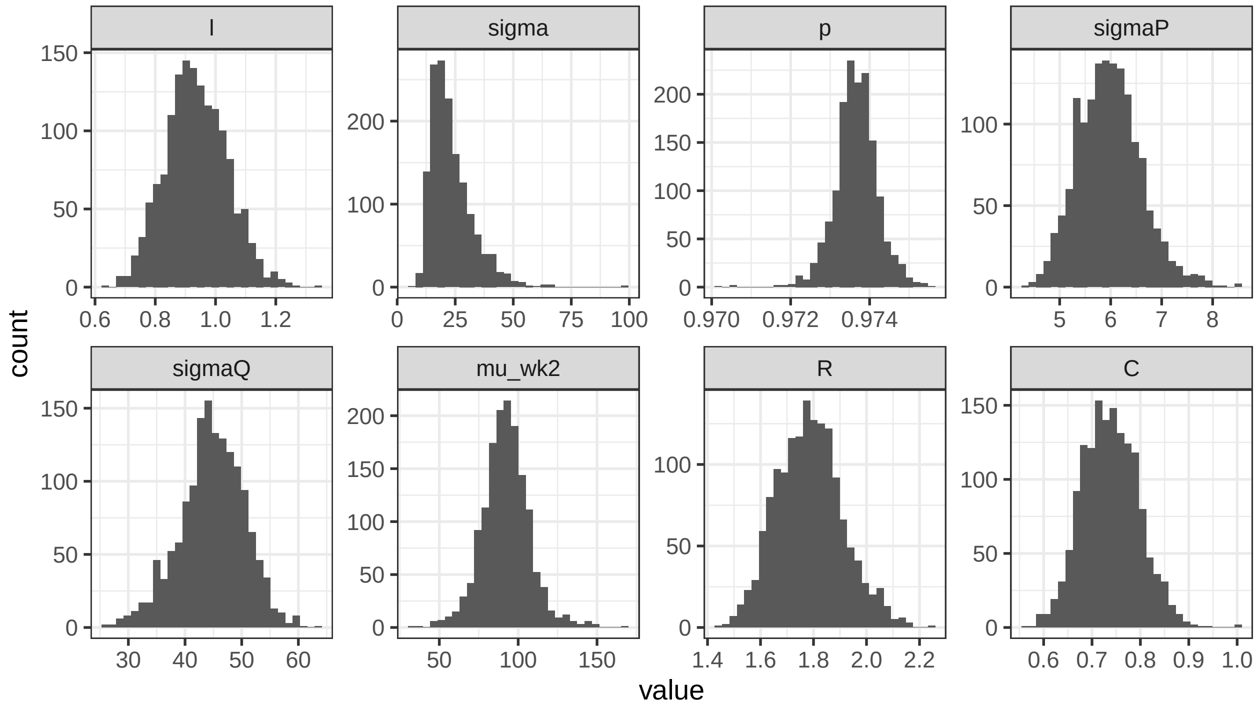
Trace plots

```
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())
```



Transforming the posterior in the original scale

```r
pp_Per =transform_post(y=data_withPred$y, fit=fit_Per)
pl_df=pp_Per[,names(fit_Per)[1:8]]
pl_df$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()
```



The noise standard deviation, σ_Q is quite large given the range of the observed inflow

```
range(data_withPred$yI_original)
```

```
[1]   0.0000 290.8027
```

WK2 + $\delta(t)$ model

Now with fit the model that accounts for model discrepancy

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

```
writeLines(readLines('STAN/Windkessel/Per/WK2_PBias__Per_dia0_PI_prior.stan'))
```

```
data_withPred$yDia=0
fit_Per_Bias = stan(file  = 'STAN/Windkessel/Per/WK2_PBias__Per_dia0_PI_prior.stan',
                    data = data_withPred,
                    chains = 3,
                    iter = 1000,
                    seed = 123
)

pp_Per_Bias =transform_post(y=data_withPred$y, fit=fit_Per_Bias)
```

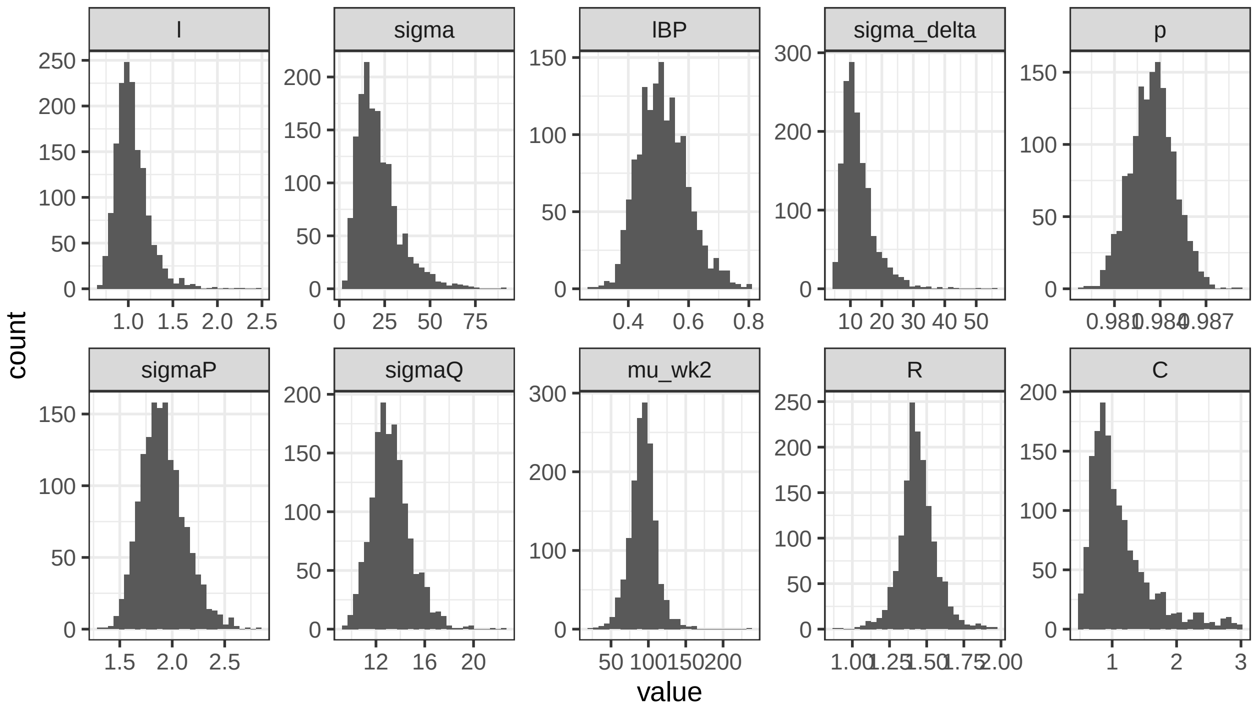
Trace plots

```
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())
```

Transforming the posterior in the original scale Transforming the posterior in the original scale

```
y = data_withPred$y
pp_Per_Bias =transform_post(y=data_withPred$y, fit=fit_Per_Bias)
```

```r
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()
```

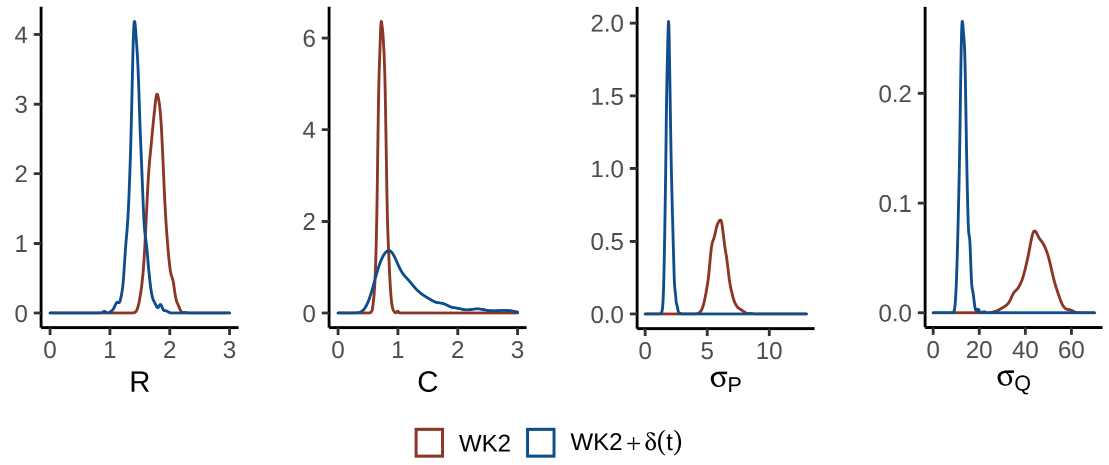


Plots in Section 5

The posterior distributions and the predictions of Section 5 are presented.

Posterior distributions of R, C, σ_P and σ_Q

```r
col_WK2 = "tomato4"
col_WK2_delta = "dodgerblue4"
post_noBias = pp_Per
post_Bias = pp_Per_Bias
plR = ggplot()+
  geom_density(data = post_noBias, aes(x=R,colour="WK2"))+
  geom_density(data = post_Bias, aes(x=R,colour="WK2_delta"))+
  xlim(0,3)+ #xlab("")+
  theme(legend.position = "none") +ylab("")+
  scale_colour_manual(values = c("WK2"=col_WK2,"WK2_delta"=col_WK2_delta),
                      name = '',
                      labels = c(expression("WK2", WK2 + delta(t))))
plC = ggplot()+
  geom_density(data = post_noBias, aes(x=C),colour=col_WK2)+
  geom_density(data = post_Bias, aes(x=C),colour=col_WK2_delta)+
  xlim(0,3)+ #xlab("")+
  theme(legend.position = c(0.8, 0.8)) +ylab("")

plPsigma = ggplot()+
  geom_density(data = post_noBias, aes(x=sigmaP),colour=col_WK2)+
  geom_density(data = post_Bias, aes(x=sigmaP),colour=col_WK2_delta)+
  xlim(0,13)+xlab(expression(sigma[P])) +ylab("")

plQsigma = ggplot()+
  geom_density(data = post_noBias, aes(x=sigmaQ),colour=col_WK2)+
  geom_density(data = post_Bias, aes(x=sigmaQ),colour=col_WK2_delta)+
  xlim(0,70)+xlab(expression(sigma[Q]))+ylab("")

pl_leg = ggplot()+
  geom_density(data = post_noBias, aes(x=R,colour="WK2"))+
  geom_density(data = post_Bias, aes(x=R,colour="WK2_delta"))+
  xlim(0,3)+ #xlab("")+
  theme(legend.position = "bottom", legend.key.size=unit(0.4,'cm'))+
  ylab("")+
  scale_colour_manual(values=c("WK2"=col_WK2,"WK2_delta"=col_WK2_delta)
                     ,name = '',
                     labels = c(expression("WK2", WK2 + delta(t))))

(pl_post = ggarrange(plR, plC,plPsigma, plQsigma, nrow = 1,
                     legend = "bottom", common.legend = TRUE,
                     legend.grob =
                       get_legend(pl_leg, position = "bottom")))
```



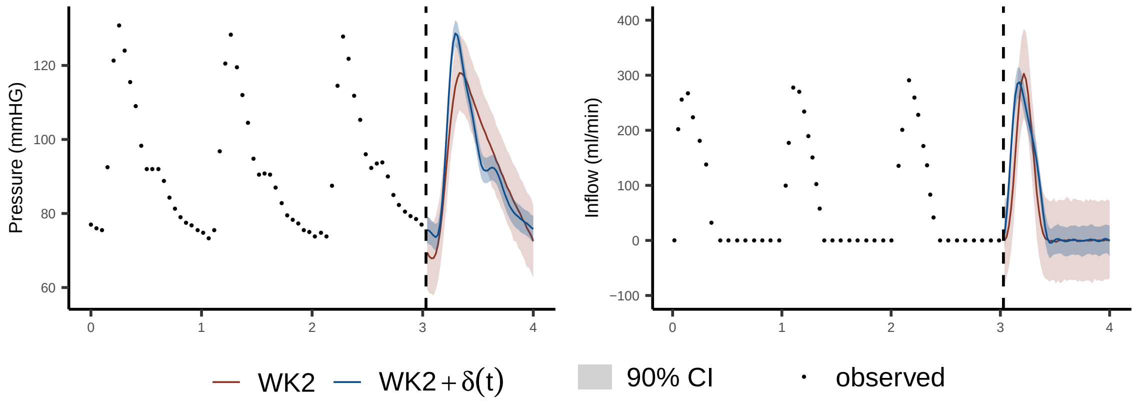
```r
# ggsave("Figures/post_real_dia0_both.pdf", plot = pl_post, width = 15, height = 5, units = "cm")
```

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.

```r
scaleFUN = function(x) sprintf("%.1f", x)
t.size = 5
l.size = 7
size.line = 0.3

obsP = data.frame(time=data_withPred$tP,
                  obs = data_withPred$yP_original)
cn = colnames(pp_Per)
yP_Per = pp_Per[,grep("y_P", cn)]
P_mean_CIs_noBias = 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_noBias$time = data_withPred$tP_pred

# Accounting for Bias
cn = colnames(pp_Per_Bias)
yP_Per_Bias = pp_Per_Bias[,grep("y_P", cn)]
P_mean_CIs_Bias = 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_Bias$time = data_withPred$tP_pred

pl_Ppred = ggplot() +
  geom_line(data=P_mean_CIs_noBias,
            aes(y=mean, x=time, linetype = "mean"),
            colour = col_WK2, size=size.line)+
  geom_ribbon(data=P_mean_CIs_noBias,
              aes(ymin=lower, ymax=upper, x=time),
              fill = col_WK2, alpha = 0.2)+
  geom_line(data=P_mean_CIs_Bias,
            aes(y=mean, x=time, linetype = "mean"),
            colour = col_WK2_delta, size=size.line)+
  geom_ribbon(data=P_mean_CIs_Bias,
              aes(ymin=lower, ymax=upper, x=time),
              fill = col_WK2_delta, alpha = 0.3)+
  geom_vline(xintercept = min(data=P_mean_CIs_noBias$time)-0.01,
             linetype="dashed")+
  geom_point(data=obsP, aes(x=time, y=obs, shape = "observed"),
             size=0.4)+
  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("Pressure (mmHG)") #+
#----------------------------
# Inflow predictions
```

```r
cn = colnames(pp_Per)
yI_Per = pp_Per[,grep("y_I", cn)]
I_mean_CIs_noBias = 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_noBias$time = data_withPred$tI_pred
obsI = data.frame(time=data_withPred$tI, obs=data_withPred$yI_original)

cn = colnames(pp_Per_Bias)
yI_Per = pp_Per_Bias[,grep("y_I", cn)]
I_mean_CIs_Bias = 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_Bias$time = data_withPred$tI_pred

pl_Ipred = ggplot() +
  geom_line(data=I_mean_CIs_noBias,
            aes(y=mean, x=time, linetype = "mean"),
            colour = col_WK2, size=size.line)+
  geom_ribbon(data=I_mean_CIs_noBias,
              aes(ymin=lower, ymax=upper, x=time),
              fill = col_WK2, alpha = 0.2)+
  geom_line(data=I_mean_CIs_Bias,
            aes(y=mean, x=time, linetype = "mean"),
            colour = col_WK2_delta, size=size.line)+
  geom_ribbon(data=I_mean_CIs_Bias,
              aes(ymin=lower, ymax=upper, x=time),
              fill = col_WK2_delta, alpha = 0.3)+
  geom_point(data= obsI,
             aes(x=time, y=obs, shape = "observed"),
             size=0.4)+
  geom_vline(xintercept=min(I_mean_CIs_noBias$time)-0.01,
             linetype="dashed")+
  theme(legend.position = "none"
        , legend.title = element_blank()
        , axis.title.x = element_blank()
        , legend.direction = "horizontal"
        , legend.background = element_rect(fill='transparent')
        , legend.key.size = unit(0.3, 'cm')
        , legend.key.height = unit(0.01, 'cm')
        , legend.text = element_text(size=6)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+ ylim(-100,400 )+
  xlab("")+ylab("Inflow (ml/min)")
#---------------------------------
pl_leg = ggplot() +
  geom_line(data=I_mean_CIs_noBias,
            aes(y=mean, x=time,colour="WK2"),
            size=size.line)+
  geom_ribbon(data=I_mean_CIs_noBias,
              aes(ymin=lower, ymax=upper, x=time, fill = "90% CI"),
              alpha = 0.2)+
  geom_line(data=I_mean_CIs_Bias,
            aes(y=mean, x=time,colour="WK2_delta"),
            size=size.line)+
  geom_point(data= obsI,
             aes(x=time, y=obs, shape = "observed"),
             size=0.4)+
  geom_vline(xintercept=min(I_mean_CIs_noBias$time)-0.01,
             linetype="dashed")+
  scale_fill_manual("",values=c("90% CI"="grey12"))+
  scale_colour_manual(values=c("WK2"=col_WK2, "WK2_delta"=col_WK2_delta),
                      name = 'mean',
                      labels = c(expression("WK2", WK2 + delta(t))))+
  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*1.5, 'cm')
        , legend.key.height = unit(0.01*2, 'cm')
        , legend.text = element_text(size=10)
        , legend.spacing.y  = unit(0.01, 'cm')
        , axis.text = element_text(size = t.size)
        , axis.title = element_text(size = l.size)
  )+ ylim(-100,400 )+
  xlab("")+ylab("Inflow (ml/min)")
(pred_both=ggarrange(pl_Ppred, pl_Ipred, nrow = 1,
                     legend = "bottom", common.legend = TRUE,
                     legend.grob = get_legend(pl_leg, position = "bottom")))
```



```
# ggsave("Figures/pred_both_same.pdf", plot = pred_both, width = 15, height = 5, units = "cm")
```

The total run time is

```
Time difference of 8.304318 mins
```

Session information

```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] 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          RColorBrewer_1.1-2 plyr_1.8.6
[21] matrixStats_0.61.0 lifecycle_1.0.1    stringr_1.4.0      munsell_0.5.0
[25] ggsignif_0.6.3     gtable_0.3.0       codetools_0.2-18   evaluate_0.14
[29] labeling_0.4.2     inline_0.3.19      knitr_1.37         callr_3.7.0
[33] fastmap_1.1.0      ps_1.6.0           parallel_4.0.3     fansi_1.0.2
[37] broom_0.7.12       Rcpp_1.0.8         backports_1.4.1    scales_1.1.1
[41] RcppParallel_5.1.5 abind_1.4-5        farver_2.1.0       gridExtra_2.3
[45] digest_0.6.29      stringi_1.7.6      rstatix_0.7.0      processx_3.5.2
[49] dplyr_1.0.7        cowplot_1.1.1      grid_4.0.3         cli_3.1.1
[53] tools_4.0.3        magrittr_2.0.2     tibble_3.1.6       car_3.0-12
[57] tidyr_1.2.0        crayon_1.4.2       pkgconfig_2.0.3    ellipsis_0.3.2
[61] prettyunits_1.1.1  assertthat_0.2.1   rmarkdown_2.11     rstudioapi_0.13
[65] R6_2.5.1           compiler_4.0.3
```