Data Science per psicologi

Corrado Caudek

2021-09-13

Indice

Indice **1**

1 Distribuzione predittiva a posteriori **3**
1.1 Schema Beta-Binomiale 3
1.2 Metodi MCMC per la distribuzione predittiva a posteriori . . . 7
1.3 Posterior predictive checks 10
Considerazioni conclusive 24

Bibliografia **25**

Capitolo 1

Distribuzione predittiva a posteriori

Oltre ad una sintesi della distribuzione a posteriori attraverso il computo di indici caratteristici e alla verifica delle ipotesi, un altro compito dell'analisi bayesiana è la predizione di nuovi dati futuri.

Definizione 1.1. La *distribuzione predittiva a posteriori* (*posterior predictive distribution*, PPD)

$$p(\tilde{y} \mid y) = \int_{\theta} p(\tilde{y} \mid \theta) p(\theta \mid y) d\theta \tag{1.1}$$

è la distribuzione di un nuovo campione di possibili osservazioni $\tilde{Y}$ avendo già osservato n manifestazioni dello stesso fenomeno $Y = y$.

La (1.1) descrive la nostra incertezza sui dati previsti futuri data la distribuzione a posteriori per θ che abbiamo ottenuto, ovvero tenendo conto della scelta del modello e della stima dei parametri mediante i dati osservati. La distribuzione predittiva a posteriori viene usata per fare inferenze predittive, cioè per prevedere la distribuzione di nuovi dati non osservati.

1.1 Schema Beta-Binomiale

Consideriamo un'altra volta il campione di pazienti clinici depressi di Zetsche et al. (2019) – si veda l'Appendice **??**. Supponiamo di volere esaminare in futuro altri 20 pazienti clinici e ci chiediamo quanti di essi ($\tilde{y} \in \{0, 1, \dots, 20\}$) manifesteranno una depressione grave.

Se vogliamo fare predizioni su $\tilde{y}$ dobbiamo innanzitutto riconoscere che i valori $\tilde{y} \in [0, 20]$ non sono tutti egualmente plausibili. Sappiamo che $\tilde{y}$ è una v.c. binomiale con distribuzione

$$p(\tilde{y} \mid \theta) = \binom{20}{\tilde{y}} \theta^{\tilde{y}} (1 - \theta)^{20 - \tilde{y}} \, . \tag{1.2}$$

La v.c. $\tilde{y}$ dipende da θ, ma θ è essa stessa una variabile casuale. Avendo osservato $y = 23$ successi in $n = 30$ prove, e avendo assunto come distribuzione a priori per θ una Beta$(2, 10)$, la distribuzione a posteriori di θ è una Beta$(25, 17)$. Per trovare la distribuzione sui possibili dati previsti futuri $\tilde{y}$ dobbiamo dunque applicare la (1.1):

$$p(\tilde{y} \mid y = 23) = \int_0^1 p(\tilde{y} \mid \theta) p(\theta \mid y = 23) d\theta \, . \tag{1.3}$$

Per il modello Beta-Binomiale, che stiamo discutendo, è possibile trovare una soluzione analitica all'equazione (1.1):

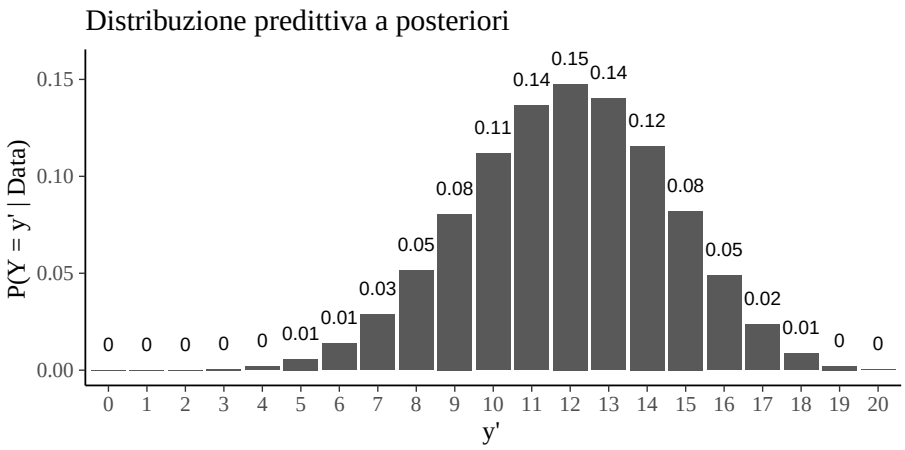
$$\begin{aligned}
p(\tilde{y} \mid y) &= \int_0^1 p(\tilde{y} \mid \theta) p(\theta \mid y) \, d\theta \\
&= \int_0^1 \binom{\tilde{n}}{\tilde{y}} \theta^{\tilde{y}} (1-\theta)^{\tilde{n}-\tilde{y}} \text{Beta}(a+y, b+n-y) \, d\theta \\
&= \binom{\tilde{n}}{\tilde{y}} \int_0^1 \theta^{\tilde{y}} (1-\theta)^{\tilde{n}-\tilde{y}} \frac{1}{B(a+y, b+n-y)} \theta^{a+y-1} (1-\theta)^{b+n-y-1} \\
&= \binom{\tilde{n}}{\tilde{y}} \frac{1}{B(a+y, b+n-y)} \int_0^1 \theta^{\tilde{y}+a+y-1} (1-\theta)^{\tilde{n}-\tilde{y}+b+n-y-1} \\
&= \binom{\tilde{n}}{\tilde{y}} \frac{B(\tilde{y}+a+y, b+n-y+\tilde{n}-\tilde{y})}{B(a+y, b+n-y)} \, .
\end{aligned} \tag{1.4}$$

Svolgendo i calcoli in R, per i dati dell'esempio otteniamo:

```
# Beta Binomial Predictive distribution function
# https://rpubs.com/FJRubio/BetaBinomialPred
BetaBinom <- Vectorize(
  function(rp){
  log_val <- lchoose(np, rp) +
    lbeta(rp+a+y, b+n-y+np-rp) -
    lbeta(a+y, b+n-y)
  return(exp(log_val))
  }
)

n <- 30
y <- 23
a <- 2
b <- 10
np <- 20
data.frame(
  heads = 0:20,
```

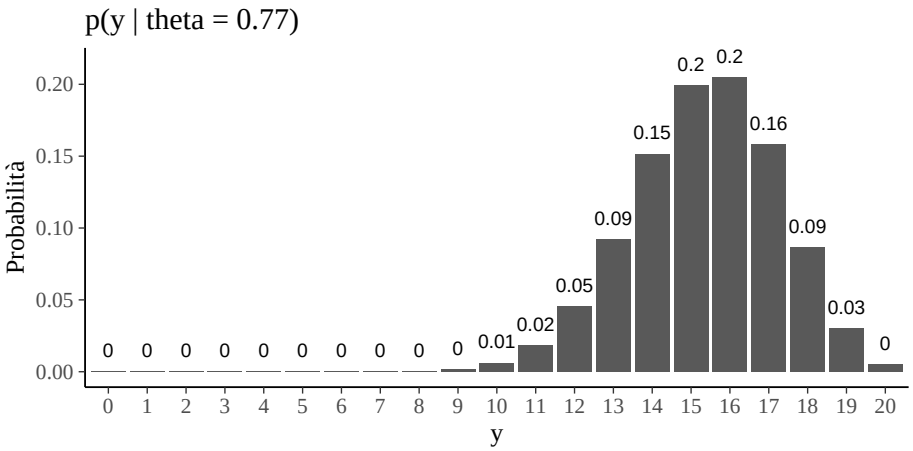
```
  pmf = BetaBinom(0:20)
) %>%
ggplot(aes(x = factor(heads), y = pmf)) +
  geom_col() +
  geom_text(
    aes(label = round(pmf, 2), y = pmf + 0.01),
    position = position_dodge(0.9),
    size = 3,
    vjust = 0
  ) +
  labs(
    title = "Distribuzione predittiva a posteriori",
    x = "y'",
    y = "P(Y = y' | Data)"
  )
```



È facile vedere che, in questo esempio, la distribuzione predittiva a posteriori $p(\tilde{y} \mid y)$ è diversa dalla binomiale di parametro $\theta = 23/30$:

```
tibble(
  heads = 0:20,
  pmf = dbinom(x = 0:20, size = 20, prob = 23/30)
) %>%
ggplot(aes(x = factor(heads), y = pmf)) +
  geom_col() +
  geom_text(
    aes(label = round(pmf, 2), y = pmf + 0.01),
    position = position_dodge(0.9),
    size = 3,
```

```
    vjust = 0
) +
labs(title = "p(y | theta = 0.77)",
     x = "y",
     y = "Probabilità")
```



In particolare, la $p(\tilde{y} \mid y)$ ha una varianza maggiore di $\text{Bin}(y \mid \theta = 0.77, n = 20)$. Questa maggiore varianza riflette le due fonti di incertezza che sono presenti nella (1.1): l'incertezza sul valore del parametro (descritta dalla distribuzione a posteriori) e l'incertezza dovuta alla variabilità campionaria (descritta dalla funzione di verosimiglianza).

Possiamo concludere la discussione di questo esempio dicendo che, nel caso di 20 nuovi pazienti clinici, ci aspettiamo di osservare 12 pazienti che manifestano una depressione severa, anche se è ragionevole aspettarci un numero compreso, diciamo, tra 8 e 16.

Una volta trovata la distribuzione predittiva a posteriori $p(\tilde{y} \mid y)$ diventa possibile rispondere a domande come: qual è la probabilità che almeno 10 dei 20 pazienti futuri mostrino una depressione grave? Rispondere a domande di questo tipo è possibile, ma richiede un po' di lavoro — non ci sono funzioni R che svolgano questi calcoli per noi. Tuttavia, non è importante imparare a risolvere problemi di questo tipo perché, in generale, anche per problemi solo leggermente più complessi di quello discusso qui, non sono disponibili espressioni analitiche della distribuzione predittiva a posteriori. Mediante simulazioni MCMC, invece, è possibile trovare una approssimazione numerica della $p(\tilde{y} \mid y)$. In tali circostanze, è più facile rispondere alle domande che ci siamo posti.

1.2 Metodi MCMC per la distribuzione predittiva a posteriori

Utilizzando la notazione di Gelman et al. (2014), chiamiamo y^{rep} i dati previsti futuri che potrebbero venire osservati se l'esperimento casuale che ha prodotto y venisse ripetuto, ovvero una realizzazione futura del modello statistico con gli stessi valori dei parametri θ che hanno prodotto y. Gelman et al. (2014) distinguono y^{rep} (repliche sotto lo stesso modello statistico) da $\tilde{y}$, che corrisponde invece ad un effettivo campione empirico di dati osservato in qualche futura occasione.

Mostreremo qui come ottenere $p(y^{rep} \mid y)$ quale stima di $p(\tilde{y} \mid y)$. Se svolgiamo l'analisi bayesiana con il metodo MCMC, $p(y^{rep} \mid y)$ può essere ottenuta nel modo seguente:

- campionare $\theta_i \sim p(\theta \mid y)$, ovvero campionare un valore del parametro dalla distribuzione a posteriori;
- campionare $y^{rep} \sim p(y^{rep} \mid \theta_i)$, ovvero campionare il valore di un'osservazione dalla funzione di verosimiglianza condizionata al valore del parametro definito nel passo precedente.

Se i due passaggi descritti sopra vengono ripetuti un numero sufficiente di volte, l'istogramma risultante approssimerà la distribuzione predittiva a posteriori che, in teoria (ma non in pratica) potrebbe essere ottenuta per via analitica (si deva il Paragrafo 1.1).

Vediamo ora come calcolare $p(y^{rep} \mid y)$ usando Stan. Qui di seguito è riportato il codice Stan per fare inferenza su una proporzione — si veda il Capitolo **??**:

```
modelString = "
data {
  int<lower=0> N;
  int<lower=0, upper=1> y[N];
}
parameters {
  real<lower=0, upper=1> theta;
}
model {
  theta ~ beta(2, 10);
  y ~ bernoulli(theta);
}
generated quantities {
  int y_rep[N];
  real log_lik[N];
  for (n in 1:N) {
    y_rep[n] = bernoulli_rng(theta);
    log_lik[n] = bernoulli_lpmf(y[n] | theta);
```

```
  }
}
"
writeLines(modelString, con = "code/betabin23-30-2-10.stan")
```

Si noti che nel nel blocco `generated quantities` sono state aggiunte le istruzioni necessarie per simulare y^{rep}, ovvero, `y_rep[n] = bernoulli_rng(theta);`.

Svolgiamo ora la simulazione. I dati dell'esempio che stiamo discutendo sono:

```
data_list <- list(
  N = 30,
  y = c(rep(1, 23), rep(0, 7))
)
```

Compiliamo il codice Stan

```
file <- file.path("code", "betabin23-30-2-10.stan")
mod <- cmdstan_model(file)
```

ed eseguiamo il campionamento MCMC:

```
fit <- mod$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = SEED,
  chains = 4L,
  parallel_chains = 4L,
  refresh = 0,
  thin = 1
)
```

Per comodità, trasformiamo l'oggetto `fit` in un oggetto di classe `stanfit`:

```
stanfit <- rstan::read_stan_csv(fit$output_files())
```

Il contenuto dell'oggetto `stanfit` si può esaminare nel modo seguente:

```
list_of_draws <- extract(stanfit)
print(names(list_of_draws))
#> [1] "theta"   "y_rep"   "log_lik" "lp__"
```

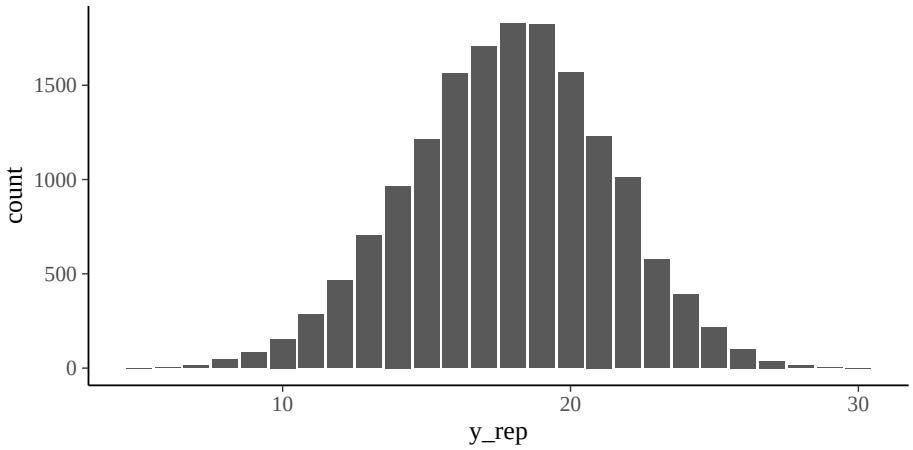
Occupiamoci ora della distribuzione predittiva a posteriori[1]. Usando l'oggetto `stanfit` creiamo `y_bern`:

[1] Un approfondimento di questa analisi statistica è fornita nell'Appendice **??**

```
y_bern <- list_of_draws$y_rep
dim(y_bern)
#> [1] 16000    30
head(y_bern)
#>
#> iterations [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#>       [1,]    0    1    0    0    0    1    1    0    1
#>       [2,]    1    0    1    1    1    1    1    1    0
#>       [3,]    1    1    1    1    1    0    1    0    0
#>       [4,]    1    0    0    1    1    1    0    1    1
#>       [5,]    0    1    1    1    1    1    1    1    0
#>       [6,]    0    0    1    1    0    1    1    0    1
#>
#> iterations [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
#>       [1,]     1     0     0     1     1     1     1     1
#>       [2,]     0     0     0     1     0     1     1     1
#>       [3,]     1     1     0     1     1     1     1     1
#>       [4,]     1     1     0     1     0     0     0     1
#>       [5,]     0     1     1     1     0     1     0     1
#>       [6,]     1     0     1     0     1     1     0     0
#>
#> iterations [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
#>       [1,]     0     0     0     0     0     1     0     0
#>       [2,]     1     1     1     0     1     1     0     0
#>       [3,]     1     1     1     1     0     1     0     1
#>       [4,]     0     0     1     1     1     1     0     0
#>       [5,]     0     1     1     1     0     1     1     1
#>       [6,]     1     0     0     0     0     1     0     1
#>
#> iterations [,26] [,27] [,28] [,29] [,30]
#>       [1,]     1     1     0     1     0
#>       [2,]     0     1     1     1     0
#>       [3,]     1     1     1     1     1
#>       [4,]     0     0     1     0     1
#>       [5,]     0     0     0     1     1
#>       [6,]     1     1     0     1     1
```

Dato che il codice Stan definisce un modello per i dati grezzi (ovvero, per ciascuna singola prova Bernoulliana del campione), ogni riga di `y_bern` include 30 colonne, ciascuna delle quali è una stima di un nuovo futuro valore possibile $y_i \in \{0, 1\}$. Per ottenere `y_rep`, ovvero, il numero previsto di "successi" in nuove future $N = 30$ prove è sufficiente calcolare la somma dei valori di ciascuna riga. Ripetendo questa operazione per tutte le 16000 righe otteniamo una stima della distribuzione predittiva a posteriori:

```
data.frame(y_rep = rowSums(y_bern)) %>%
  ggplot(aes(x = y_rep)) +
  stat_count()
```



Si noti che la simulazione di `y_rep` assume che l'ampiezza del campione di dati futuri sia uguale all'ampiezza del campione di dati osservati — nel caso presente $n = 30$.

1.3 Posterior predictive checks

La distribuzione predittiva a posteriori viene utilizzata per eseguire i cosiddetti *Posterior Predictive Checks* (PPC). I PPC vengono utilizzati per valutare l'*accuratezza predittiva* del modello, ovvero per confrontare con metodi grafici la distribuzione dei dati osservati y con la stima della distribuzione predittiva a posteriori $p(y^{rep} \mid y)$. Confrontando visivamente gli aspetti chiave dei dati previsti futuri y^{rep} e dei dati osservati y possiamo determinare se il modello è adeguato. Se il modello si adatta bene ai dati, la distribuzione di y^{rep} è molto simile alla distribuzione dei dati osservati. In altre parole, i dati osservati devono risultare plausibili alla luce della distribuzione predittiva a posteriori.

Oltre al confronto tra le distribuzioni di y e di y^{rep} è anche possibile un confronto tra la distribuzione di varie statistiche descrittive, i cui valori sono calcolati su diversi campioni y^{rep}, e le corrispondenti statistiche descrittive calcolate sui dati osservati. Vengono solitamente considerate statistiche descrittive quali la media, la varianza, la deviazione standard, il minimo o il massimo. Ma confronti di questo tipo sono possibili per qualunque statistica descrittiva. Questi confronti sono appunto chiamati PPC.

Per l'esempio presente, una volta eseguito il campionamento MCMC e ottenuto un oggetto di classe `stanfit`, è possibile usare le funzionalità del pacchetto `bayesplot` per eseguire i PPC. Nel caso presente, il campione di da-

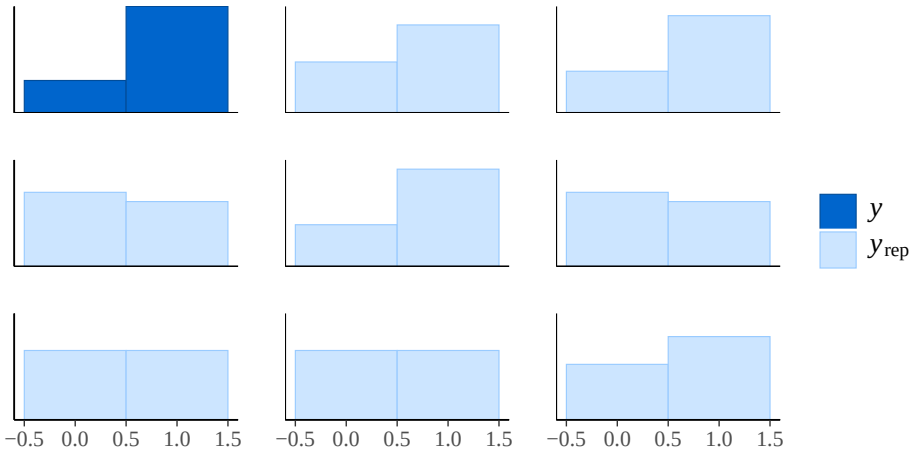
ti ha dimensioni esigue, per cui i PPC rifletteranno la grande incertenzza dell'inferenza.

Dall'oggetto `stanfit` estraiamo y^{rep}:

```
y_rep <- as.matrix(stanfit, pars = "y_rep")
dim(y_rep)
#> [1] 16000    30
```

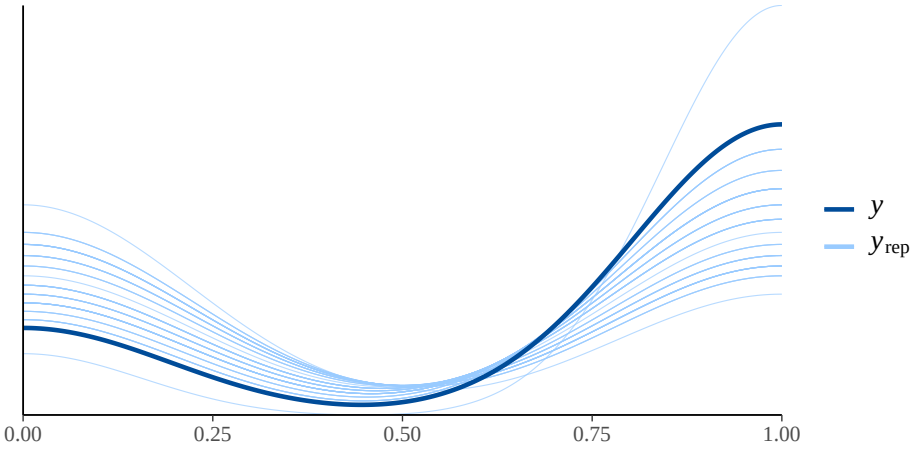
Qui sotto esaminiamo la distribuzione della y insieme alla distribuzione di 8 campioni y^{rep}:

```
ppc_hist(data_list$y, y_rep[1:8, ], binwidth = 1)
```



La corrispondenza tra le distribuzioni della y e di y^{rep} è solo parziale. Il confronto è più facile se sovrapponiamo graficamente i kernel density plot della y e di y^{rep} (qui usiamo 50 campioni y^{rep}):

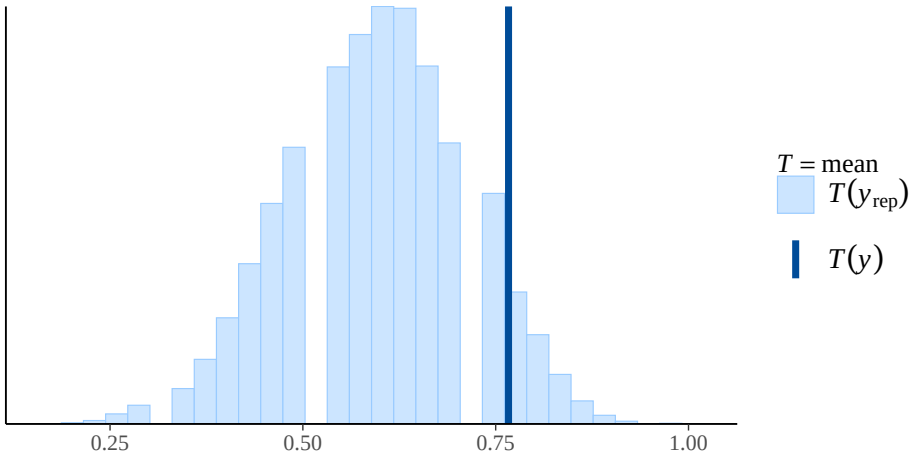
```
ppc_dens_overlay(data_list$y, y_rep[1:50, ])
```



Anche in questo caso c'è una corrispondenza solo approssimativa tra l'istogramma lisciato della y e quello di y^{rep} — ciò è dovuto al fatto che il campione è molto piccolo.

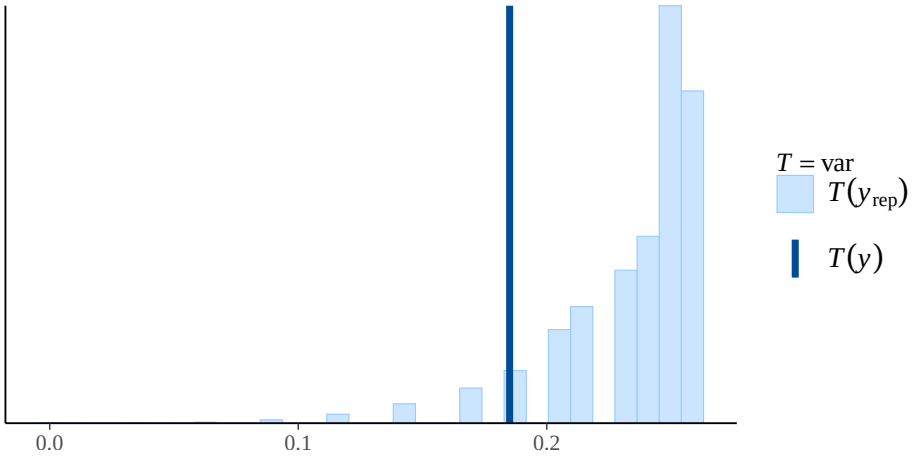
La distribuzione predittiva a posteriori è comunque in grado di rappresentare abbastanza bene la media di y:

```
ppc_stat(data_list$y, y_rep, stat = "mean")
```



Lo stesso si può dire della varianza:

```
ppc_stat(data_list$y, y_rep, stat = "var")
```



Nell'esempio successivo considereremo un campione più grande e vedremo come i PPC possano fornirci delle indicazioni sulla mancanza di adattamento del modello ai dati.

1.3.1 PPC per il modello di Poisson

Le istruzioni R qui utilizzate sono state recuperate dalla seguente pagina web. Nell'esempio discusso da Jonah Gabry e Aki Vehtari vengono usati i seguenti dati:

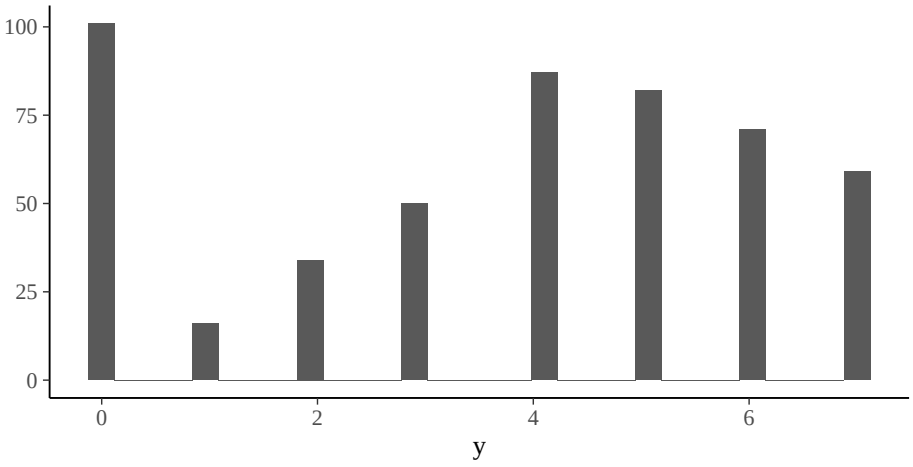
```
y <- c(0L, 3L, 5L, 0L, 4L, 7L, 4L, 2L, 3L,
       6L, 7L, 0L, 0L, 3L, 7L, 5L, 5L, 0L,
       4L, 0L, 4L, 4L, 6L, 3L, 7L, 5L, 3L,
       0L, 0L, 2L, 0L, 1L, 0L, 1L, 5L, 4L,
       4L, 2L, 3L, 6L, 4L, 5L, 0L, 7L, 7L,
       4L, 4L, 4L, 0L, 6L, 1L, 5L, 6L, 5L,
       6L, 7L, 3L, 6L, 2L, 3L, 0L, 2L, 0L,
       6L, 6L, 0L, 3L, 4L, 4L, 5L, 5L, 0L,
       5L, 7L, 5L, 5L, 6L, 4L, 2L, 3L, 4L,
       6L, 4L, 6L, 6L, 4L, 0L, 6L, 5L, 5L,
       7L, 0L, 1L, 6L, 7L, 0L, 5L, 0L, 0L,
       5L, 6L, 5L, 1L, 0L, 7L, 1L, 2L, 6L,
       5L, 4L, 0L, 4L, 0L, 4L, 4L, 6L, 3L,
       0L, 0L, 3L, 3L, 4L, 2L, 5L, 3L, 4L,
       3L, 2L, 5L, 2L, 4L, 4L, 0L, 2L, 7L,
       5L, 7L, 5L, 5L, 7L, 7L, 0L, 4L, 6L,
       0L, 4L, 6L, 7L, 4L, 0L, 4L, 1L, 5L,
       0L, 3L, 5L, 7L, 6L, 0L, 5L, 5L, 6L,
       7L, 6L, 7L, 3L, 4L, 3L, 7L, 7L, 2L,
       5L, 4L, 5L, 5L, 0L, 6L, 2L, 4L, 5L,
```

```
       4L, 0L, 0L, 5L, 5L, 7L, 7L, 0L, 3L,
       0L, 3L, 3L, 6L, 1L, 4L, 2L, 0L, 4L,
       7L, 5L, 5L, 0L, 3L, 7L, 0L, 6L, 6L,
       4L, 1L, 6L, 7L, 6L, 0L, 3L, 6L, 4L,
       7L, 0L, 5L, 5L, 4L, 0L, 0L, 2L, 4L,
       6L, 0L, 5L, 0L, 2L, 7L, 2L, 7L, 5L,
       4L, 6L, 2L, 4L, 0L, 4L, 0L, 0L, 3L,
       5L, 4L, 3L, 5L, 5L, 7L, 7L, 0L, 6L,
       4L, 5L, 1L, 5L, 3L, 5L, 5L, 5L, 0L,
       2L, 7L, 6L, 2L, 3L, 2L, 5L, 4L, 7L,
       6L, 7L, 3L, 3L, 4L, 4L, 6L, 4L, 6L,
       7L, 1L, 5L, 6L, 3L, 3L, 6L, 3L, 4L,
       0L, 7L, 0L, 3L, 6L, 5L, 0L, 0L, 0L,
       5L, 4L, 4L, 0L, 4L, 7L, 5L, 5L, 3L,
       3L, 0L, 0L, 5L, 4L, 0L, 7L, 6L, 0L,
       6L, 2L, 0L, 6L, 1L, 0L, 4L, 0L, 4L,
       3L, 0L, 4L, 5L, 5L, 7L, 6L, 6L, 5L,
       4L, 7L, 0L, 6L, 4L, 7L, 7L, 5L, 0L,
       1L, 4L, 7L, 6L, 4L, 5L, 4L, 7L, 2L,
       5L, 2L, 6L, 3L, 2L, 7L, 4L, 3L, 4L,
       6L, 6L, 6L, 6L, 7L, 1L, 0L, 0L, 7L,
       7L, 4L, 2L, 4L, 5L, 5L, 7L, 4L, 1L,
       7L, 6L, 5L, 6L, 5L, 4L, 0L, 0L, 7L,
       0L, 0L, 5L, 6L, 6L, 3L, 6L, 0L, 0L,
       0L, 4L, 4L, 3L, 0L, 7L, 5L, 4L, 2L,
       7L, 0L, 4L, 0L, 0L, 2L, 4L, 5L, 0L,
       4L, 2L, 5L, 2L, 0L, 6L, 6L, 3L, 6L,
       0L, 2L, 5L, 0L, 0L, 0L, 6L, 0L, 0L,
       6L, 5L, 4L, 6L, 4L, 5L, 5L, 4L, 0L,
       3L, 4L, 3L, 3L, 5L, 3L, 4L, 5L, 7L,
       0L, 0L, 1L, 4L, 6L, 3L, 5L, 7L, 6L,
       6L, 5L, 0L, 5L, 4L, 0L, 0L, 2L, 6L,
       0L, 6L, 0L, 4L, 5L, 6L, 3L, 4L, 2L,
       3L, 4L, 0L, 5L, 0L, 0L, 0L, 0L, 3L,
       4L, 7L, 6L, 7L, 7L, 3L, 4L, 4L, 7L,
       4L, 5L, 2L, 5L, 6L)

N <- length(y)
```

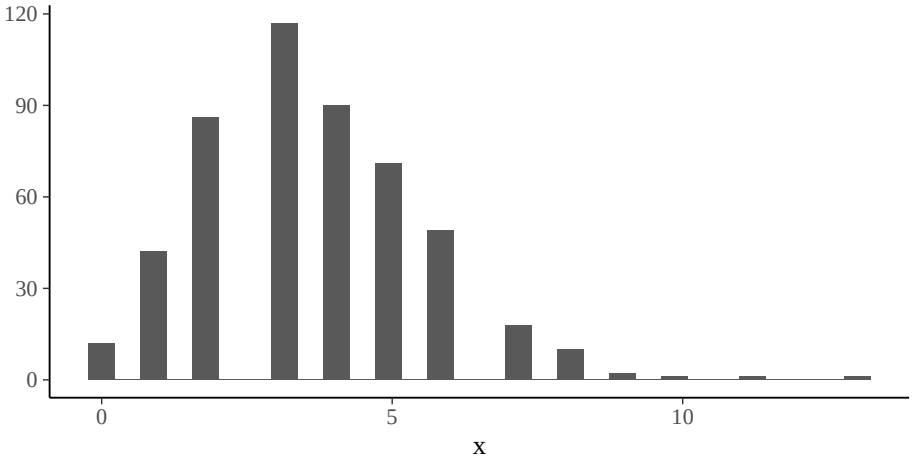
Per questi dati sembra appropriato un modello di Poisson.

```
qplot(y)
```



Un istogramma di un campione casuale della stessa ampiezza di y tratto dalla distribuzione di Poisson è il seguente:

```
x <- rpois(N, mean(y))
qplot(x)
```



È chiaro però che i due istogrammi sono molto diversi.

Anche se sospettiamo che non sarà un buon modello per questi dati, è comunque una buona idea iniziare adattando ai dati il modello più semplice, ovvero quello di Poisson. Partendo da lì possiamo poi cercare di capire in che modo il modello è inadeguato.

```
modelString <- "
data {
  int<lower=1> N;
  int<lower=0> y[N];
}
parameters {
  real<lower=0> lambda;
}
model {
  lambda ~ exponential(0.2);
  y ~ poisson(lambda);
}
generated quantities {
  int y_rep[N];
  for (n in 1:N) {
    y_rep[n] = poisson_rng(lambda);
  }
}
"
writeLines(modelString, con = "code/code_poisson.stan")
```

Creiamo un oggetto di tipo `list` dove inserire i dati:

```
data_list <- list(
  y = y,
  N = length(y)
)
```

Adattando il modello ai dati

```
file <- file.path("code", "code_poisson.stan")
mod <- cmdstan_model(file)
fit <- mod$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = SEED,
  chains = 4L,
  parallel_chains = 4L,
  refresh = 0,
  thin = 1
)
```

otteniamo la seguente stima del parametro λ:

```
fit$summary(c("lambda"))
#> # A tibble: 1 x 10
#>   variable  mean median     sd    mad    q5   q95  rhat
#>   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>
#> 1 lambda    3.66   3.66 0.0839 0.0837  3.52  3.80  1.00
#> # ... with 2 more variables: ess_bulk <dbl>, ess_tail <dbl>
```

Confrontiamo λ con la media di y:

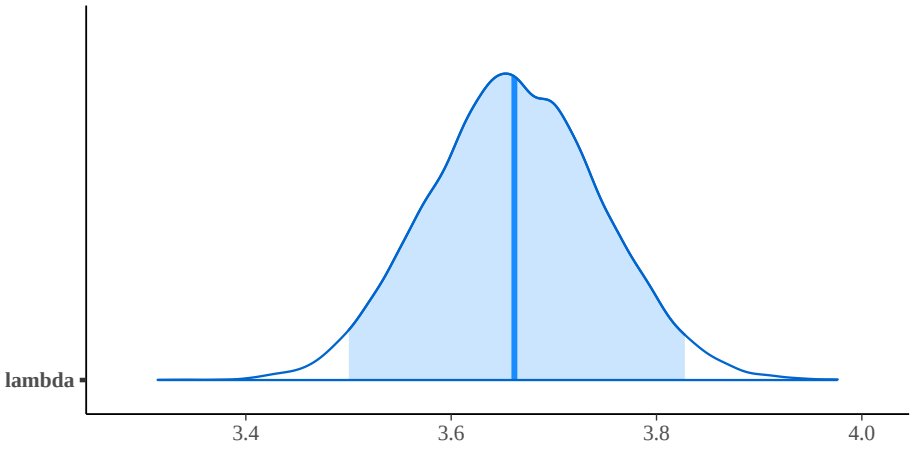
```
mean(y)
#> [1] 3.662
```

Anche se trova la media giusta, il modello non è comunque adeguato a prevedere le altre proprietà della y. Trasformiamo `fit` in un oggetto `stanfit`:

```
stanfit <- rstan::read_stan_csv(fit$output_files())
```

La distribuzione a posteriori di λ è

```
lambda_draws <- as.matrix(stanfit, pars = "lambda")
mcmc_areas(lambda_draws, prob = 0.95) # color 95% interval
```

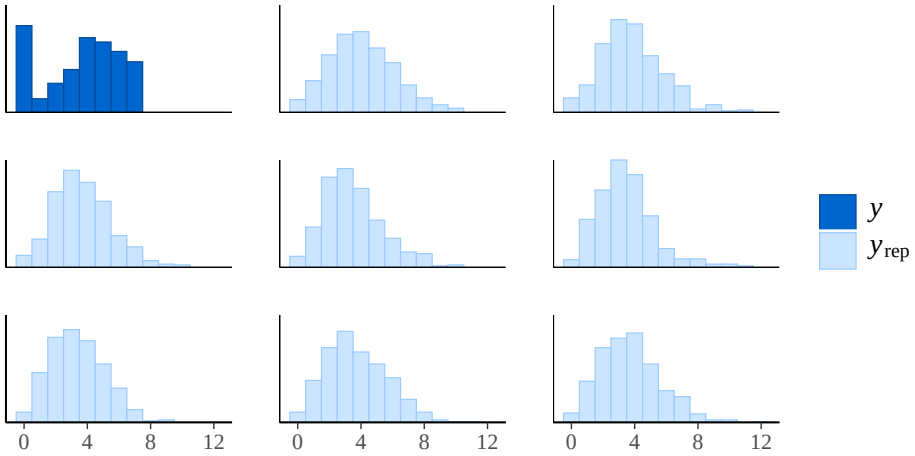


Estraiamo y^{rep} dall'oggetto `stanfit`:

```
y_rep <- as.matrix(stanfit, pars = "y_rep")
dim(y_rep)
#> [1] 16000   500
```

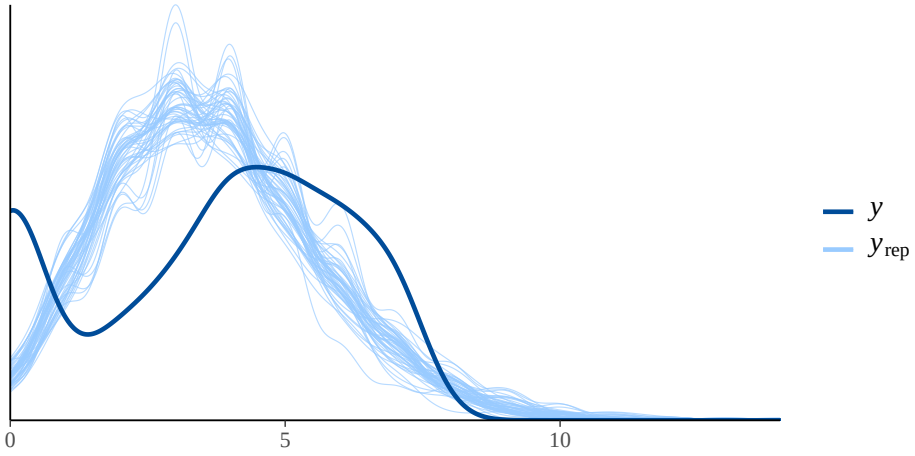
Il confronto tra l'istogramma della y e gli istogrammi di diversi campioni y^{rep} mostra una scarsa corrispondenza tra i due:

```
ppc_hist(y, y_rep[1:8, ], binwidth = 1)
```



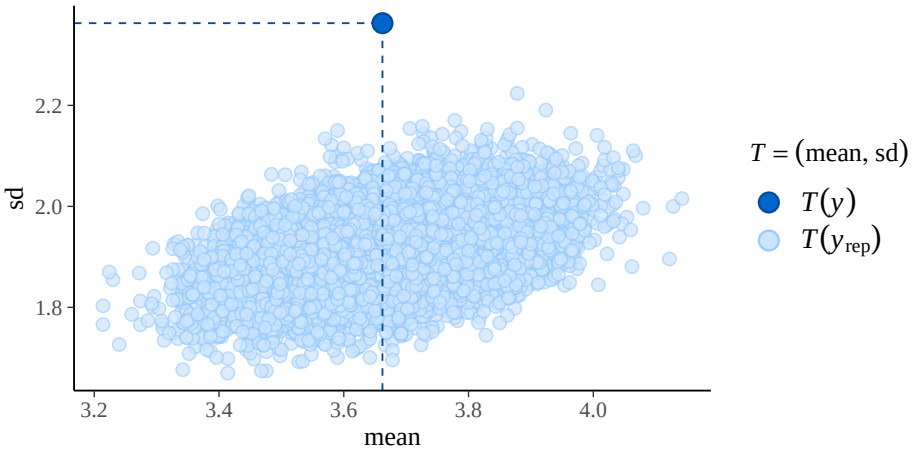
Alla stessa conclusione si giunge tramite un confronto tra la funzione di densità empirica della y e quella di diversi campioni y^{rep}:

```
ppc_dens_overlay(y, y_rep[1:50, ])
```



Eseguiamo ora i PPC per la media e la deviazine standard:

```
ppc_stat_2d(y, y_rep, stat = c("mean", "sd"))
```



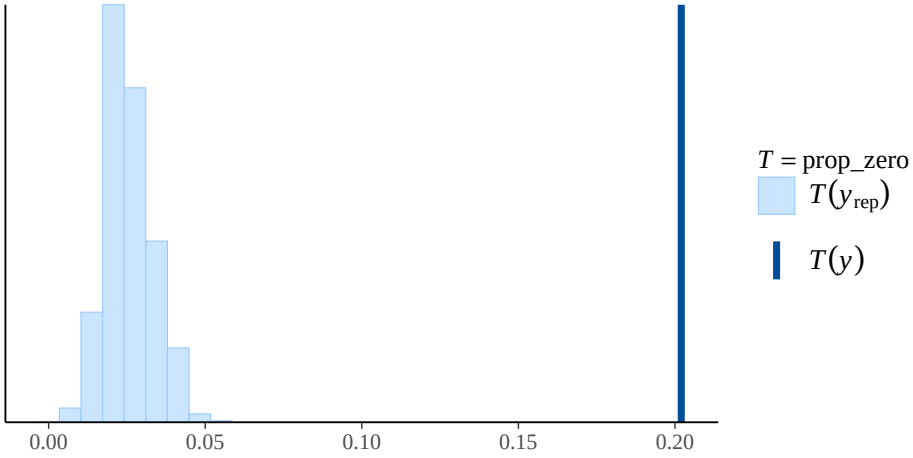
Mentre la media viene riprodotta accuratamente dal modello (come avevamo visto sopra), ciò non è vero per la deviazione stanard dei dati. La domanda è quale sia l'origine di questa mancanza di adattamento.

Calcoliamo ora la proporzione di zeri in y e nei campioni y^{rep}.

```
prop_zero <- function(x) mean(x == 0)
print(prop_zero(y))
#> [1] 0.202
```

Eseguendo il PPC sulla proporzione di zeri

```
ppc_stat(y, y_rep, stat = "prop_zero")
```



notiamo che il modello non è assolutamente in grado di catturare la proporzione di casi nei quali la variabile Y assume il valore 0. In altri termini, i dati presentano un'inflazione di valori 0 rispetto a quelli che sono previsti da un modello di Poisson. Questo è un problema che si verifica spesso nei dati empirici.

Per ovviare al problema dell'inflazione di valori pari a 0 è possibile definire un modello di tipo "hurdle" che consente la presenza di una proporzione di valori pari a 0 maggiore di quanto normalmente previsto dalla distribuzione di Poisson. Senza entrare nei dettagli di come questo viene fatto, Gabry e Vehtari definiscono il seguente modello:

```
modelString2 <- "
data {
  int<lower=1> N;
  int<lower=0> y[N];
}
transformed data {
  int U = max(y);  // upper truncation point
}
parameters {
  real<lower=0,upper=1> theta; // Pr(y = 0)
  real<lower=0> lambda; // Poisson rate parameter (if y > 0)
}
model {
  lambda ~ exponential(0.2);

  for (n in 1:N) {
    if (y[n] == 0) {
      target += log(theta);  // log(Pr(y = 0))
    } else {
      target += log1m(theta);  // log(Pr(y > 0))
      y[n] ~ poisson(lambda) T[1,U];  // truncated poisson
    }
  }
}
generated quantities {
  real log_lik[N];
  int y_rep[N];
  for (n in 1:N) {
    if (bernoulli_rng(theta)) {
      y_rep[n] = 0;
    } else {
      int w;  // temporary variable
      w = poisson_rng(lambda);
      while (w == 0 || w > U)
        w = poisson_rng(lambda);
```

```
      y_rep[n] = w;
    }
    if (y[n] == 0) {
      log_lik[n] = log(theta);
    } else {
      log_lik[n] = log1m(theta)
    + poisson_lpmf(y[n] | lambda)
    - log_diff_exp(poisson_lcdf(U | lambda),
                   poisson_lcdf(0 | lambda));
    }
  }
}
"
writeLines(modelString2, con = "code/code_poisson_hurdle.stan")
```

Adattiamo il modello ai dati:

```
file2 <- file.path("code", "code_poisson_hurdle.stan")
mod2 <- cmdstan_model(file2)

fit2 <- mod2$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = SEED,
  chains = 4L,
  parallel_chains = 4L,
  refresh = 0,
  thin = 1
)
```

In questo caso otteniamo una stima di λ diversa da quella ottenuta in precedenza:

```
fit2$summary(c("lambda", "theta"))
#> # A tibble: 2 x 10
#>   variable  mean median     sd    mad    q5   q95  rhat
#>   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>
#> 1 lambda   5.31   5.30  0.163  0.164  5.05  5.58   1.00
#> 2 theta    0.203  0.203 0.0177 0.0179 0.175 0.233  1.00
#> # ... with 2 more variables: ess_bulk <dbl>, ess_tail <dbl>
```

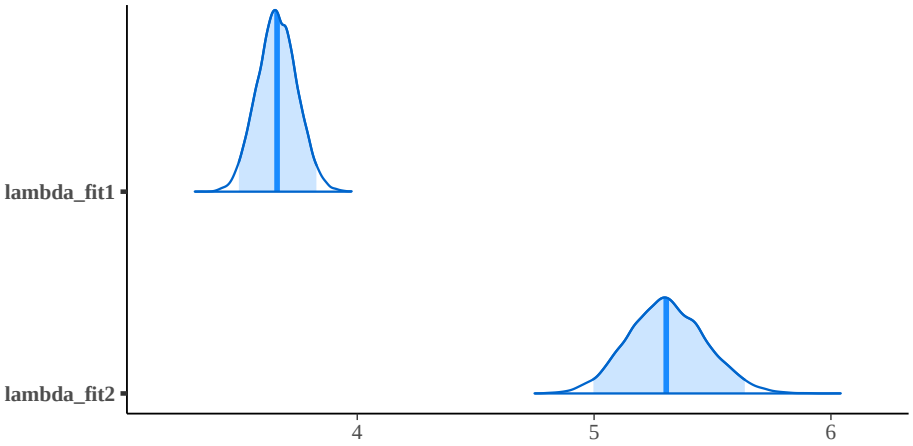
Si noti che il parametro θ viene usato per modellizzare l'eccesso di valori 0.

Eseguiamo un confronto tra le distribuzioni a posteriori di λ per i due modelli si ottiene nel modo seguente:

```
stanfit2 <- rstan::read_stan_csv(fit2$output_files())
```

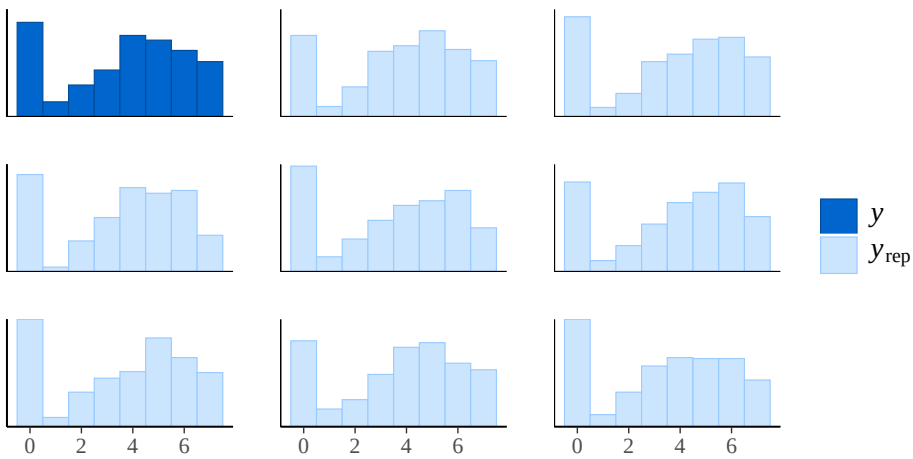
```
lambda_draws2 <- as.matrix(stanfit2, pars = "lambda")

lambdas <- cbind(lambda_fit1 = lambda_draws[, 1],
                 lambda_fit2 = lambda_draws2[, 1])
mcmc_areas(lambdas, prob = 0.95) # color 95% interval
```



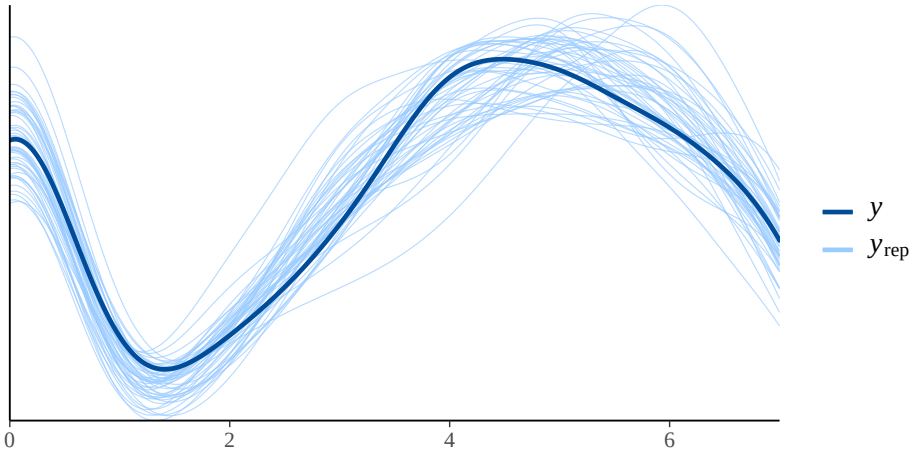
Rifacciamo i grafici esaminati in precedenza (e alcuni altri), ma questa volta eatraendo y^{rep} da `fit2`:

```
y_rep2 <- as.matrix(stanfit2, pars = "y_rep")
ppc_hist(y, y_rep2[1:8, ], binwidth = 1)
```

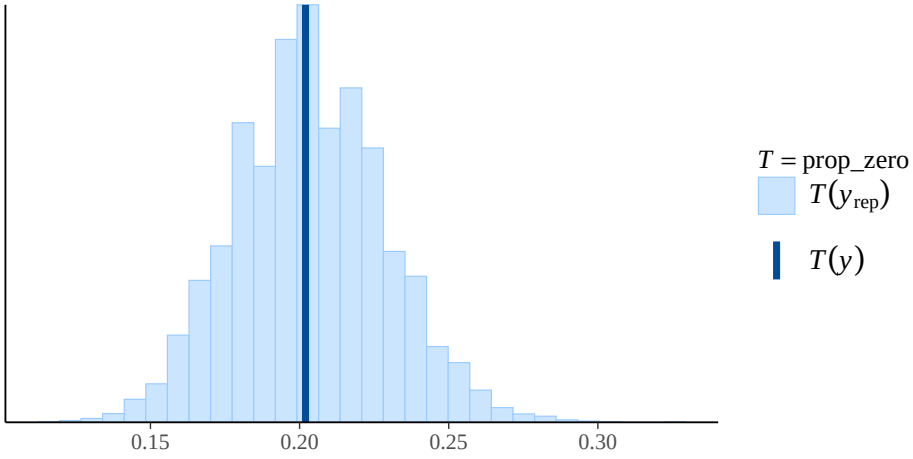


In questo caso la distribuzione di y^{rep} è molto simile alla distribuzione di y.

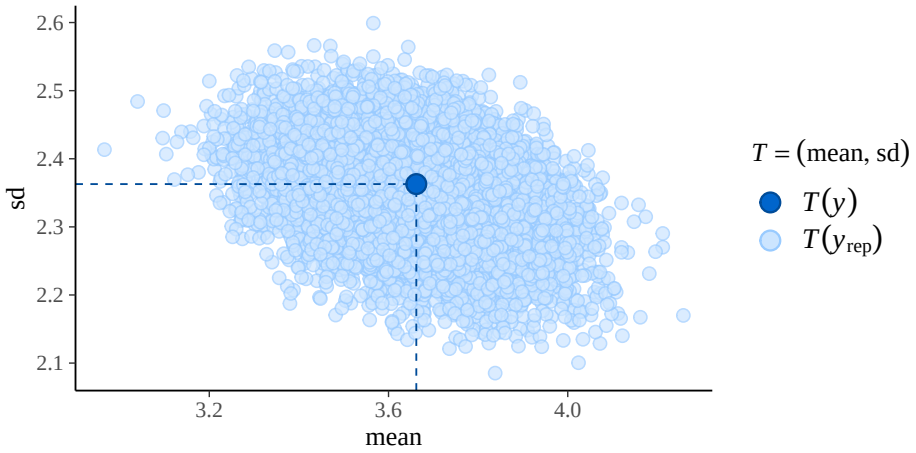
```
ppc_dens_overlay(y, y_rep2[1:50, ])
```




```
ppc_stat(y, y_rep2, stat = "prop_zero")
```




```
ppc_stat_2d(y, y_rep2, stat = c("mean", "sd"))
```



In conclusione, l'accuratezza predittiva del modello "hurdle" è chiaramente migliore di quella del modello di Poisson.

Considerazioni conclusive

Questo capitolo abbiamo discusso i controlli predittivi a posteriori. A questo proposito è necessario notare un punto importante: i controlli predittivi a posteriori, quando suggeriscono un buon adattamento del modello alle caratterische dei dati previsti futuri y^{rep}, non forniscono una forte evidenza della capacità del modello di generalizzarsi a nuovi campioni di dati. Infatti, una tale evidenza sulla generalizzabilità del modello può essere solo fornita da studi di *cross-validation*, ovvero da studi nei quali viene utilizzato un *nuovo* campione di dati. D'altra parte, invece, se i PPC mostrano un cattivo adattamento del modello ai dati previsti futuri, questo fornisce una forte evidenza di una errata specificazione del modello.

Bibliografia

Gelman, A., Hwang, J., and Vehtari, A. (2014). Understanding predictive information criteria for bayesian models. *Statistics and Computing*, 24(6):997–1016.

Zetsche, U., Bürkner, P.-C., and Renneberg, B. (2019). Future expectations in clinical depression: Biased or realistic? *Journal of Abnormal Psychology*, 128(7):678–688.