homework12

Homework 12

yc3356 Yi Chen

1. Review the following case study, focusing on the model:

http://mc-stan.org/users/documentation/case-studies/lotka-volterra-predator-prey.html#data-lynx-and-hare-pelts-in-canada (http://mc-stan.org/users/documentation/case-studies/lotka-volterra-predator-prey.html#data-lynx-and-hare-pelts-in-canada)

```
# load the packages
library(rstan)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.18.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(ggplot2)
library(gridExtra)
library(knitr)
library(reshape)
library(tufte)
library(bayesplot)
```

```
## This is bayesplot version 1.6.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(deSolve)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

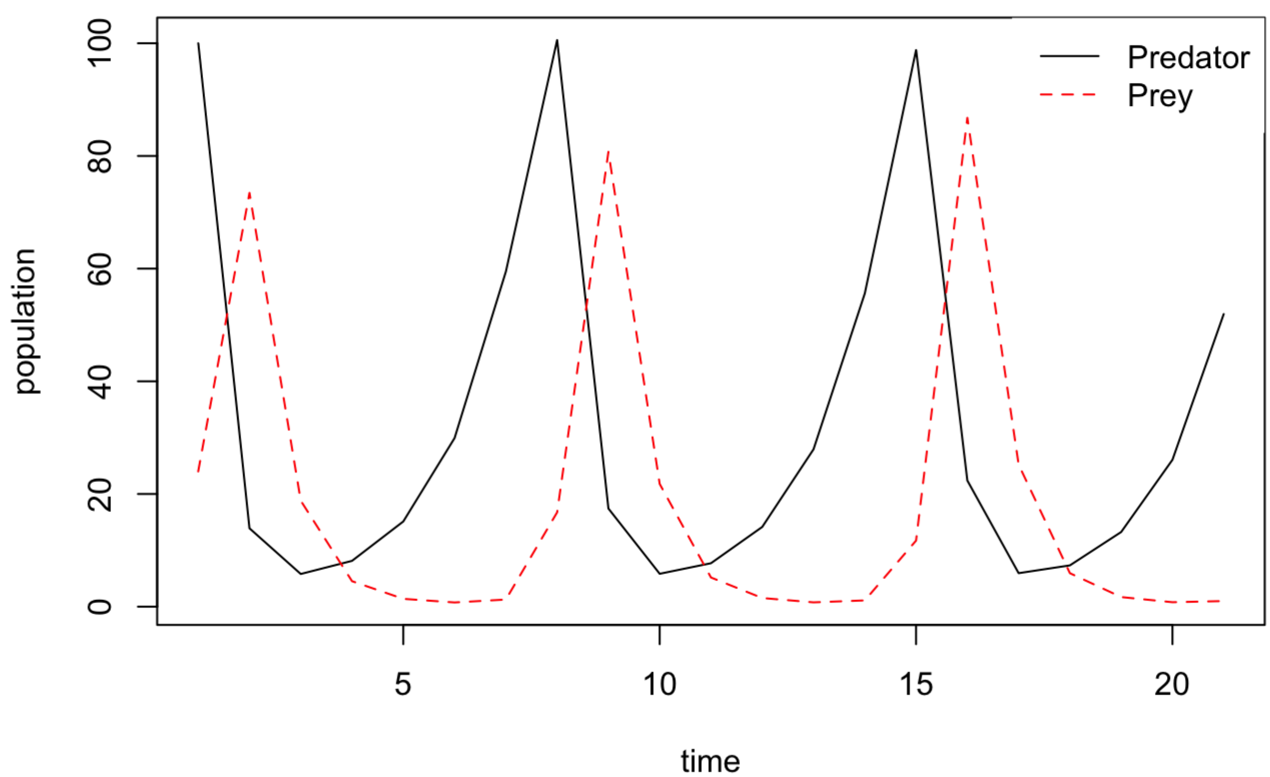
a. Simulate fake data and check that the model recovers the parameters. Feel free to simplify the model as necessary.

```
# simulate the fake data based on the machanistic model

set.seed(123)
LotVmod <- function (Time, State, Pars) {
    with(as.list(c(State, Pars)), {
        dx = x*(alpha - beta*y)
        dy = -y*(gamma - delta*x)
        return(list(c(dx, dy)))})
}

Pars <- c(alpha = rnorm(1,1,0.5), beta = rnorm(1,0.05, 0.05), gamma = rnorm(1,1,0.5), de
lta = rnorm(1,0.05, 0.05))
State <- c(x = 100, y = 24)
Time <- seq(0, 20, by = 1)
out <- as.data.frame(ode(func = LotVmod, y = State, parms = Pars, times = Time))

matplot(out[,-1], type = "l", xlab = "time", ylab = "population")
legend("topright", c("Predator", "Prey"), lty = c(1,2), col = c(1,2), box.lwd = 0)
```



```r
writeLines(readLines("lotka-volterra.stan"))
```

```
## Warning in readLines("lotka-volterra.stan"): incomplete final line found on
## 'lotka-volterra.stan'
```

```
## functions {
##   real[] dz_dt(real t,       // time
##                real[] z,     // system state {prey, predator}
##                real[] theta, // parameters
##                real[] x_r,   // unused data
##                int[] x_i) {
##     real u = z[1];
##     real v = z[2];
##
##     real alpha = theta[1];
##     real beta = theta[2];
##     real gamma = theta[3];
##     real delta = theta[4];
##
##     real du_dt = (alpha - beta * v) * u;
##     real dv_dt = (-gamma + delta * u) * v;
##     return { du_dt, dv_dt };
##   }
## }
## data {
##   int<lower = 0> N;           // number of measurement times
##   real ts[N];                 // measurement times > 0
##   real y_init[2];             // initial measured populations
##   real<lower = 0> y[N, 2];    // measured populations
## }
## parameters {
##   real<lower = 0> theta[4];   // { alpha, beta, gamma, delta }
##   real<lower = 0> z_init[2];  // initial population
##   real<lower = 0> sigma[2];   // measurement errors
## }
## transformed parameters {
##   real z[N, 2]
##     = integrate_ode_rk45(dz_dt, z_init, 0, ts, theta,
##                          rep_array(0.0, 0), rep_array(0, 0),
##                          1e-5, 1e-3, 5e2);
## }
## model {
##   theta[{1, 3}] ~ normal(1, 0.5);
##   theta[{2, 4}] ~ normal(0.05, 0.05);
##   sigma ~ lognormal(-1, 1);
##   z_init ~ lognormal(log(10), 1);
##   for (k in 1:2) {
##     y_init[k] ~ lognormal(log(z_init[k]), sigma[k]);
##     y[ , k] ~ lognormal(log(z[, k]), sigma[k]);
##   }
## }
## generated quantities {
##   real y_init_rep[2];
##   real y_rep[N, 2];
##   for (k in 1:2) {
##     y_init_rep[k] = lognormal_rng(log(z_init[k]), sigma[k]);
##     for (n in 1:N)
##       y_rep[n, k] = lognormal_rng(log(z[n, k]), sigma[k]);
```

```
##   }
## }
```

```
set.seed(123)
y <- out[,c('x','y')]
fake_data <- list(N=nrow(out), ts = 1:nrow(out), y_init = c(out$x[1],out$y[2]), y=y)
comp_model_f <- stan_model('lotka-volterra.stan')
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/
## Users/yi/Desktop/study/subjects/bayesian-data-analysis/homework/homework17/
## lotka-volterra.stan'
```

```
fit_model_f <- sampling(comp_model_f, data = fake_data, seed = 123)
fit_model_f
```

```
## Inference for Stan model: lotka-volterra.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                mean se_mean    sd  2.5%    25%    50%    75%  97.5%
## theta[1]       0.80    0.00  0.04  0.73   0.77   0.80   0.82   0.88
## theta[2]       0.05    0.00  0.01  0.03   0.04   0.05   0.06   0.08
## theta[3]       1.46    0.00  0.11  1.25   1.39   1.46   1.54   1.69
## theta[4]       0.04    0.00  0.00  0.04   0.04   0.04   0.04   0.05
## z_init[1]     73.49    0.09  3.62 66.81  70.98  73.32  75.74  81.12
## z_init[2]      2.61    0.01  0.53  1.74   2.24   2.56   2.93   3.82
## sigma[1]       0.10    0.00  0.02  0.07   0.09   0.10   0.11   0.14
## sigma[2]       0.82    0.00  0.13  0.62   0.73   0.80   0.89   1.12
## z[1,1]        95.34    0.10  5.76 83.96  91.55  95.39  99.24 106.92
## z[1,2]        32.91    0.15  6.98 21.84  27.99  31.99  36.93  49.38
## z[2,1]        12.83    0.02  0.94 10.99  12.21  12.81  13.44  14.68
## z[2,2]        53.71    0.35 13.24 33.37  44.54  51.62  60.92  84.42
## z[3,1]         5.95    0.00  0.21  5.54   5.80   5.94   6.08   6.37
## z[3,2]        17.09    0.07  3.21 11.79  14.84  16.70  18.99  24.32
## z[4,1]         8.16    0.01  0.33  7.54   7.94   8.15   8.37   8.84
## z[4,2]         5.24    0.02  0.88  3.74   4.63   5.17   5.77   7.22
## z[5,1]        15.48    0.01  0.55 14.43  15.11  15.46  15.83  16.59
## z[5,2]         1.97    0.01  0.36  1.37   1.72   1.93   2.17   2.79
## z[6,1]        32.03    0.02  0.99 30.12  31.35  32.00  32.67  34.02
## z[6,2]         1.20    0.01  0.24  0.81   1.03   1.18   1.34   1.76
## z[7,1]        66.46    0.07  2.65 61.47  64.71  66.35  68.17  71.85
## z[7,2]         2.09    0.01  0.41  1.42   1.80   2.04   2.33   3.03
## z[8,1]       102.93    0.07  4.72 93.49  99.85 102.89 105.97 112.64
## z[8,2]        21.76    0.08  3.93 15.24  18.99  21.34  24.08  30.49
## z[9,1]        16.98    0.02  0.92 15.18  16.37  16.97  17.56  18.80
## z[9,2]        60.81    0.41 15.34 37.28  50.09  58.50  69.18  96.41
## z[10,1]        6.03    0.00  0.23  5.60   5.88   6.03   6.18   6.49
## z[10,2]       20.41    0.09  3.88 14.02  17.70  19.95  22.69  29.26
## z[11,1]        7.56    0.01  0.28  7.04   7.38   7.55   7.73   8.14
## z[11,2]        6.18    0.02  1.02  4.45   5.46   6.09   6.78   8.40
## z[12,1]       13.97    0.01  0.45 13.09  13.68  13.97  14.26  14.87
## z[12,2]        2.22    0.01  0.40  1.56   1.95   2.18   2.45   3.11
## z[13,1]       28.74    0.01  0.76 27.25  28.23  28.73  29.23  30.27
## z[13,2]        1.23    0.01  0.24  0.83   1.06   1.21   1.37   1.80
## z[14,1]       59.93    0.05  2.19 55.82  58.47  59.83  61.35  64.49
## z[14,2]        1.74    0.01  0.34  1.18   1.51   1.71   1.94   2.56
## z[15,1]      103.84    0.08  4.33 95.52 100.98 103.83 106.59 112.62
## z[15,2]       14.11    0.05  2.59  9.76  12.28  13.86  15.67  19.99
## z[16,1]       23.60    0.03  1.83 20.23  22.35  23.48  24.73  27.47
## z[16,2]       66.55    0.46 17.13 40.27  54.60  63.95  75.63 106.76
## z[17,1]        6.29    0.00  0.26  5.79   6.12   6.28   6.45   6.81
## z[17,2]       24.37    0.11  4.85 16.39  20.99  23.80  27.10  35.17
## z[18,1]        7.06    0.01  0.26  6.55   6.89   7.06   7.23   7.59
## z[18,2]        7.32    0.02  1.23  5.24   6.47   7.20   8.06  10.05
## z[19,1]       12.65    0.01  0.48 11.71  12.33  12.65  12.96  13.60
## z[19,2]        2.53    0.01  0.45  1.77   2.22   2.50   2.80   3.54
## z[20,1]       25.80    0.01  0.90 23.99  25.22  25.80  26.39  27.59
## z[20,2]        1.29    0.01  0.25  0.87   1.11   1.26   1.43   1.87
```

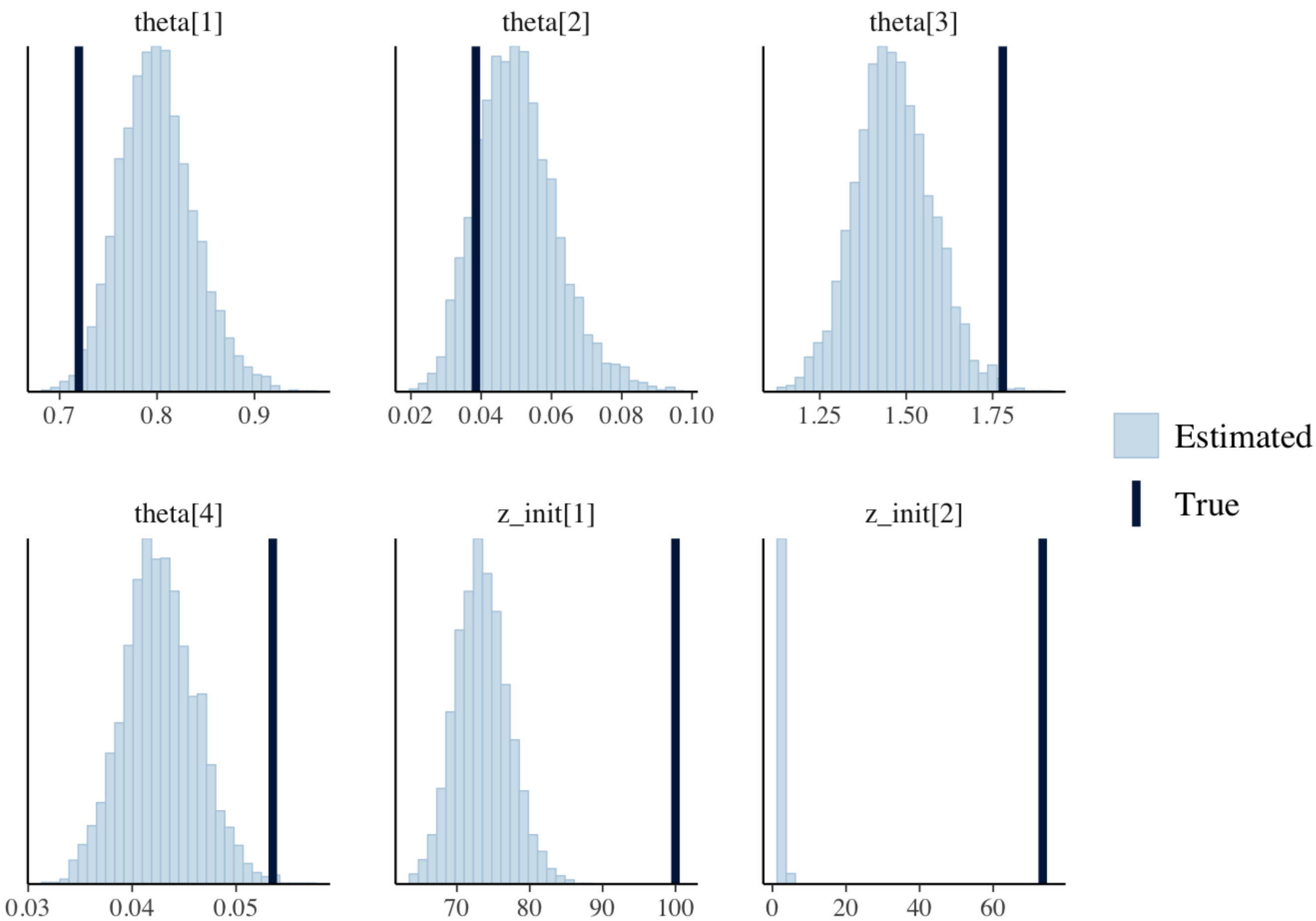
```
## z[21,1]         53.95    0.04   2.28 49.67  52.40  53.88  55.46  58.50
## z[21,2]          1.52    0.01   0.31  1.02   1.31   1.48   1.69   2.24
## y_init_rep[1]   73.78    0.15   8.37 59.02  68.05  73.23  78.88  91.84
## y_init_rep[2]    3.68    0.07   4.36  0.44   1.48   2.57   4.31  13.89
## y_rep[1,1]      95.75    0.20  11.53 74.96  87.97  95.10 102.86 120.67
## y_rep[1,2]      46.64    1.00  62.00  5.90  18.19  32.39  56.21 166.29
## y_rep[2,1]      12.92    0.03   1.60 10.00  11.84  12.85  13.90  16.29
## y_rep[2,2]      76.28    1.46  88.91  9.04  29.71  51.46  92.60 280.44
## y_rep[3,1]       5.99    0.01   0.65  4.78   5.57   5.95   6.38   7.39
## y_rep[3,2]      24.11    0.44  27.96  3.19   9.60  16.47  28.61  91.99
## y_rep[4,1]       8.20    0.01   0.88  6.63   7.60   8.15   8.76  10.08
## y_rep[4,2]       7.15    0.13   7.68  0.98   2.93   4.99   8.67  25.15
## y_rep[5,1]      15.58    0.03   1.67 12.59  14.45  15.50  16.62  19.20
## y_rep[5,2]       2.78    0.06   3.68  0.36   1.08   1.89   3.32  10.45
## y_rep[6,1]      32.18    0.06   3.34 26.12  29.88  31.96  34.25  39.39
## y_rep[6,2]       1.68    0.03   2.07  0.22   0.68   1.18   2.01   6.03
## y_rep[7,1]      66.85    0.13   7.21 53.53  61.95  66.56  71.31  81.91
## y_rep[7,2]       2.95    0.05   3.17  0.37   1.19   2.06   3.63  10.92
## y_rep[8,1]     103.24    0.18  11.51 82.07  95.65 102.70 110.05 128.36
## y_rep[8,2]      30.92    0.56  34.93  4.28  12.33  21.44  36.97 108.51
## y_rep[9,1]      17.05    0.03   1.96 13.52  15.69  16.94  18.22  21.24
## y_rep[9,2]      87.00    1.99 124.49 11.00  32.73  58.85 103.46 310.35
## y_rep[10,1]      6.04    0.01   0.66  4.87   5.60   6.00   6.45   7.49
## y_rep[10,2]     27.82    0.48  29.90  3.59  11.32  19.83  33.79 102.07
## y_rep[11,1]      7.60    0.01   0.82  6.12   7.05   7.56   8.10   9.36
## y_rep[11,2]      8.66    0.15   9.02  1.24   3.61   6.10  10.20  31.90
## y_rep[12,1]     14.00    0.02   1.48 11.27  13.03  13.92  14.88  17.13
## y_rep[12,2]      3.12    0.06   3.35  0.42   1.24   2.16   3.78  11.77
## y_rep[13,1]     28.88    0.05   2.99 23.51  26.87  28.72  30.66  34.99
## y_rep[13,2]      1.80    0.03   2.01  0.24   0.72   1.22   2.17   6.66
## y_rep[14,1]     60.17    0.11   6.44 48.32  55.90  59.84  64.06  73.63
## y_rep[14,2]      2.49    0.05   2.96  0.33   0.98   1.72   2.99   9.14
## y_rep[15,1]    104.71    0.18  11.56 83.64  96.99 104.11 111.92 128.95
## y_rep[15,2]     19.71    0.35  21.28  2.56   7.83  13.66  24.11  72.62
## y_rep[16,1]     23.74    0.05   3.07 18.28  21.66  23.56  25.60  30.48
## y_rep[16,2]     93.13    1.66 103.01 11.69  37.09  65.08 112.23 341.64
## y_rep[17,1]      6.32    0.01   0.69  5.06   5.86   6.29   6.74   7.80
## y_rep[17,2]     34.48    0.65  38.95  4.65  13.88  24.23  41.95 122.64
## y_rep[18,1]      7.09    0.01   0.76  5.72   6.56   7.04   7.55   8.77
## y_rep[18,2]     10.34    0.20  12.74  1.33   4.11   7.12  12.20  38.84
## y_rep[19,1]     12.73    0.02   1.35 10.27  11.82  12.65  13.53  15.67
## y_rep[19,2]      3.56    0.07   3.89  0.45   1.41   2.45   4.33  13.54
## y_rep[20,1]     25.91    0.04   2.74 20.67  24.10  25.78  27.59  31.51
## y_rep[20,2]      1.80    0.03   2.06  0.24   0.72   1.25   2.11   6.62
## y_rep[21,1]     53.97    0.09   5.75 43.28  50.10  53.83  57.57  66.33
## y_rep[21,2]      2.16    0.04   2.64  0.29   0.84   1.48   2.55   7.88
## lp__            23.45    0.06   2.14 18.40  22.25  23.77  25.02  26.63
##                n_eff Rhat
## theta[1]        1009    1
## theta[2]        1448    1
## theta[3]        1003    1
## theta[4]         995    1
## z_init[1]       1643    1
## z_init[2]       2130    1
```

```
## sigma[1]       2301    1
## sigma[2]       2787    1
## z[1,1]         3630    1
## z[1,2]         2164    1
## z[2,1]         3268    1
## z[2,2]         1438    1
## z[3,1]         3998    1
## z[3,2]         2104    1
## z[4,1]         2487    1
## z[4,2]         2889    1
## z[5,1]         2943    1
## z[5,2]         2389    1
## z[6,1]         2858    1
## z[6,2]         2060    1
## z[7,1]         1536    1
## z[7,2]         2145    1
## z[8,1]         4321    1
## z[8,2]         2327    1
## z[9,1]         3217    1
## z[9,2]         1415    1
## z[10,1]        3637    1
## z[10,2]        1955    1
## z[11,1]        2293    1
## z[11,2]        2748    1
## z[12,1]        2185    1
## z[12,2]        2373    1
## z[13,1]        4119    1
## z[13,2]        2060    1
## z[14,1]        1827    1
## z[14,2]        2207    1
## z[15,1]        3168    1
## z[15,2]        2776    1
## z[16,1]        4047    1
## z[16,2]        1408    1
## z[17,1]        3078    1
## z[17,2]        1885    1
## z[18,1]        2424    1
## z[18,2]        2640    1
## z[19,1]        2290    1
## z[19,2]        2376    1
## z[20,1]        4455    1
## z[20,2]        2049    1
## z[21,1]        2784    1
## z[21,2]        2253    1
## y_init_rep[1]  3108    1
## y_init_rep[2]  3396    1
## y_rep[1,1]     3462    1
## y_rep[1,2]     3835    1
## y_rep[2,1]     3614    1
## y_rep[2,2]     3693    1
## y_rep[3,1]     3999    1
## y_rep[3,2]     3956    1
## y_rep[4,1]     3894    1
## y_rep[4,2]     3692    1
```

```
## y_rep[5,1]      3643    1
## y_rep[5,2]      3846    1
## y_rep[6,1]      3525    1
## y_rep[6,2]      3669    1
## y_rep[7,1]      3129    1
## y_rep[7,2]      3459    1
## y_rep[8,1]      4019    1
## y_rep[8,2]      3884    1
## y_rep[9,1]      3929    1
## y_rep[9,2]      3897    1
## y_rep[10,1]     4236    1
## y_rep[10,2]     3855    1
## y_rep[11,1]     3758    1
## y_rep[11,2]     3736    1
## y_rep[12,1]     3767    1
## y_rep[12,2]     3566    1
## y_rep[13,1]     4145    1
## y_rep[13,2]     3909    1
## y_rep[14,1]     3155    1
## y_rep[14,2]     3853    1
## y_rep[15,1]     3999    1
## y_rep[15,2]     3717    1
## y_rep[16,1]     4235    1
## y_rep[16,2]     3871    1
## y_rep[17,1]     3954    1
## y_rep[17,2]     3588    1
## y_rep[18,1]     3077    1
## y_rep[18,2]     3871    1
## y_rep[19,1]     4094    1
## y_rep[19,2]     3563    1
## y_rep[20,1]     3888    1
## y_rep[20,2]     3614    1
## y_rep[21,1]     3837    1
## y_rep[21,2]     4012    1
## lp__            1201    1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 25 18:27:23 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
posterior_alpha_beta <- as.matrix(as.matrix(fit_model_f, pars = c('theta','z_init')))
true_alpha_beta <- c(Pars,c(out$x[1],out$y[2]))
mcmc_recover_hist(posterior_alpha_beta, true = true_alpha_beta)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



For my fake data:

The inite value did not cover very well. But the most the thetas can cover in the posterior thetas. The Rhat of all the parameters works very well.

b. In two or three sentences, discuss the strengths and weaknesses of the model. How might the model be expanded?

The model works very well. Here are just some extension I will try in this model: The model assume that the population is only depend on the Predator-Prey Population. No other systematic effect exist. However, this may not true make the sigma term not the real normal. I will try to add a offset term to measure all the other effects.

2.

a. Fit the model to the real data and perform model checking and/or validation (Chapters 6 and 7 of BDA).

```
lynx_hare_df <-  read.csv("hudson-bay-lynx-hare.csv",comment.char="#")
N <- length(lynx_hare_df$Year) - 1
ts <- 1:N
y_init <- c(lynx_hare_df$Hare[1], lynx_hare_df$Lynx[1])
y <- as.matrix(lynx_hare_df[2:(N + 1), 2:3])
y <- cbind(y[ , 2], y[ , 1]); # hare, lynx order
lynx_hare_data <- list(N, ts, y_init, y)
model <- stan_model("lotka-volterra.stan")
```

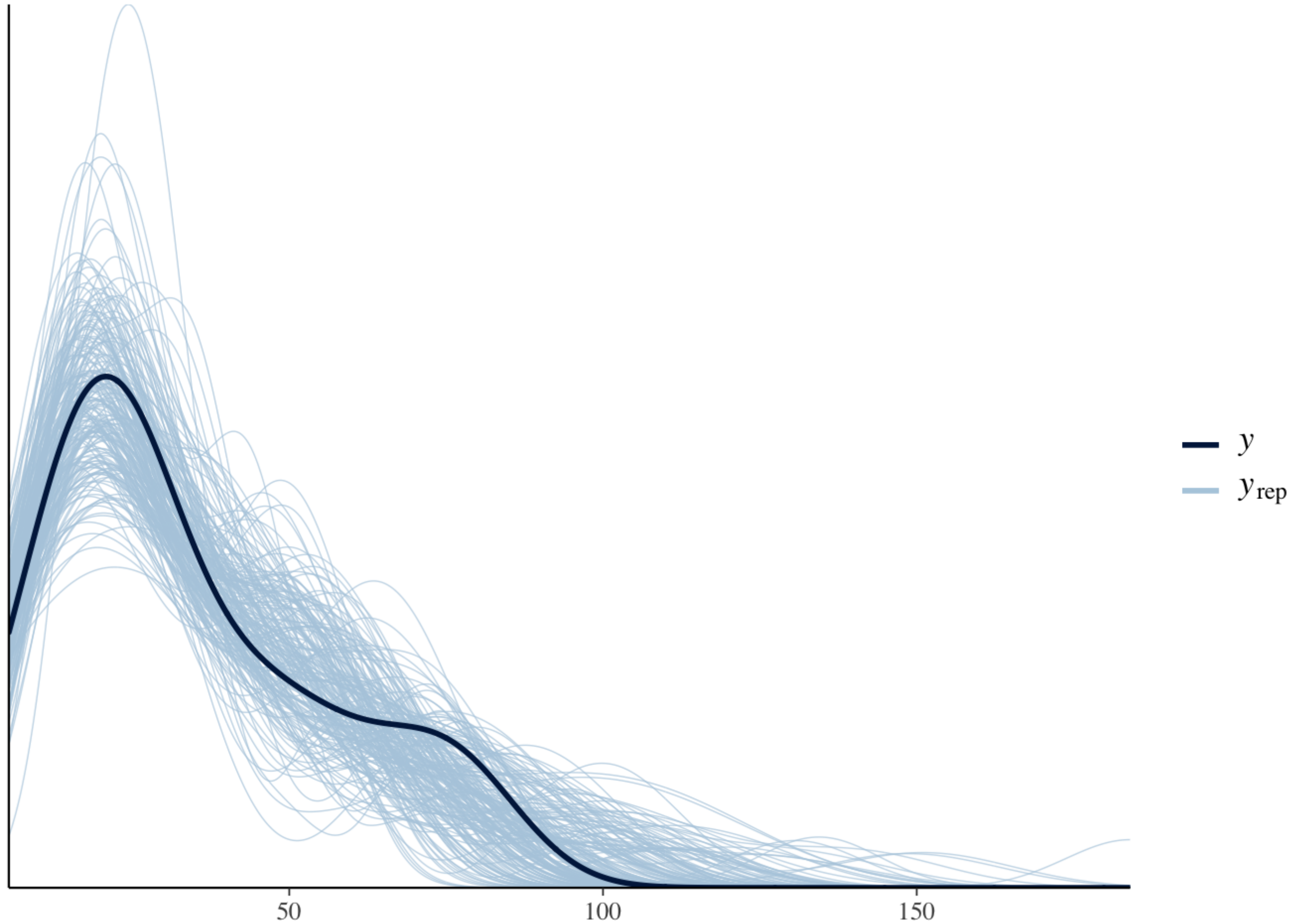
```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/
## Users/yi/Desktop/study/subjects/bayesian-data-analysis/homework/homework17/
## lotka-volterra.stan'
```

```
fit <- sampling(model, data = lynx_hare_data, seed = 123)
print(fit, pars=c("theta", "sigma", "z_init"), probs=c(0.1, 0.5, 0.9), digits = 3)
```

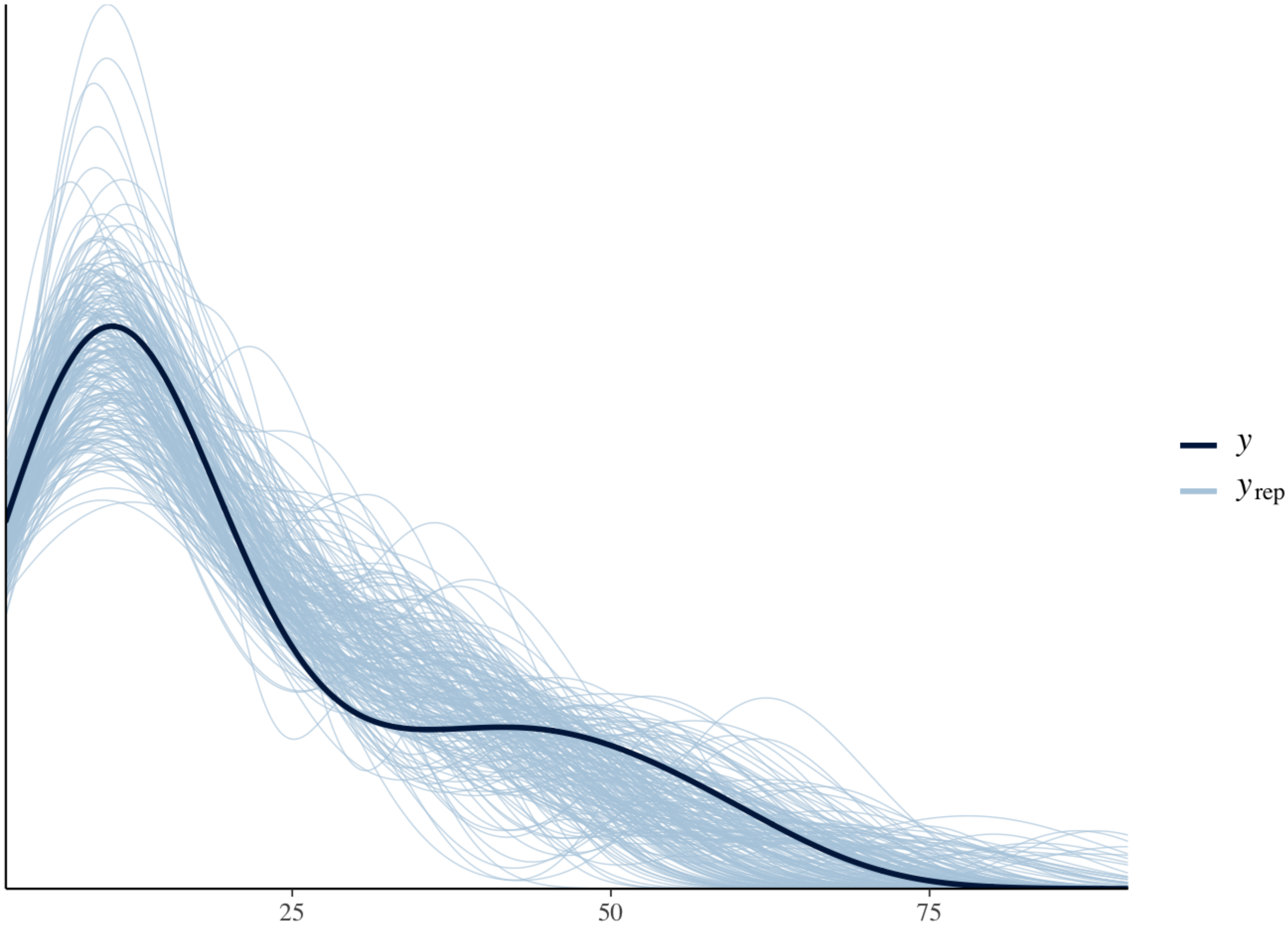
```
## Inference for Stan model: lotka-volterra.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean    sd    10%    50%    90% n_eff  Rhat
## theta[1]   0.545   0.002 0.064  0.465  0.542  0.630  1076 1.002
## theta[2]   0.028   0.000 0.004  0.022  0.027  0.033  1195 1.001
## theta[3]   0.803   0.003 0.092  0.692  0.797  0.926   993 1.002
## theta[4]   0.024   0.000 0.004  0.020  0.024  0.029  1062 1.001
## sigma[1]   0.250   0.001 0.045  0.200  0.243  0.307  2537 1.001
## sigma[2]   0.252   0.001 0.044  0.200  0.245  0.309  2692 1.000
## z_init[1] 33.956   0.057 2.856 30.415 33.908 37.630  2474 1.000
## z_init[2]  5.933   0.012 0.535  5.273  5.912  6.614  2095 1.002
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 25 18:28:01 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
y_rep_1 <- as.matrix(as.matrix(fit, pars = "y_rep")[1:200,1:20])
y_rep_2 <- as.matrix(as.matrix(fit, pars = "y_rep")[1:200,21:40])

ppc_dens_overlay(y = y[,1], yrep = y_rep_1)
```

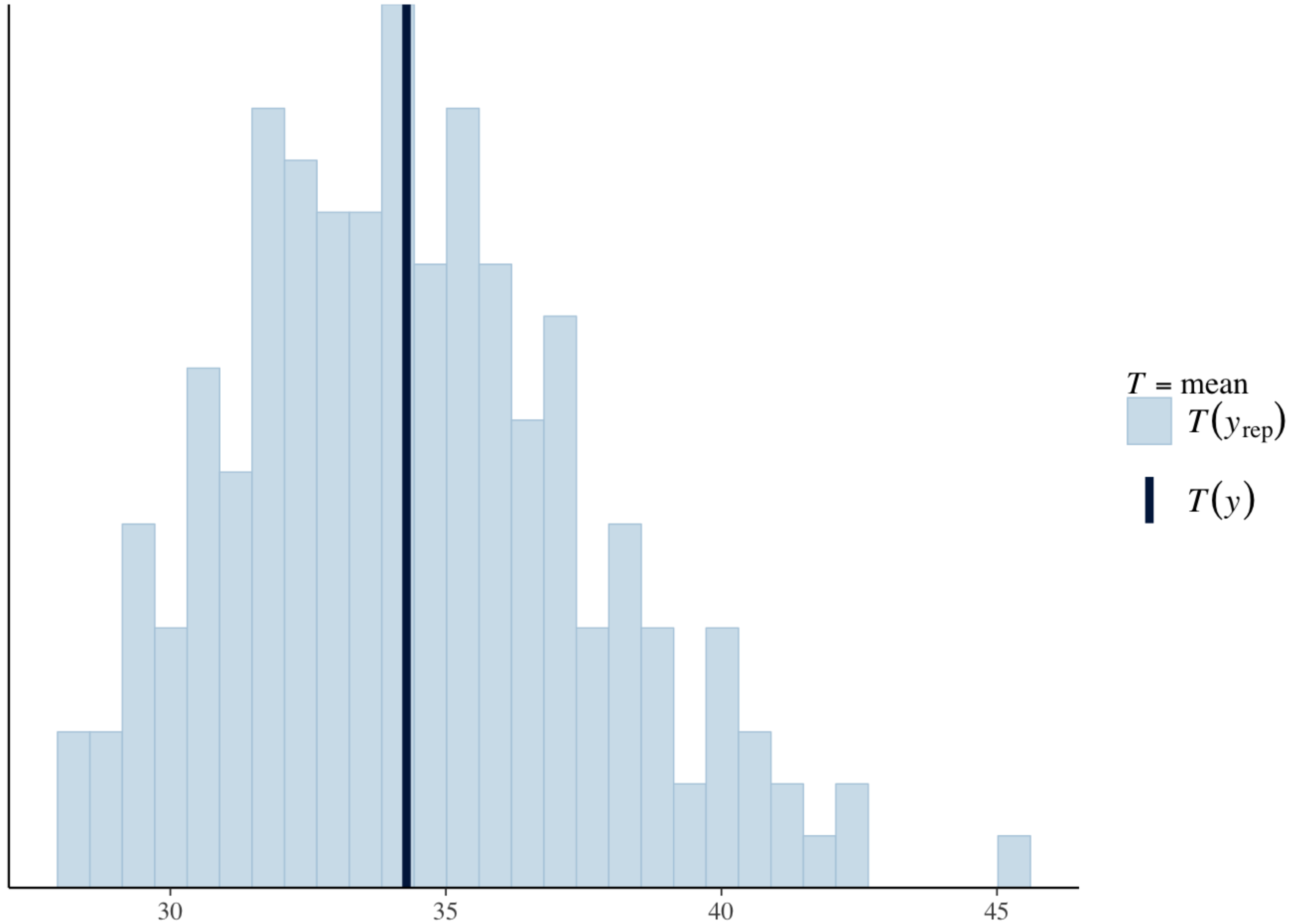



```
ppc_dens_overlay(y = y[,2], yrep = y_rep_2)
```



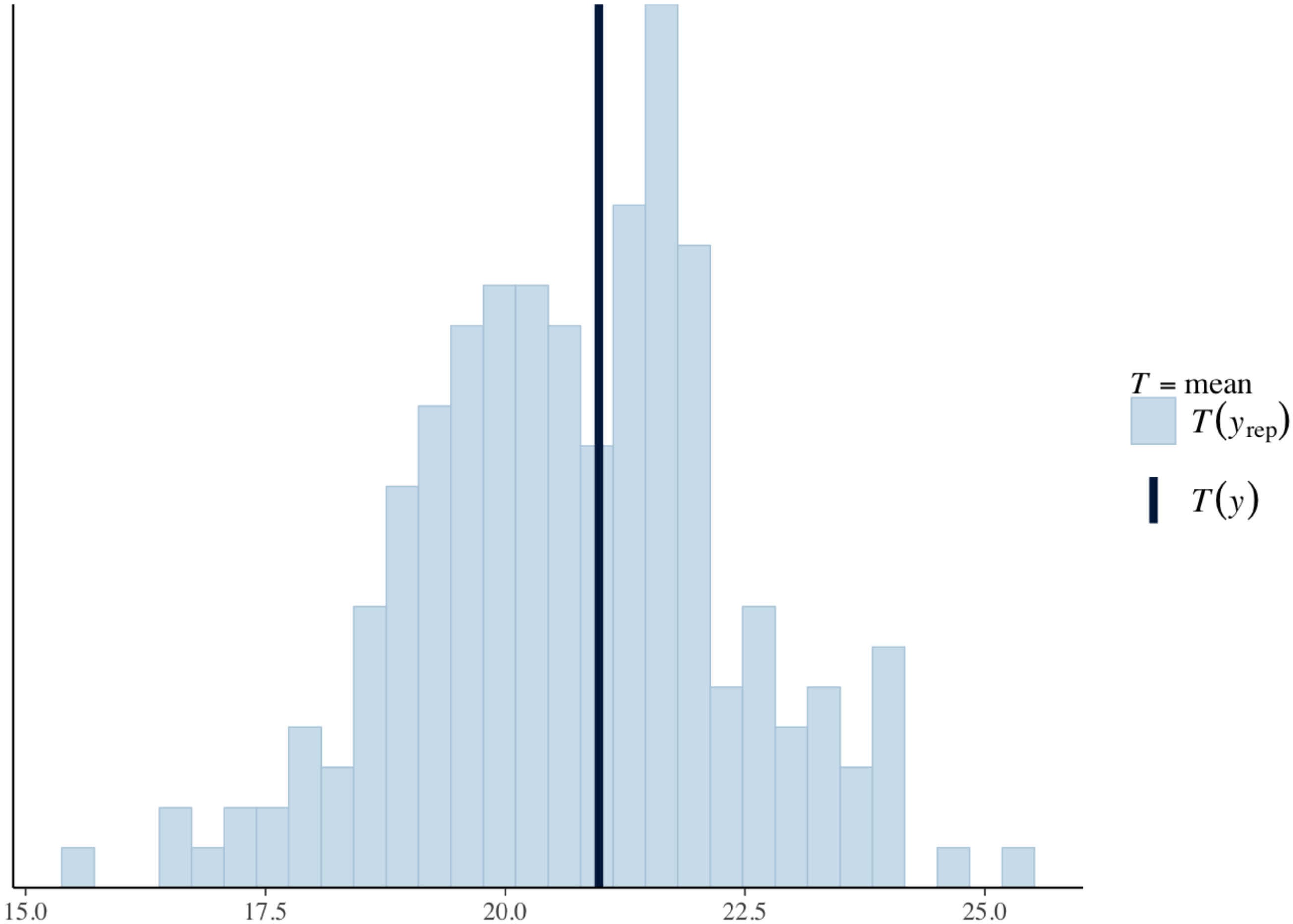

```
ppc_stat(y = y[,1], yrep = y_rep_1, stat = "mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



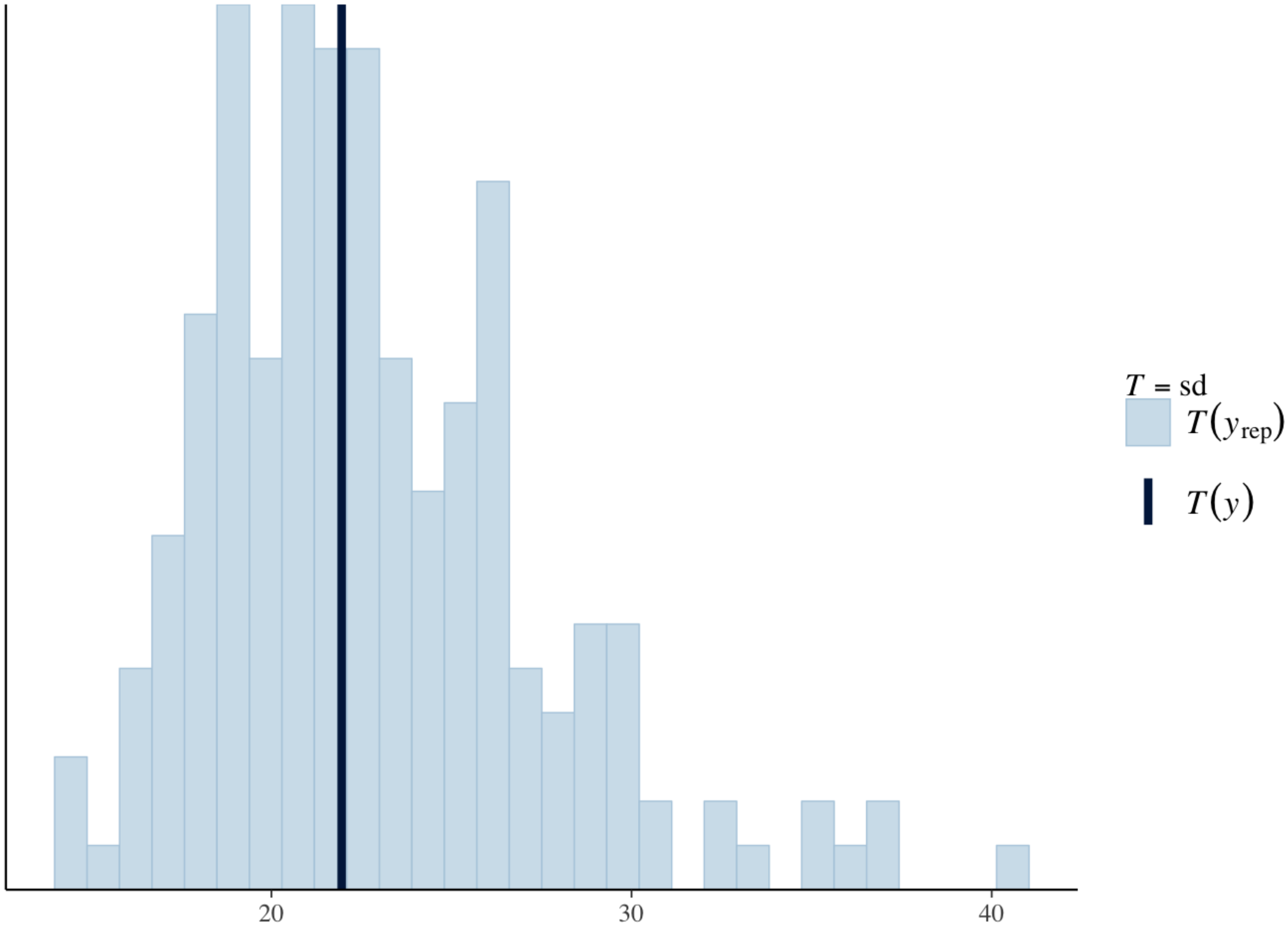

```
ppc_stat(y = y[,2], yrep = y_rep_2, stat = "mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



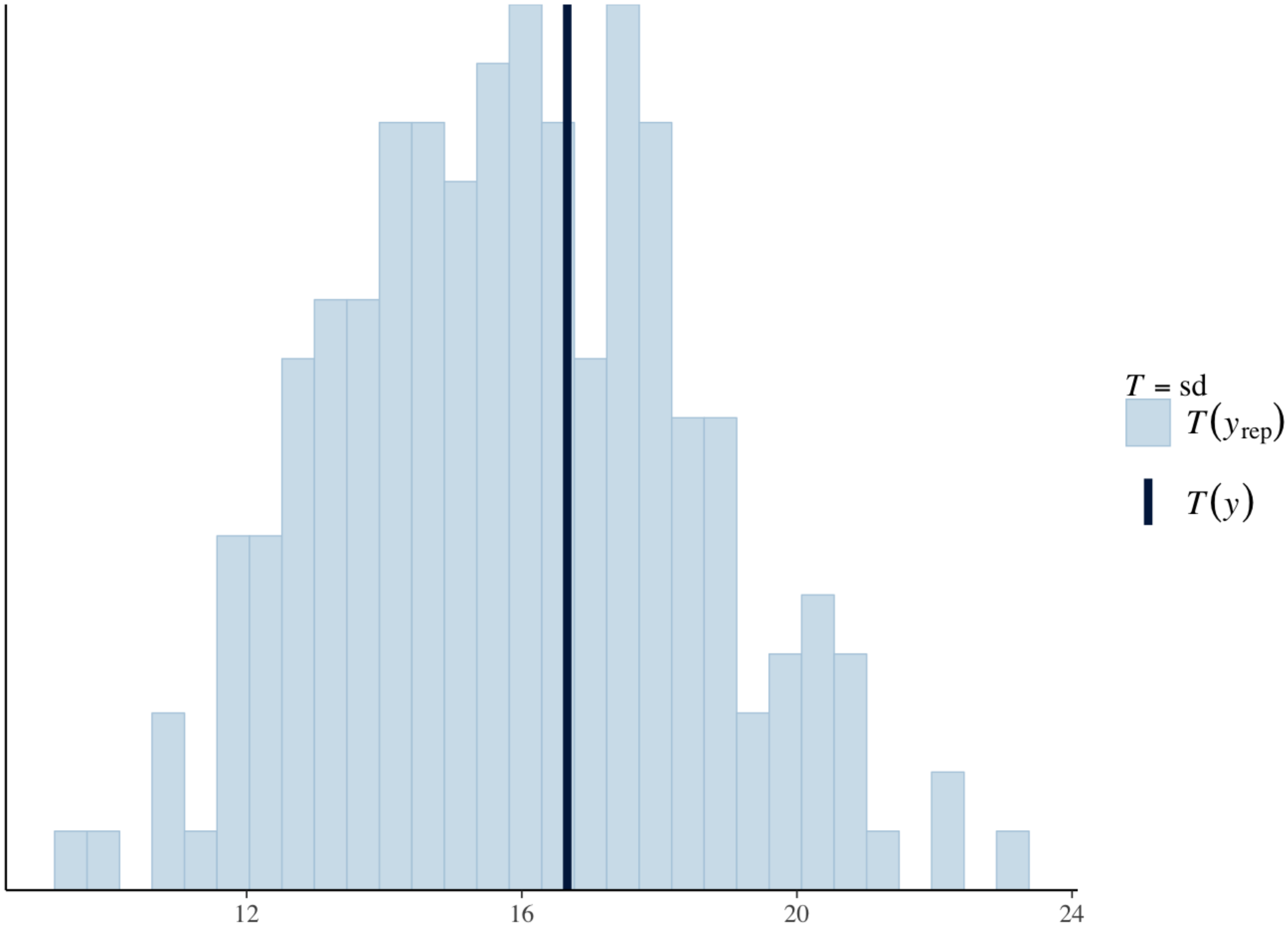

```
ppc_stat(y = y[,1], yrep = y_rep_1, stat = "sd")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

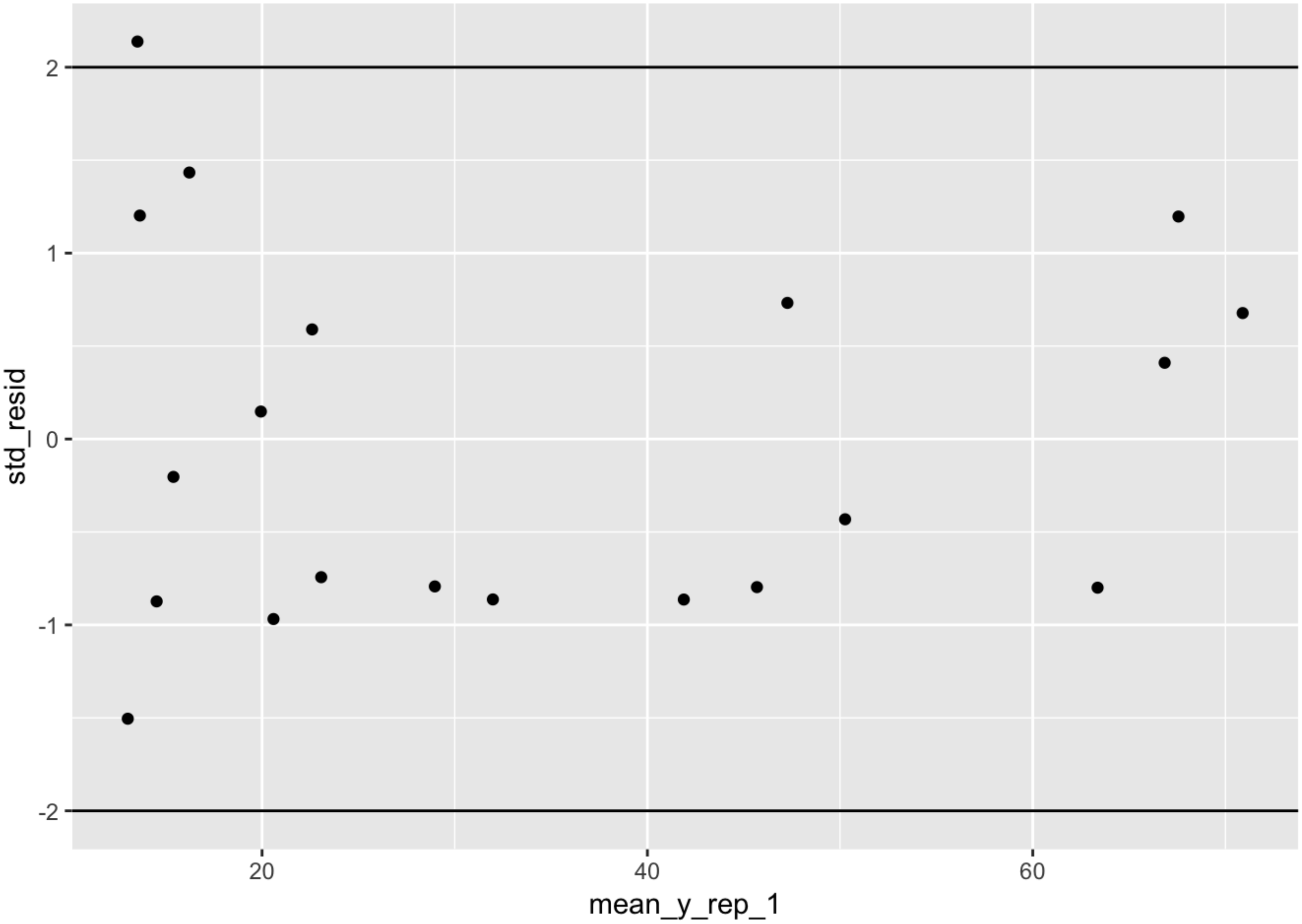



```
ppc_stat(y = y[,2], yrep = y_rep_2, stat = "sd")
```

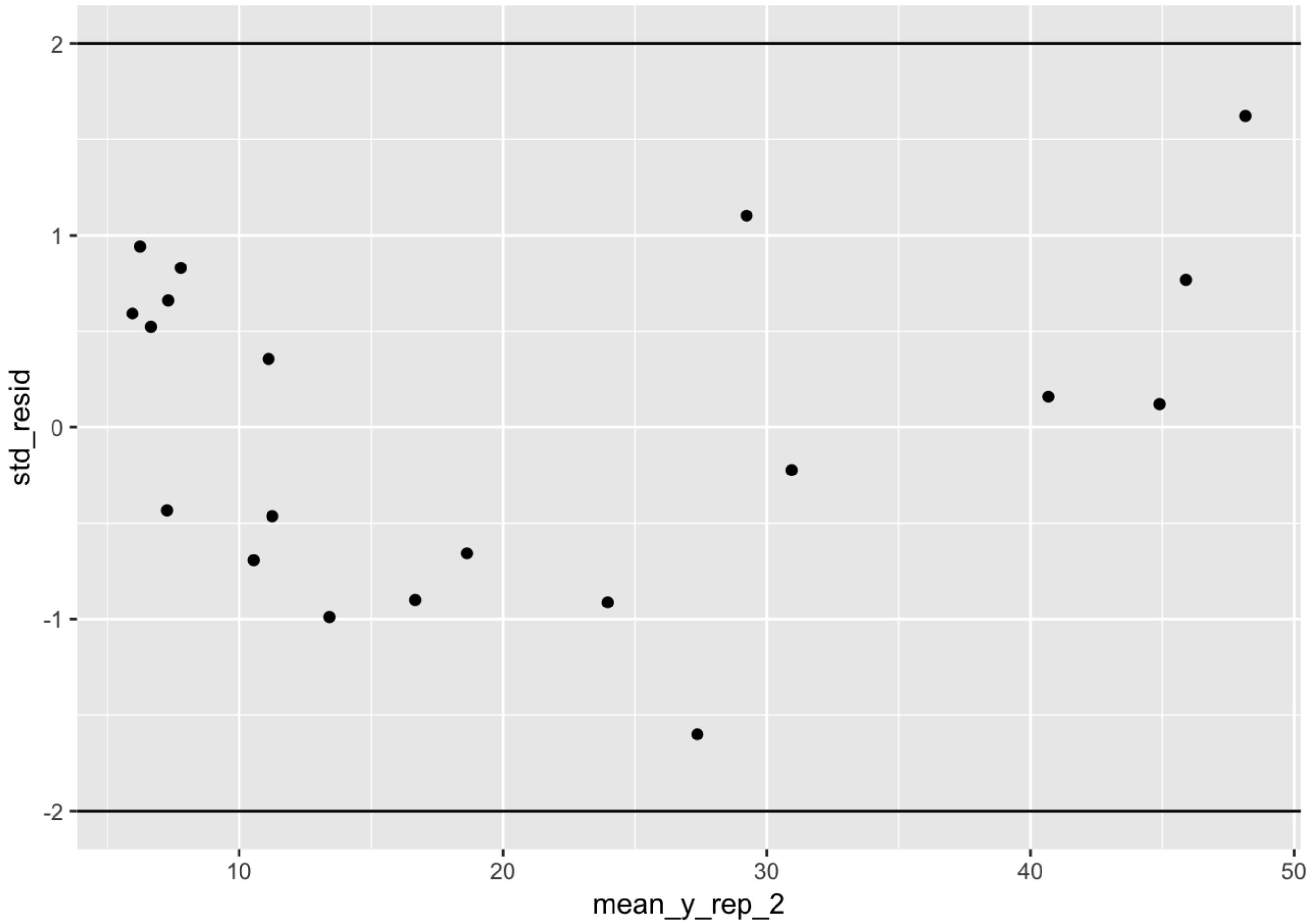
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mean_y_rep_1 <- colMeans(y_rep_1)
std_resid <- (y[,1] - mean_y_rep_1) / sqrt(mean_y_rep_1)
qplot(mean_y_rep_1, std_resid) + hline_at(2) + hline_at(-2)
```




```
mean_y_rep_2 <- colMeans(y_rep_2)
std_resid <- (y[,2] - mean_y_rep_2) / sqrt(mean_y_rep_2)
qplot(mean_y_rep_2, std_resid) + hline_at(2) + hline_at(-2)
```



b. Expand the model as discussed in 1.b./class and interpret the results.

```
## for the fake data
set.seed(123)
y <- out[,c('x','y')]
fake_data <- list(N=nrow(out), ts = 1:nrow(out), y_init = c(out$x[1],out$y[2]), y=y)

comp_model_f <- stan_model('model_extend.stan')
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/
## Users/yi/Desktop/study/subjects/bayesian-data-analysis/homework/homework17/
## model_extend.stan'
```

```
## hash mismatch so recompiling; make sure Stan code ends with a blank line
```

```
fit_model_f <- sampling(comp_model_f, data = fake_data, seed = 1234)
```

```
## Warning in validityMethod(object): The following variables have undefined
## values: y_init_rep[1],The following variables have undefined values:
## y_init_rep[2],The following variables have undefined values: y_rep[1,1],The
## following variables have undefined values: y_rep[2,1],The following
## variables have undefined values: y_rep[3,1],The following variables have
## undefined values: y_rep[4,1],The following variables have undefined values:
## y_rep[5,1],The following variables have undefined values: y_rep[6,1],The
## following variables have undefined values: y_rep[7,1],The following
## variables have undefined values: y_rep[8,1],The following variables have
## undefined values: y_rep[9,1],The following variables have undefined values:
## y_rep[10,1],The following variables have undefined values: y_rep[11,1],The
## following variables have undefined values: y_rep[12,1],The following
## variables have undefined values: y_rep[13,1],The following variables
## have undefined values: y_rep[14,1],The following variables have undefined
## values: y_rep[15,1],The following variables have undefined values:
## y_rep[16,1],The following variables have undefined values: y_rep[17,1],The
## following variables have undefined values: y_rep[18,1],The following
## variables have undefined values: y_rep[19,1],The following variables
## have undefined values: y_rep[20,1],The following variables have undefined
## values: y_rep[21,1],The following variables have undefined values:
## y_rep[1,2],The following variables have undefined values: y_rep[2,2],The
## following variables have undefined values: y_rep[3,2],The following
## variables have undefined values: y_rep[4,2],The following variables have
## undefined values: y_rep[5,2],The following variables have undefined values:
## y_rep[6,2],The following variables have undefined values: y_rep[7,2],The
## following variables have undefined values: y_rep[8,2],The following
## variables have undefined values: y_rep[9,2],The following variables have
## undefined values: y_rep[10,2],The following variables have undefined
## values: y_rep[11,2],The following variables have undefined values:
## y_rep[12,2],The following variables have undefined values: y_rep[13,2],The
## following variables have undefined values: y_rep[14,2],The following
## variables have undefined values: y_rep[15,2],The following variables
## have undefined values: y_rep[16,2],The following variables have undefined
## values: y_rep[17,2],The following variables have undefined values:
## y_rep[18,2],The following variables have undefined values: y_rep[19,2],The
## following variables have undefined values: y_rep[20,2],The following
## variables have undefined values: y_rep[21,2]. Many subsequent functions
## will not work correctly.
```

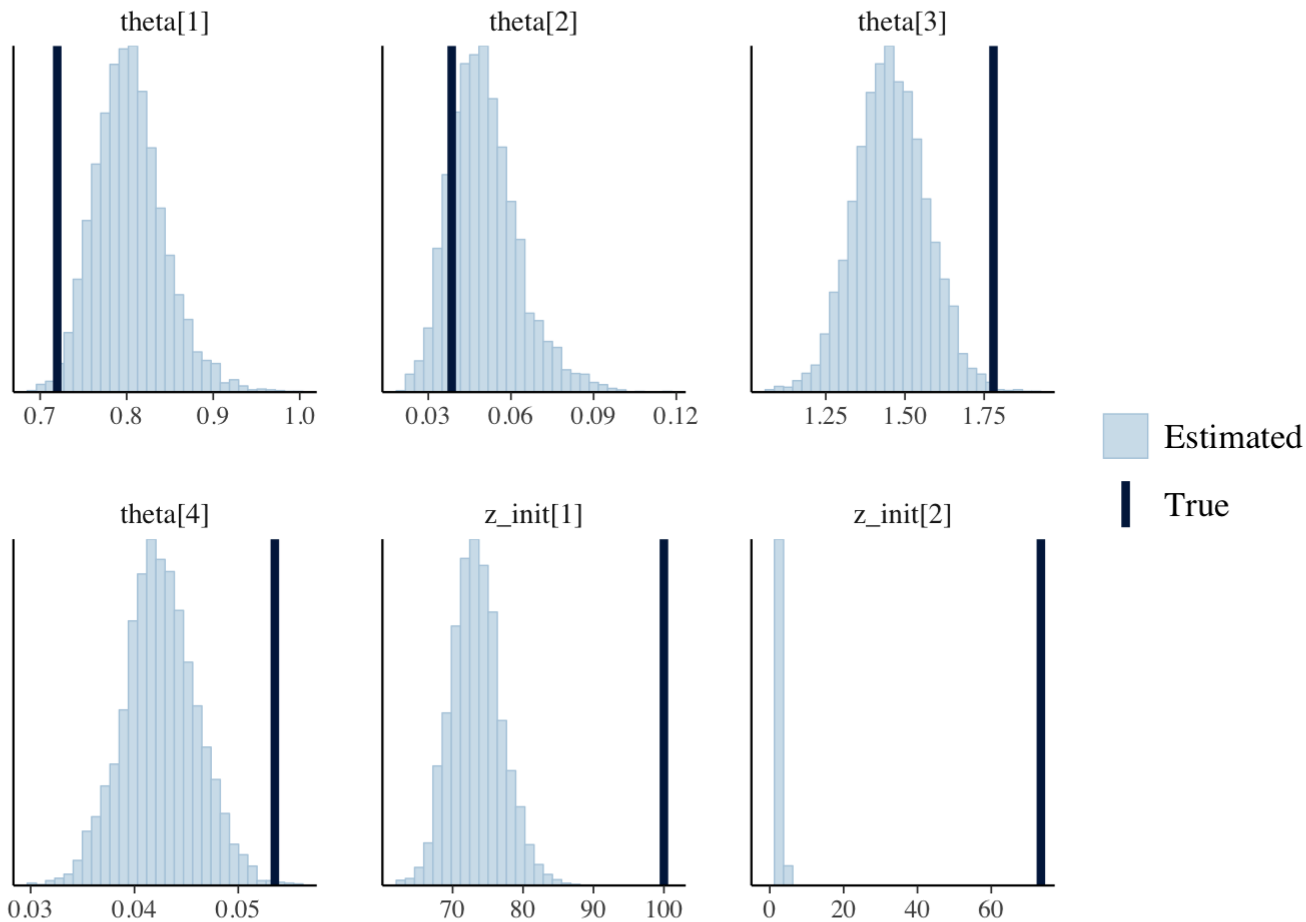
```
## Warning: There were 1496 divergent transitions after warmup. Increasing adapt_delta a
bove 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 517 transitions after warmup that exceeded the maximum treedepth.
Increase max_treedepth above 10. See
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
posterior_alpha_beta <- as.matrix(as.matrix(fit_model_f, pars = c('theta','z_init')))
true_alpha_beta <- c(Pars,c(out$x[1],out$y[2]))
mcmc_recover_hist(posterior_alpha_beta, true = true_alpha_beta)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
writeLines(readLines("model_extend.stan"))
```

```
## Warning in readLines("model_extend.stan"): incomplete final line found on
## 'model_extend.stan'
```

```
## functions {
##   real[] dz_dt(real t,       // time
##                real[] z,     // system state {prey, predator}
##                real[] theta, // parameters
##                real[] x_r,   // unused data
##                int[] x_i) {
##     real u = z[1];
##     real v = z[2];
##
##     real alpha = theta[1];
##     real beta = theta[2];
##     real gamma = theta[3];
##     real delta = theta[4];
##
##     real du_dt = (alpha - beta * v) * u;
##     real dv_dt = (-gamma + delta * u) * v;
##     return { du_dt, dv_dt };
##   }
## }
## data {
##   int<lower = 0> N;            // number of measurement times
##   real ts[N];                  // measurement times > 0
##   real y_init[2];              // initial measured populations
##   real<lower = 0> y[N, 2];     // measured populations
## }
## parameters {
##   real<lower = 0> theta[4];   // { alpha, beta, gamma, delta }
##   real<lower = 0> z_init[2];  // initial population
##   real<lower = 0> sigma[2];   // measurement errors
##   vector[N] offset;
## }
## transformed parameters {
##   real z[N, 2]
##     = integrate_ode_rk45(dz_dt, z_init, 0, ts, theta,
##                          rep_array(0.0, 0), rep_array(0, 0),
##                          1e-5, 1e-3, 5e2);
##   for (k in 1:2) {
##     for (n in 1:N){
##            z[N, 2] = z[N, 2] + offset[n];
##     }
##   }
## }
## model {
##   theta[{1, 3}] ~ normal(1, 0.5);
##   theta[{2, 4}] ~ normal(0.05, 0.05);
##   sigma ~ lognormal(-1, 1);
##   z_init ~ lognormal(log(10), 1);
##   offset ~ cauchy(0,1);
##   for (k in 1:2) {
##     y_init[k] ~ lognormal(log(z_init[k]), sigma[k]);
##     y[ , k] ~ lognormal(log(z[, k]), sigma[k]);
##   }
## }
```

```
## generated quantities {
##   real y_init_rep[2];
##   real y_rep[N, 2];
##
##   for (k in 1:2) {
##     y_init_rep[k] = lognormal_rng(log(z_init[k]) , sigma[k]);
##     for (n in 1:N)
##       y_rep[n, k] = lognormal_rng(log(z[n, k]), sigma[k]);
##   }
## }
```

```
lynx_hare_df <-  read.csv("hudson-bay-lynx-hare.csv",comment.char="#")
N <- length(lynx_hare_df$Year) - 1
ts <- 1:N
y_init <- c(lynx_hare_df$Hare[1], lynx_hare_df$Lynx[1])
y <- as.matrix(lynx_hare_df[2:(N + 1), 2:3])
y <- cbind(y[ , 2], y[ , 1]) # hare, lynx order
lynx_hare_data <- list(N, ts, y_init, y)
model <- stan_model("model_extend.stan")
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/
## Users/yi/Desktop/study/subjects/bayesian-data-analysis/homework/homework17/
## model_extend.stan'
```

```
fit <- sampling(model, data = lynx_hare_data, seed = 123)
```

```
## Warning: There were 43 divergent transitions after warmup. Increasing adapt_delta abo
ve 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 7 transitions after warmup that exceeded the maximum treedepth. I
ncrease max_treedepth above 10. See
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(fit,  digits = 3)
```

```
## Inference for Stan model: model_extend.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                 mean se_mean    sd    2.5%    25%    50%    75%   97.5%
## theta[1]       0.533   0.001 0.060   0.420  0.493  0.530  0.571   0.660
## theta[2]       0.027   0.000 0.004   0.020  0.024  0.027  0.029   0.036
## theta[3]       0.823   0.002 0.091   0.661  0.763  0.817  0.878   1.025
## theta[4]       0.025   0.000 0.004   0.019  0.022  0.024  0.027   0.033
## z_init[1]     34.234   0.047 2.825  28.877 32.298 34.197 36.090  39.867
## z_init[2]      5.694   0.009 0.501   4.791  5.353  5.669  6.009   6.770
## sigma[1]       0.248   0.001 0.043   0.181  0.218  0.243  0.271   0.348
## sigma[2]       0.242   0.001 0.043   0.174  0.211  0.235  0.267   0.341
## offset[1]      0.164   0.083 3.804  -6.513 -0.854  0.023  0.998   8.436
## offset[2]      0.109   0.113 4.215  -6.844 -0.828  0.039  0.957   6.517
## offset[3]      0.063   0.097 3.960  -7.574 -0.914  0.013  0.900   7.715
## offset[4]      0.355   0.273 5.858  -8.130 -0.963  0.023  1.059  10.386
## offset[5]      0.277   0.193 5.389  -7.225 -0.885  0.025  1.012   9.687
## offset[6]      0.090   0.051 2.794  -5.615 -0.858  0.016  0.896   6.590
## offset[7]      0.034   0.197 5.055  -8.142 -0.930  0.026  0.982   8.329
## offset[8]      0.009   0.103 4.116  -8.116 -0.948 -0.010  0.944   7.747
## offset[9]     -0.097   0.179 4.821  -6.675 -0.875  0.003  0.986   6.979
## offset[10]     0.172   0.145 4.918  -7.567 -0.883  0.026  0.975  10.009
## offset[11]    -0.057   0.190 4.711  -8.309 -0.949  0.003  1.016   7.368
## offset[12]     0.119   0.130 4.638  -7.272 -0.926  0.011  0.941   7.532
## offset[13]     0.350   0.182 5.554  -7.241 -0.923  0.026  0.985  10.395
## offset[14]    -0.069   0.074 3.392  -6.903 -0.935 -0.017  0.871   6.135
## offset[15]    -0.018   0.254 5.862  -7.531 -0.946  0.015  1.000   9.580
## offset[16]     0.364   0.221 4.838  -6.604 -0.872  0.051  1.001   9.214
## offset[17]     0.069   0.102 5.031  -7.469 -0.917  0.007  1.001   8.608
## offset[18]    -0.171   0.215 5.612  -8.807 -0.912  0.039  0.960   7.704
## offset[19]     0.047   0.075 3.448  -6.281 -0.886  0.022  0.931   6.145
## offset[20]    -0.065   0.125 4.199  -8.845 -0.945 -0.022  0.925   8.384
## z[1,1]        49.484   0.083 4.417  41.383 46.387 49.301 52.263  58.551
## z[1,2]         6.933   0.010 0.631   5.803  6.503  6.894  7.326   8.266
## z[2,1]        65.893   0.137 6.718  53.785 61.265 65.524 70.176  80.203
## z[2,2]        12.623   0.022 1.440  10.042 11.637 12.562 13.522  15.761
## z[3,1]        65.819   0.136 7.178  53.097 60.869 65.312 70.232  81.126
## z[3,2]        29.488   0.055 3.436  23.130 27.189 29.364 31.670  36.514
## z[4,1]        38.386   0.067 4.162  30.946 35.584 38.016 40.852  47.433
## z[4,2]        47.462   0.090 4.973  38.365 44.153 47.253 50.509  58.097
## z[5,1]        19.161   0.028 1.786  15.944 17.950 19.063 20.237  23.027
## z[5,2]        40.789   0.074 4.071  33.473 38.046 40.555 43.274  49.553
## z[6,1]        13.316   0.021 1.192  11.171 12.518 13.242 14.058  15.853
## z[6,2]        26.295   0.037 2.250  22.167 24.744 26.167 27.715  31.138
## z[7,1]        13.003   0.021 1.164  10.877 12.214 12.943 13.732  15.530
## z[7,2]        15.881   0.020 1.235  13.624 15.045 15.863 16.641  18.477
## z[8,1]        15.751   0.023 1.286  13.309 14.871 15.707 16.550  18.485
## z[8,2]         9.912   0.014 0.803   8.398  9.371  9.898 10.414  11.581
## z[9,1]        21.473   0.024 1.487  18.598 20.489 21.461 22.397  24.457
## z[9,2]         6.860   0.011 0.595   5.763  6.450  6.844  7.234   8.118
## z[10,1]       30.960   0.032 1.996  27.079 29.628 30.921 32.209  35.006
## z[10,2]        5.707   0.009 0.505   4.788  5.361  5.683  6.024   6.773
```

```
## z[11,1]        45.074   0.064  3.468   38.665 42.798 44.889 47.227  52.273
## z[11,2]         6.324   0.010  0.568    5.280  5.938  6.305  6.670   7.516
## z[12,1]        62.167   0.124  6.087   51.258 58.016 61.767 65.963  75.131
## z[12,2]        10.341   0.016  1.040    8.439  9.652 10.292 10.975  12.635
## z[13,1]        69.091   0.148  7.374   55.688 64.091 68.565 73.695  84.557
## z[13,2]        23.700   0.042  2.728   18.866 21.844 23.563 25.335  29.507
## z[14,1]        46.210   0.078  4.782   37.137 43.062 45.915 49.230  56.253
## z[14,2]        44.869   0.082  4.837   35.835 41.558 44.583 47.926  54.787
## z[15,1]        22.502   0.034  2.297   18.173 20.948 22.417 23.953  27.333
## z[15,2]        44.251   0.083  4.471   36.095 41.248 44.026 46.941  53.901
## z[16,1]        14.133   0.022  1.311   11.746 13.253 14.067 14.956  16.865
## z[16,2]        29.880   0.044  2.739   24.966 27.981 29.734 31.542  35.710
## z[17,1]        12.767   0.021  1.151   10.702 11.988 12.699 13.478  15.283
## z[17,2]        18.151   0.023  1.551   15.328 17.102 18.056 19.102  21.395
## z[18,1]        14.781   0.023  1.310   12.366 13.896 14.734 15.586  17.663
## z[18,2]        11.158   0.015  0.966    9.418 10.504 11.115 11.761  13.228
## z[19,1]        19.685   0.027  1.696   16.547 18.550 19.631 20.737  23.305
## z[19,2]         7.466   0.012  0.658    6.273  7.018  7.430  7.868   8.940
## z[20,1]        28.111   0.039  2.492   23.461 26.453 27.995 29.642  33.427
## z[20,2]         9.362   0.041  2.322    5.773  7.731  9.056 10.642  14.799
## y_init_rep[1] 35.215   0.151  9.537   20.140 28.653 34.049 40.448  56.963
## y_init_rep[2]  5.878   0.026  1.582    3.372  4.776  5.707  6.737   9.539
## y_rep[1,1]     51.314   0.225 13.878   29.275 41.710 49.832 59.061  81.856
## y_rep[1,2]      7.179   0.032  1.926    4.148  5.834  6.966  8.216  11.709
## y_rep[2,1]     68.207   0.305 18.844   38.802 55.363 65.798 78.225 110.610
## y_rep[2,2]     13.040   0.057  3.648    7.328 10.588 12.504 15.085  21.138
## y_rep[3,1]     67.958   0.320 19.043   37.980 54.828 65.487 78.453 114.076
## y_rep[3,2]     30.432   0.132  8.453   17.017 24.580 29.555 35.088  49.944
## y_rep[4,1]     39.667   0.181 11.075   22.433 32.137 38.139 45.301  65.957
## y_rep[4,2]     49.042   0.216 13.406   27.579 39.715 47.347 56.600  79.963
## y_rep[5,1]     19.757   0.086  5.392   11.252 15.997 19.074 22.590  32.635
## y_rep[5,2]     41.901   0.181 11.158   24.080 34.014 40.535 48.118  67.204
## y_rep[6,1]     13.736   0.061  3.778    7.848 11.180 13.236 15.754  22.432
## y_rep[6,2]     27.152   0.117  7.305   15.727 22.157 26.126 31.124  43.893
## y_rep[7,1]     13.363   0.058  3.700    7.544 10.818 12.896 15.303  22.046
## y_rep[7,2]     16.285   0.071  4.366    9.449 13.325 15.725 18.583  26.368
## y_rep[8,1]     16.269   0.071  4.402    9.298 13.166 15.788 18.676  26.402
## y_rep[8,2]     10.162   0.043  2.663    5.961  8.329  9.832 11.611  16.335
## y_rep[9,1]     22.248   0.094  5.927   12.736 18.065 21.573 25.524  35.584
## y_rep[9,2]      7.028   0.030  1.896    4.105  5.747  6.791  8.032  11.423
## y_rep[10,1]    31.936   0.131  8.428   18.324 26.192 30.829 36.635  51.430
## y_rep[10,2]     5.899   0.026  1.610    3.369  4.772  5.687  6.788   9.562
## y_rep[11,1]    46.251   0.199 12.401   26.682 37.638 44.702 53.176  74.537
## y_rep[11,2]     6.535   0.028  1.755    3.782  5.360  6.301  7.459  10.838
## y_rep[12,1]    64.296   0.296 17.658   36.245 51.903 62.160 74.064 105.746
## y_rep[12,2]    10.643   0.044  2.832    6.150  8.718 10.297 12.132  17.102
## y_rep[13,1]    71.022   0.314 19.872   39.814 57.516 68.428 81.549 114.979
## y_rep[13,2]    24.454   0.108  6.822   13.844 19.771 23.572 28.028  40.029
## y_rep[14,1]    47.776   0.219 13.544   26.966 38.254 45.978 54.998  80.587
## y_rep[14,2]    45.893   0.196 12.340   25.736 37.418 44.400 52.743  73.989
## y_rep[15,1]    23.136   0.100  6.235   12.700 18.753 22.575 26.694  37.230
## y_rep[15,2]    45.374   0.192 12.222   25.847 37.037 43.965 52.015  72.804
## y_rep[16,1]    14.633   0.063  3.935    8.345 11.883 14.110 16.916  23.545
## y_rep[16,2]    30.737   0.134  8.229   17.864 25.100 29.756 35.066  50.070
```

```
## y_rep[17,1]   13.246   0.058  3.613   7.634 10.744 12.748 15.194  21.804
## y_rep[17,2]   18.740   0.080  5.100  10.853 15.252 18.028 21.404  30.735
## y_rep[18,1]   15.300   0.065  4.122   8.867 12.382 14.745 17.637  25.426
## y_rep[18,2]   11.476   0.049  3.061   6.657  9.351 11.125 13.069  18.651
## y_rep[19,1]   20.383   0.092  5.718  11.498 16.431 19.746 23.389  33.953
## y_rep[19,2]    7.648   0.035  2.049   4.440  6.226  7.345  8.742  12.508
## y_rep[20,1]   28.983   0.128  8.068  16.110 23.356 27.931 33.320  47.211
## y_rep[20,2]    9.692   0.063  3.567   4.612  7.196  9.089 11.337  18.377
## lp__           5.788   0.384  7.453 -10.870  1.403  6.458 11.104  18.229
##              n_eff  Rhat
## theta[1]       1722 1.002
## theta[2]       1881 1.002
## theta[3]       1633 1.002
## theta[4]       1669 1.001
## z_init[1]      3683 1.000
## z_init[2]      2968 1.000
## sigma[1]       2925 1.000
## sigma[2]       3401 1.001
## offset[1]      2079 1.000
## offset[2]      1402 1.000
## offset[3]      1661 1.000
## offset[4]       460 1.003
## offset[5]       782 1.004
## offset[6]      3025 1.000
## offset[7]       658 1.008
## offset[8]      1609 1.003
## offset[9]       722 1.003
## offset[10]     1157 1.001
## offset[11]      612 1.009
## offset[12]     1276 1.004
## offset[13]      927 1.005
## offset[14]     2092 1.001
## offset[15]      533 1.005
## offset[16]      479 1.009
## offset[17]     2411 1.001
## offset[18]      684 1.002
## offset[19]     2091 1.000
## offset[20]     1123 1.002
## z[1,1]         2806 1.000
## z[1,2]         4045 1.000
## z[2,1]         2395 1.001
## z[2,2]         4370 1.000
## z[3,1]         2766 1.001
## z[3,2]         3938 1.000
## z[4,1]         3806 0.999
## z[4,2]         3027 1.001
## z[5,1]         4064 1.000
## z[5,2]         3008 1.001
## z[6,1]         3377 1.001
## z[6,2]         3739 1.000
## z[7,1]         3072 1.002
## z[7,2]         3878 0.999
## z[8,1]         3264 1.001
## z[8,2]         3087 0.999
```

```
## z[9,1]         3882 1.000
## z[9,2]         2709 1.000
## z[10,1]        3992 1.000
## z[10,2]        2876 1.000
## z[11,1]        2921 1.000
## z[11,2]        3532 1.000
## z[12,1]        2417 1.000
## z[12,2]        4119 1.000
## z[13,1]        2493 1.001
## z[13,2]        4224 1.000
## z[14,1]        3736 1.000
## z[14,2]        3485 1.000
## z[15,1]        4448 1.000
## z[15,2]        2873 1.001
## z[16,1]        3628 1.001
## z[16,2]        3806 1.001
## z[17,1]        3068 1.002
## z[17,2]        4429 1.001
## z[18,1]        3342 1.001
## z[18,2]        4107 1.000
## z[19,1]        3857 1.001
## z[19,2]        3145 1.000
## z[20,1]        4045 1.000
## z[20,2]        3140 1.000
## y_init_rep[1]  3996 1.000
## y_init_rep[2]  3614 1.000
## y_rep[1,1]     3788 1.000
## y_rep[1,2]     3684 1.000
## y_rep[2,1]     3822 1.000
## y_rep[2,2]     4069 0.999
## y_rep[3,1]     3544 1.000
## y_rep[3,2]     4074 1.000
## y_rep[4,1]     3727 1.000
## y_rep[4,2]     3860 1.000
## y_rep[5,1]     3966 1.000
## y_rep[5,2]     3796 1.001
## y_rep[6,1]     3873 1.000
## y_rep[6,2]     3884 1.000
## y_rep[7,1]     4005 1.000
## y_rep[7,2]     3768 1.000
## y_rep[8,1]     3880 1.001
## y_rep[8,2]     3763 1.001
## y_rep[9,1]     4007 1.000
## y_rep[9,2]     3961 0.999
## y_rep[10,1]    4126 1.000
## y_rep[10,2]    3758 1.000
## y_rep[11,1]    3888 0.999
## y_rep[11,2]    3955 1.000
## y_rep[12,1]    3560 0.999
## y_rep[12,2]    4141 1.000
## y_rep[13,1]    4009 1.000
## y_rep[13,2]    3980 1.001
## y_rep[14,1]    3826 1.000
## y_rep[14,2]    3965 0.999
```

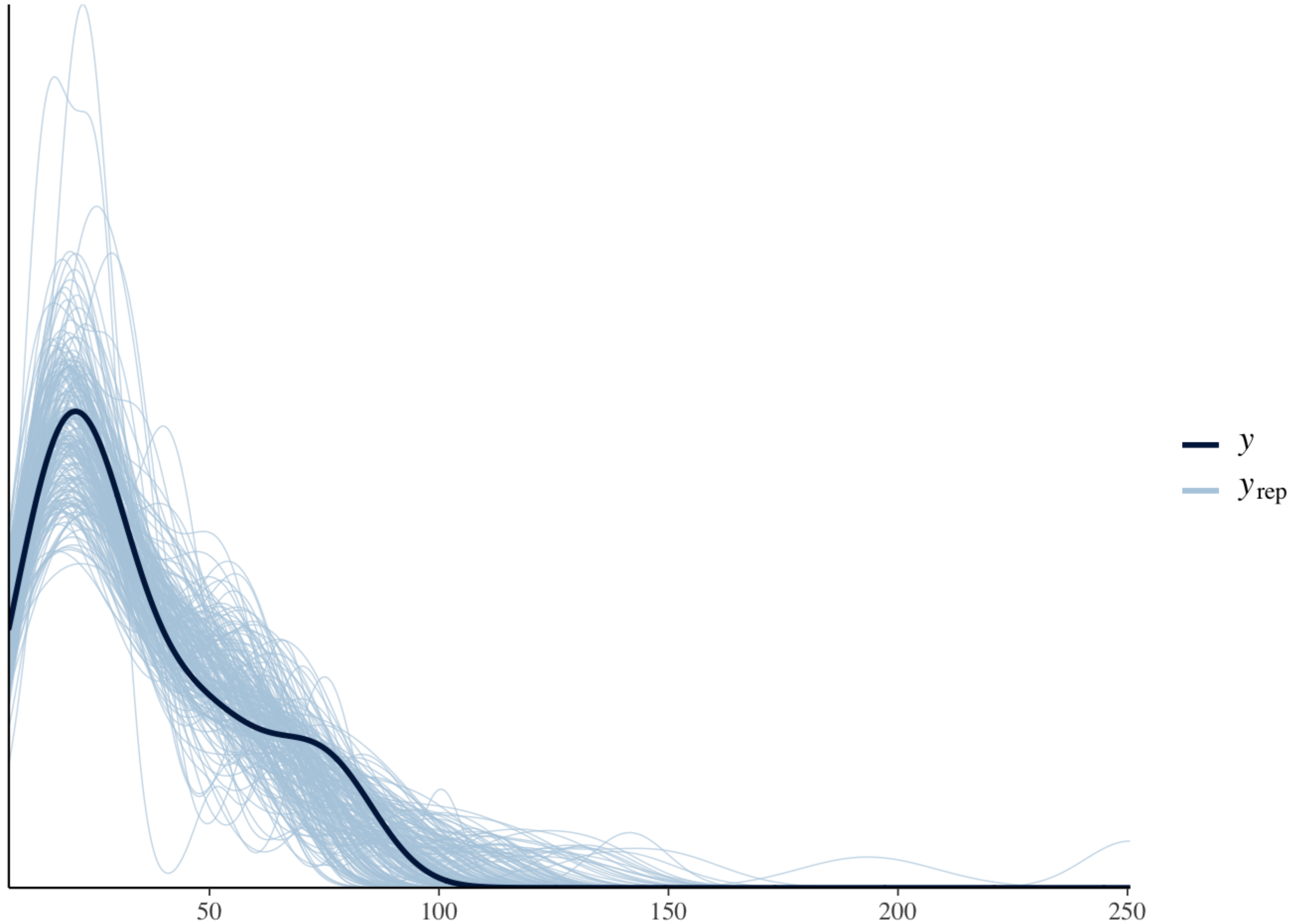
```
## y_rep[15,1]    3883 1.000
## y_rep[15,2]    4033 1.001
## y_rep[16,1]    3936 1.000
## y_rep[16,2]    3775 1.001
## y_rep[17,1]    3875 1.000
## y_rep[17,2]    4036 1.000
## y_rep[18,1]    3996 1.000
## y_rep[18,2]    3949 1.001
## y_rep[19,1]    3890 1.001
## y_rep[19,2]    3485 1.000
## y_rep[20,1]    3984 1.000
## y_rep[20,2]    3205 1.000
## lp__            378 1.005
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 25 18:49:31 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

The improvement in this extension is that, we can measurem the effect of the systematical influence from the factors outside.

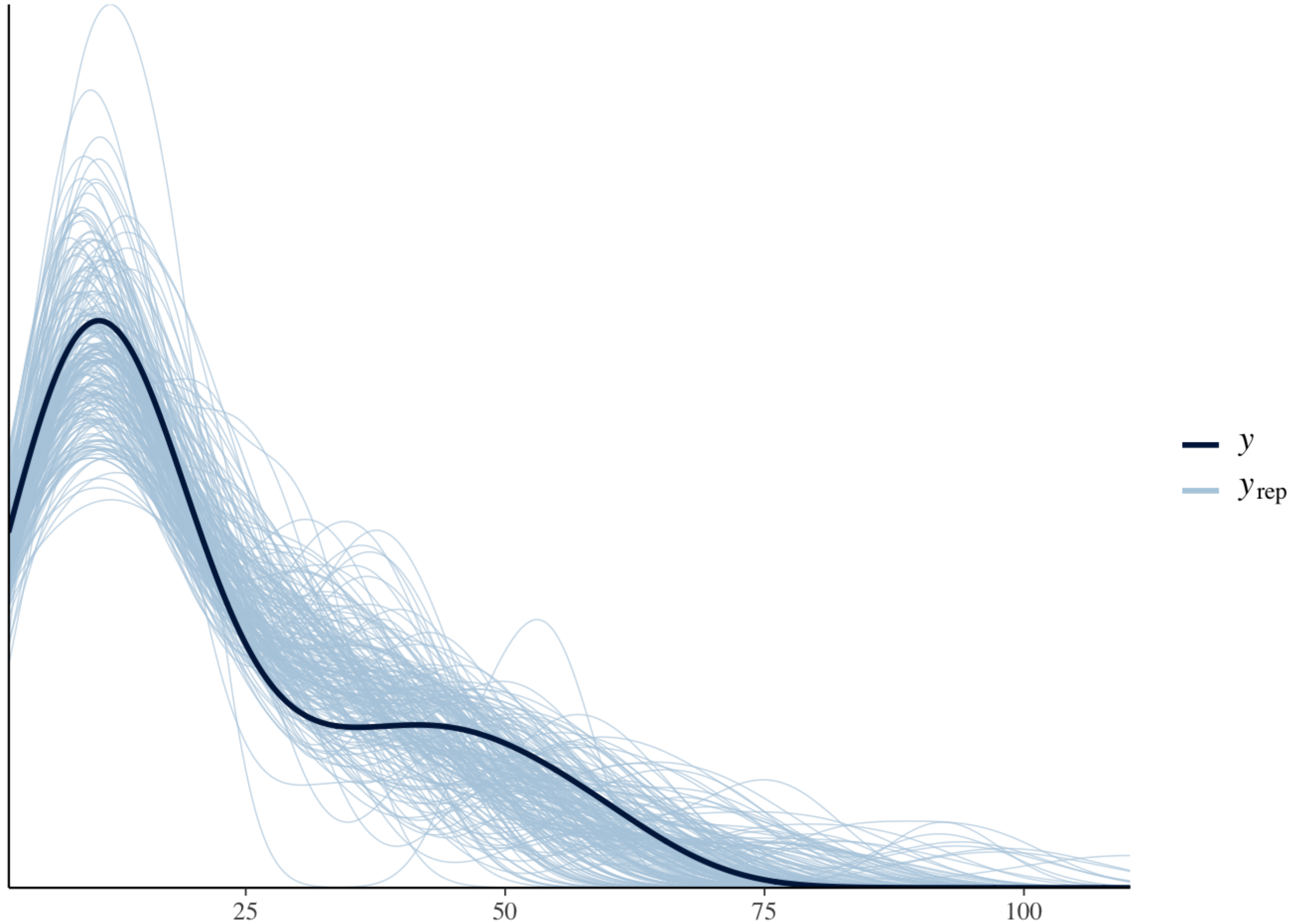
```
lynx_hare_df <-  read.csv("hudson-bay-lynx-hare.csv",comment.char="#")
N <- length(lynx_hare_df$Year) - 1
ts <- 1:N
y_init <- c(lynx_hare_df$Hare[1], lynx_hare_df$Lynx[1])
y <- as.matrix(lynx_hare_df[2:(N + 1), 2:3])
y <- cbind(y[ , 2], y[ , 1]); # hare, lynx order

y_rep_1 <- as.matrix(as.matrix(fit, pars = "y_rep")[1:200,1:20])
y_rep_2 <- as.matrix(as.matrix(fit, pars = "y_rep")[1:200,21:40])

ppc_dens_overlay(y = y[,1], yrep = y_rep_1)
```

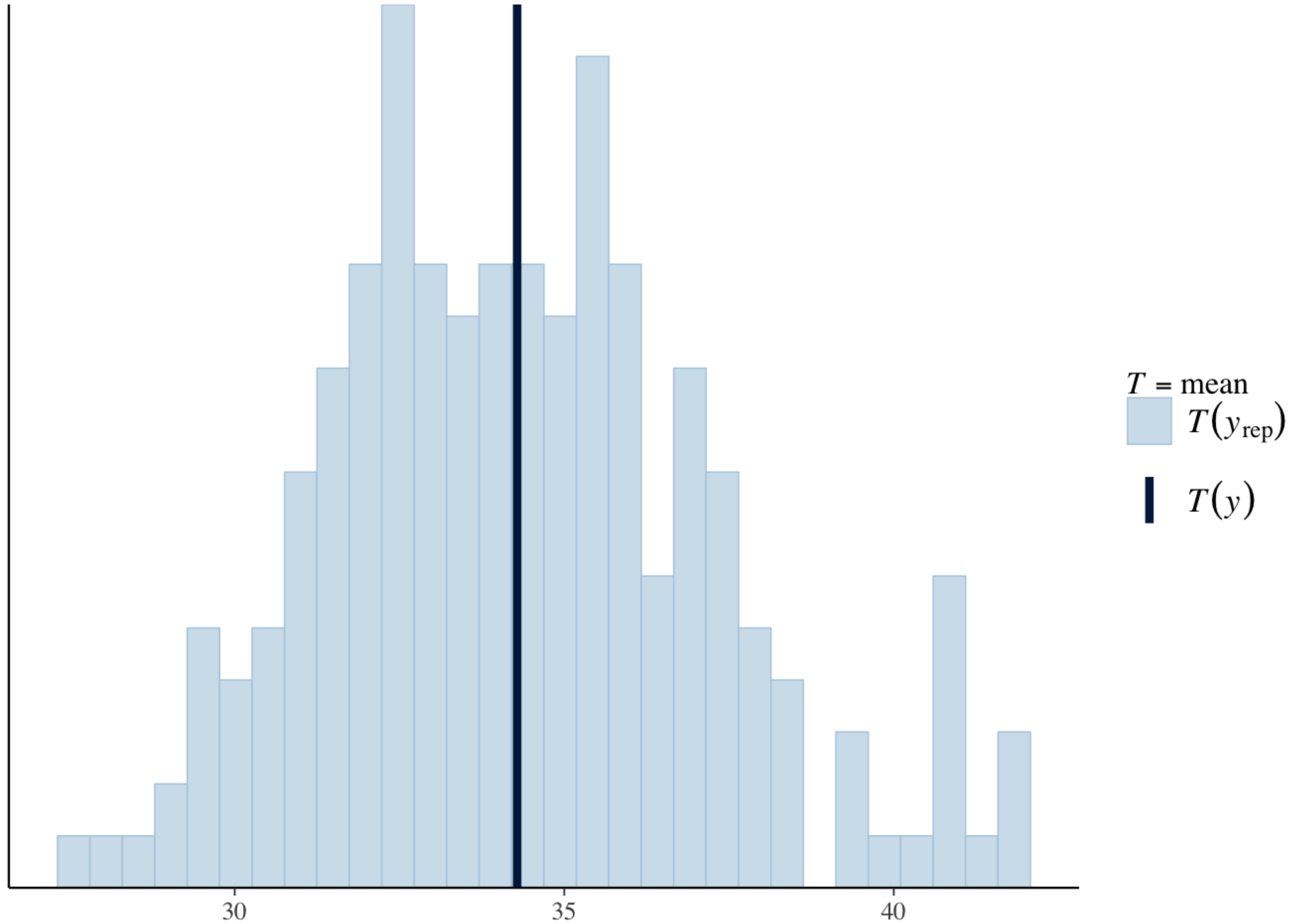



```
ppc_dens_overlay(y = y[,2], yrep = y_rep_2)
```



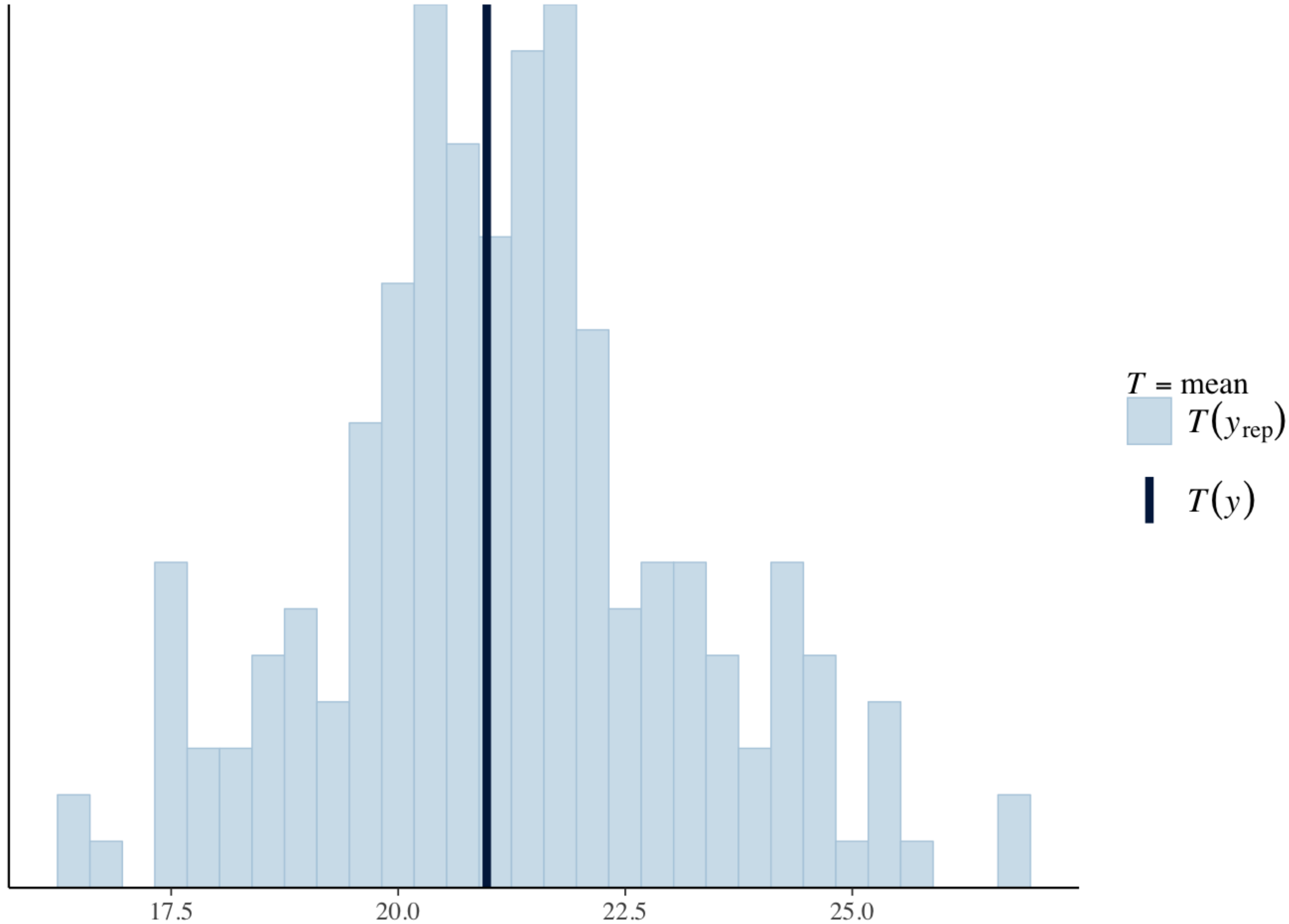

```
ppc_stat(y = y[,1], yrep = y_rep_1, stat = "mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



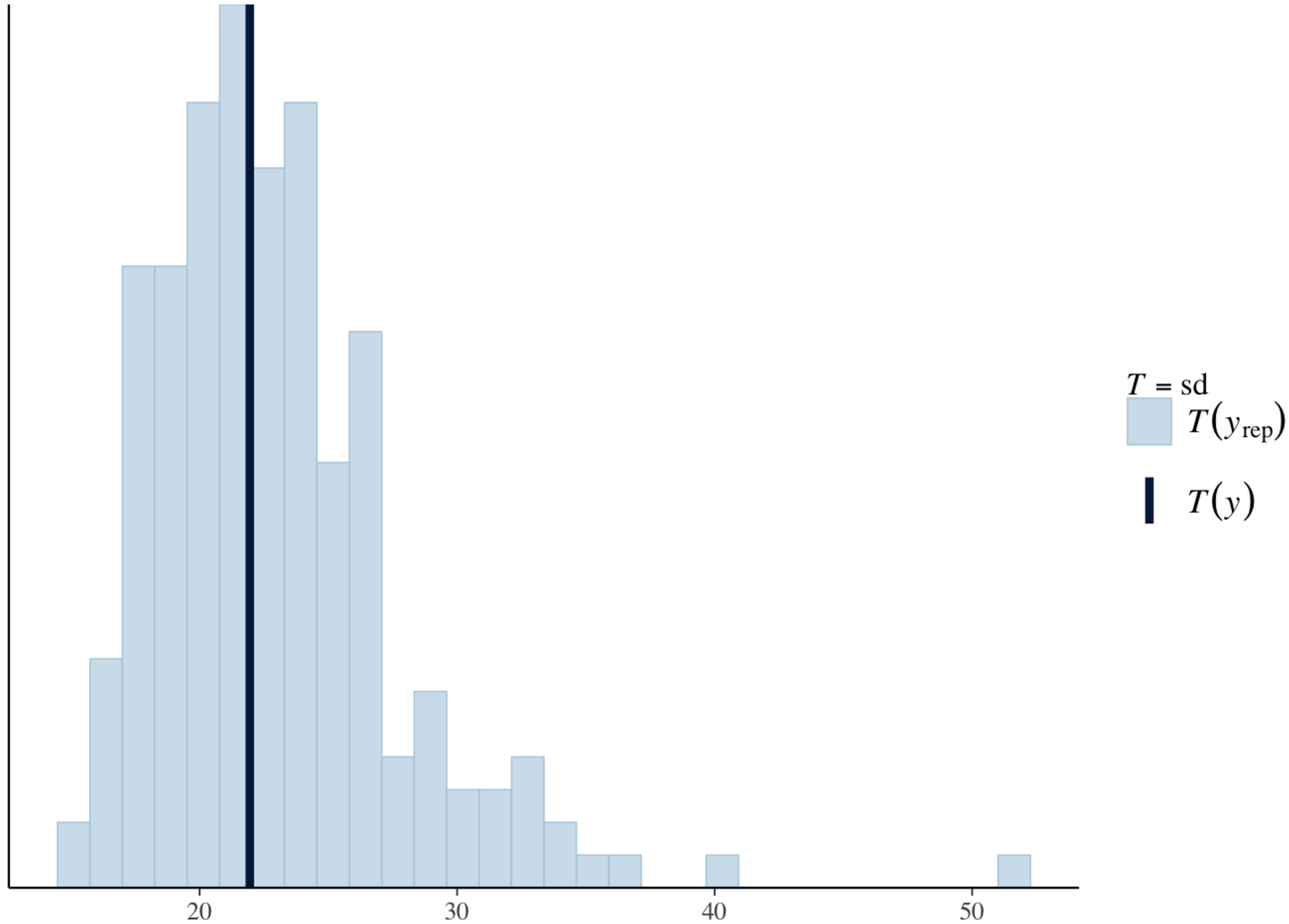

```r
ppc_stat(y = y[,2], yrep = y_rep_2, stat = "mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



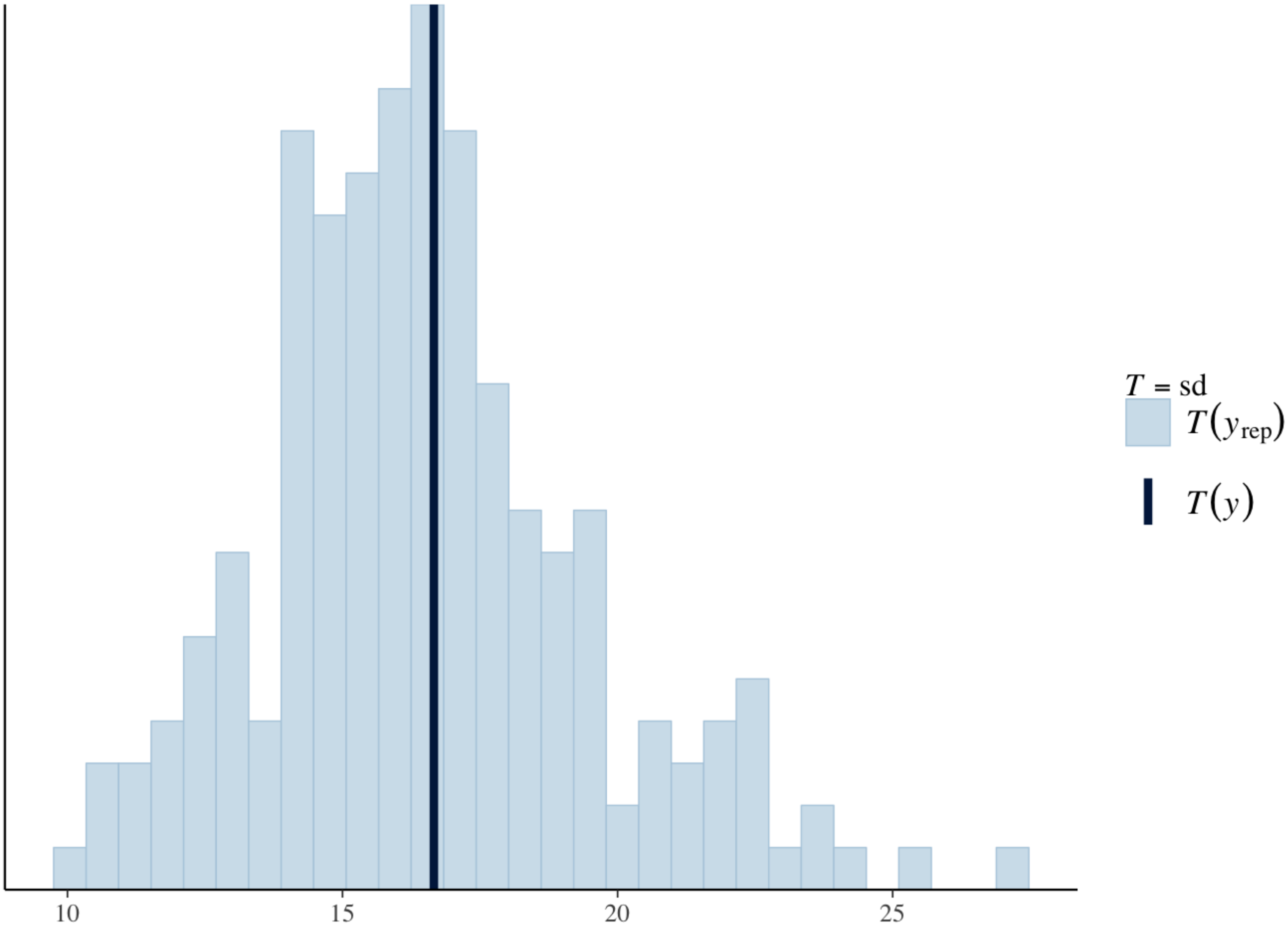

```
ppc_stat(y = y[,1], yrep = y_rep_1, stat = "sd")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

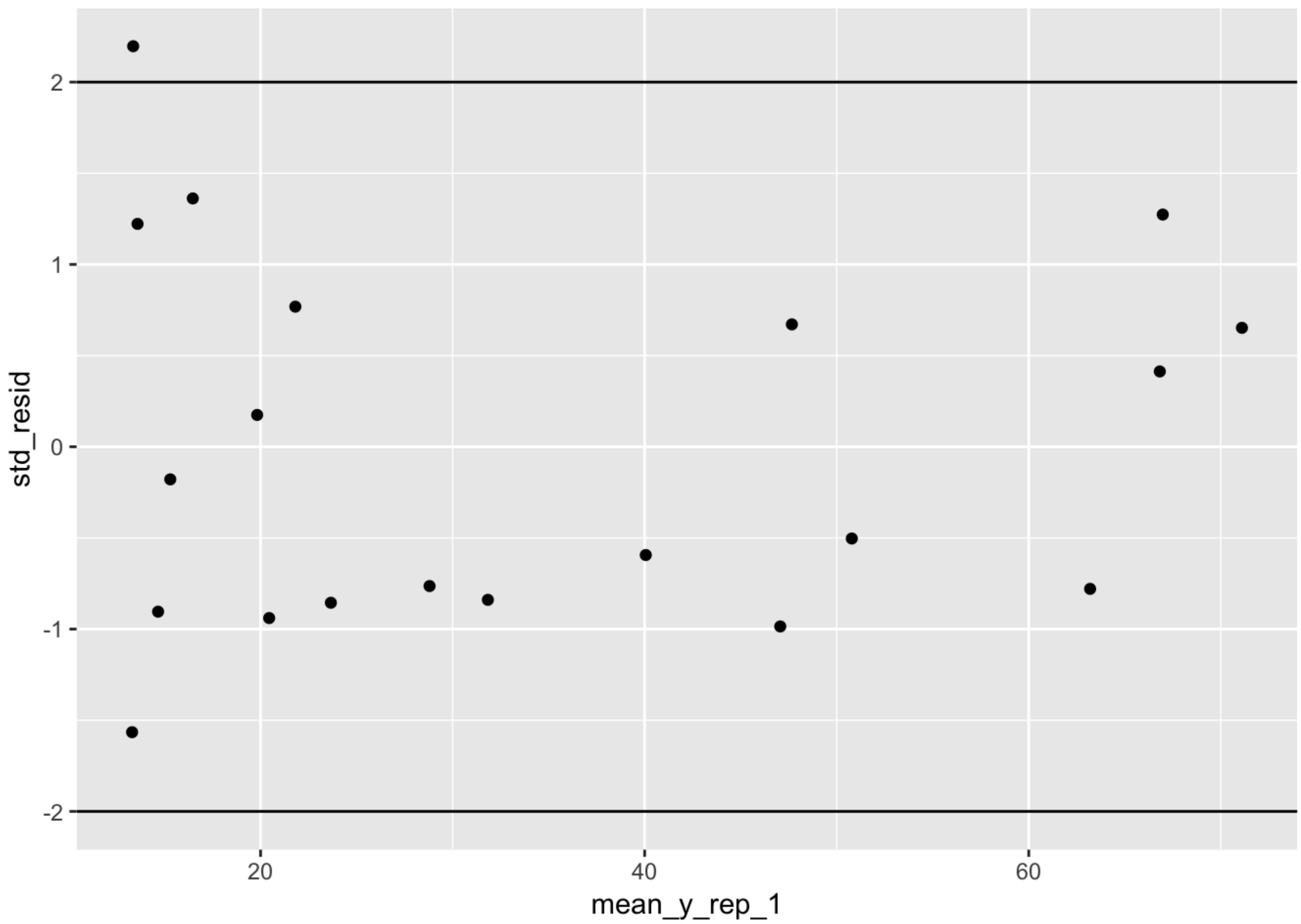



```
ppc_stat(y = y[,2], yrep = y_rep_2, stat = "sd")
```

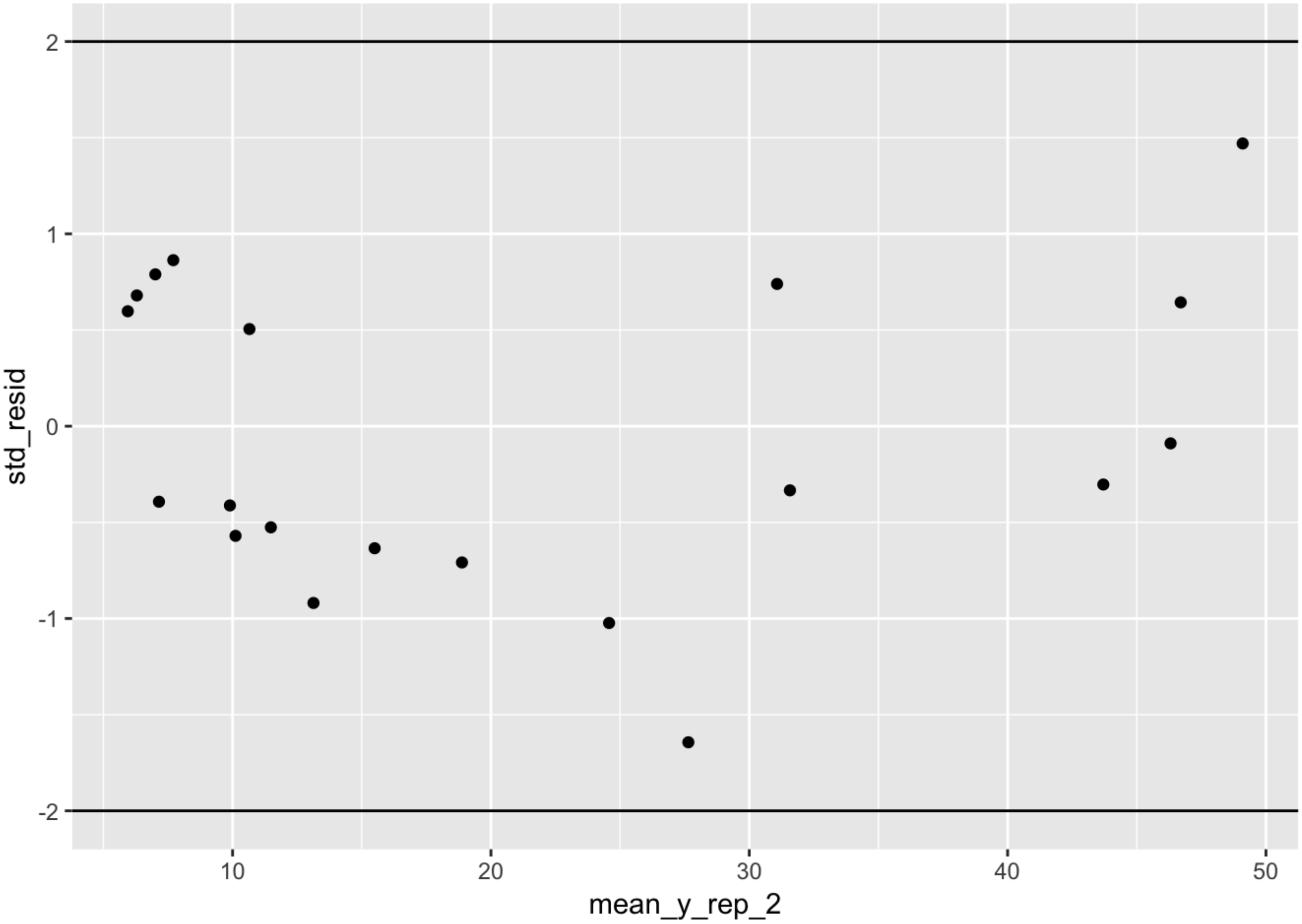
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
mean_y_rep_1 <- colMeans(y_rep_1)
std_resid <- (y[,1] - mean_y_rep_1) / sqrt(mean_y_rep_1)
qplot(mean_y_rep_1, std_resid) + hline_at(2) + hline_at(-2)
```




```
mean_y_rep_2 <- colMeans(y_rep_2)
std_resid <- (y[,2] - mean_y_rep_2) / sqrt(mean_y_rep_2)
qplot(mean_y_rep_2, std_resid) + hline_at(2) + hline_at(-2)
```


2
1
0
-1
-2
std_resid
10
20
30
40
50
mean_y_rep_2