8.4 predictive simulation

Chris Parrish

July 3, 2016

Contents

8.4 predictive simulation

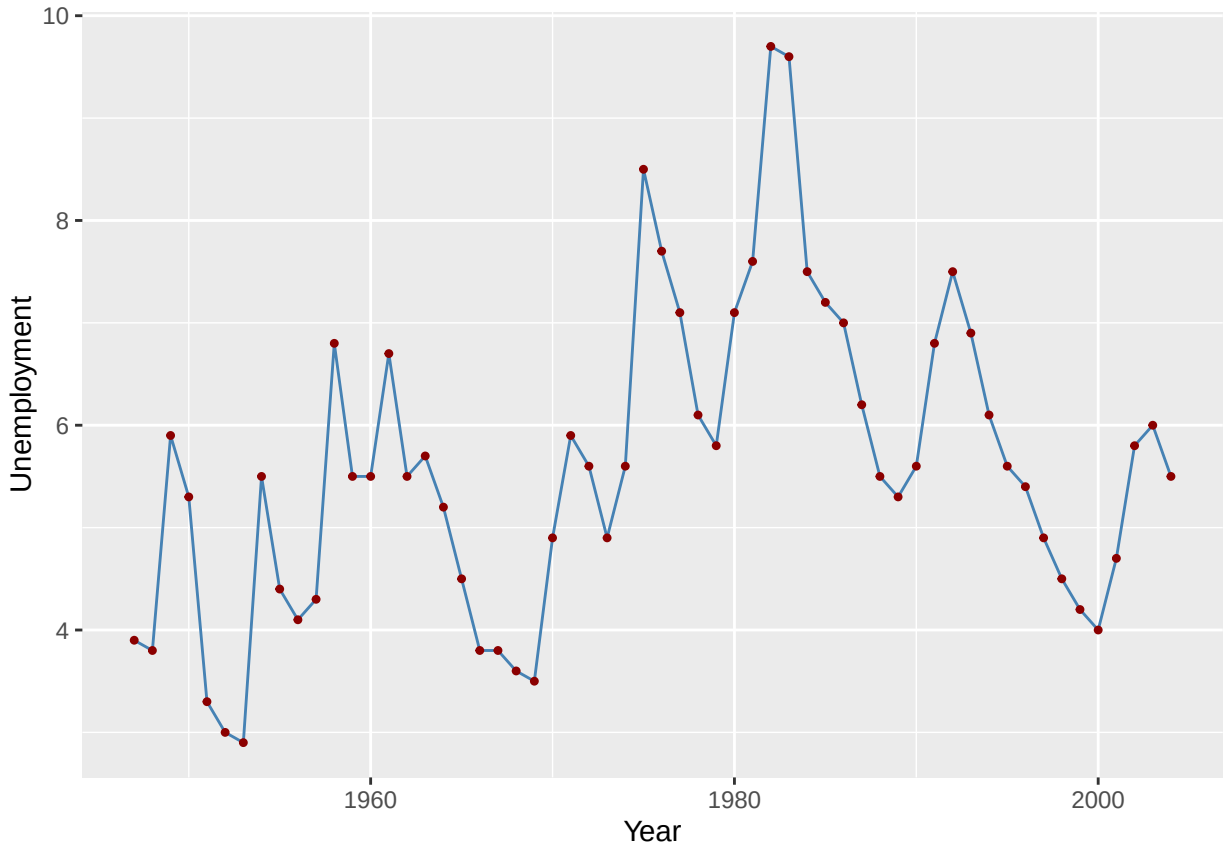
reference:
- ARM chapter 08, github

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
library(reshape2)        # for melt
```

fake-data simulation

```
unemployment <- read.table ("unemployment.dat", header=TRUE)
year <- unemployment$year
y <- unemployment$unemployed.pct
```

```
## Plot of the unemployment rate
frame1 = data.frame(year=year,y=y)
p1 <- ggplot(frame1,aes(x=year,y=y)) +
      geom_line(color = "steelblue") +
      geom_point(shape = 20, color = "darkred") +
      scale_y_continuous("Unemployment") +
      scale_x_continuous("Year") +
      theme_gray()
print(p1)
```



model

unemployment.stan

```
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] y_lag;
}
parameters {
  vector[2] beta;
  real<lower=0> sigma;
}
model {
  y ~ normal(beta[1] + beta[2] * y_lag, sigma);
}
```

fit

```
## Fitting a 1st-order autogregression
## lm(y ~ y_lag)
source("unemployment.data.R", echo = TRUE)
```

```
##
## > "y" <- c(3.8, 5.9, 5.3, 3.3, 3, 2.9, 5.5, 4.4, 4.1,
```

```
## +     4.3, 6.8, 5.5, 5.5, 6.7, 5.5, 5.7, 5.2, 4.5, 3.8, 3.8, 3.6,
## +     3.5, 4.9, 5.9, 5.6, 4. .... [TRUNCATED]
##
## > "y_lag" <- c(3.9, 3.8, 5.9, 5.3, 3.3, 3, 2.9, 5.5,
## +     4.4, 4.1, 4.3, 6.8, 5.5, 5.5, 6.7, 5.5, 5.7, 5.2, 4.5, 3.8,
## +     3.8, 3.6, 3.5, 4.9, 5.9 .... [TRUNCATED]
##
## > "N" <- 57
```

```
dataList.1 <- c("N","y_lag","y")
unemployment.sf1 <- stan(file='unemployment.stan', data=dataList.1,
                         iter=1000, chains=4)
print(unemployment.sf1)
```

```
## Inference for Stan model: unemployment.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta[1]   1.47    0.02 0.52   0.45   1.14   1.47   1.79   2.52   558    1
## beta[2]   0.74    0.00 0.09   0.56   0.69   0.74   0.80   0.92   553    1
## sigma     1.01    0.00 0.10   0.85   0.94   1.00   1.07   1.23   908    1
## lp__    -28.20    0.05 1.26 -31.52 -28.81 -27.86 -27.26 -26.79   571    1
##
## Samples were drawn using NUTS(diag_e) at Sun Jul 10 08:20:00 2016.
## 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 estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  0.9 0.9 1.1 1.1
```

simulate new data $\tilde{y}$

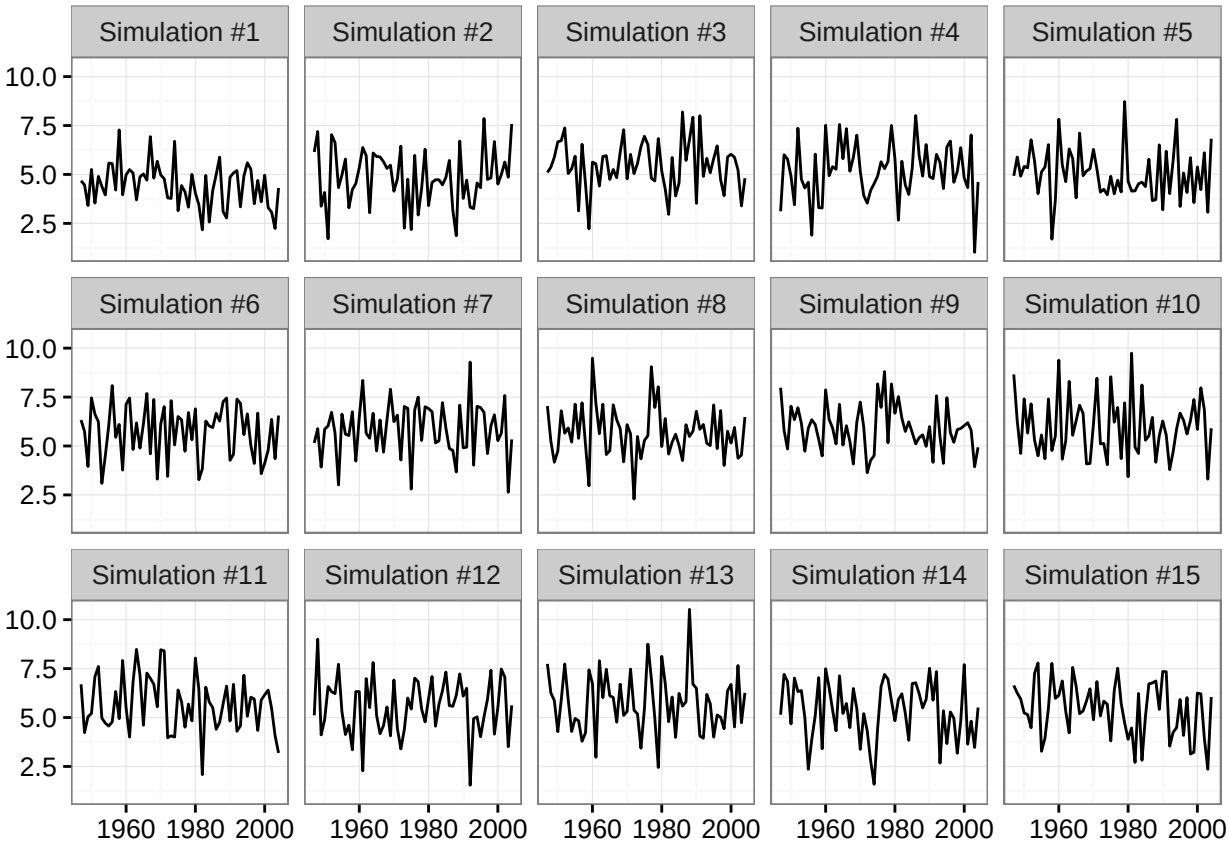
```
## Simulating replicated datasets
lag.post <- extract(unemployment.sf1)
b.hat <- colMeans(lag.post$beta)
s.hat <- mean(lag.post$sigma)

n.sims <- length(year) + 1
n <- 16
y.rep <- array (NA, c(n.sims, n))
for (s in 1:n.sims){
  y.rep[s,1] <- y[1]
  for (t in 2:n){
    prediction <- c (1, y.rep[s,t-1]) %*% b.hat
    y.rep[s,t] <- rnorm (1, prediction, s.hat)
  }
}
```

```
## Including uncertainty in the estimated parameters
n <- 15
n.sims <- length(year)
```

```
y.rep2 <- array (NA, c(n.sims, n))
for (s in 1:n.sims){
  for (t in 1:n){
    prediction <-  c (1, y.rep[s+1,t]) %*% lag.post$beta[s,]
    y.rep2[s,t] <- rnorm (1, prediction, lag.post$sigma[s])
  }
}
```

```
## Plot of simulated unemployment rate series
y.new <- melt(y.rep2)
y.new$Var2 <- factor(y.new$Var2, levels=c('1','2','3','4','5','6','7','8','9','10','11','12','13','14',
labels=c('Simulation #1','Simulation #2','Simulation #3','Simulation #4','Simulation #5',
        'Simulation #6','Simulation #7','Simulation #8','Simulation #9','Simulation #10',
        'Simulation #11','Simulation #12','Simulation #13','Simulation #14','Simulation #15'))
frame2 = data.frame(y.new=y.new$value, year=year, Var2=y.new$Var2)
p2 <- ggplot(frame2, aes(y=y.new, x=year)) +
      theme_bw() +
      geom_line() +
      facet_wrap( ~ Var2, ncol=5) +
      theme(axis.title.y = element_blank(), axis.title.x=element_blank())
print(p2)
```



generate more simulated data for model check

```
n.sims <- 1000
n <- length (y)
y.rep <- array (NA, c(n.sims, n))
for (s in 1:n.sims){
  y.rep[s,1] <- y[1]
  for (t in 2:n){
    prediction <- c (1, y.rep[s,t-1]) %*% b.hat
    y.rep[s,t] <- rnorm (1, prediction, s.hat)
  }
}
```

model check

```
## Numerical model check
Test <- function (y){
  n <- length (y)
  y.lag <- c (NA, y[1:(n-1)])
  y.lag2 <- c (NA, NA, y[1:(n-2)])
  sum (sign(y-y.lag) != sign(y.lag-y.lag2), na.rm=TRUE)
}

n.sims <- 1000
print (Test (y))
```

```
## [1] 22
```

```
test.rep <- rep (NA, n.sims)
for (s in 1:n.sims){
  test.rep[s] <- Test (y.rep[s,])
}

# print (mean (test.rep > Test(y)))
print (quantile (test.rep, c(.05,.5,.95)))
```

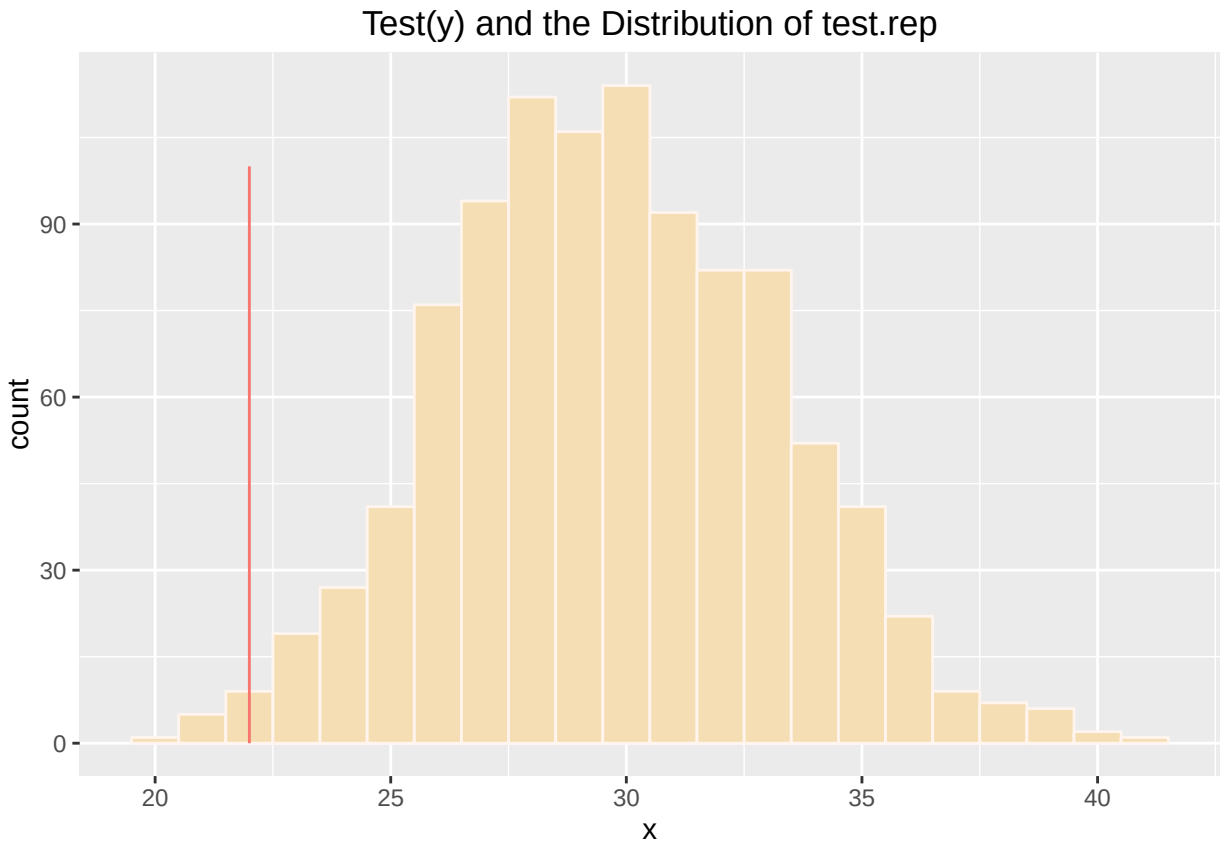
```
##  5% 50% 95%
##  24  30  35
```

```
mean(test.rep > Test(y))
```

```
## [1] 0.985
```

figure

```
data1 <- data.frame(x = test.rep)
data2 <- data.frame(x1 = Test(y), x2 = Test(y), y1 = 0, y2 = 100)
ggplot(data1, aes(x)) +
  geom_histogram(binwidth = 1, color = "seashell", fill = "wheat") +
  geom_segment(data = data2, aes(x = x1, y = y1, xend = x2, yend = y2,
  color = "salmon")) +
  theme(legend.position="none") +
  labs(title = "Test(y) and the Distribution of test.rep")
```


Test(y) and the Distribution of test.rep
count
90
60
30
0
20
25
30
35
40
x