Bayesian Learning: Lab 4

Mohsen Pirmoradiyan, Ahmed Alhasan

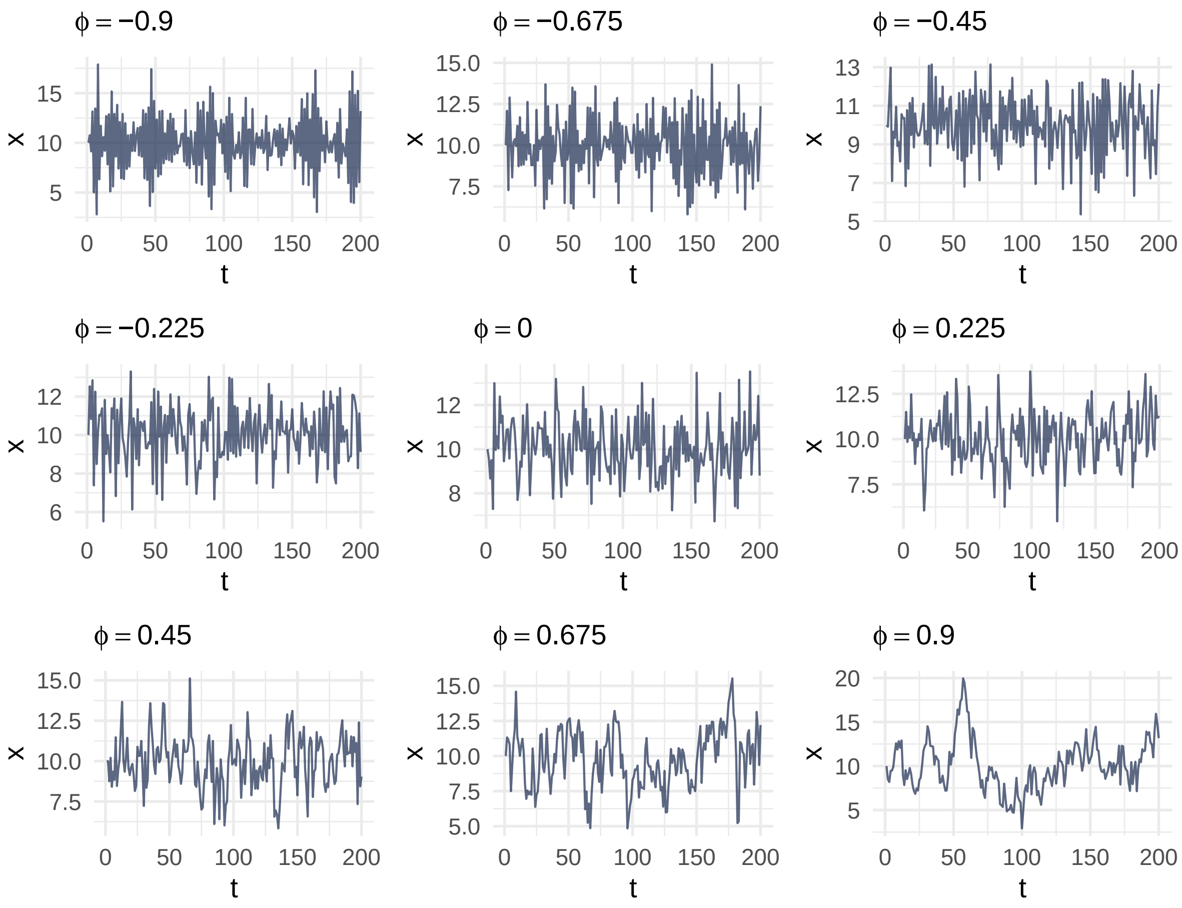
2020-05-17

1. ***Time series models in Stan.***

(a) Write a function in R that simulates data from the AR(1)-process

$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \;\; \epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$$

for given values of μ, ϕ and σ^2. Start the process at $x_1 = \mu$ and then simulate values for x_t for t = 2, 3 . . . , T and return the vector $x_{1:T}$ containing all time points. Use $\mu = 10$, $\sigma^2 = 2$ and T = 200 and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stable). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

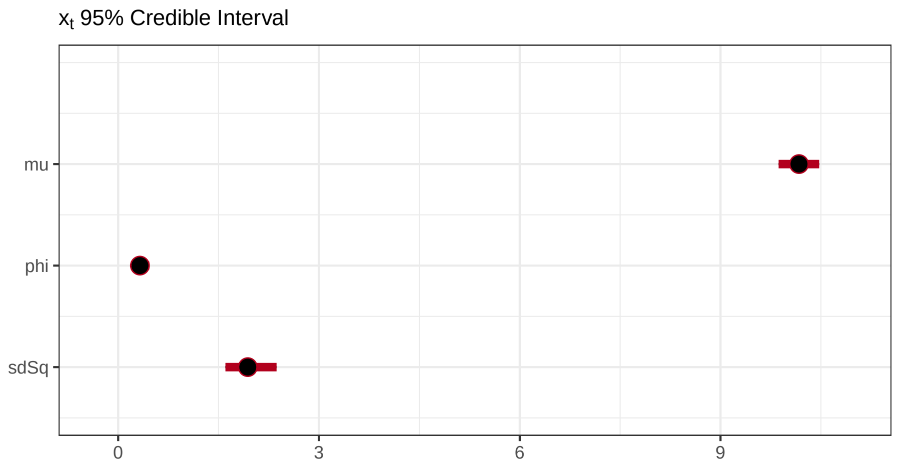


- When ϕ approches 1 the value of x_t get a larger contribution from the previous point x_{t-1} compared with the white noise ϵ_t, therefore we see a correlation between x_t values.
- When $\phi = 0$ the value of x_t is just the error ϵ_t around the mean μ i.e.($x_t = \mu + \epsilon_t$).

- When ϕ approches -1 the value of x_t also get a larger contribution from the previous term x_{t-1} compared with the white noise ϵ_t but this time the middle term $\phi(x_{t-1} - \mu)$ will have opposite sign for each successive points in oscillations
- When $\phi = 1$ the variance of x_t depends on time lag t, so that the variance of the series diverges to infinity as t goes to infinity, and same thing for $\phi = -1$ but this time with overlapping signs as explained above.

(b) Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.95$. Now, treat your simulated vectors as synthetic data, and treat the values of μ, ϕ and σ^2 as unknown and estimate them using MCMC. Implement Stan-code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan reference manual, and note the different parameterization used here.]

(i) Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?



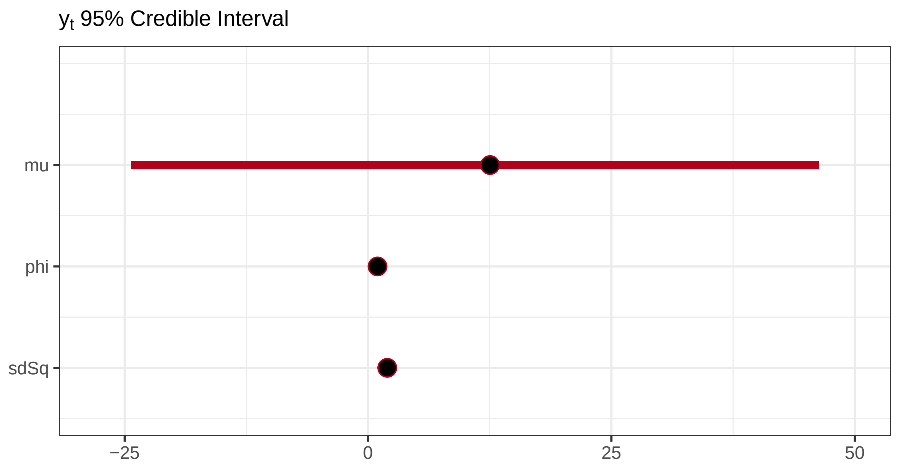


Table 1: x 95% Credible Interval

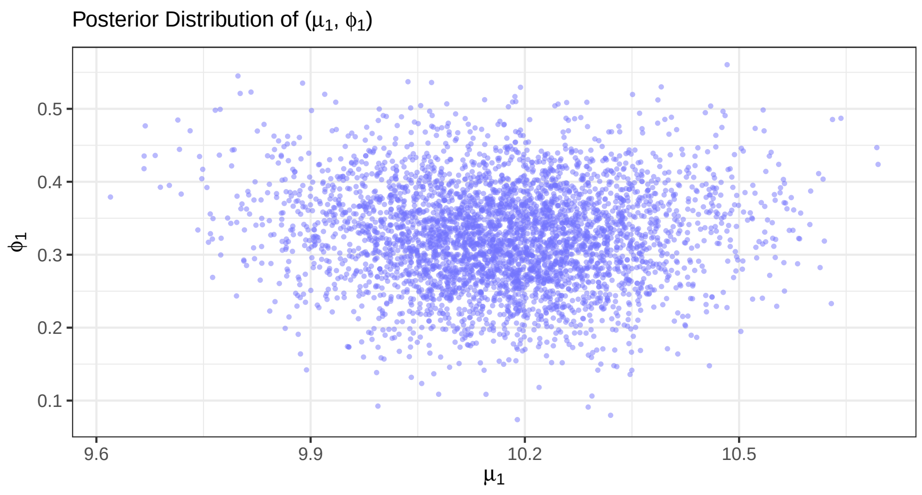
	mean	sd	2.5%	97.5%
mu	10.1696978	0.1501162	9.8682298	10.4743754
phi	0.3250343	0.0687915	0.1897776	0.4652665
sdSq	1.9485131	0.1995242	1.6021056	2.3657323

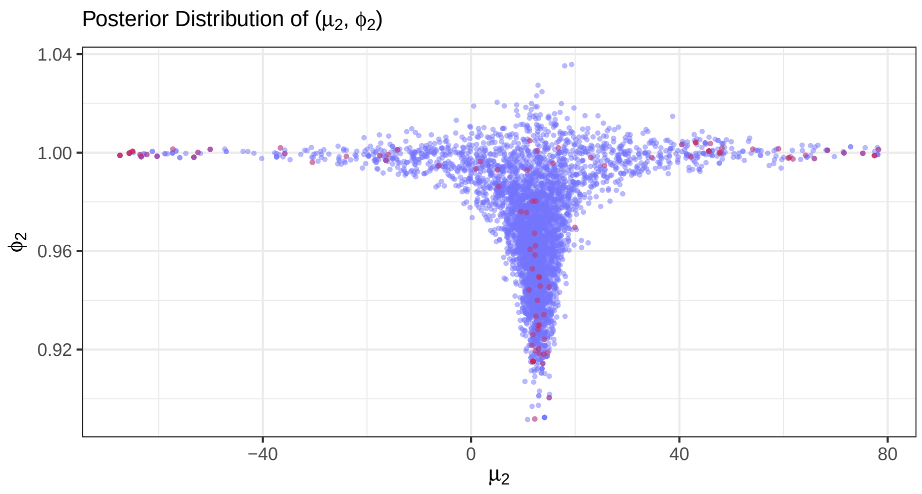
Table 2: y 95% Credible Interval

	mean	sd	2.5%	97.5%
mu	11.9075564	15.4229117	-24.3263274	46.324317
phi	0.9724014	0.0242491	0.9219734	1.007356
sdSq	1.9928782	0.2076022	1.6387584	2.450743

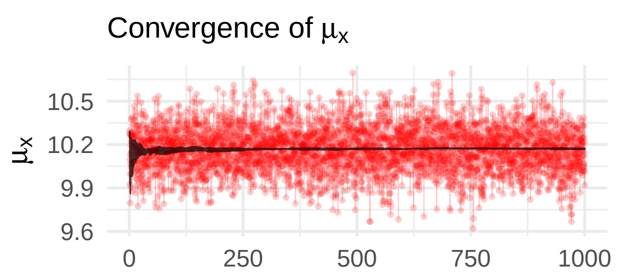
- From the plots and the tables it seems harder to estimate μ_y because it has larger credible interval

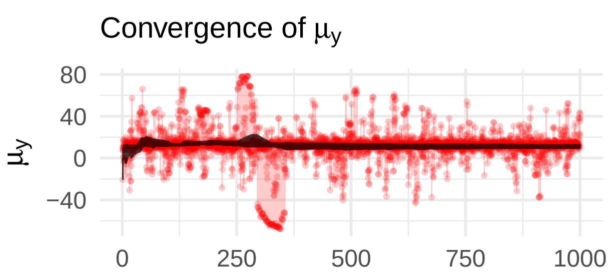
(ii) For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

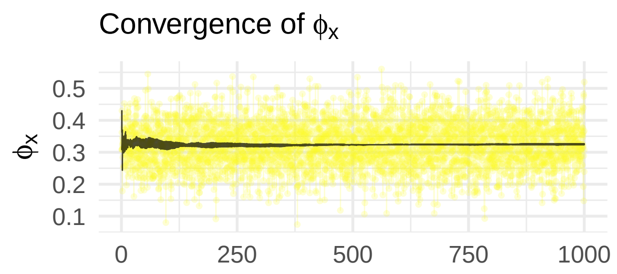


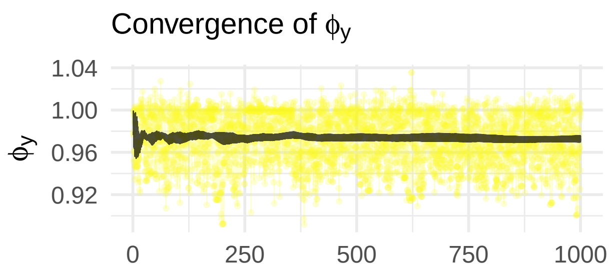


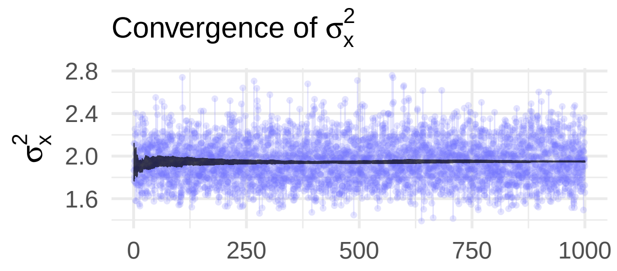
- The red dots in the second plot are the divergent transitions which can be explained by the highly varying posterior curvature causing HMC trajectory to depart from the true trajectory
- In another stan model where we used the constraints '<lower = -1, upper = 1>' as constraints for ϕ, most of the divergent transitions disappeared which also explain the high divergence in this model is because the series variance became too large for the HMC to keep track of

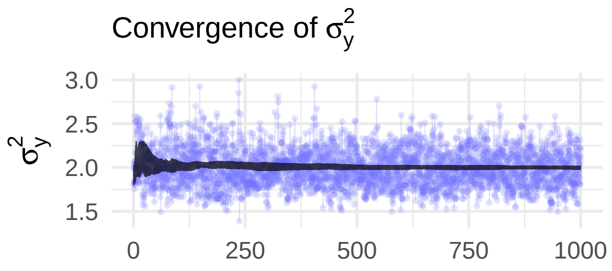

Convergence of μ_x
μ_x
10.5
10.2
9.9
9.6
0
250
500
750
1000

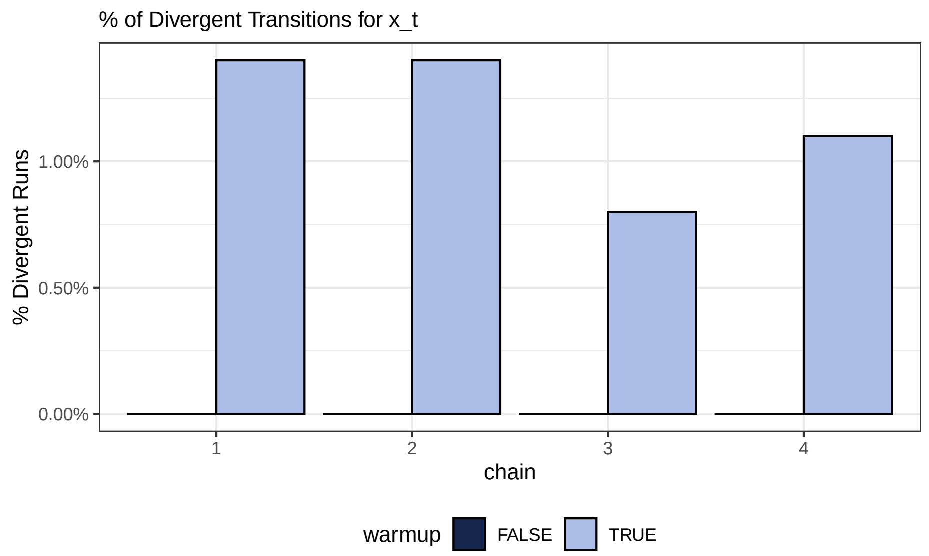

Convergence of μ_y
μ_y
80
40
0
−40
0
250
500
750
1000

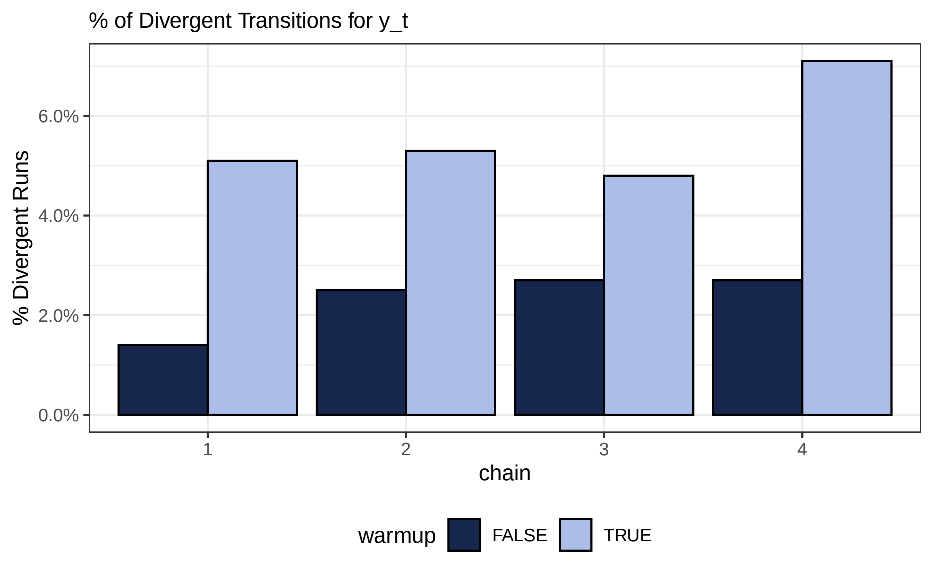

Convergence of ϕ_x
ϕ_x
0.5
0.4
0.3
0.2
0.1
0
250
500
750
1000


Convergence of ϕ_y
ϕ_y
1.04
1.00
0.96
0.92
0
250
500
750
1000


Convergence of σ_x^2
σ_x^2
2.8
2.4
2.0
1.6
0
250
500
750
1000


Convergence of σ_y^2
σ_y^2
3.0
2.5
2.0
1.5
0
250
500
750
1000


% of Divergent Transitions for x_t
% Divergent Runs
1.00%
0.50%
0.00%
1
2
3
4
chain
warmup
FALSE
TRUE

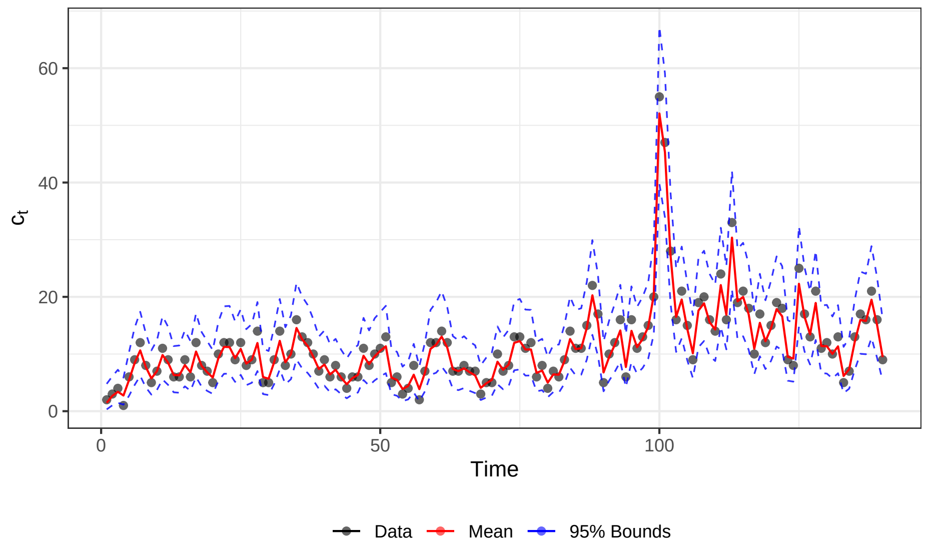


- The percentage of the divergent transtions when $\phi_y = 0.95$ is too high for the posterior estimates to be trusted

(c) The data `campy.dat` contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. Assume that the number of infections c_t at each time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, that is

$$c_t|x_t \sim Poisson(exp(x_t))$$

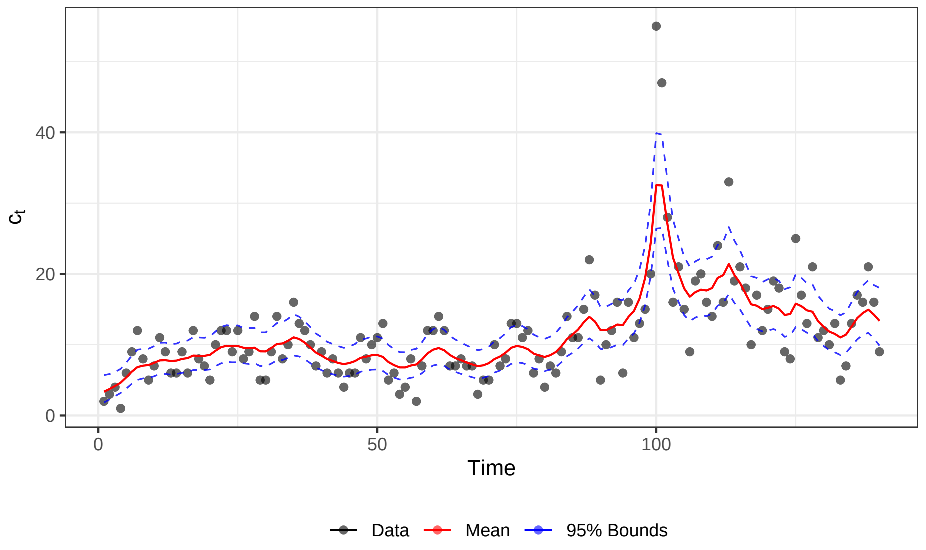
where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice. Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = \exp(x_t)$ over time. [Hint: Should x_t be seen as data or parameters?]



- Here because now we are working on a hierarchical model, Stan will integrate out all the parameters that are conditioned on x_t:

$$p(c_t|x_t) = \int\int\int p\left(c_t, \mu, \phi, \sigma^2|x_t\right) d\mu\, d\phi\, d\sigma^2$$

(d) Now, assume that we have a prior belief that the true underlying intensity θ_t varies more smoothly than the data suggests. Change the prior for σ^2 so that it becomes informative about that the AR(1)-process increments ϵ_t should be small. Re-estimate the model using Stan with the new prior and produce the same plot as in c). Has the posterior for θ_t changed?



- The posterior of c_t has changed because we used a more informative prior that gives a strong belief on where c_t is by reducing and increasing the hyper-parameters (σ^2 and ν respectively) of the model variance.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, fig.align = "center", warning = FALSE, out.width = "70%", fig.height=4
knitr::read_chunk("R_Code.r")
data {
  int<lower=0> N;
  vector[N] x;
}

parameters {
  real mu;
  real phi;
  real<lower=0> sdSq;
}

model {
  mu ~ normal(0, 1000);
  phi ~ uniform(-1000, 1000);
  sdSq ~ uniform(0, 1000000);

  x[2:N] ~ normal(mu + phi * (x[1:(N - 1)] - mu), sqrt(sdSq));
}

data {
  int<lower=0> N;
  int y[N,1];
}

parameters {
  real mu;
  real<lower = -1, upper = 1> phi;
  real<lower=0> sdSq;
  vector[N] x;
}

model {
  mu ~ normal(0, 2);
  phi ~ uniform(-1, 1);
  sdSq ~ scaled_inv_chi_square(1, 5);

  x[2:N] ~ normal(mu + phi * (x[1:(N-1)] - mu), sqrt(sdSq));
  y[1:N,1] ~ poisson(exp(x[1:N]));
}

data {
  int<lower=0> N;
  int y[N,1];
}

parameters {
  real mu;
  real<lower = -1, upper = 1> phi;
  real<lower=0> sdSq;
  vector[N] x;
}

model {
  mu ~ normal(0, 2);
  phi ~ uniform(-1, 1);
  sdSq ~ scaled_inv_chi_square(140, 0.1);

  x[2:N] ~ normal(mu + phi * (x[1:(N-1)] - mu), sqrt(sdSq));
  y[1:N,1] ~ poisson(exp(x[1:N]));
}

#setwd("E:/1. Workshop/11. Bayesian Learning/1. Labs/Lab 4")
setwd("C:/Users/WizzCon/Desktop/Machine Learning/1. Workshop/11. Bayesian Learning/1. Labs/Lab 4")
#################################################################################
## Time series models in Stan.
#################################################################################
## A

phi <- seq(-0.9,0.9, length.out = 9)

AR <- function(mu, sdSq, phi, T){
  X <- data.frame(matrix(NaN, nrow = T, ncol = length(phi)))
  X[1,] <- mu

  for(i in phi){
    for(t in 2:T){
      e <- rnorm(1, mean = 0, sd = sqrt(sdSq))
      X[t,which(phi == i)] <- mu + i * (X[t-1,which(phi == i)] - mu) + e
    }
  }
  return(data.frame(t = 1:T, X))
}

X <- AR(mu = 10, sdSq = 2, phi = phi, T = 200)
#colnames(X) <- c("t", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9")

library(ggplot2)
library("latex2exp")
p1 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,2]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = -0.9$"), x = "t", y = "x") +
  theme_minimal()

p2 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,3]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = -0.675$"), x = "t", y = "x") +
  theme_minimal()

p3 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,4]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = -0.45$"), x = "t", y = "x") +
  theme_minimal()

p4 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,5]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = -0.225$"), x = "t", y = "x") +
  theme_minimal()

p5 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,6]), color = "#16264c", alpha = 0.7, size = 0.4) +
```

```r
  labs(subtitle = TeX("$\\phi = 0$"), x = "t", y = "x") +
  theme_minimal()

p6 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,7]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = 0.225$"), x = "t", y = "x") +
  theme_minimal()

p7 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,8]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = 0.45$"), x = "t", y = "x") +
  theme_minimal()

p8 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,9]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = 0.675$"), x = "t", y = "x") +
  theme_minimal()

p9 <- ggplot(X) +
  geom_line(aes(x = X[,1], y = X[,10]), color = "#16264c", alpha = 0.7, size = 0.4) +
  labs(subtitle = TeX("$\\phi = 0.9$"), x = "t", y = "x") +
  theme_minimal()


library(gridExtra)
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow = 3)
## B_i

X <- AR(mu = 10, sdSq = 2, phi = c(0.3,0.95), T = 200)
suppressPackageStartupMessages(library(rstan))

# This step is optional, but it can result in compiled Stan programs that execute
# much faster than they otherwise would. Simply paste the following into R once

# dotR <- file.path(Sys.getenv("HOME"), ".R")
# if (!file.exists(dotR)) dir.create(dotR)
# M <- file.path(dotR, ifelse(.Platform$OS.type == "windows", "Makevars.win", "Makevars"))
# if (!file.exists(M)) file.create(M)
# cat("\nCXX14FLAGS=-O3 -march=native -mtune=native",
#     if( grepl("^darwin", R.version$os)) "CXX14FLAGS += -arch x86_64 -ftemplate-depth-256" else
#       if (.Platform$OS.type == "windows") "CXX11FLAGS=-O3 -march=corei7 -mtune=corei7" else
#         "CXX14FLAGS += -fPIC",
#     file = M, sep = "\n", append = TRUE)

# If you ever need to change anything with your C++ toolchain configuration, you can execute:
# M <- file.path(Sys.getenv("HOME"), ".R", ifelse(.Platform$OS.type == "windows", "Makevars.win", "Makevar
# file.edit(M)

# if you are using rstan locally on a multicore machine and have plenty of RAM
# to estimate your model in parallel, at this point execute or we can set the number
# of cores in the stan() function
options(mc.cores = parallel::detectCores())

# allows you to automatically save a bare version of a compiled Stan program
# to the hard disk so that it does not need to be recompiled (unless you change it).
rstan_options(auto_write = TRUE)
```

```
n_chains <- 4
n_cores  <- 4
warmups  <- 1000
iters    <- 2000
max_treedepth <- 10
accept_rate   <- 0.8

fit1 <- stan(file = "Other/Stan/AR1.stan", model_name = "AR1", data = list(N = dim(X)[1], x = X[,2]),
             chains = n_chains, cores = n_cores, warmup = warmups, iter = iters,
             control = list(max_treedepth = max_treedepth, adapt_delta = accept_rate))

fit2 <- stan(file = "Other/Stan/AR1.stan", model_name = "AR1", data = list(N = dim(X)[1], x = X[,3]),
             chains = n_chains, cores = n_cores, warmup = warmups, iter = iters,
             control = list(max_treedepth = max_treedepth, adapt_delta = accept_rate))


x_t <- extract(fit1, permuted = TRUE, inc_warmup = FALSE)
# The summaries for the parameters shown by the print method
# are calculated using only post-warmup draws.

y_t <- extract(fit2, permuted = TRUE, inc_warmup = FALSE)

plot(fit1, ci_level = 0.95) +
  labs(subtitle = TeX("$\\x_t$ 95% Credible Interval"), x = "", y = "") +
  theme_bw()


plot(fit2, ci_level = 0.95) +
  labs(subtitle = TeX("$\\y_t$ 95% Credible Interval"), x = "", y = "") +
  theme_bw()

library(knitr)
kable(summary(fit1, pars = c("mu", "phi", "sdSq"), probs = c(0.025, 0.975))$summary[,c(1,3,4,5)],
             caption = "x 95% Credible Interval")
kable(summary(fit2, pars = c("mu", "phi", "sdSq"), probs = c(0.025, 0.975))$summary[,c(1,3,4,5)],
             caption = "y 95% Credible Interval")
## B_ii

getPlotData <- function(model, n_chains, iters, warmups){
  post_warmup <- iters - warmups

  chains <- NULL
  for(chain in 1:n_chains){
    Cumsum <- apply(As.mcmc.list(model)[[chain]], 2, function(x)cumsum(x)/ seq(1,post_warmup))
    colnames(Cumsum) <- c("mu_cumsum", "phi_cumsum", "sdSq_cumsum", "lp_cumsum")
    single_chain <- cbind(xGrid = 1:post_warmup, As.mcmc.list(model)[[chain]], Cumsum,
                          div = get_sampler_params(model, inc_warmup = FALSE)[[chain]][,5])
    chains <- rbind(chains, single_chain)
  }

  return(as.data.frame(chains))
}

x_plotData <- getPlotData(fit1, n_chains, iters, warmups)
y_plotData <- getPlotData(fit2, n_chains, iters, warmups)
x_divDraws   <- data.frame(mu = x_plotData$mu[which(x_plotData$div == 1)],
                         phi = x_plotData$phi[which(x_plotData$div == 1)])
```

```r
y_divDraws   <- data.frame(mu = y_plotData$mu[which(y_plotData$div == 1)],
                           phi = y_plotData$phi[which(y_plotData$div == 1)])


##### Joint Density Plots #####
ggplot(x_plotData)+
  geom_point(aes(x = mu, y = phi), color = "#7475FD", alpha = 0.5, size = 0.6) +
  geom_point(data = x_divDraws, aes(x = mu, y = phi), color = "#F64769", alpha = 0.3, size = 0.7) +
  labs(subtitle = TeX("Posterior Distribution of ($\\mu_1$, $\\phi_1$)"),
       x = TeX("$\\mu_1$"), y = TeX("$\\phi_1$")) +
  theme_bw()

ggplot(y_plotData)+
  geom_point(aes(x = mu, y = phi), colour = "#7475FD", alpha = 0.5, size = 0.6) +
  geom_point(data = y_divDraws, aes(x = mu, y = phi), colour = "red", alpha = 0.3, size = 0.7) +
  labs(subtitle = TeX("Posterior Distribution of ($\\mu_2$, $\\phi_2$)"),
       x = TeX("$\\mu_2$"), y = TeX("$\\phi_2$")) +
  theme_bw()
##### x Convergance Plots #####
gp1 <- ggplot(x_plotData) +
  geom_line(aes(x = xGrid, y = mu), color = "red", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = mu), color = "red", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = mu_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\mu_x$"), x = "", y = TeX('$\\mu_x$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

gp2 <- ggplot(x_plotData) +
  geom_line(aes(x = xGrid, y = phi), color = "#FCFA2922", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = phi), color = "#FCFA2922", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = phi_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\phi_x$"), x = "", y = TeX('$\\phi_x$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

gp3 <- ggplot(x_plotData) +
  geom_line(aes(x = xGrid, y = sdSq), color = "#7475FD", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = sdSq), color = "#7475FD", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = sdSq_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\sigma_x^2$"), x = "", y = TeX('$\\sigma_x^2$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

gp4 <- ggplot(y_plotData) +
  geom_line(aes(x = xGrid, y = mu), color = "red", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = mu), color = "red", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = mu_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\mu_y$"), x = "", y = TeX('$\\mu_y$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

gp5 <- ggplot(y_plotData) +
  geom_line(aes(x = xGrid, y = phi), color = "#FCFA2922", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = phi), color = "#FCFA2922", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = phi_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\phi_y$"), x = "", y = TeX('$\\phi_y$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

gp6 <- ggplot(y_plotData) +
  geom_line(aes(x = xGrid, y = sdSq), color = "#7475FD", alpha = 0.2, size = 0.2) +
  geom_point(aes(x = xGrid, y = sdSq), color = "#7475FD", alpha = 0.2, size = 0.5) +
  geom_line(aes(x = xGrid, y = sdSq_cumsum), color = "black", alpha = 0.7, size = 0.2) +
  labs(subtitle = TeX("Convergence of $\\sigma_y^2$"), x = "", y = TeX('$\\sigma_y^2$')) +
  theme_minimal() +
  theme(legend.position="bottom", legend.title = element_blank())

grid.arrange(gp1, gp4, gp2, gp5, gp3, gp6, ncol = 2)
# The following code is taken from this blog with slight adjustment
# https://www.weirdfishes.blog/blog/fitting-bayesian-models-with-stan-and-r/
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(dplyr))
diagnostics_1 <- get_sampler_params(fit1) %>%
  set_names(1:n_chains) %>%
  map_df(as_tibble,.id = 'chain') %>%
  group_by(chain) %>%
  mutate(iteration = 1:length(chain)) %>%
  mutate(warmup = iteration <= warmups)

diagnostics_1 %>%
  group_by(warmup, chain) %>%
  summarise(percent_divergent = mean(divergent__ >0)) %>%
  ggplot() +
  geom_col(aes(chain, percent_divergent, fill = warmup),
           position = 'dodge', color = 'black') +
  scale_y_continuous(labels = scales::percent, name = "% Divergent Runs") +
  scale_fill_manual(values = c("#16264c", "#ACBEE7")) +
  labs(subtitle = "% of Divergent Transitions for x_t") +
  theme_bw() +
  theme(legend.position="bottom")

diagnostics_2 <- get_sampler_params(fit2) %>%
  set_names(1:n_chains) %>%
  map_df(as_tibble,.id = 'chain') %>%
  group_by(chain) %>%
  mutate(iteration = 1:length(chain)) %>%
  mutate(warmup = iteration <= warmups)

diagnostics_2 %>%
  group_by(warmup, chain) %>%
  summarise(percent_divergent = mean(divergent__ >0)) %>%
  ggplot() +
  geom_col(aes(chain, percent_divergent, fill = warmup),
           position = 'dodge', color = 'black') +
  scale_y_continuous(labels = scales::percent, name = "% Divergent Runs") +
  scale_fill_manual(values = c("#16264c", "#ACBEE7")) +
  labs(subtitle = "% of Divergent Transitions for y_t") +
  theme_bw() +
  theme(legend.position="bottom")
## C

campy <- read.table("Other/Data/campy.dat", header=TRUE)

n_chains <- 4
```

```r
n_cores  <- 4
warmups  <- 1000
iters    <- 4000
max_treedepth <- 10
accept_rate   <- 0.8

ct_fit <- stan(file = "Other/Stan/AR2.stan", model_name = "AR2", data = list(N = dim(campy)[1], y = campy)
               chains = n_chains, cores = n_cores, warmup = warmups, iter = iters,
               control = list(max_treedepth = max_treedepth, adapt_delta = accept_rate))



ct <- extract(ct_fit)
ct_summ <- summary(ct_fit)$summary

mean_ci <- exp(summary(ct_fit, probs = c(0.025, 0.975))$summary[c(4:143),c(1,4,5)])
ctPlotData <- data.frame(Time = 1:nrow(campy),
                         Data  = campy$c,
                         Mean  = mean_ci[,1],
                         Lower = mean_ci[,2],
                         Upper = mean_ci[,3])

ggplot(ctPlotData, aes(x = Time)) +
  geom_point(aes(y = Data, col = "Data"), alpha = 0.6) +
  geom_line(aes(y = Mean, col = "Mean"), size = 0.5) +
  geom_line(aes(y = Lower, col = "95% Bounds"), lty = 2, size = 0.4, alpha = 0.8) +
  geom_line(aes(y = Upper), color = "blue", lty = 2, size = 0.4, alpha = 0.8) +
  scale_color_manual(breaks = c("Data", "Mean", "95% Bounds"),
                     values = c("black", "red", "blue")) +
  labs(y = TeX("$\\c_t$")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())
## D

ct2_fit <- stan(file = "Other/Stan/AR3.stan", model_name = "AR3", data = list(N = dim(campy)[1], y = campy
               chains = n_chains, cores = n_cores, warmup = warmups, iter = iters,
               control = list(max_treedepth = max_treedepth, adapt_delta = accept_rate))



ct2 <- extract(ct2_fit)
ct2_summ <- summary(ct2_fit)$summary

mean_ci_2 <- exp(summary(ct2_fit, probs = c(0.025, 0.975))$summary[c(4:143),c(1,4,5)])
ct2PlotData <- data.frame(Time = 1:nrow(campy),
                          Data  = campy$c,
                          Mean  = mean_ci_2[,1],
                          Lower = mean_ci_2[,2],
                          Upper = mean_ci_2[,3])

ggplot(ct2PlotData, aes(x = Time)) +
  geom_point(aes(y = Data, col = "Data"), alpha = 0.6) +
  geom_line(aes(y = Mean, col = "Mean"), size = 0.5) +
  geom_line(aes(y = Lower, col = "95% Bounds"), lty = 2, size = 0.4, alpha = 0.8) +
  geom_line(aes(y = Upper), color = "blue", lty = 2, size = 0.4, alpha = 0.8) +
  scale_color_manual(breaks = c("Data", "Mean", "95% Bounds"),
                     values = c("black", "red", "blue")) +
  labs(y = TeX("$\\c_t$")) +
  theme_bw() +
  theme(legend.position="bottom", legend.title = element_blank())
```