Contents

Chapter 1

Introduction to the Bayesian framework

Remark 1. The topics of this block are contained in Chapters 2 and 3 of *Lambert B. (2018) A Student's Guide to Bayesian Statistics, Sage.*
Another reference is *Hoff, P. D. (2009). A first course in Bayesian statistical methods. Springer Science & Business Media. ISBN: 978-0-387-92299-7.* For this part section chapter 1 (1.1 - 1.4) and 2(2.1 - 2.6) while for remarks on important statistical distributions see pages 253-258.

1.1 Introduction to Bayesian reasoning

1.1.1 The concept of event

Definition 1.1.1 (Event)**.** In an experimental situation, an event is something physical/observable, stated by a proposition, which after the experiment has been conducted can be either true (T, event occurred) or false (F).

Example 1.1.1 (Counterexample)**.** The proportion of heads when tossing a coin is not an event.

1.1.2 Probability of an event

Remark 2. Two main idea/interpretation of probability are available.

Definition 1.1.2 (**Objective probability** (*logical* or *frequentist*))**.** Probability is a physical property of the event.

Important remark 1 (Critique). The definition is linked tot he concept of repeatable events but *repeatable events do non exist*: only past experiences under similar conditions do exist.

Definition 1.1.3 (**Subjective probability**)**.** Probability is the measure of plausibility that an individual assigns to an (uncertain) event.
The probability of an event E, for a certain individual (in a certain moment) is the price $P(E) = p$ that he considers right to pay to participate at a bet where he will win 1 if E occurs or 0 if it doesn't.

Remark 3. In this definition, probability:

- is not a physical property of the event, rather a formalization of the beliefs (and information) that the individual possesses about the event;
- can be different for different individuals (thus the term "subjective");

Remark 4. It is important to state that also what is *not observable* may receive a probability

Remark 5. The classical statistician does not accept subjective probability, since the last is not related to the concept of frequency.

1.1.3 Coherence of subjective probability

Definition 1.1.4 (Coherence of subjective probability)**.** A probability assessment about the n events $E_1, E_2, \ldots, E_n$ is said to be *coherent* if no combination of bets on these events allows a sure/positive win (independently on the events E_i, $i = 1, \ldots, n$ that actually occur).

NB: These n events are alternative I think, think a partition ...

Remark 6. Necessary and sufficient condition for coherence of the subjective probability is expressed by the following theorem.

Theorem 1.1.1. *A necessary and sufficient condition for $P(E)$ coherence is that $0 \leq P(E) \leq 1$. In particular, if $P(E) = 0$ the event is impossible, while if $P(E) = 1$ the event is said certain.*

Proof. Let $p = P(E)$ the *price* and let assume to bet S about the occurrence of E . When

NB: per il gioco dobbiamo pagare p per singolo euro di vincita, se scommettiamo su S di vincita dobbiamo pagare pS

- E occurs, the gain obtained by the bet is the wager/win minus the price for the wager itself:

$$W(E) = S - pS = S(1 - p) = S(1 - P(E))$$

- E does not occur, the gain is negative (just the price paid):

$$W(\bar{E}) = -pS = -P(E) \cdot S$$

How choosing p and S in order avoiding to obtain a sure win (positive earning in any case)?

- if $p < 0$ it would be enough to bet a positive wager $S > 0$ to guarantee a sure win.
- if $p > 1$ it would be enough to bet a negative wager $S < 0$ to guarantee the sure win.

Thus it follows that to avoid a sure win it must be that $0 \leq P(E) \leq 1$.
Furthermore:

- if the event E is certain, it's payoff is $W(E) = S(1 - p)$: the only way to avoid a sure win is to set $W(E) = 0$, by fixing $P(E) = 1$.

- if E is impossible, $\bar{E}$ is certain, its payoff is $W(\bar{E}) = -pS$: in order to avoid sure wins it has to be $W(\bar{E}) = 0$, from which $p = P(E) = 0$ (case of impossible events).

□

Theorem 1.1.2. *The probability of the union of many events (incompatible when considered in couples) is the sum of their probabilities.*

Proof. Omitted □

1.1.4 The axiomatic Kolmogorov framework

Remark 7. Allows any computations regarding probabilities, exploiting the analogy between probability and measure, and between mathematical expectation and integration according to Lebesgue.
It needs at least the definition of an algebra. or better, of a σ-algebra

Definition 1.1.5 (Algebra)**.** A class A of subsets of Ω is an algebra if:

1. $\Omega \in A$;
2. if $E_i \in A$ then $\bar{E}_i \in A$;
3. *finite additivity*: $\cup_{i=0}^{n} E_i \in A$ for events that are incompatible two by two

Definition 1.1.6 (σ-algebra)**.** Has the same two first properties, but the third consists of complete rather than finite additivity: $\cup_{i=0}^{\infty} E_i \in A$.

Remark 8. Complete additivity cannot be derived from finite additivity with the application of a limit.

1.1.5 Differences between classical/Bayesian statistics

Parameter of interest for inference:

- frequentist: fixed in the population but unknown
- bayesian: unknown but not necessarily fixed, can be expressed as random variable. In the Bayesian approach, parameters can be viewed from two perspectives. Either are truly *varying*, or we view our knowledge about the parameters as imperfect. The fact that we obtain different etimates of parameters from different studies can be taken to reflext either of these two views

Interpretation/concept of probability:

- frequentist: as relative frequence occurred in repeated experiments. Thus can be applied to events that can be reèeated
- bayesian: degree of credibility on a certain hypothesis. Thus can be applied to unique events and series of events.

Experiment assumption:

- frequentist: **independence** between units

- bayesian: **exchangeability** between units

Statistical inference is based on:

- frequentist: idea that the experiment may be repeated and that a sampling distribution of a statistic of interest (that varies in the set of possible samples, sampling space) can be obtained at least theoretically.
  Some critiques:
  - regarding statistic distribution decisions are taken on the basis of something that never will be observed: we have only one sample.
  - regarding p-values we calculate the probability of obtaining a sample as extreme as, or more extreme than, the one actually obtained, assuming a certain hypothesis to be true. Eg in a test on mean height of people (null 1.50) we calculate hte probability of obtaining a height greater or equal to 2.5m, assumin the null to be true. However we did not actuially witness any individual with a height greater than 2.5. So in frequentist inference we must invent fictitious samples to test a hypothesis
- Final/posterior distribution: the *main assumption* is that prior/initial probability distribution can/has to be be assigned.
  A critique is that different starting point could yield to different final results. A countercritique is that any analysis involve subjectivity (which can be more or less evident/transparent)[1]

Parameter estimation:

- frequentist: needs a theory of estimation. Eg for likelihood based inference the likelihood function provides all information contained in the sample and, for that specific sample, it measures the plausibility of the various alternatives for expressing how the phenomenon is. However (critique):
  - Several techniques of classical statistics do not respect the likelihood principle, that states that two experiments give the same information if the corresponding likelihood functions differ for a multiplicative constant.
  - The estimate may depend on how the experiment is developed
- bayesian: needs description/synthesis of the final/posterior distribution.

Role of personal evaluations:

- frequentist: the choice of the experiment is needed, as well as the choice of the procedures to adopt. Personal evaluations remain external (they are not quantified: the problem *appears* as dealt in an objective way.)
- bayesian: all knowledge is formally incorporated in the prior/initial distribution of parameter of interest

https://towardsdatascience.com/statistics-are-you-bayesian-or-frequentist-4943f953f21b

[1] Eg: the simple linear regression model is often used, without justification, in applied Frequentist analyses. This model makes assumptions about the relationships between the dependent and independent variables that may, or may not, be true. In a Bayesian approach, we more typically build our models from the ground up, meaning that we are more aware of the assumptions inherent in the approach.

1.1.6 Probability distributions vs likelihoods

Example 1.1.2 (Binomial Case: Difference between Probability Distributions and Likelihoods)**.** Consider the throw of 10 coins with a given probability of head θ. Table 1.1 describes the probability of all possible outcomes (obtain from 0 to 10 heads) using a binomial distribution for some different values of the parameter θ:

- each line contains the probability distribution of the outcomes from 0 to 10 for different values of the parameter (the values of each line sum to 1)
- each column likelihood (it is a function, not a probability distribution) of some values of the parameters for the possible different results (the sum of the probabilities of each column is not 1)

	Y=0	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10	Sum
0	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
0.1	0.349	0.387	0.194	0.057	0.011	0.001	0.000	0.000	0.000	0.000	0.000	1.000
0.2	0.107	0.268	0.302	0.201	0.088	0.026	0.006	0.001	0.000	0.000	0.000	1.000
0.3	0.028	0.121	0.233	0.267	0.200	0.103	0.037	0.009	0.001	0.000	0.000	1.000
0.4	0.006	0.040	0.121	0.215	0.251	0.201	0.111	0.042	0.011	0.002	0.000	1.000
0.5	0.001	0.010	0.044	0.117	0.205	0.246	0.205	0.117	0.044	0.010	0.001	1.000
0.6	0.000	0.002	0.011	0.042	0.111	0.201	0.251	0.215	0.121	0.040	0.006	1.000
0.7	0.000	0.000	0.001	0.009	0.037	0.103	0.200	0.267	0.233	0.121	0.028	1.000
0.8	0.000	0.000	0.000	0.001	0.006	0.026	0.088	0.201	0.302	0.268	0.107	1.000
0.9	0.000	0.000	0.000	0.000	0.000	0.001	0.011	0.057	0.194	0.387	0.349	1.000
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000
Sum	1.491	0.829	0.906	0.910	0.909	0.909	0.909	0.910	0.906	0.829	1.491	

Table 1.1: Probability vs likelihood

Important remark 2. The examples concerning events are very simple/intuitive. When probabilities of events are substituted by the probabilities of random variables and the notion of statistical model is introduced, things get a little more complicated.

1.2 Bayes theorem for events

Theorem 1.2.1 (Compound probability theorem)**.** *The conditional probability of one event E_1 given another E_2 can be written as*

$$P(E_1|E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)} = \frac{P(E_1)P(E_2|E_1)}{P(E_2)}$$

Remark 9. In the case of independence, the relationship becomes even simpler.

Remark 10. Bayes theorem starts from the compound probability theorem by changing the denominator.

Theorem 1.2.2 (Bayes theorem)**.** *Considering a finite partition $\{H_i\}$ $i = 1, \ldots, I$ of the certain event, and an event B with probability $P(B) > 0$.*

	D	$\bar{D}$
T	0.95	0.02
$\bar{T}$	0.05	0.98
	1	1

Table 1.2: Experiment and conditional distributions

The probabilities of B conditional on the H_i are known: $P(B|H_i)$, $i = 1, \dots, I$. Then

$$P(H_i|B) = \frac{P(H_i)P(B|H_i)}{P(B)} = \frac{P(H_i)P(B|H_i)}{\sum_{i=1}^{I} P(H_i)P(B|H_i)}.$$

1.2.1 Examples on (discrete) prior for events

1.2.1.1 Rare disease test (asymmetric/informative prior)

NB prof: cambio notazione T test e D disease per mia maggior facilità memorizzazione risp A e B

Remark 11. If an individual is positive to the medical test for a disease, which is the probability that she/he is sick? Much of the answer depends on the contribution of the prior/*non modifiable* situation

Example 1.2.1 (Rare disease)**.** The event D is suffering from a rare disease, with $P(D) = 0.001$ (so $P(\bar{D}) = 0.999$). The prevalence of the disease $P(D)$ is our prior/starting point, the *state of the world*: it cannot be changed (neither $P(\bar{D})$).
The event T is being positive at a medical test for that disease. Technology can provide probabilities of T that are conditional on being sick or not where hold:

$$P(T|D) + P(\bar{T}|D) = 1$$
$$P(T|\bar{D}) + P(\bar{T}|\bar{D}) = 1$$

An experiment T is performed conditional on D and results are organized in tables like 1.2 where two distributions are available (conditional on the two states of the world) in columns.
Now some naming:

NB prof: qui nelle originali false positive e negative erano invertiti

- $P(\bar{T}|D)$ is the probability of false positive: the disease is present but the test is negative (the individuals are lost for following tests). It is the second type error β (in this case $\beta = 0.05$).
- $P(T|D)$ is the *sensitivity* $1 - \beta$ (or power elsewhere); here $1 - \beta = 0.95$
- $P(T|\bar{D})$ is the probability of false positives: the result of the test is positive even if the individual has not the disease; it's the first type error α (here $\alpha = 0.02$).
- $P(\bar{T}|\bar{D})$ is the *specificity* $1 - \alpha$ (here $1 - \alpha = 0.98$).

The quantity $P(D|T)$ (probability of being diseased given a positive test, also said positive predictive value) is computed using Bayes theorem as:

$$P(D|T) = \frac{P(D)P(T|D)}{P(T)}$$

	D	$\bar{D}$
T	0.99	0.005
$\bar{T}$	0.01	0.995
	1	1

Table 1.3: Techology improvement

In order to compute the denominator $P(T)$ at (i.e. the probability of being positive to the test, irrespective of the state of the world) the weighted sum of the probabilities of being sick for all possible states of the world has to be computed (weighs are $P(D)$ and $P(\bar{D})$ respectively). In this way the possible states of the world have been *integrated out* of the formula.

$$\begin{aligned} P(T) &= P(D)P(T|D) + P(\bar{D})P(T|\bar{D}) \\ &= 0.001 \cdot 0.95 + 0.999 \cdot 0.02 \\ &= 0.02093 \end{aligned}$$

We thus have that

$$P(\bar{T}) = 1 - P(T) = 1 - 0.02093 = 0.9791$$

and

$$\begin{aligned} P(D|T) &= \frac{P(D)P(T|D)}{P(T)} = \frac{0.001 \cdot 0.95}{0.02093} = 0.04539 \\ P(\bar{D}|T) &= \frac{P(\bar{D})P(T|\bar{D})}{P(T)} = \frac{0.999 \cdot 0.02}{0.02093} = 0.9546 \end{aligned}$$

Note that $0.04539 + 0.9546 = 1$, coherently.

Example 1.2.2 (Techology improvements)**.** If $P(A) = 0.001$ and $P(\bar{A}) = 0.999$ remain unchanged, while technology improves as in the conditional distribution of table 1.3, the results become

$$\begin{aligned} P(D|T) &= 0.1654 \\ P(\bar{D}|T) &= 0.8346 \end{aligned}$$

So probability of being diseased given a positive test is increased due to a greater confidence in the testing system (among the positives

Example 1.2.3 (Less rare disease)**.** If disease is less rare, i.e. $P(D) = 0.01$

- under the first hypothesis on technological development we have $P(D|T) = 0.324$ and $P(\bar{D}|T) = 0.676$,
- under the second hypothesis on technological development $P(D|T) = 0.666$ and $P(\bar{D}|T) = 0.333$.

So in order to obtain $P(D|T) > 0.5$, the disease cannot be too rare and technological development must be very high.

	C	$\bar{C}$
Compatible DNA	0.999	0.02
Not compatible DNA	0.001	0.98
	1	1

Table 1.4: genetic test performance

1.2.1.2 DNA test for a crime (ignorance prior))

Example 1.2.4 (Crime and genetic tests)**.** A crime has been committed and there's a suspected: the event C is "the suspected committed the crime". Initially no one knows whether the suspected committed the crime or not so our prior (*ignorance prior*) is

$$P(C) = P(\bar{C}) = 0.5$$

A genetic test is available and in situation like this its performance are reported in table 1.4.
What is needed is $P(C|\text{compatible DNA})$ which can be obtained via the Bayes theorem as

$$P(C|\text{compatible DNA}) = 0.9804$$

In other words, if the DNA of the suspected is compatible, the probability that the suspected committed the crime strongly increases compared to $P(C) = 0.5$.

1.2.2 Prior odd ratios, Bayes factor

Some definitions using the notation of D (disease) and T (test)

$$\text{Prior odds ratio in favor} = \frac{P(D)}{1 - P(D)} = \frac{P(D)}{P(\bar{D})}$$
$$\text{Prior odds ratio against} = \frac{P(\bar{D})}{P(D)}$$

We're interested in:

$$\text{Posterior odds ratio in favor } = \frac{P(D|T)}{P(\bar{D}|T)}$$

to compute it note first that

$$P(D|T) = \frac{P(D)P(T|D)}{P(D)P(T|D) + P(\bar{D})P(T|\bar{D})}$$
$$P(\bar{D}|T) = \frac{P(\bar{D})P(T|\bar{D})}{P(D)P(T|D) + P(\bar{D})P(T|\bar{D})}$$

Since the denominators are the same their computation is not needed and thus

$$\text{Posterior odds ratio in favor} = \frac{P(D|T)}{P(\bar{D}|T)} = \frac{P(D)P(T|D)}{P(\bar{D})P(T|\bar{D})} = \frac{P(D)}{P(\bar{D})}r$$

The ratio:

$$r = \frac{P(T|D)}{P(T|\bar{D})}$$

is known as **Bayes factor**; since we are speaking of events, it depends on data and not on priors.
So posterior odds ratio in favor can be written as prior odds ratio in favor (which contains info on the prior only) times the Bayes factor (which contains info on the data only).
Note that if $P(D) = 0.5$ (*ignorance prior*), the prior odds ratio in favor is 1 and all the decision depends on the Bayes factor/data.

Example 1.2.5 (Rare disease (continued))**.** If $P(D) = 0.001$:

$$\text{Prior odds ratio in favor} = \frac{P(D)}{P(\bar{D})} = \frac{0.001}{0.999} = 0.001001$$
$$\text{Prior odds ratio against} = \frac{P(\bar{D})}{P(D)} = \frac{0.999}{0.001} = 999$$

The Bayes factor is

$$r = \frac{P(T|D)}{P(T|\bar{D})} = \frac{0.95}{0.02} = 47.5$$

Here the Bayes factor assumes a high > 1 value, the hypothesis of disease is seconded by/after conducing the experiment; the value of the bayes factor is something similar to hypothesis testing based on only the sample as frequentist do[2].
The posterior odds ratio is

$$\frac{P(D|T)}{P(\bar{D}|T)} = \frac{0.0045}{0.955} = 0.04712$$

Example 1.2.6 (Crime and DNA test (continued))**.** We have

$$\text{Prior odds ratio in favor} = \frac{P(C)}{P(\bar{C})} = 1$$
$$\text{Posterior odds ratio in favor} = \frac{P(C)}{P(\bar{C})} \cdot r = 1\frac{P(\text{compatible DNA}|C)}{P(\text{compatible DNA}|\bar{C})}$$
$$= \frac{0.999}{0.002} = 50.02.$$

1.3 The statistical model as the basic element for inference

1.3.1 Definition of a statistical model

Among the basic statistical models some examples can be considered.

[2]Actual testing is not used so much in Bayesian statistics because Bayesian says that everything is included in the posterior distribution.

1. Experiment with known probability of success: Bernoulli distribution.
2. Model for repeated measures: normal distribution.
3. Time of functioning of homogeneous apparatuses: exponential distribution.
4. Non parametric models.
5. Sampling without replacement.
6. Inverse sampling.

The case of *inverse sampling* deserves some comments. It illustrates how the way of conducting the experiment may determine the statistical distribution to consider.
The proportion of a characteristic in a population can be managed via the Binomial distribution (that models X as the number of successes in n trials); also X can be the number of failures before obtaining n successes leading to the following distribution (negative binomial)

$$p(x|n,\theta) = \binom{n+x-1}{x}\theta^n(1-\theta)^x$$

Support of X: set of natural numbers that are $\geq x$. $\mathbb{N} = \{x, x+1, \ldots\}$, with $0 \leq \theta \leq 1$.

Now it turns out that n is not necessarily an integer (in negative binomial can be a real number);

- if that is the case the distribution is known as Pascal distribution (special, most famous, case of negative binomial)
- if furthermore $n = 1$ then $p(x|\theta) = \theta(1-\theta)^x$, and we have a Geometric distribution.

Remark 12. Alternative ways of writing a binomial coefficient can be found in the literature, since

$$\binom{n+x-1}{x} = \binom{n-1}{n-x-1}.$$

1.4 Probability brush up

Remark 13. Idea è mettere qui in un unico posto tutti i richiami di probabilità da usare nel seguito.

1.4.1 Transformation of random variables

Proposition 1.4.1. *If X is a continuous random variable, g a monotonic function (strictly increasing or decreasing), the density function of the random variable $g(X)$, $f_{g(X)}$, is obtained as:*

$$f_{g(X)}(x) = f_X(g^{-1}(x)) \cdot \left|\frac{\partial g^{-1}(x)}{\partial x}\right| \tag{1.1}$$

1.4.2 Some recalls from the likelihood theory

The probability distribution for any variable $p(x|\theta)$ may be seen as a likelihood $L(\theta; x)$ (for sample size of $n = 1$ and considering the function as dependent on the parameter, rather than the data).
Its logarithmic transformation is

$$\log L(\theta; x) = \log p(x|\theta) = l(\theta; x).$$

now:

- the Fisher information, i.e. the information provided by an experiment, is is a probabilistic concepts that is defined over all x values, and it refers to the experiment before it is performed and is defined as

$$I(\theta; x) = -E\left[\frac{\partial^2 l(\theta; x)}{\partial \theta^2}\right] = E\left\{\left[\frac{\partial l(\theta; x)}{\partial \theta}\right]^2\right\}$$

  Thus the Fisher Information is − expectation of the second derivative of loglikelihood or the expectation of the square of the first derivative.
  For n observations the Fisher information becomes $I(\theta; x_1, \ldots . x_n) = n \cdot I(\theta; x)$.

- the "score" random variable is defined as $\frac{\partial l(\theta;x)}{\partial \theta}$. Its expected value is

$$E\left[\frac{\partial l(\theta; x)}{\partial \theta}\right] = 0.$$

  Its variance is, consequently, the Fisher information:

$$V\left[\frac{\partial l(\theta; x)}{\partial \theta}\right] = E\left\{\left[\frac{\partial l(\theta; x)}{\partial \theta}\right]^2\right\} = I(\theta; x).$$

  So, being the Fisher Information the expectation of the square of the first derivative, it is just the expectation of the square of the score.

The term information comes from the ML process where it is used to measure the peak in the likelihood function. We measure the peadedness in the likelihood by calcuylating the magnitude of its second derivative at the maximum likelihood estimates. This because the first derivative represent the gradiend, whereas the second the rate of change of the gradient, a measure of curvature. The more curved the likelihood, the more confident we are in the estimates and any conclusion drawn from them.
Note that frequentist inference is not based on probability distributions, since it's based on likelihoods. This contrasts with bayesian inference that relies on probability distributions.

1.4.3 χ^2 as Gamma distribution

Remark 14. χ^2 distribution is a special case of Gamma; in this Section we see how one can pass from a $X \sim \chi^2_\nu$ to a $Gamma(\alpha, \beta)$.

Important remark 3 (First parametrization). In case of positive support random variable, we say that X is distributed as χ^2 with ν degrees of freedom $X \sim \chi^2_\nu$ if:

$$p(X|\nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} x^{\nu/2-1} \exp\left(-\frac{1}{2}x\right)$$

where $E(X|\nu) = \nu$ and $V(X|\nu) = 2\nu$.
If we create a new variable by applying the transformation

$$Y = \frac{X}{S} = S^{-1}X$$

then $X = SY$ and $\frac{\partial X}{\partial Y} = S$ (Jacobian of the transformation). Thus

$$Y \sim S^{-1}\chi^2_\nu$$

with

$$E(Y|\nu) = S^{-1}\nu$$
$$V(Y|\nu) = S^{-2}2\nu$$

For the density of Y (we substitute X with SY and multiply by the Jacobian):

$$\begin{aligned} p(Y|\nu) &= \frac{1}{2^{\nu/2}\Gamma(\nu/2)} (Sy)^{\nu/2-1} \exp\left(-\frac{1}{2}Sy\right) S \\ &= \frac{S^{\nu/2}}{2^{\nu/2}\Gamma(\nu/2)} (y)^{\nu/2-1} \exp\left(-\frac{1}{2}Sy\right) \end{aligned}$$

In order to recognize it as a Gamma distribution with parameters α, λ, let's set

$$S = 2\lambda \implies \lambda = S/2$$
$$\alpha = \nu/2 \implies \nu = 2\alpha$$

obtaining

$$\begin{aligned} p(Y|\alpha,\lambda) &= \frac{(2\lambda)^\alpha}{2^\alpha\Gamma(\alpha)} (y)^{\alpha-1} \exp\left(-\frac{1}{2}2\lambda y\right) \\ &= \frac{\lambda^\alpha}{\Gamma(\alpha)} y^{\alpha-1} \exp\left(-\lambda y\right) \end{aligned}$$

which is the first parametrization with, thus

$$E(Y|\alpha,\lambda) = \alpha/\lambda$$
$$V(Y|\alpha,\lambda) = \alpha/\lambda^2$$

Important remark 4 (Second parametrization). If we set $\beta = 1/\lambda$ we find the well-known two-parameters Gamma distribution $Gamma(\alpha, \beta)$

$$p(Y|\alpha,\beta) = \frac{1}{\beta^\alpha\Gamma(\alpha)} y^{\alpha-1} \exp\left(-\frac{y}{\beta}\right)$$

with moments

$$E(Y|\alpha,\beta) = \alpha\beta$$
$$V(Y|\alpha,\beta) = \alpha\beta^2$$

Easily, via a simple substitution, one retrieves the moments of Y defined as a χ^2:

$$E\left(Y|\frac{\nu}{2},\frac{2}{S}\right) = \frac{\nu}{S}$$
$$V\left(Y|\frac{\nu}{2},\frac{2}{S}\right) = \frac{2\nu}{S^2}$$

The transformed variable Y coming from X is distributed as $Y \sim S^{-1}\chi^2_\nu$, in this case, since $S^{-1} = \beta/2$ and $\nu = 2\alpha$ we get

$$Gamma(\alpha,\beta) = \frac{\beta}{2}\chi^2_{2\alpha}$$

We can find the moments of the χ^2 with S^{-1} and ν and those of the Gamma with α and β, obtaining the same results.

Important remark 5 (One-parameter Gamma distribution). This special case $Y \sim Ga(\alpha)$ is derived from the second parametrization, having set $\beta = \lambda = 1$:

$$p(Y|\alpha) = \frac{1}{\Gamma(\alpha)}y^{\alpha-1}\exp(-y)$$

that is also $Gamma(\alpha) = \frac{1}{2}\chi^2_{2\alpha}$.

1.4.4 Inverse χ^2 distribution

If $X \sim \chi^2_\nu$ e $Y \sim S^{-1}\chi^2_\nu$ **TODO**: to check yet

$$p(X|\nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}x^{\nu/2-1}\exp\left(-\frac{1}{2}x\right), \qquad X > 0$$
$$p(Y|\nu) = \frac{S^{\nu/2}}{2^{\nu/2}\Gamma(\nu/2)}y^{\nu/2-1}\exp\left(-\frac{1}{2}Sy\right), \qquad Y > 0.$$

The inverse χ^2 is such that $\frac{1}{X} \sim \chi^2_\nu$ or in other words $X \sim \chi^{-2}_\nu$

$$p(X|\nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}x^{-\nu/2-1}\exp\left(-\frac{1}{2}x^{-1}\right), \qquad X > 0$$

but also such that $\frac{1}{Y} \sim S^{-1}\chi^2_\nu$ or in other words $Y \sim S\chi^{-2}_\nu$

$$p(Y|\nu) = \frac{S^{\nu/2}}{2^{\nu/2}\Gamma(\nu/2)}y^{-\nu/2-1}\exp\left(-\frac{1}{2}\frac{S}{y}\right), \qquad Y > 0.$$

Important remark 6. Remember that in the density of $\chi^{-2}_{\nu-1}$, Y has exponent $-\frac{\nu-1}{2} - 1 = -\frac{\nu+1}{2}$, so we can write

$$p(Y|\nu) = \frac{S^{\nu/2}}{2^{\nu/2}\Gamma(\nu/2)}y^{-(\nu+1)/2}\exp\left(-\frac{1}{2}\frac{S}{y}\right), \qquad Y > 0.$$

that is a $\chi^{-2}_{\nu-1}$.

Chapter 2

From prior to posterior distribution

Remark 15. The topics of this block are contained in Chapters 4, 5,6, 7 and 8 of Lambert. Other references in Hoff, chapters 3 and 5[1]

Remark 16. In this section we move from events (and their probability) to random variables (and their distribution function); the aim is to adapt all the machinery of bayesian stuff to do inference.
We need to adapt bayes theorem, the usage of prior for parameters of interest and likelihood to arrive at a posterior distribution of the parameter of interest.
I have an idea of the state of the world (prior). Classical statistics is based on the fact that something fixed is in the population and how to use data to estimate that; the parameter is unknown but fixed. In Bayesian stats there's uncertainty on the state of the world/parameter, so we can assume a probability of states of the world, the prior.
After fixing the prior something is done (experiment) to see if our idea changes.

TODO: da aggiungere Hoff: before vs after in exchangeability and introduction of choose in binary model (sufficient stat) pag 35, kernel density e posterior density pag 38

[1]Hoff, P. D. (2009). *A first course in Bayesian statistical methods.* Springer Science & Business Media. ISBN: 978-0-387-92299-7
The notes illustrate the passage from the prior distribution of the parameter to the posterior. This argument is developed in **Chapter 3** of the textbook, where several further topics are introduced, that will be deepened later in these notes, after the present block.
In this block of notes, we focus on the cases of **a)** discrete prior and likelihood **b)** continuous prior and discrete likelihood, where the discrete likelihood is the binomial, the continuous prior is the Beta. See how the textbook sketches the topic for the binomial likelihood in *Section 3.1 - The binomial model.*
The relevance of predictive distributions is illustrated in *Section 2.3* of the textbook.
Also the normal univariate model is illustrated in this block of notes. In the textbook this topic appears in **Chapter 5**, where the case of normal continuous prior and normal continuous likelihood is developed (continuous normal posterior): *Section 5.1 - The normal model* and *Section 5.2 - Inference for the mean, conditional on the variance.*
Hoff's **Chapter 3** deals with *conjugacy* in *Section 3.1* (page 38), the whole *Section 3.3 - Exponential families and conjugate priors.*
The link between the distribution χ^2 and the two parameters Gamma, together with the case of normal gamma prior and discrete Poisson likelihood (continuous gamma posterior) are developed in *Section 3.2 - The Poisson model.*
The topic of credibility intervals is developed in the whole *Section 3.1.2 - Confidence regions* and is not developed in this block.
Important statistical distributions. For remarks on important statistical distributions see pages 253-258.

However in passing from probability of events to statistical distributions something occurs/changes.

2.1 Bayes theorem for random variables

NB prof: qui cambio notazione uniformando le varie componenti

Important remark 7. We can express the passage from the prior to the posterior as:

$$p(\theta|x) = \frac{p(\theta) \cdot p(x|\theta)}{p(x)} \propto p(\theta) \cdot p(x|\theta)$$

where

- $p(\theta)$ is the prior probability distribution for a parameter of interest
- $p(x|\theta)$ is the likelihood of the current observation/sample given the value asumed by θ
- the denominator $p(x)$ is a constant that only depends on the data (it's averaged on all the possible value of the parameter θ, which are at least two[2])

Important remark 8. In the following we specify this expression for the discrete and continuous case.

2.1.1 Discrete parameter and discrete data

Here θ has k possible values, $\theta_1, \ldots, \theta_k$, each with a certain prior probability that sums to the unit:

$$\sum_{i=1}^{k} p(\theta_i) = 1$$

For a sample of size n, we have:

$$p(\theta_i|\mathbf{x}) = \frac{p(\theta_i) \cdot p(\mathbf{x}|\theta_i)}{\sum_{i=1}^{k} p(\theta_i) \cdot p(\mathbf{x}|\theta_i)} \propto p(\theta_i) \cdot p(\mathbf{x}|\theta_i) = p(\theta_i) \cdot p(x_1|\theta_i) \cdot \ldots \cdot p(x_n|\theta_i)$$

where in the last passage the likelihood has been rewritten as a product (of individual likelihood/densities), since we assume independence of the observation (actually we do not need observation to be independent, they could be correlated, but we do not deal with this case).
The posterior distribution is a probability distribution:

$$\sum_{i}^{k} p(\theta_i|\mathbf{x}) = 1.$$

Important remark 9. We do not develop the case of discrete parameter and continuous data. This can be used when prior are expressed by experts' considerations, or to choosing/compare among a discrete set of models; each model may receive an a priori probability and can be supported by an experiment.

[2] If the prior was Dirac there would not be the needs to perform an experiment

2.1.2 Continuous parameter and discrete data

Parameter θ follows a (prior) probability distribution $p(\theta)$ which integrates to 1. For samples of size n, we can write

$$\begin{aligned} p(\theta|\mathbf{x}) &= \frac{p(\theta)\cdot p(\mathbf{x}|\theta)}{\int p(\tilde{\theta})\cdot p(\mathbf{x}|\tilde{\theta})\,\mathrm{d}\tilde{\theta}} \propto p(\theta)p(\mathbf{x}|\theta) \\ &= p(\theta)p(x_1|\theta)\dots p(x_n|\theta) \end{aligned}$$

Also in this case the likelihood can be written in a simpler way under the independence condition.

2.2 Exchangeability

In Bayesian statistics the concept of exchangeability (of a set of random variables) become important

NB prof: sezione introdotta prendendo da Hoff 2.7 e 2.8

Definition 2.2.1 (Exchangeability)**.** Let $X_1,\dots,X_n$ be a sequence of random variablew and $p(x_1,\dots,p_n)$ its joint density.
We say $X_1,\dots,X_n$ are exchangeable if $p(x_1,\dots,x_n) = p(x_{\pi_1},\dots,x_{\pi_n})$ for all permutations π of $\{1,\dots,n\}$.

Remark 17. Roughly speaking, $X_1,\dots,X_n$ are exchangeable if the subscript label convey no information about the outcomes.

Remark 18. What is the relationship between exchangeability and iid? It can be proved that

Theorem 2.2.1 (DeFinetti)**.** *The random variable $X_1,\dots,X_n$ are exchangeable for all n if and only if they depend on a common unknown parameter having distribution $\theta\sim p(\theta)$, but conditionally of its assumed variable are iid, that is*

$$\begin{cases} \theta \sim p(\theta) \\ X_1,\dots,X_n|\theta \text{ are iid} \end{cases} \iff X_1,\dots,X_n \text{ are exchangeable for all } n$$

Proof. Omitted □

Important remark 10. If exchangeability holds we can assume, conditionally on the parameter defining the random variable distributions, that these random variable are iid.
In the application of bayes theorem this helps in writing likelihood which can be written as product of individual density (using the common parameter), that is assuming (conditional) independence.

Important remark 11. When is the condition $X_1,\dots,X_n$ are exchangeable for all n reasonable?
For this condition to hold we must have *exchangeability* and *repeatability*:

- exchangeability will hold if the labels convey no informations (eg not a time)
- situations in which repeatability is reasonable include the following:
  - $X_1,\dots,X_n$ are outcomes of a repeatable experiment

- $X_1, \ldots, X_n$ are sampled from a finite population with relacement
- $X_1, \ldots, X_n$ are sampled from an infinite population without relacement

Thus in the classical case, if $X_1, \ldots, X_n$ are exchangeable and sampled from a finite population of size $N >> n$ without replacement, then they can be modeled as approximately being iid

Remark 19. Starting from the next session we see some one-parameter models; these areclass of sampling distributions that is indexed by a single unknown parameter such as binomial, normal and poisson models

2.3 Inference for a proportion

Remark 20. Here we see the how to apply the bayes theorem with random variable: we have a prior distribution for the parameter of interest (proportion) and we conduct an experiment; finally we obtain a posterior distribution for the parameter.
We tackle the case using different priors for the parameter (discrete or continuous) and considering an experiment which produces dichotomic data

2.3.1 Discrete prior

Important remark 12. In both case we start from a prior of four possible proprotions, with uniform probability:

$$\boldsymbol{\theta} = (0.2; 0.4; 0.6; 0.8)$$
$$p(\theta_j) = 1/4 \quad \forall j = 1, \ldots, 4$$

2.3.1.1 Sample of $n = 1$

Example 2.3.1 (One replication of the experiment)**.** The probability distribution for this unique observation, that is also the likelihood for the parameter, is the Bernoulli

$$p(X = x|\theta) = \theta^x (1 - \theta)^{1-x}$$

If

- $X = 1$ is observed then $p(X = 1|\theta_j) = \theta_j$, for each possible value of θ.
  The passages we do to modify our prior opinion about any possible value of the parameter after observing $X = 1$ is resumed in table 2.1.
  So it turns out that a single positive answer makes 4 times more plausible the that the population proportion is 0.8 rather than 0.2.
- $X = 0$ is observed then $p(X = 0|\theta_j) = 1 - \theta_j$.
  The passages we do to modify our prior opinion about any possible value of the parameter after observing $X = 0$ is resumed in table 2.2.
  Thus a single negative answer makes 4 times less plausible the statement that the population proportion is 0.8 rather than 0.2.

θ_j	$p(\theta_j)$	$p(X=1\|\theta_j)$	$p(\theta_j)p(X=1\|\theta_j)$	$p(\theta\|X=1)$
0.2	0.25	0.2	0.05	0.10
0.4	0.25	0.4	0.1	0.20
0.6	0.25	0.6	0.15	0.30
0.8	0.25	0.8	0.20	0.40
Sum	1		$p(x)=0.5$	1

Table 2.1: Experiment with single unit extraction with $X=1$.

θ_j	$p(\theta_j)$	$p(X=0\|\theta_j)$	$p(\theta_j)p(X=0\|\theta_j)$	$p(\theta\|X=0)$
0.2	0.25	0.8	0.2	0.40
0.4	0.25	0.6	0.15	0.30
0.6	0.25	0.4	0.1	0.20
0.8	0.25	0.2	0.05	0.10
Sum	1		$p(x)=0.5$	1

Table 2.2: Experiment with single unit extraction with $X=0$.

2.3.1.2 Sample of $n = 20$

Example 2.3.2 (n replications of the experiment)**.** Instead of performing only one trial, we have $n = 20$. The probability distribution of the number x of successes among n observations is the binomial:

$$p(x|n,\theta) = \binom{n}{x}\theta^x(1-\theta)^{n-x}$$

In case we observe $x = 15$ successes out of $n = 20$ trials, the likelihood of θ_j is:

$$p(X=15, n=20|\theta_j) = \binom{20}{15}\theta_j^{15}(1-\theta_j)^5, \quad j = 1\ldots,4$$

NB prof: cambiata notazione della likelihood a $p(X = 15, n = 20|\theta_j)$ per stare consistent p(data|param)

The passages we do to modify our prior opinion about any possible value of the parameter after observing $x = 15$ successes out of $n = 20$ trials is resumed in table 2.3 (where the two central columns were multiplied by 10^{-7} in order to be readable, all are very low probability values).

θ_j	$p(\theta_j)$	$p(15,20\|\theta_j)\times 10^{-7}$	$p(\theta_j)p(15,20\|\theta_j)\times 10^{-7}$	$p(\theta_j\|15,20)$
0.2	0.25	0.00	0.000	0.000
0.4	0.25	0.83	0.201	0.005
0.6	0.25	48.10	12.025	0.298
0.8	0.25	112.60	28.150	0.697
Sum	1		40.376	1

Table 2.3: Experiment with $n = 20$

```r
# riproduzione della tabella, ignorando il 10^{-7}
theta <- seq(0.2, 0.8, 0.2)
prior <- rep(0.25, 4)
lik   <- sapply(theta, function(t) dbinom(x = 15, size = 20, prob = t))
num   <- prior * lik
den   <- sum(num)
posterior <- num/den ## sum(posterior) ==1

round(cbind(theta, prior, lik, num, posterior), digits = 3)

##      theta prior   lik   num posterior
## [1,]   0.2  0.25 0.000 0.000     0.000
## [2,]   0.4  0.25 0.001 0.000     0.005
## [3,]   0.6  0.25 0.075 0.019     0.298
## [4,]   0.8  0.25 0.175 0.044     0.697

## probabilmente non torna nelle colonne centrali perché io ho ignorato
## esponente, prof ha ignorato coefficiente binomiale

## check with expected value of posterior distribution
sum(theta * posterior) ## not exactly the same as ML estimate, btw

## [1] 0.7383344
```

Comparison with pure ML estimation For comparison's sake, let's look at the ML estimator of the proportion (using only the sample, not the prior). Here we could ignore the binomial coefficient (as done by prof) in the optimization (since it's just a constant which does not depend on θ). The derivation goes like:

NB prof: nelle note originali vi è un typo algebrico all'ultimo passaggio

$$
\begin{aligned}
L(\theta|x,n) &= \binom{n}{x}\theta^x(1-\theta)^{n-x} \\
\log L(\theta;x,n) &= \log\binom{n}{x} + x\log\theta + (n-x)\log(1-\theta) \\
\frac{\partial \log L(\theta;x,n)}{\partial\theta} &= x\frac{1}{\theta} + (n-x)\frac{1}{1-\theta}(-1) = \frac{x}{\theta} - \frac{n-x}{1-\theta} = \frac{x(1-\theta)-(n-x)\theta}{\theta} \\
&= \frac{x - x\theta - n\theta + x\theta}{\theta(1-\theta)} = \frac{x-n\theta}{\theta(1-\theta)}
\end{aligned}
$$

Thus by putting $\frac{\partial \log L}{\partial\theta} = 0$ we derive the ML estimator, which as we know is:

$$\hat{\theta} = \frac{x}{n}$$

Applied to our sample is $\hat{\theta} = \frac{15}{20} = 0.75$ (the result is somewhat between the parameter having the posterior higher probability).
This value, a point estimate, differs from what will be derived when passing for priors to posteriors.

2.3.2 Continuous prior

Remark 21. The prior knowledge for the proportion θ of the population can be expressed as a continuous distribution using the Beta.

2.3.2.1 Beta distribution reminder

TODO: Da portare nel probability brush up?

Definition 2.3.1. Characterized by two parameters, $a > 0$ and $b > 0$, defined by:

$$p(X = x|a, b) = \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 \tilde{x}^{a-1}(1-\tilde{x})^{b-1}\,\mathrm{d}\tilde{x}} I_{(0,1)}(x)$$

and contains the Beta function at the denominator

$$\int_0^1 x^{a-1}(1-x)^{b-1}\,\mathrm{d}x = B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

which contains the Gamma function that, for a positive integers n, is defined as

$$\Gamma(n) = (n-1)!$$

Important remark 13. The first two moments of the Beta distribution are

$$E(X|a,b) = \frac{a}{(a+b)}$$
$$V(X|a,b) = \frac{ab}{(a+b+1)(a+b)^2}$$

Remark 22. The variance can be written differently, by noticing that:

NB prof: qui nell'originale due volte a al numeratore

$$\begin{aligned} V(X|a,b) &= \frac{ab}{(a+b+1)(a+b)^2} = \frac{a}{a+b}\frac{b}{a+b}\frac{1}{a+b+1} \\ &= \underbrace{\frac{a}{a+b}}_{E(X|a,b)} \underbrace{\left(1 - \frac{a}{a+b}\right)}_{1-E(X|a,b)} \frac{1}{a+b+1} \\ &= \frac{E(X|a,b)\,(1-E(X|a,b))}{(a+b+1)}. \end{aligned} \tag{2.1}$$

Example 2.3.3. When $a = b = 1$, the $Beta(1,1)$ coincides with a $U(0,1)$

2.3.2.2 Uniform prior

Remark 23. We start with a $U(0,1)$ prior, that is also $Beta(1,1)$.[3]

Let us assume a uniform prior in the interval (0,1), i.e. $U(0,1)$:

$$p(\theta|a=1,b=1) = \begin{cases} 1 & 0 < \theta < 1 \\ 0 & \text{otherwise} \end{cases}$$

[3]The example that is developed below is analogous to the *happiness data* example of Hoff, *Chapter 3 - One-parameter models, Section 3.1 - The binomial model.*

	Prior	Experiment	Posterior
Successes	1	15	16
Failures	1	5	6
Total	2	20	22

Table 2.4: Virtual and actual sample for this case

for which the moments are (by adapting the Beta formulas with $a = b = 1$):

$$E(\theta|a = 1, b = 1) = 1/2 = 0.5 \qquad V(\theta|a = 1, b = 1) = 1/12 = 0.083$$

As before in the experiment we observe $x = 15$ successes out of $n = 20$ trials. The expression of the likelihood is also the same (below the binomial coefficient remains), with continuos domain from 0 to 1:

$$p(X = 15, n = 20|\theta) = \binom{20}{15}\theta^{15}(1-\theta)^5$$

The posterior distribution of θ is defined in this case by having an integral over all its possible values at the denominator

$$p(\theta|a, b, X, n) = \frac{p(\theta|a, b)p(x, n|\theta)}{\int_0^1 p(\tilde{\theta}|a, b)p(x, n|\tilde{\theta})\ \mathrm{d}\tilde{\theta}} = \frac{p(\theta|1, 1)p(15, 20|\theta)}{\int_0^1 p(\tilde{\theta}|1, 1)p(15, 20|\tilde{\theta})\ \mathrm{d}\tilde{\theta}}$$

Now considering the case of $U(0, 1) = Beta(1, 1)$ prior (for which the density is $p(\theta) = 1, \forall\theta \in [0, 1]$, experiment with $X = 15$ and $n = 20$, the posterior becomes:

$$\begin{aligned} p(\theta|1, 1, 15, 20) &= \frac{1 \cdot \binom{20}{15}\theta^{15}(1-\theta)^5}{\int_0^1 1 \cdot \binom{20}{15}\tilde{\theta}^{15}(1-\tilde{\theta})^5\ \mathrm{d}\tilde{\theta}} = \frac{\theta^{15}(1-\theta)^5}{\int_0^1 \tilde{\theta}^{15}(1-\tilde{\theta})^5\ \mathrm{d}\tilde{\theta}} \\ &= \frac{\theta^{16-1}(1-\theta)^{6-1}}{\int_0^1 \tilde{\theta}^{16-1}(1-\tilde{\theta})^{6-1}\ \mathrm{d}\tilde{\theta}} \\ &= Beta(16, 6) \end{aligned}$$

Virtual and actual sample So, it turns out that if the prior is a $Beta(1, 1)$ ($U(0, 1)$) and in the experiment we have 15 successes and 5 failures, the posterior becomes a $Beta(16, 6)$. This gives an interpretation to the parameter a, b of the distribution which can be seen as sample size (before is virtual, after experiment is virtual + actual) of the units having failures and successes (tab 2.4).
To express a prior $U(0, 1)$ information, that is a $Beta(1, 1)$, 2 virtual cases are enough (1 success and 1 failure); enough to model a distribution with that expected value and variance. (The concept of virtual sample will be summarized also below.)

Prior vs Posterior Now let's compare moments and shapes of the prior and posterior distributions to appreciate the knowledge benefit of the experiment: before it we knew nothing (we knew θ can be in 0,1 but no more than that). In:

- table 2.5 the moments of the two distribution are compared and an improvement can be observed: the posterior variance is approximately $1/10$

	Prior $Beta(1,1)$ $(U(0,1))$	Posterior $Beta(16,6)$
$E(\theta)$	$E(\theta\|1,1) = 1/2 = 0.5$	$E(\theta\|16,6) = \frac{a}{a+b} = \frac{16}{22} = 0.7272$
$V(\theta)$	$V(\theta\|1,1) = 1/12 = 0.083$	$V(\theta\|16,6) = \frac{ab}{(a+b+1)(a+b)^2} = \frac{16 \cdot 6}{(16+6+1)(16+6)^2} = 0.008624$

Table 2.5: Prior vs posterior moments in uniform(0,1)-binomial case

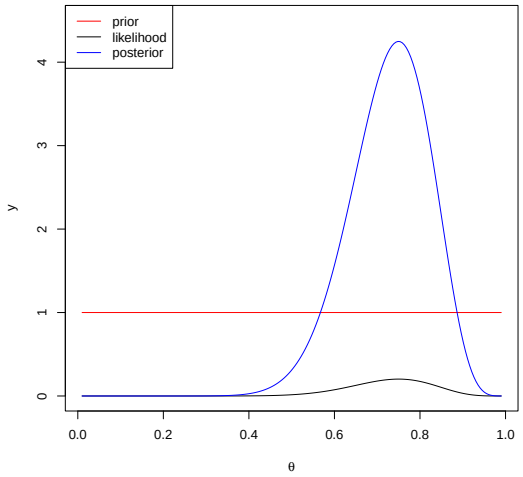


Figure 2.1: Uniform prior and binomial likelihood case

of the prior variance (this is due to the consideration of both the prior and the experiment information)

- fig 2.1 we can plot the distribution to shows the passage from the prior to the posterior through the experiment

Conjugacy introduction

Remark 24. [From BDA3] The property that the posterior distribution follows the same parametric form as the prior distribution (in this case the Beta) is called *conjugacy*.
Here we can say that the **beta prior** distribution is a **conjugate family** for the *binomial likelihood*.
The conjugate family is mathematically convenient in that the posterior distribution follows a known parametric form

2.3.2.3 Generic beta prior

A more generic prior $Beta(a,b)$ (with $a,b > 0$ to be choosen to have a certain expected value and variance of θ distribution) is expressed, as usual as

$$p(\theta|a,b) = \frac{\theta^{a-1}(1-\theta)^{b-1}}{\int_0^1 \tilde{\theta}^{a-1}(1-\tilde{\theta})^{b-1}\,\mathrm{d}\tilde{\theta}} I_{(0,1)}(\theta) \propto \theta^{a-1}(1-\theta)^{b-1}$$

The binomial likelihood is:

$$p(x, n|\theta) = \binom{n}{x}\theta^x(1-\theta)^{n-x} \propto \theta^x(1-\theta)^{n-x}$$

The posterior distribution applying Bayes theorem is

$$\begin{aligned} p(\theta|a, b, x, n) &= \frac{p(\theta|a, b) \cdot p(x, n|\theta)}{\int_0^1 p(\tilde{\theta}|a, b) \cdot p(x, n|\tilde{\theta})\ \mathrm{d}\tilde{\theta}} \propto p(\theta|a, b)p(x, n|\theta) \\ &\propto \theta^{a-1}(1-\theta)^{b-1} \cdot \theta^x(1-\theta)^{n-x} \\ &= \theta^{a+x-1}(1-\theta)^{b+(n-x)-1} \end{aligned}$$

Remark 25. The last term is the *kernel* of a $Beta(a + x, b + (n - x))$ density (that is its numerator, ignoring the constant denominator $B(a+r, b+(n-x))$).

Remark 26. [Miei ragionamenti] Abbiamo trovato che il kernel della posteriori è quello di una beta con parametri $a + r$, $b + n - r$;

- sappiamo che differisce dalla distribuzione della posteriori per una costante,
- sappiamo però che la posteriori è una distribuzione con integrale a 1

quindi la costante deve essere la costante di una beta con tali parametri e dunque la posteriori è effettivamente una beta con tali parametri

Remark 27. [From wikipedia] In statistics, especially in Bayesian statistics, the kernel of a probability density function (pdf) or probability mass function (pmf) is the form of the pdf or pmf in which any factors that are not functions of any of the variables in the domain are omitted.[1] Note that such factors may well be functions of the parameters of the pdf or pmf. These factors form part of the normalization factor of the probability distribution, and are unnecessary in many situations. For example, in pseudo-random number sampling, most sampling algorithms ignore the normalization factor. In addition, in Bayesian analysis of conjugate prior distributions, the normalization factors are generally ignored during the calculations, and only the kernel considered. At the end, the form of the kernel is examined, and if it matches a known distribution, the normalization factor can be reinstated. Otherwise, it may be unnecessary (for example, if the distribution only needs to be sampled from).

Important remark 14. So we have seen the generalized rules for cooking up a beta prior based on a and b, and a binomial likelihood based on r, n (where the special case of uniform was tackled before), ending up in a beta posterior with parameters depeding on prior and likelihood.
The next step is just changing the prior to accomodate different hypotheses on previous knowledge.

2.3.2.4 Beta with prespecified expected value/variance

Consider another prior, based on conjectures on the expected value and the variance

$$\begin{cases} E(\theta|a, b) = 0.4 \\ V(\theta|a, b) = 0.01 \end{cases}$$

	Prior	Experiment	Posterior
Successes	9.2	15	24.2
Failures	13.8	5	18.8
Total	23	20	43

Table 2.6: Sample sizes of prior (virtual) and experiment (actual)

To obtain a, b we set a system of equation[4] and exploit the alternative definition of variance of the beta distribution (eq 2.1)

$$\begin{cases} E(\theta|a,b) = \frac{a}{a+b} = 0.4 \\ V(\theta|a,b) = \frac{E(\theta|a,b)(1-E(\theta|a,b))}{a+b+1} = 0.01 \end{cases} \quad \begin{cases} \frac{a}{a+b} = 0.4 \\ \frac{0.4 \cdot 0.6}{a+b+1} = 0.01 \end{cases} \quad \dots \begin{cases} a = 9.2 \\ b = 13.8 \end{cases}$$

So in order to express those prior information (in terms mean and variance) we need $a = 9.2$ successes and $b = 13.8$ failures in the prior "virtual sample" and thus the prior information corresponds to 23 virtual cases.
The experiment is kept the same as before: 20 trials and 15 successes, as well as the binomial likelihood.
Thus:

- the complete sample has 43 cases (tab 2.6)
- the posterior distribution is a Beta, with parameters

$$\begin{aligned} a' &= a + x = 9.2 + 15 = 24.2 \\ b' &= b + n - x = 13.8 + 20 - 15 = 18.8 \end{aligned}$$

  and moments

$$\begin{aligned} E(\theta|a',b') &= \frac{a'}{a'+b'} = 0.562 \\ V(\theta|a',b') &= \frac{(a' \cdot b')}{(a'+b'+1) \cdot (a'+b')^2} = 0.0056 \end{aligned}$$

- the distributions plot (fig 2.2) shows the passage from the prior to the posterior through the experiment and that the variability of the posterior is smaller that in the first case.

2.3.2.5 Precise/informative, indifference, beta prior

Considering another prior, based on the following conjectures on the expected value and the (small) variance

$$\begin{cases} E(\theta|a,b) = 0.5 \\ V(\theta|a,b) = 0.001 \end{cases}$$

[4] Full development:

$$\begin{cases} \frac{a}{a+b} = 0.4 \\ \frac{0.4 \cdot 0.6}{a+b+1} = 0.01 \end{cases} \quad \begin{cases} a = 0.4a + 0.4b \\ a + b + 1 = 24 \end{cases} \quad \begin{cases} b = \frac{3}{2}a \\ a = 23 - b \end{cases} \quad \begin{cases} b = \frac{3}{2}(23 - b) \\ a = 23 - b \end{cases} \quad \begin{cases} b = \frac{69}{5} = 13.8 \\ a = 23 - 13.8 = 9.2 \end{cases}$$

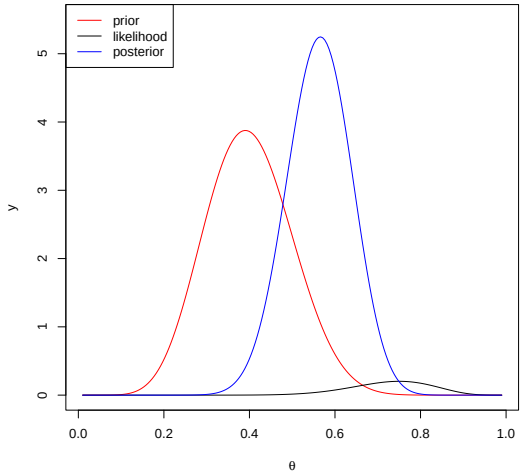


Figure 2.2: General beta-binomial example

These hypotheses yields to the following parameters for the prior

$$a = 124.5$$
$$b = 124.5$$

Thus the prior expected value 0.5 (and variance 0.001) is equivalent to 124.5 virtual successes out of 249 virtual cases. Adopting the same experiment/likelihood:

- the complete sample becomes composed of 269 cases (2.7)
- parameters of the posterior are

  $$a' = 124.5 + 15 = 139.5$$
  $$b' = 124.5 + 20 - 15 = 129.5$$

  so it is a $Beta(139.5; 129.5)$ with moments

  $$E(\theta|a', b') = 0.518$$
  $$V(\theta|a', b') = 0.000925$$

  To note that the ratio (posterior variance/prior variance) is near to 1, since the experiment adds very few cases to the virtual ones:

  $$\frac{V(\theta|a', b')}{V(\theta|a, b)} = \frac{0.000925}{0.001} = 0.99.$$

- finally, the plot (fig 2.3) shows the passage from the prior to the posterior through the experiment.

	Prior	Experiment	Posterior
Successes	124.5	15	139.5
Failures	124.5	5	129.5
Total	249	20	269

Table 2.7: Virtual sample with precise/indifferent prior

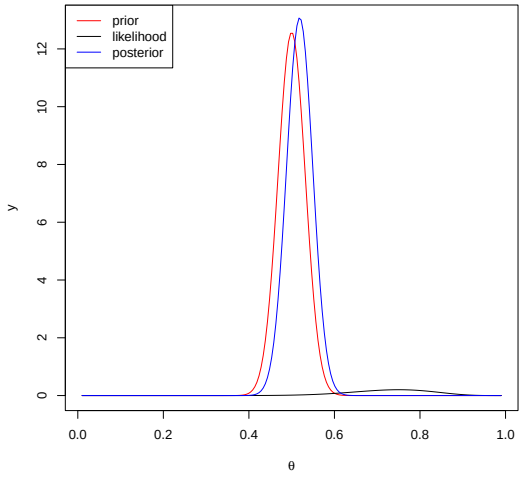


Figure 2.3: Experiment with precise uninformative prior

2.3.2.6 Virtual sample sum up

Important remark 15. In essence[5] , in the previous examples:

1. for a conjecture on the 0.5 expected value, without considering its variability, a prior of 1 case out of 2 is sufficient (tab 2.4)

2. for a conjecture on specific expected value and variances, a prior of 23 virtual cases is needed (tab 2.6)

3. a very precise conjecture on an indifference expected value needs a prior of many virtual cases (tab 2.7)

2.3.2.7 Expected values of posterior, prior and likelihood

Through the Bayes theorem, one can pass from a prior $Beta(a, b)$ to a posterior $Beta(a', b')$ with $a' = a + x$ and $b' = b + n - x$. Therefore some algebrical manipulations can highlight one fact

$$\begin{aligned} E(\theta|a', b') &= \frac{a'}{a' + b'} = \frac{a + x}{a + x + b + n - x} = \frac{a + x}{a + b + n} = \frac{a}{a + b + n} + \frac{x}{a + b + n} \\ &= \frac{a}{a + b + n}\frac{a + b}{a + b} + \frac{x}{a + b + n}\frac{n}{n} = \frac{a + b}{a + b + n}\frac{a}{a + b} + \frac{n}{a + b + n}\frac{x}{n} \\ &= \frac{a + b}{a + b + n}E(\theta|a, b) + \frac{n}{a + b + n}\bar{x} \end{aligned}$$

The last passage show how the posterior distribution expected value can be seen as a weighted mean of the prior $E(\theta|a, b)$ and the sample mean from the experiment $\bar{x}$, with weights proportional to the virtual sample and sample size respectively.

2.4 Inference for a mean

NB prof: cambio notazione

Here we consider the case of a continuous random variable for which our parameter of interest is the mean θ (assuming known/given variance of the variable), with the following assumptions. For:

- the **prior**, we assume that the population mean could be approximatively normally distributed/described, with mean θ_0 and variance τ_0^2

$$\begin{aligned} &\theta|\theta_0, \tau_0^2 \sim N(\theta_0, \tau_0^2) \\ &p(\theta|\theta_0, \tau_0^2) = \frac{1}{\sqrt{2\pi\tau_0^2}} \exp\left[-\frac{1}{2\tau_0^2}(\theta - \theta_0)^2\right] \propto \exp\left[-\frac{1}{2\tau_0^2}(\theta - \theta_0)^2\right] \end{aligned}$$

- the **likelihood** we have sampled $n > 1$ units coming from independent normals distribution with unknown mean θ and *known* variance σ^2,

[5] This topic is also dealt in Hoff, Section 3.3.1, pages 38-39 for the binomial model, and later, in Section 3.2. for the Poisson model.

$\{X_1, \ldots, X_n|\theta, \sigma^2\} \sim i.i.d.\ N(\theta, \sigma^2)$, thus the likelihood

$$p(\mathbf{x}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x_i-\theta)^2\right]$$
$$\propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\theta)^2\right]$$

- the **posterior** can be derived as follows:

$$p(\theta|\theta_0, \tau_0^2, \mathbf{x}) \propto p(\theta|\theta_0, \tau_0^2)\, p(\mathbf{x}|\theta)$$
$$\propto \exp\left[-\frac{1}{2\tau_0^2}(\theta-\theta_0)^2\right] \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\theta)^2\right]$$
$$= \exp\left\{-\frac{1}{2}\left[\frac{1}{\tau_0^2}(\theta^2+\theta_0^2-2\theta\theta_0) + \frac{1}{\sigma^2}\left(\sum_{i=1}^{n}x_i^2 + n\theta^2 - 2\theta\sum_{i=1}^{n}x_i\right)\right]\right\}$$
$$= \exp\left\{-\frac{1}{2}\left[\frac{\theta^2}{\tau_0^2} + \frac{\theta_0^2}{\tau_0^2} - \frac{2\theta\theta_0}{\tau_0^2} + \frac{\sum_{i=1}^{n}x_i^2}{\sigma^2} + \frac{n\theta^2}{\sigma^2} - \frac{2\theta\sum_{i=1}^{n}x_i}{\sigma^2}\right]\right\}$$

Now at this point, remembering it's a function of θ we gather terms with θ^2, θ and all the remaining stuff which will be avoided by proportionality

$$= \exp\left\{-\frac{1}{2}\left[\theta^2 \underbrace{\left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)}_{a} - 2\theta \underbrace{\left(\frac{\theta_0}{\tau_0^2} + \frac{n\bar{x}}{\sigma^2}\right)}_{b} + \underbrace{\frac{\theta_0^2}{\tau_0^2} + \frac{\sum_{i=1}^{n}x_i^2}{\sigma^2}}_{c}\right]\right\}$$
$$\propto \exp\left[-\frac{1}{2}\big(a\theta^2 - 2b\theta\big)\right]$$

Now, following Hoff, let's see if posterior takes the form/kernel of a normal:

$$p(\theta|\theta_0, \tau_0^2, \mathbf{x}) \propto \exp\left[-\frac{1}{2}\big(a\theta^2 - 2b\theta\big)\right] = \exp\left[-\frac{1}{2}a\left(\theta^2 - \frac{2b\theta}{a}\right)\right]$$
$$= \exp\left[-\frac{1}{2}a\left(\theta^2 - \frac{2b\theta}{a} + \frac{b^2}{a^2} - \frac{b^2}{a^2}\right)\right] = \exp\left[-\frac{1}{2}a\left(\theta^2 - \frac{2b\theta}{a} - \frac{b^2}{a^2}\right) + \frac{1}{2}\frac{b^2}{a}\right]$$
$$\propto \exp\left[-\frac{1}{2}a\left(\theta - \frac{b}{a}\right)^2\right] = \exp\left[-\frac{1}{2}\left(\frac{\theta - \frac{b}{a}}{1/\sqrt{a}}\right)^2\right]$$

This function is the kernel of a normal with mean b/a and $1/\sqrt{a}$ standard deviation, so posterior being a probability distribution will be normal

$$\theta|\theta_0, \tau_0^2, \mathbf{x} \sim N(\theta_1, \tau_1^2)$$

with θ_1 and τ_1^2 posterior mean and variance

$$\theta_1 = \frac{b}{a} = \frac{\frac{1}{\tau_0^2}\theta_0 + \frac{n}{\sigma^2}\bar{x}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}$$
$$\tau_1^2 = \frac{1}{a} = \frac{1}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}$$

Important remark 16. Some remarks:

1. the inverse of variance is often referred as **precision** of a random variable (that is how close we are to the centre). Let the following be the prior, sampling and posterior precisions:

$$\begin{aligned} &\text{prior precision:} && \frac{1}{\tau_0^2} \\ &\text{sampling precision:} && \frac{1}{\sigma^2} \\ &\text{posterior precision:} && \frac{1}{\tau_1^2} = \frac{1}{\tau_0^2} + \frac{n}{\sigma^2} = \underbrace{\frac{1}{\tau_0^2}}_{\text{prior}} + n \underbrace{\frac{1}{\sigma^2}}_{\text{sampling}} \end{aligned} \tag{2.2}$$

2. we can define as **weight** (thinkable as *importance of the prior*) the ratio between prior and posterior precision:

$$\omega = \frac{\frac{1}{\tau_0^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} = \frac{\frac{1}{\tau_0^2}}{\frac{\sigma^2 + n\tau_0^2}{\tau_0^2\sigma^2}} = \frac{1}{\tau_0^2} \cdot \frac{\tau_0^2\sigma^2}{\sigma^2 + n\tau_0^2} = \frac{\sigma^2}{\sigma^2 + n\tau_0^2}$$

   Complement weight to sum up to 1 (importance of data/likelihood)

$$1 - \omega = \frac{\frac{n}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} = \frac{n\tau_0^2}{\sigma^2 + n\tau_0^2}$$

3. we can **rewrite posterior expected value and variance** as

$$\theta_1 = \frac{\frac{1}{\tau_0^2}\theta_0 + \frac{n}{\sigma^2}\bar{x}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} = \theta_0 \underbrace{\left(\frac{\frac{1}{\tau_0^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} \right)}_{\omega} + \bar{x} \underbrace{\left(\frac{\frac{n}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} \right)}_{1-\omega}$$

$$\begin{aligned} \tau_1^2 &= \frac{1}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} = \frac{1}{\frac{\sigma^2 + n\tau_0^2}{\tau_0^2\sigma^2}} = \frac{\tau_0^2\sigma^2}{\sigma^2 + n\tau_0^2} = \sigma^2 \cdot \frac{\tau_0^2}{\sigma^2 + n\tau_0^2} \\ &= \frac{\sigma^2}{n} \cdot \frac{n\tau_0^2}{\sigma^2 + n\tau_0^2} = \frac{\sigma^2}{n}(1 - \omega) \end{aligned} \tag{2.3}$$

   and overall **posterior distribution** as

$$\theta | \theta_0, \tau_0^2, \mathbf{x} \sim \ N\left(\omega \cdot \theta_0 + (1 - \omega) \cdot \bar{x}, \frac{\sigma^2}{n}(1 - \omega) \right)$$

   We can note from 2.3 that posterior variance is smaller than the one of the ML estimator for the mean ($\frac{\sigma^2}{n}$).

4. **decreasing importance of prior** If $\tau_0^2 \to \infty$ or $n \to \infty$ then

$$\omega = \frac{\sigma^2}{\sigma^2 + n\tau_0^2} \to 0$$

   and the posterior will no longer depend on the prior (its parameters disappear) and moments of the posterior coincide with the ML estimates.

$$\theta | \theta_0, \tau_0^2, \mathbf{x} \sim N\left(\bar{x}, \frac{\sigma^2}{n} \right)$$

Example 2.4.1. In case $\tau_0^2 = \frac{\sigma^2}{m}$ with $m > 1$

- the prior distribution of the mean becomes

$$\theta \sim N\left(\theta_0, \frac{\sigma^2}{m}\right)$$

  where m can be seen as the number of observation which contributed to the prior distribution definition, the so-called *virtual sample precision*;

- weights for prior importance ω becomes

$$\omega = \frac{\frac{m}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{m}{\sigma^2}} = \frac{m}{m+n}$$
$$1 - \omega = \frac{n}{m+n}$$

- posterior distribution simplifies to

$$\theta_1 = \theta_0 \frac{m}{m+n} + \bar{x}\frac{n}{m+n}$$
$$\tau_1 = \frac{\sigma^2}{n}\frac{n}{m+n} = \frac{\sigma^2}{m+n}$$
$$\theta | x_1, \ldots, x_n \sim N\left(\theta_0 \frac{m}{m+n} + \bar{x}\frac{n}{m+n}, \frac{\sigma^2}{m+n}\right)$$

2.5 Inference for a count

Remark 28. Some measurements (eg number of friends) have values that are whole numbers. For these phenomenon the simplest probability model of the measurement is the Poisson model.
In a bayesian context thus, the likelihood depends only on one parameter of the Poisson distribution, the mean $\lambda > 0$. Now we switch to the common notation to θ to mean λ as parameter of interest.
A prior distribution for θ that is "natural" is the Gamma distribution.

Important remark 17 (Gamma-Poisson model). We assume:

- a Gamma **prior** distribution for mean count θ. Has positive only support (which is what we want) $\theta > 0$ and depends on two positive shape parameters $\alpha > 0, \lambda > 0$ (ie $G(\alpha, \lambda)$)

NB: Two parametrization are used: this is preferred

$$p(\theta|\alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \cdot \exp(-\lambda\theta) \cdot \theta^{\alpha-1}$$
$$\propto \exp(-\lambda\theta) \cdot \theta^{\alpha-1}$$

  In this parametrization:

$$E(\theta) = \frac{\alpha}{\lambda}$$
$$V(\theta) = \frac{\alpha}{\lambda^2}$$

- a Poisson data generating process. Conditionally on unknown mean θ each exchangeable/independent rv is distributed according to

$$p(X = x|\theta) = \frac{\exp(-\theta) \cdot \theta^x}{x!}$$

and the **likelihood** for a sample of size n:

$$\begin{aligned} p(\mathbf{x}|\theta) &= \prod_{i=1}^{n} \frac{\exp(-\theta) \cdot \theta^{x_i}}{x_i!} = \frac{\exp(-n\theta) \cdot \theta^{\sum_{i=1}^n x_i}}{\prod_{i=1}^n x_i!} \\ &\propto \exp(-n\theta) \cdot \theta^{\sum_{i=1}^n x_i} \end{aligned}$$

- the **posterior** is a Gamma as well, indeed

$$\begin{aligned} p(\theta|\alpha, \lambda, \mathbf{x}) &\propto p(\theta|\alpha, \lambda) \cdot p(\mathbf{x}|\theta) \\ &\propto \exp(-\lambda\theta)\theta^{\alpha-1} \cdot \exp(-n\theta)\theta^{\sum_{i=1}^n x_i} \\ &= \exp(-\theta(\lambda + n)) \cdot \theta^{\alpha+\sum_{i=1}^n x_i - 1} \end{aligned}$$

where at last we recognize the kernel of a gamma $G(\alpha', \lambda')$, where

$$\begin{aligned} \alpha' &= \alpha + \sum_{i=1}^{n} x_i \\ \lambda' &= \lambda + n \end{aligned}$$

for which

$$\begin{aligned} E(\theta|\alpha, \lambda, x) &= \frac{\alpha'}{\lambda'} = \frac{\alpha + \sum_{i=1}^n x_i}{\lambda + n} \\ V(\theta|\alpha, \lambda, x) &= \frac{\alpha'}{\lambda'^2} = \frac{\alpha + \sum_{i=1}^n x_i}{(\lambda + n)^2} \end{aligned}$$

Remark 29. Looking at the posterior expectation we can interpret the parameters of prior and likelihood:

- λ is interpreted as number of prior observations (while n as sample size of the experiment)
- α is interpreted as sum of counts from the λ prior observations (while $\sum_{i=1}^n x_i$ is for the experiment)

Thus the posterior expected value is a weighted mean between prior mean and experiment sample mean (with weights based on number of observations).

Example 2.5.1. [Birth rate] A Survey was conducted to estimate the mean number of children women without (θ_1) and with (θ_2) bachelor degree.
Let's assume that the prior for that mean are distributed both according

$$\theta_1, \theta_2 \sim Ga(\alpha = 2, \lambda = 1)$$

with a common expected value of 2 children per woman.
The survey gathered data on number of children in the two groups ($n_1 = 111$

women without degree, $n_2 = 44$ women with), and the mean of children per woman was higher in the without bachelor degree mothers:

$$n_1 = 111, \sum_{i=1}^{n_1} Y_{i,1} = 217 \implies \bar{Y}_1 = 1.95$$
$$n_2 = 44, \sum_{i=1}^{n_2} Y_{i,2} = 66 \implies \bar{Y}_1 = 1.50$$

Assuming a Poisson model is appropriate to describe/synthetize the empirical distribution, a posterior distribution for the two parameters are easily two gammas

$$\theta_1 | \left\{ n_1 = 111, \sum_{i=1}^{n_1} Y_{i,1} = 217 \right\} \sim Ga(2 + 217, 1 + 111) = Ga(219, 112)$$
$$\theta_2 | \left\{ n_2 = 44, \sum_{i=1}^{n_2} Y_{i,2} = 66 \right\} \sim Ga(2 + 66, 1 + 44) = Ga(68, 45)$$

The posterior means for θ_1, θ_2 becomes $219/112 = 1.95$ and $68/45 = 1.51$, so we moved way far from the initial 2 mean value for the women with degree.

Remark 30 (Alternative parametrization of the Gamma). Using the second parametrization:

- for the prior density we define $\beta = \frac{1}{\lambda}$ and thus the density becomes (eg $G(\alpha, \beta)$):

$$p(\theta|\alpha, \beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} \theta^{\alpha-1} \exp\left(-\frac{\theta}{\beta}\right)$$

  For this parametrization

$$E(\theta) = \alpha\beta$$
$$V(\theta) = \alpha\beta^2$$

- the posterior using the second parametrization

$$\begin{aligned} p(\theta|\alpha, \beta, x) &\propto e^{-n\theta} \theta^{\sum_{i=1}^n x_i} \cdot \theta^{\alpha-1} e^{-\frac{\theta}{\beta}} \\ &= e^{-\theta(n+\frac{1}{\beta})} \cdot \theta^{\alpha+\sum_{i=1}^n x_i - 1} \end{aligned}$$

  If we substitute

$$\beta' = \left(n + \frac{1}{\beta}\right)^{-1} = \left(\frac{\beta n + 1}{\beta}\right)^{-1} = \frac{\beta}{\beta n + 1}$$

  we recognize the kernel of a Gamma, $G(\alpha', \beta')$ where

$$\alpha' = \alpha + \sum_{i=1}^n x_i$$
$$\beta' = \frac{\beta}{\beta n + 1}$$

for which

$$E(\theta|\alpha,\beta,x) = \left(\alpha + \sum_{i=1}^{n} x_i\right)\left(\frac{\beta}{\beta n+1}\right)$$
$$V(\theta|\alpha,\beta,x) = \left(\alpha + \sum_{i=1}^{n} x_i\right)\left(\frac{\beta}{\beta n+1}\right)^2$$

Important remark 18. Looking, for instance, at the expectations, the two parametrizations are equivalent since

$$\frac{\beta}{\beta n+1} = \frac{1}{n+\lambda}.$$

2.6 Natural conjugate distributions

Remark 31. We have seen, among other, that beta prior distribution and binomial sampling model lead to a beta posterior distribution.
To reflect this we say that *the class of beta priors is conjugate for the binomial sampling model*.
It is desirable to have a prior such that the posterior has a tractable form and is algebraically convenient/known.

Definition 2.6.1 (Conjugacy)**.** A class $\mathcal{P}$ of prior distributions for θ is called conjugate for a sampling model $p(\mathbf{x}|\theta)$ if the posterior is in the same class of distributions

$$p(\theta) \in \mathcal{P} \implies p(\theta|\mathbf{x}) \in \mathcal{P}$$

Remark 32. In other words, *conjugacy* is the property that the posterior distribution follows the same parametric form as the prior distribution.

2.6.1 Sufficient statistics

Remark 33. Sufficiency is a property of a statistic/function T (e.g. the sum of cases) computed on a sample dataset $(x_1, \ldots, x_n)$, in relation to a parametric model.
Informally speaking, a sufficient statistic contains all of the information that the dataset provides about the model parameters.

Remark 34. The following theorem provides a characterization of a sufficient statistic

Theorem 2.6.1 (Fisher-Neyman factorization theorem)**.** *A statistic $t = T(\mathbf{x})$ is sufficient for θ given the sample $\mathbf{x}$ if and only if there are functions f and g such that the likelihood can be rewritten as*

$$p(\mathbf{x}|\theta) = g(\mathbf{x})f(t|\theta)$$

Remark 35. Often we have $g(\mathbf{x}) = 1$, so only $f(t|\theta)$ remainsand thus $p(\theta|\mathbf{x}) = p(\theta|t)$

Important remark 19 (Sufficient statistic). In the bayesian framework being sufficient implies that

$$p(\theta|\mathbf{x}) = p(\theta|t) \propto p(\theta)p(t|\theta)$$

Remark 36. Only likelihoods that admit sufficient statistics are considered.

2.6.2 One-parameter exponential family

Definition 2.6.2. A density belongs to the one-parameter exponential family if (equivalently):

- for one observation, if it can be expressed in the form:

$$p(x|\theta) = g(x)h(\theta)\exp\left[t(x)\Psi(\theta)\right]$$

- for n independent observations, if the likelihood of the sample can be expressed as:

$$p(\mathbf{x}|\theta) \propto h(\theta)^n \exp\left[\sum t(x_i)\Psi(\theta)\right]$$

  where:

  - $g(x)$ can be omitted since and keep proportionality, not being a function of the parameter
  - $\sum t(x_i)$ is a sufficient statistic for θ (should follow from Neyman's factorization theorem)

2.6.2.1 Examples

Remark 37. The following examples show how different distributions belong to the exponential family.

Example 2.6.1 (Normal with known variance σ^2)**.**

$$\begin{aligned} p(x|\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{1}{2\sigma^2}(x-\theta)^2\right] \\ &= \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)}_{g(x)}\underbrace{\exp\left(-\frac{\theta^2}{2\sigma^2}\right)}_{h(\theta)}\underbrace{\exp\left(\frac{x\theta}{\sigma^2}\right)}_{\exp[t(x)\Psi(\theta)]} \end{aligned}$$

Example 2.6.2 (Normal with known mean μ)**.**

$$p(x|\theta) = \underbrace{\frac{1}{\sqrt{2\pi}}}_{g(x)}\underbrace{\frac{1}{\sqrt{\theta}}}_{h(\theta)}\underbrace{\exp\left\{-\frac{1}{2\theta}(x-\mu)^2\right\}}_{\exp[\Psi(\theta)t(x)]}$$

Example 2.6.3 (Poisson)**.**

$$\begin{aligned} p(x|\theta) &= \frac{e^{-\theta}\theta^x}{x!} = \frac{e^{-\theta}\exp(x\log\theta)}{x!} \\ &= \underbrace{\frac{1}{x!}}_{g(x)}\underbrace{\exp(-\theta)}_{h(\theta)}\underbrace{\exp\left[x\log\theta\right]}_{\exp[t(x)\Psi(\theta)]} \end{aligned}$$

Example 2.6.4 (Binomial with known n)**.**

$$\begin{aligned} p(x|\theta) &= \binom{n}{x}\theta^x(1-\theta)^{n-x} \\ &= \binom{n}{x}(1-\theta)^n\theta^x(1-\theta)^{-x} \end{aligned}$$

considering that

$$\begin{aligned} \theta^x(1-\theta)^{-x} &= \exp\log\left[\theta^x(1-\theta)^{-x}\right] = \exp\log\left[\left(\frac{\theta}{1-\theta}\right)^x\right] \\ &= \exp\left[x\log\left(\frac{\theta}{1-\theta}\right)\right] \end{aligned}$$

then

$$p(x|\theta) = \underbrace{\binom{n}{x}}_{g(x)}\underbrace{(1-\theta)^n}_{h(\theta)}\underbrace{\exp\left[x\log\frac{\theta}{1-\theta}\right]}_{\exp[t(x)\Psi(\theta)]}$$

Example 2.6.5 (Exponential)**.**

$$p(x|\theta) = \theta\exp(-\theta x) = \underbrace{1}_{g(x)}\underbrace{\theta}_{h(\theta)}\underbrace{\exp(-\theta x)}_{\exp[\Psi(\theta)t(x)]}$$

2.6.2.2 Relationship between conjugacy and exponential family

Remark 38. There's a relation between exponential family and conjugacy

Important remark 20. If the experiment produces data belonging to a distribution of the exponential family, having thus likelihood

$$p(\mathbf{x}|\theta) \propto h(\theta)^n\exp\left[\sum t(x_i)\Psi(\theta)\right]$$

the conjugate prior $\mathcal{P}$ belongs to the family with density

$$p(\theta) \propto h(\theta)\exp\{\tau\Psi(\theta)\}$$

where τ stresses the fact that no observation related to the experiment can be considered.

Example 2.6.6. Some examples of experiments with likelihood belonging to the one parameter exponential family (and depend on the sufficient statistic for the parameter) are reported in table 2.8

2.6.3 Two-parameters exponential family

Definition 2.6.3. A probability density belongs to the two parameters exponential family:

- for one observation, if it can be expressed in the form:

$$p(x|\theta,\varphi) = L(\theta,\varphi;x) = g(x)h(\theta,\varphi)\exp\left[t(x)\Psi(\theta,\varphi) + u(x)\chi(\theta,\varphi)\right],$$

Likelihood	Conjugate Prior	Case/Naming
Binomial	Beta	Beta-Binomial
Normal (known variance)	Normal	Normal-Normal
Normal (known mean)	Inverse-gamma	Normal-Inverse-gamma
Poisson	Gamma	Gamma-Poisson

Table 2.8: Likelihood and conjugate priors: notable examples.

- for n independent observations if the likelihood of the sample $p(x|\theta,\varphi)$ can be expressed as

$$L(\theta,\varphi;x) \propto h(\theta,\varphi)^n \exp\left[\sum t(x_i)\Psi(\theta,\varphi) + \sum u(x_i)\chi(\theta,\varphi)\right],$$

  where $g(x)$ is not considered since it is not a function of the parameter.

Remark 39. The relation between exponential family and conjugacy becomes the following

Important remark 21. In this case the family of conjugate densities has the form:

$$p(\theta,\varphi) \propto h(\theta,\varphi)^n \exp\left[\tau\Psi(\theta,\varphi) + \nu\chi(\theta,\varphi)\right]$$

Important remark 22. Here $(\sum t(x_i), \sum u(x_i))$ is a sufficient statistic for the bidimensional vector (θ,φ) given x.

Chapter 3

Interval estimation, prediction, hypothesis testing

The topics of this block are contained in Chapters 7, 10 and 18 of Lambert B. (2018) A Student's Guide to Bayesian Statistics, Sage:[1]

3.1 Credibility intervals

Remark 40. It is often desirable to identify regions of the parameter space that are likely to contain the true value of the parameter.

Remark 41. In the Bayesian framework, the information to exploit for inference is contained in the posterior distribution, where the parameter of interest θ is a random variable.
Aside expected value of the posterior distribution (which can be thought as estimate of parameter of interest), the construction of *posterior intervals* allows to communicate uncertainty on the estimate.
Such intervals are usually called **Bayesian confidence intervals** or **credibility intervals**[2]

3.1.1 Credibility vs confidence (frequentist) intervals

Remark 42. In general to construct regions of the parameter space that are likely to contain the true value of the parameter frequentist and bayesian behaves differently:

[1] Credibility Intervals (Section 7.7 Intervals of uncertainty: (7.7.1 Failings of the Frequentist confidence interval; 7.7.2 Credible intervals; 7.7.3 Reconciling the difference between confidence and credible intervals) High Posterior Density Region: Section 7.7.2 (Treasure hunting: the central posterior and highest density intervals) Hypothesis Testing Chapter 10 - Evaluation of model fit and hypothesis testing (See, in particular, Section 10.6 Marginal likelohoods and Bayes Factor).

In Hoff, Credibility intervals (often named posterior confidence intervals) are discussed throughout the textbook, starting from "Comparison to non-Bayesian methods" in Chapter 1. Special emphasis to HDPR is given in Chapter 3 (Section 3.1.2: Confidence regions). Hypothesis testing is a topic that is not developed in the textbook.

[2] The term "credibility" was introduced in Edwards, Lindman *et al.* (1963).

- bayesian after observing the sample $X = x$ (or say $x_1, \dots, x_n$) can construct an interval $[l(x), u(x)]$ such that the probability that $l(x) < \theta < u(x)$ is large
- frequentist costruct a rule that will provide a random interval which will contains the true parameter with given probability. Once constructed the parameter will belong to the interval or not

Definition 3.1.1 (Bayesian coverage)**.** An interval $[l(x), u(x)]$ based on the observed data $X = x$ (or say $x_1, \dots, x_n$) has 95% Bayesian coverage for θ if

$$\mathbb{P}\left(l(x) < \theta < u(x) | X = x\right) = 0.95$$

Remark 43. The interpretation of this interval is that it describes your information about the location of the true value of θ after we've observed $X = x$ (or say $x_1, \dots, x_n$).

Remark 44. Classical statistics arrives at similar conclusions, but builds random intervals with probability $1 - \alpha$ of containing the fixed but unknown value of the parameter θ.

Definition 3.1.2 (Frequentist coverage)**.** A random interval $[l(X), u(X)]$ has 95% frequentist coverage for θ if, before data are gathered

$$\mathbb{P}\left(L(x) < \theta < U(x) | \theta\right) = 0.95$$

Remark 45. In a sense, the frequentist and bayesian notions of coverage describe pre and post-experimental coverage respectively

Remark 46. In the frequentist approach, once observed the sample and plug the data in the confidence interval formula $[l(x), u(x)]$ is obtained and then

$$\mathbb{P}\left(l(x) < \theta < u(x) | \theta\right) = \begin{cases} 0 & \theta \notin [l(x), u(x)] \\ 1 & \theta \in [l(x), u(x)] \end{cases}$$

This hightlights the lack of post-experimental interpretation of frequentist coverage.
Although this may make the frequentist interpretation seem lacking, it's still useful in many situations. Suppose we are running a large number of unrelated experiments and creating a confidence interval for each one of them: if our intervals each have 95% frequentist coversage probability, we can expect that 95% of our intervals will contain the correct parameter value.

Remark 47. The following example shows i guess how in special cases where the computation is actually identical the interpretation differs.

Example 3.1.1. Let $(x_1, \dots, x_n | \theta)$ be a sample from $N(\theta, 1)$, if the prior for θ is *uninformative* (later in these notes), the posterior is

$$\theta | \overline{x} \sim N\left(\overline{x}, \frac{\sigma_0^2}{n}\right) = N\left(\overline{x}, \frac{1}{n}\right)$$

where $\sigma_0^2 = 1$ and $\overline{x}$ is a sufficient statistic for θ. In this situation:

- the symmetric 95% credibility interval for θ is

$$0.95 = P\left\{\overline{x} - \frac{2}{\sqrt{n}} \leq \theta \leq \overline{x} + \frac{2}{\sqrt{n}} | \overline{x}\right\}.$$

  the interpretation of this interval is *direct* since θ has a probability distribution and it is conditional to the data through $\overline{x}$; $1-\alpha$ is a probability statement made directly on the quantity of interest.

- the frequentist confidence interval would be

$$0.95 = P\left\{\overline{x} - \frac{2}{\sqrt{n}} \leq \theta \leq \overline{x} + \frac{2}{\sqrt{n}} | \theta\right\}.$$

  that is conditioned on θ. Probability statements can be done only on $\overline{x}$, that is the only observed random variable.
  The interpretation of a (frequentist) confidence interval refers to *indirect* considerations on the true value of the parameter, it is based on data that are not observed and produced by a large number of repetitions.

Remark 48. Can a confidence interval have the same Bayesian and frequentist coverage probability? Hartigan showed that (for types of intervals shown in Hoff), confidence interval procedure that fives 95% bayesian coverage will have approximately 95% frequentist coverage as well, at least asymptotically.
Particularly an interval that has 95% Bayesian coverage has frequentist coverage that is

$$\mathbb{P}\left(L(x) < \theta < U(x) | \theta\right) = 0.95 + \epsilon_n$$

where $|\epsilon_n| < \frac{a}{n}$ for some constant a, so as $n \to infty$ the term become negligible.

3.1.2 Bayesian methods for credibility intervals

Important remark 23. Two different approaches are used:

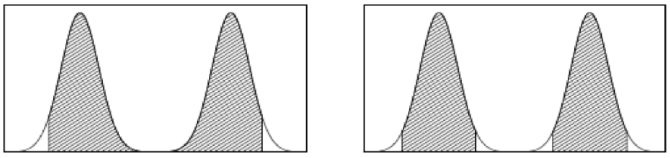
- **posterior quantiles**: the chosen extremes of the interval are the quantiles of the posterior distribution such that the posterior probabilities in both tails are $\alpha/2$ (thus way obtaining a $100(1-\alpha)\%$ level interval);
- **highest posterior density region (HPDR)**: in this case, besides containing the $100(1-\alpha)\%$ posterior probability, the interval has also to satisfy the request of containing most of the distribution and the constraint that the density within the range is never less than in the external areas. The idea is useful in the case of distributions that are strongly asymmetric and/or multimodal, as shown in figure 3.1

3.1.2.1 Quantiles interval estimation

Considering the cumulative (posterior) distribution function $F(\theta | x_1, \ldots, x_n)$, for a fixed value α, we can find an interval (a, b) such that:

$$F(b) - F(a) = P\left\{a < \theta < b | x_1, \ldots, x_n\right\} = 1 - \alpha$$

The (a, b) interval is called credibility interval for θ at credibility level α.



Hypothetical density for which the 95% central interval and 95% highest posterior density region dramatically differ: (a) central posterior interval, (b) highest posterior density region.

Figure 3.1: A mixture situation

Example 3.1.2. Suppose that in the case of example 2.5.1 we want to find the 95% credibility interval for the two parameters (θ_1 and θ_2) which have a posterior distributions $\theta_1 \sim Ga(219, 112)$, $\theta_2 \sim Ga(68, 45)$, and expected value 1.95 and 1.51 respectively. With R

```
quantiles <- c(0.025, 0.975)
qgamma(p = quantiles, 219, 112)    # without bachelor degree

## [1] 1.704943 2.222679

qgamma(p = quantiles, 68, 45)      # with bachelor degree

## [1] 1.173437 1.890836
```

3.1.2.2 Highest Posterior Density Region (HPDR)

The HPDR is a specific credibility interval (the smallest) for which the following properties hold:

1. $F(b) - F(a) = 1 - \alpha(0.95)$
2. if $h(\theta|x_1 \dots x_n)$ is the posterior density, for $a \leq \theta \leq b$, $h(\theta|x_1 \dots x_n)$ has the highest value with respect to any other interval for which property 1 holds.

Extending to more than one dimension, instead of a credibility interval we refer to a multidimensional credibility region R (region with minimum volume). The above properties become:

1. $P(\theta \in R|\mathbf{x}_1 \dots \mathbf{x}_n) = 1 - \alpha$
2. $\forall \theta_1 \in R$ and $\theta_2 \in R \;\; h(\theta_1|\mathbf{x}_1 \dots \mathbf{x}_n) \geq h(\theta_2|\mathbf{x}_1 \dots \mathbf{x}_n)$.

Theorem 3.1.1 (Box and Tiao,1973)**.** *The Highest Posterior Density Interval/Region always exists and it is unique for all the intervals/regions of level* $(1 - \alpha)$ *in which the posterior density is not uniform in any interval/region of the space of* θ *.*

Proof. omitted □

prof: Sezione inotta da Hoff, pag 234, prove da verificare

HDPR identification in general/practice In general a close formula for HDPR can be tricky to find (unless we're in a special case as unimodality below), but if we can generate random data from the posterior distribution (Hoff example page 234), the following step (using R) allows to determine which values of θ are contained in the HDPR:

1. simulate a sample from the posterior density (eg: `r*` function)
2. compute estimates of the posterior density: if posterior is known use otherwise use `density` command
3. then normalize the density values so they sum to 1 (obtaining thus a proper distribution)
4. sort these discrete probabilities in decreasing order
5. find the first value such that the cumulative sum of the sorted values exceeds $(1-\alpha)$; the HPD region includes all values of θ which have a discretized probability greater than this cutoff

Example 3.1.3. We calculate the HPDR for the posterior distribution of the birth rate of women without bachelor degree

```
set.seed(1)

## data
alpha <- 219
lambda <- 112

## quantile
quantiles <- c(0.025, 0.975)
qgamma(quantiles, alpha, lambda)
```

```
## [1] 1.704943 2.222679
```

```
## HDPR using known density function ("dgamma")
thetas <- rgamma(n = 1000, alpha, lambda)          # theta/posterior extraction
post_density <- dgamma(x = thetas, alpha, lambda) # density of theta
post_prob <- post_density/sum(post_density)
db <- data.frame(thetas, post_prob)
db <- db[order(db$post_prob, decreasing = TRUE), ]
db$cumprob <- cumsum(db$post_prob)
range(db$thetas[db$cumprob < 0.95])                # unimodal, we can use range
```

```
## [1] 1.767418 2.135385
```

```
## HDPR using "density" instead (kernel density estimate)
post_density <- density(thetas)                          # only changes
db2 <- data.frame(thetas = post_density$x,               # only changes
                  post_prob = post_density$y / sum(post_density$y))
db2 <- db2[order(db2$post_prob, decreasing = TRUE), ]    # same as above from now on
db2$cumprob <- cumsum(db2$post_prob)
range(db2$thetas[db2$cumprob < 0.95])
```

```
## [1] 1.699463 2.217133
```

Example 3.1.4 (Mixture/multimodal)**.** Suppose the posterior for one parameter is a mixture of two distribution. HDPR can be computed as follows and depicted in figure 3.2

```
set.seed(1)
rmixture <- function(n = 10000){
  mu1 <- 0
  mu2 <- 5
  sd  <- 1
  pi  <- 0.5
  a <- rnorm(n, mean = mu1, sd = sd)
  b <- rnorm(n, mean = mu2, sd = sd)
  ifelse(runif(n) < 0.5, a, b)
}

posterior <- rmixture()
hist(posterior, breaks = 100)

## write a function that does it all in the general case
hpdr <- function(posterior, level = 0.95){
  post_density <- density(posterior)
  db2 <- data.frame(thetas = post_density$x,
                    post_prob = post_density$y / sum(post_density$y))
  db2 <- db2[order(db2$post_prob, decreasing = TRUE), ]    # same as above from now on
  db2$cumprob <- cumsum(db2$post_prob)
  db2$in_hpdr <- db2$cumprob < 0.95
  db2 <- db2[order(db2$thetas), c("thetas", "in_hpdr")]
  # search for changes in ci belonging
  db2$change <- abs(c(0, diff(db2$in_hpdr))) > 0
  db2[db2$change, 'thetas']
}

(ci <- hpdr(posterior))

## [1] -2.017002  2.160098  2.903965  7.023845

abline(v = ci, col = 'red', lty = 'dashed')
```

Example 3.1.5 (Asimmetric posterior)**.** Supposing for a proportion a uniform prior distribution and an experiment of 2 successes among 10 unit extracted. The posterior is $\theta|suminx_i = 2 \sim \text{Beta}\,(()1+2, 1+8)$. Let's plot the distribution and compute the intervals in fig 3.3

```
succ <- 3
fail <- 9

## quantiles interval
(quantile_int <- qbeta(c(0.025, 0.975), succ, fail))

## [1] 0.06021773 0.51775585
```

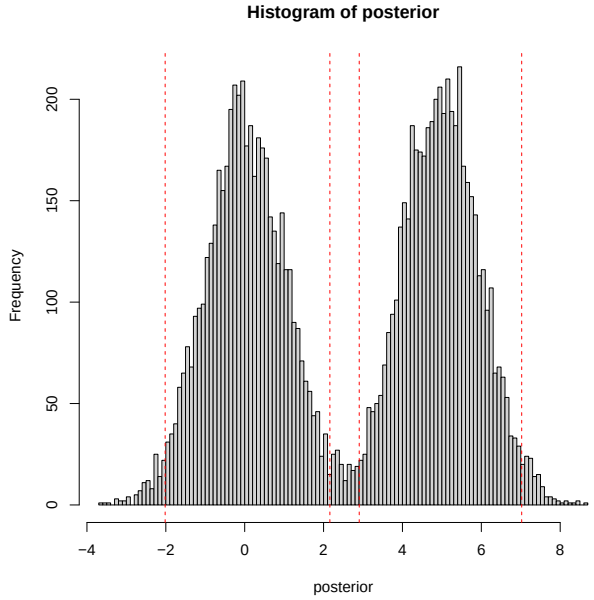


Figure 3.2: Mixture and hpdr

```
## hpdr
posterior_sim <- rbeta(10000, succ, fail)
(hpdr_int <- hpdr(posterior_sim))

## [1] 0.03949387 0.49114634

## plot
plot_fun(function(x) dbeta(x = x, succ, fail),
         from = 0, to = 0.8, cartesian_plane = FALSE,
         xlab = expression(theta), ylab = "P(theta|x)")
abline(v = quantile_int, col = 'red')
abline(v = hpdr_int, col = 'blue')
legend('topright', legend = c('quantiles', 'HPDR'),
       col = c("red", "blue"), lty = 1)
```

HPDR identification with unimodality Lets refer to the univariate and unimodal case (unimodality helps since we hope to find a single maximum within the interval (a, b)) and use method of the Lagrange multipliers.

The Lagrangian function is:

$$\mathcal{L} = (b-a) + \lambda \left\{ \int_a^b h(\theta|x_1, \ldots, x_n) d\theta - (1-\alpha) \right\}$$

where λ is the cost associated to not satisfying the constraint. The quantity within brackets is to be minimized.

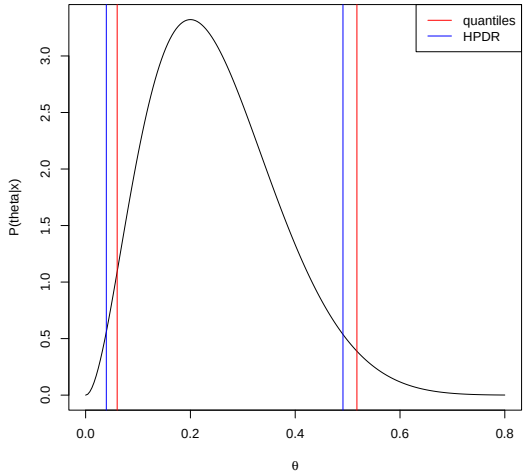


Figure 3.3: Asymmetric posterior

We know that $[H(\theta)]_a^b = H(b) - H(a)$ and that $h(\theta) = \frac{\partial H(\theta)}{\partial \theta}$. So:

$$\frac{\partial \mathcal{L}}{\partial a} = -1 - \lambda h(a|x_1, \ldots, x_n) + 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = 1 + \lambda h(b|x_1, \ldots, x_n) + 0$$

In order to verify the first-order condition we set these derivatives equal to 0 and find the critical point in the interval (a, b).

$$-1 - \lambda h(a|x_1, \ldots, x_n) = 0 \quad ; \quad h(a|x_1, \ldots, x_n) = -\frac{1}{\lambda}$$

$$1 + \lambda h(b|x_1, \ldots, x_n) = 0 \quad ; \quad h(b|x_1, \ldots, x_n) = -\frac{1}{\lambda}$$

Being a probability function, $h(\cdot)$ must be always positive hence λ must be negative ($\lambda < 0$).

To understand if the critical value is a minimum or a maximum we compute the second derivatives, that are contained in a 2×2 table. The diagonal elements are:

$$\frac{\partial^2 \mathcal{L}}{\partial a^2} = -\lambda \frac{\partial h(a|x_1, \ldots, x_n)}{\partial a} \quad \text{and} \quad \frac{\partial^2 \mathcal{L}}{\partial b^2} = -\lambda \frac{\partial h(b|x_1, \ldots, x_n)}{\partial b}$$

The out-of-diagonal elements are equal to zero since:

$$\frac{\partial^2 \mathcal{L}}{\partial a \partial b} = \frac{\partial^2 \mathcal{L}}{\partial b \partial a} = 0.$$

The matrix of second derivatives is diagonal.

Since it must be $\lambda < 0$ both diagonal elements of the matrix of second derivatives of the Lagrangian function are positive, the Hessian matrix is definite positive. As an immediate consequence, the critical point identifies a minimum for the interval (a, b).

If we drop the assumption of unimodality the process of identification of the HPDR is not straightforward.

3.2 Prediction

NB prof: riorganizzata usando teoria di gelman con notazione rivista, e aggiungendo un esempio di hoff

Remark 49. In this section we study the probability distribution for a new observation; these are called *predictive distributions*.

3.2.1 Predictive distributions

Remark 50. To make inferences about a new observation we can do it basically in two moments: before or after the experiment.
This originates actually two possible distribution for this new observation, the *prior predictive distribution* and the *posterior predictive distribution*.
Now let's consider the general case where the single unknown parameter θ can assume infinite values

Definition 3.2.1 (Prior predictive distribution)**.** We want to make prediction for a new observation Z; we're actually interested in its distribution, which is

$$P(Z) \overset{(1)}{=} \int P(Z, \theta)\, \mathrm{d}\theta \overset{(2)}{=} \int P(\theta)P(Z|\theta)\, \mathrm{d}\theta$$

where:

- in (1) can be thought as marginalization of the joint distribution of Z, θ for any value of θ
- where in turn, (2), the joint distribution can be rewritten as product of prior distribution $P(\theta)$ and conditional distribution $P(Z|\theta)$

Remark 51. Things changes a bit if we set ourself in the post experiment moment and we can condition $P(Z)$ on the its information $P(Z|x_1, \ldots, x_n)$ retrieved on the observed sample. Adding the conditioning to the step above we are interested in the "marginal" of $P(Z|x_1, \ldots, x_n)$

Definition 3.2.2 (Posterior predictive distribution)**.**

$$\begin{aligned} P(Z|x_1, \ldots x_n) &\overset{(1)}{=} \int P(Z, \theta|x_1, \ldots, x_n)\, \mathrm{d}\theta \\ &\overset{(2)}{=} \int P(Z|\theta, x_1, \ldots, x_n) \cdot P(\theta|x_1, \ldots, x_n)\, \mathrm{d}\theta \end{aligned}$$

where

- (1) again we write "the marginal" as integration of the joint of Z, θ for any possible value of the added parameter θ

- (2) the joint of Z, θ can be rewritten conditioning on θ as well and weighting using the posterior distribution/probability for the new conditioning parameter θ

At this moment if we can assume *conditional independence* between Z and the sample, (that is $Z \perp\!\!\!\perp x_1, \dots, x_n | theta$, eg if we sample a new independent unit, not say a time/geographical series), then we can further have that $P(Z|\theta, x_1, \dots, x_n) = P(Z|\theta)$ and the equation simplify further

$$P(Z|x_1, \dots x_n) = \int P(Z|\theta) \cdot P(\theta|x_1, \dots, x_n) \, \mathrm{d}\theta$$

Remark 52. In general when casually speaking about "predictive distribution", one likely means the posterior predictive distribution.

3.2.2 Examples

NB prof: integrato Hoff pag 40, 47

Example 3.2.1 (Beta binomial)**.** Let $x_1, \dots, x_n$ be the outcomes from a sample of n binary random variable/observation, a beta prior for θ, and let Z be a new observation, coming from the sampe population, yet to be observed, that can be either $Z = 1$ or $Z = 0$.
We're interested in $P(Z = 1|x_1, \dots, x_n)$. We have:

$$\begin{aligned}
P(Z = 1|x_1, \dots, x_n) &= \int P(Z = 1, \theta|x_1, \dots, x_n) \, \mathrm{d}\theta \\
&= \int P(Z = 1|\theta, x_1, \dots, x_n) \cdot P(\theta|x_1, \dots, x_n) \, \mathrm{d}\theta \\
&\overset{(1)}{=} \int P(Z = 1|\theta) \cdot P(\theta|x_1, \dots, x_n) \, \mathrm{d}\theta \\
&= \int \theta \cdot P(\theta|x_1, \dots, x_n) \, \mathrm{d}\theta \\
&= \mathbb{E}\left[\theta|x_1, \dots, x_n\right] = \frac{a + \sum_{i=1}^n x_i}{a + b + n}
\end{aligned}$$

where in (1) we assumed $Z, x_1, \dots, x_n$ are independent.
We know as well that

$$P(Z = 0|x_1, \dots, x_n) = 1 - P(Z = 1|x_1, \dots, x_n) = 1 - \frac{a + \sum_{i=1}^n x_i}{a + b + n} = \dots = \frac{b + \sum_{i=1}^n (1 - x_i)}{a + b + n}$$

Remark 53. In this example we see how in general the posterior predictive distribution:

1. does not depend on any unknown quantity (θ is ruled out by marginalization); if it did, we would not be able to use it to make predictions
2. does depend on previously observed data, this because $x_1, \dots, x_n$ gives information on θ which in turn gives information about Z. It would be bad if Z were independent of $X_1, \dots, X_n$ it would mean that we could never infer anything about the unsampled population from the smple cases

Example 3.2.2 (Experiment of $n = 1$, discrete uniform prior)**.** Coming back to example 2.3.1, a binary experiment (with outcome $x = 1$ or $x = 0$) is performed. As we've seen, the experiment induces two different posterior distributions, each conditional to the occurred realization.
The single future observation Z may be $Z = 1$ or $Z = 0$.
The task of prediction aim to develop $p(Z|x)$ ruling out the values of the parameters θ; in the case of discrete distribution for θ

$$p(Z|x) \overset{(1)}{=} \sum_j p(Z, \theta_j|x) \overset{(2)}{=} \sum_j p(Z|\theta_j, x)p(\theta_j|x) \overset{(3)}{=} \sum_j p(Z|\theta_j)p(\theta_j|x)$$

where:

- (1) again we write the marginal on Z as a sum on the joint Z, θ_j
- (3) computations lie on the very important hypothesis that future and past observations are independent *conditionally* on the values θ_j, that is $Z \perp\!\!\!\perp x|\theta_j$, so that $P(Z|\theta_j, x) = P(Z|\theta_j)$ meaning that the distribution of a future observation is the same of the generic past observation. If that holds, in this case all the random variables in the example Z, X are ruled by the Bernoulli distribution:

$$P(Z|\theta) = \theta^z(1-\theta)^{1-z}$$

In the numerical example 2.3.1, the parameter θ has 4 possible values $(0.2; 0.4; 0.6; 0.8)$.
The probability of a future success given a success in the experiment is

$$\begin{aligned} P(Z = 1|x = 1) &= \sum_j p(Z = 1|\theta_j)p(\theta_j|x = 1) \\ &\overset{(1)}{=} \sum_j \theta_j p(\theta_j|x = 1) \\ &= 0.2 \times 0.1 + 0.4 \times 0.2 + 0.6 \times 0.3 + 0.8 \times 0.4 \\ &= 0.02 + 0.08 + 0.18 + 0.32 = 0.6 \end{aligned}$$

where in (1) remembering that the probability of a future success conditional on the values of the parameters is $P(Z = 1|\theta_j) = \theta_j$. On the other hand, the probability of a future failure given a success in the experiment is

$$\begin{aligned} P(Z = 0|x = 1) &= \sum_j p(Z = 0|\theta_j)p(\theta_j|x = 1) \\ &= \sum_j (1 - \theta_j)p(\theta_j|x = 1) \\ &= 0.8 \times 0.1 + 0.6 \times 0.2 + 0.4 \times 0.3 + 0.2 \times 0.4 \\ &= 0.08 + 0.12 + 0.12 + 0.08 = 0.4 \end{aligned}$$

Note that as expected said it must be $P(Z = 1|x = 1) + P(Z = 0|x = 1) = 1$.
Similarly it must be that (not verified/shown) $p(Z = 1|x = 0) + p(Z = 0|x = 0) = 1$

Example 3.2.3. [Gamma-poisson] As seen in section 2.5, for count data if we assume a parameter of interest with prior $\theta \sim Gamma(\alpha, \lambda)$, an experiment $x_1, \dots, x_n|\theta \sim \text{Pois}\,(()\theta)$ then the posterior of the parameter is $\theta|x_1, \dots, x_n \sim Gamma(\alpha + \sum_{i=1}^{n} x_i, \lambda + n)$.
Which is the **predictive probability distribution** for the count of a new observation Z? It turns out (see hoff pag 47) it is **negative binomial** with parameters $(\alpha + \sum_{i=1}^{n} x_i, \lambda + n)$

Example 3.2.4. [Numerical example] In the birth rate example we have the two posterior distribution which are $\theta_1 \sim Ga(219, 112)$, $\theta_2 \sim Ga(68, 45)$.
The predictive probability distribution for new observations are respectively $NBinom(219, 112)$ and $NBinom(68, 45)$.
To obtain them in R we must pay attention to the second parameter of `dnbinom` function which must be inserted as α/λ not λ. In fig **??** the posterior distribution for θ (left) and posterior predictive distribution for number of children (right)

```
par(mfrow = c(1, 2))

## plot 1
xlim <- c(0, 5)
plot_fun(f = function(x) dgamma(x = x, shape = 2, rate = 1),
         from = 0, to = 5, cartesian_plane = FALSE, ylim = c(0, 3),
         main = 'Prior and posterior distribution for theta',
         xlab = "theta", ylab = 'Density')
plot_fun(f = function(x) dgamma(x = x, shape = 219, rate = 112),
         from = 0, to = 5, add = TRUE, col = 'red', cartesian_plane = FALSE)
plot_fun(f = function(x) dgamma(x = x, shape = 68, rate = 45),
         from = 0, to = 5, add = TRUE, col = 'blue', cartesian_plane = FALSE)
legend('topright',
       legend = c("prior", "Less than BD", "BD or higher"),
       lty = "solid",
       col = c("black","red", "blue"))

## plot 2
n_children <- 0:10
round(p_nc_nobs <- dnbinom(n_children, size = 219, mu = 219/112), 3)

##  [1] 0.143 0.277 0.269 0.176 0.086 0.034 0.011 0.003 0.001 0.000 0.000

round(p_nc_bs   <- dnbinom(n_children, size = 68, mu = 68/45), 3)

##  [1] 0.224 0.332 0.249 0.126 0.049 0.015 0.004 0.001 0.000 0.000 0.000

plot(x = n_children, y = p_nc_nobs, type = 'h', ylim = c(0, 0.35),
     main = 'Posterior predictive distributions', col = "red")
points(x = n_children + 0.1, y = p_nc_bs  , type = 'h', col = 'blue')
legend('topright',
       legend = c("Less than BD","BD or higher"),
       lty = "solid",
       col = c("red","blue"))
```

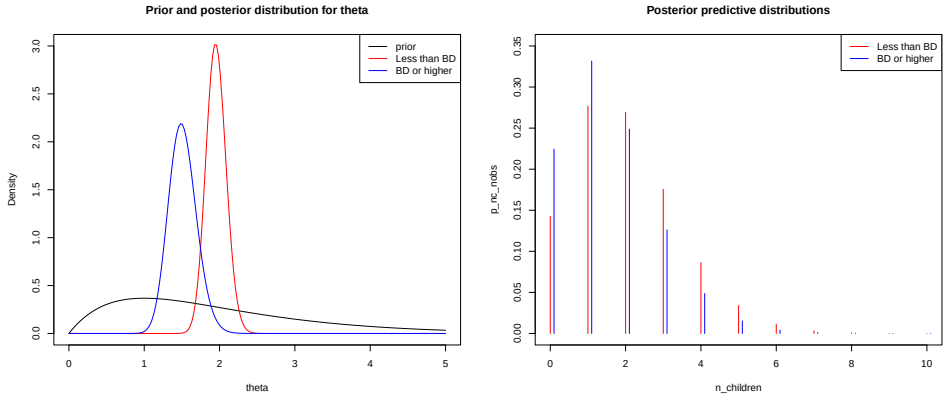


Figure 3.4: Posterior distribution of parameter and posterior predictive distributions for number of children

Remark 54. for the negative binomial with parameter $(\alpha + \sum_{i=1}^{n} x_i, \lambda + n)$ we have

$$E(Z|x_1,\dots,x_n) = \frac{\alpha + \sum_{i=1}^{n} x_i}{\lambda + n} = E(\theta|x_1,\dots,x_n)$$

$$Var(Z|x_1,\dots,x_n) = \frac{\alpha + \sum_{i=1}^{n} x_i}{\lambda + n}\frac{\lambda + n + 1}{\lambda + n} = Var(\theta|x_1,\dots,x_n)\cdot(\lambda + n + 1) = E(\theta|x_1,\dots,x_n)\cdot\frac{\lambda + n + 1}{\lambda + n}$$

Focusing on the last equation, the predictive variance, this can be seen as a measure of our uncertainty in prediction on a new unit Z from the population. Uncertainty stems from uncertainty about the population (which is not a Dirac) and variability in sampling.
For:

- large n, uncertainty about θ is small, being $\frac{\lambda+n+1}{\lambda+n} \approxeq 1$ and uncertainty about Z stems primarily from sampling variability, which for the poisson model is equal to θ.
- small n uncertainty in Z includes the uncertainty in θ and so the total uncertainty is larger than just the sampling variability, that is $\frac{\lambda+n+1}{\lambda+n} > 1$

3.3 Hypothesis testing

3.3.1 Classical hypothesis

NB prof: lievemente riorganizzato, aggiunto da lee/wikipedia

Important remark 24 (History).

- First proposal: Pearson (1892)
- Fishers proposal: a non Bayesian rule based on Popper conception, relying on hypotheses falsification. According to this principle, it is not possible to establish criteria for accepting hypotheses. The hypothesis to be tested does not receive any probability (hypotheses are not random variables in

the frequentist context).
According to this viewpoint, the refusal of a null hypothesis does not favor any alternative: the decision is postponed until information is improved.

- Proposal of the p-value and its popularity in the era of computers. The p-value is the probability of observing the value that is actually obtained, or an even more extreme value, under the null hypothesis. This deserves a discussion.

- Introduction of the idea of alternative hypothesis by Neyman and Pearson. The innovation of the proposal is the computation of the ratio of the likelihood of the null hypothesis out of the likelihood of the alternative hypothesis. In this case, the power of a test can be computed.

- Also a decision based hypothesis testing has been proposed, which assesses the consequences of accepting the alternatives.

In classical hypothesis testing theory the main question is whether the alternative hypothesis has to be considered or not.

classical hypothesis testing is set on fixing hypothesis Definition of first and second type errors. Definition of a rejection region R; balancing first and second type errors when choosing the rejection region.

NB prof: ripreso da wikipedia

Important remark 25 (Performing a frequentist hypothesis test in practice). The typical steps involved in performing a frequentist hypothesis test in practice are:

1. Define a hypothesis (claim which is testable using data).

2. Select a relevant statistical test with associated test statistic T.

3. Derive the distribution of the test statistic under the null hypothesis from the assumptions. In standard cases this will be a well-known result. For example, the test statistic might follow a Student's t distribution with known degrees of freedom, or a normal distribution with known mean and variance.

4. Select a significance level (α), the maximum acceptable false positive rate. Common values are 5% and 1%.

5. Compute from the observations the observed value t_{obs} of the test statistic T.

6. Decide to either reject the null hypothesis in favor of the alternative or not reject it. The Neyman-Pearson decision rule is to reject the null hypothesis H_0 if the observed value t_{obs} is in the critical region, and not to reject the null hypothesis otherwise.

Important remark 26 (Problems with frequentist methods in hypothesis testing). We can mention:

- difficulty of understanding the real meaning of preassigned significance levels (boh non chiaro, è la prob di errore che accettiamo per poter rifiutare la nulla)

- inadequacy of the frequentist method for a point null hypothesis, since a sample may be found having a so great size to induce the refusal of the null hypothesis
- p-value depends on values that have not been observed (we add the probability of a more extreme result, which actually has not been observed): thus an hypothesis that is possibly true (H_0) may be refused because it did not predict (extremer) results that actually did not occur;
- p-value depends on the sample size: if it tends to infinity, the p-value tends to 0.

3.3.2 Bayesian hypothesis testing

Regarding the value assumed by θ we can define two competing situation by defining two disjoint set, Θ_0, Θ_1, to which θ can belong

$$\Theta_0 \cup \Theta_1 = \Theta, \quad \Theta_0 \cap \Theta_1 = \emptyset$$

The different competing hypotheses than can formalized as

$$H_0 : \theta \in \Theta_0$$
$$H_1 : \theta \in \Theta_1$$

We can define the posterior probabilities of that hypotheses

$$p_0 = P(\theta \in \Theta_0 | x_1, \ldots, x_n)$$
$$p_1 = P(\theta \in \Theta_1 | x_1, \ldots, x_n)$$

and decide between H_0, and H_1 accordingly.

3.3.3 Posterior OR factorization, Bayes factor

Important remark 27. Bayesian hypothesis testing aims at measuring how much the experimental evidence makes the main hypothesis, H_0, stronger than the alternative one H_1, by comparing posterior probability using the posterior odd in favour of H_0

Definition 3.3.1 (Posterior odd in favour of H_0)**.** Defined as the ratio between the posterior probability of H_0 over the posterior of H_1

$$\frac{p_0}{p_1} = \frac{P(\theta \in \Theta_0 | x_1, \ldots, x_n)}{P(\theta \in \Theta_1 | x_1, \ldots, x_n)} = \frac{P(H_0 | x_1, \ldots, x_n)}{P(H_1 | x_1, \ldots, x_n)}$$

Important remark 28 (Bayesian decision). The decision is taken according to the value of the posterior odds in favor of H_0: if $\frac{p_0}{p_1} > 1$, then H_0 is accepted. By the fact that Bayesians assign probabilities to hypotheses H_0 and H_1: significance levels α need not to be specified to make a choice.

Remark 55. It's called odds ratio because if the probabilities of the two competing hypotheses $p_0 + p_1 = 1$ (considered that $\Theta_0 \cup \Theta_1 = \Theta$, $\Theta_0 \cap \Theta_1 = \emptyset$), the ratio of the two probabilities is said odds in favor of the numerator (H_0).

Remark 56. Now let's see which are the components of the posterior odds ratio, by exploding the posterior probabilities using Bayes theorem

Important remark 29 (Posterior odds ratio factorization).

The posterior odds ratio may be developed as follows. Considering two *mutually exclusive* alternative hypotheses H_0 and H_1, as we did, following quantities can be computed:

$$P(\theta \in \Theta_0|x_1,\dots,x_n) = \frac{P(\theta \in \Theta_0)P(x_1,\dots,x_n|\theta \in \Theta_0)}{P(\theta \in \Theta_0)P(x_1,\dots,x_n|\theta \in \Theta_0) + P(\theta \in \Theta_1)P(x_1,\dots,x_n|\theta \in \Theta_1)}$$
$$P(\theta \in \Theta_1|x_1,\dots,x_n) = \frac{P(\theta \in \Theta_1)P(x_1,\dots,x_n|\theta \in \Theta_1)}{P(\theta \in \Theta_0)P(x_1,\dots,x_n|\theta \in \Theta_0) + P(\theta \in \Theta_1)P(x_1,\dots,x_n|\theta \in \Theta_1)}$$

If we combine them in the posterior odds ratio we get a simplification given that the denominator is common (this is due to the fact that Θ is partitioned in Θ_1 and Θ_2):

$$\frac{p_0}{p_1} = \underbrace{\frac{P(\theta \in \Theta_0|x_1,\dots,x_n)}{P(\theta \in \Theta_1|x_1,\dots,x_n)}}_{\text{Posterior OR}} = \underbrace{\frac{P(\theta \in \Theta_0)}{P(\theta \in \Theta_1)}}_{\text{Prior OR}} \cdot \underbrace{\frac{P(x_1,\dots,x_n|\theta \in \Theta_0)}{P(x_1,\dots,x_n|\theta \in \Theta_1)}}_{\text{Bayes factor}}$$

where thus the posterior odds in favour of H_0 is factorized in the prior odds in favour of H_0 times the so-called *Bayes factor*, which is the ratio of likelihood under the two concurrent hypotheses. By rearranging algebrically we have another interpretation of Bayes factor:

$$\text{Bayes factor} = \frac{\text{Posterior OR}}{\text{Prior OR}}$$

which is how change the odds in favour of H_0 after the experiment. Often the prior odds ratio is posed equal to 1.

Remark 57. The Bayes factor (that is the ratio between the posterior odds and the prior odds) depends only on the sample data: it indicates how much data are in favor of a model rather than another. It is an important feature for assessing, with respect to prior evaluations, the change of opinion after an experiment.
It is employed also as a tool for model selection: since the prior odds ratio is often 1, for an objectivist Bayesian the Bayes factor is a way to perform model comparison.

Remark 58. If the competing hypotheses are more than 2 instead, the one with greatest posterior probability can be chosen. In general the hypothesis with highest posterior probability is chosen.

Remark 59. We now see some cases where hypotheses can be *simple*, for which Θ_0 (or Θ_0) is composed of a single value θ_0 (respectively θ_1) or composite (more than one possible value).

3.3.4 Simple hypotheses

Remark 60. If both the competing hypotheses are simple, it turns out that the Bayes factor = Likelihood ratio.
The likelihood ratio is the frequentist test statistic for comparing simple hypotheses (Neyman Pearson Lemma)

prof: fatto tutti i li per esteso non conando la statistica sufte per avere il caso rale (eg lee p4.4.4 pag

Example 3.3.1. Supposing that we extract from a normal population $X \sim N(\theta, 1)$ with unknown parameter of interest and unitary variance. Let

$$H_0 : \theta = 0$$
$$H_1 : \theta = 1$$

so we have 2 competing hypotheses, both simple. Finally, we suppose that $P(H_0) = P(H_1) = 0.5$.
The densities under H_0 $(\mu = \theta_0 = 0, \sigma = \sigma^2 = 1)$

$$f_{H_0}(x) = \frac{1}{\sqrt{2\pi \cdot 1}} \exp\left[-\frac{1}{2}\left(\frac{x-0}{1}^2\right)\right] = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}x^2\right]$$

while under H_1 $(\mu = \theta_1 = 1, \sigma = \sigma^2 = 1)$

$$f_{H_1}(x) = \frac{1}{\sqrt{2\pi \cdot 1}} \exp\left[-\frac{1}{2}\left(\frac{x-1}{1}^2\right)\right] = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}x^2 - \frac{1}{2} + x\right]$$

The likelihood ratio from the sample/experiment is

$$\begin{aligned}
\frac{P(x_1, \dots, x_n|\theta_0)}{P(x_1, \dots, x_n|\theta_1)} = \frac{\prod_{i=1}^n f_{H_0}(x)}{\prod_{i=1}^n f_{H_1}(x)} &= \frac{\prod_{i=1}^n \frac{1}{\sqrt{2\pi \cdot}} \exp\left[-\frac{1}{2}x^2\right]}{\prod_{i=1}^n \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}x^2 - \frac{1}{2} + x\right]} \\
&= \prod_{i=1}^n \exp\left[-\frac{1}{2}x_i^2 + \frac{1}{2}x_i^2 + \frac{1}{2} - x_i\right] \\
&= \exp\left[\sum_{i=1}^n \left(\frac{1}{2} - x_i\right)\right] = \exp\left(\frac{n}{2} - \overline{x} \cdot n\right) \\
&= \exp\left[n\left(\frac{1}{2} - \overline{x}\right)\right]
\end{aligned}$$

Note that the "baricentre" is $\frac{1}{2}$ which is halfway between θ_0 and θ_1 (if $\overline{x} < \frac{1}{2}$ the term between parenthesis positive and likelihood ratio greater than 1, showing experimental evidence in favour of H_0).
Going back to the posterior odds ratio we have:

$$\frac{P(H_0|x_1, \dots, x_n)}{P(H_1|x_1, \dots, x_n)} = \frac{P(H_0)}{P(H_1)} \cdot \frac{P(x_1, \dots, x_n|\theta_0)}{P(x_1, \dots, x_n|\theta_1)} = 1 \cdot \exp\left[n\left(\frac{1}{2} - \overline{x}\right)\right] = \exp\left[n\left(\frac{1}{2} - \overline{x}\right)\right]$$

If in the experiment with $n = 10$ we have $\overline{x} = 2$ (nearest to alternative hypothesis parameter) Bayes factor and posterior OR become:

$$\frac{P(H_0|x_1, \dots, x_n)}{P(H_1|x_1, \dots, x_n)} = \exp(-15)$$

so the posterior odds is a very small value, which induces to reject H_0 in favor of H_1.

Remark 61. As we'll see then, the equality

$$\text{Bayes factor} = \text{Likelihood ratio}$$

should hold only for simple null and alternative hypotheses.

3.3.5 Simple null and composite alternative

Important remark 30. In this case we have that

$$H_0 : \theta = \theta_0, \quad \Theta_0 = \{\theta_0\}$$
$$H_1 : \theta \in \Theta_1, \quad \Theta_1 = \{\theta_1, \ldots\}$$

and at the denominator, for the likelihood of alternative hypothesis, we need to average for possible values/probabilities of each $\theta \in \Theta_1$, obtaining the posterior odds:

$$\begin{aligned}\frac{P(H_0|x_1,\ldots,x_n)}{P(H_1|x_1,\ldots,x_n)} &= \frac{P(\theta=\theta_0)\cdot P(x_1,\ldots,x_n|\theta_0)}{P(\theta\in\Theta_1)\cdot P(x_1,\ldots,x_n|\theta\in\Theta_1)} \\ &= \frac{P(\theta=\theta_0)}{P(\theta\in\Theta_1)}\cdot\frac{P(x_1,\ldots,x_n|\theta_0)}{\int_{\theta\in\Theta_1} g_1(\theta)\cdot P(x_1,\ldots,x_n|\theta)\,\mathrm{d}\theta}\end{aligned}$$

where

- $g_1(\theta)$ is defined to be a proper/normalized density (eg it integrate to 1 in Θ_1). Thus it's defined as:

  $$g_1(\theta) = \frac{P(\theta)}{P(\theta\in\Theta_1)}$$

  where at the numerator we have the prior density of θ and at denominator the total probability that θ falls in the Θ_1 region.

- we start to see the dependence of the calculation of the bayes factor from the prior of θ through $P(\theta)$ at previous point (so it's not correct to say that Bayes factor depends on experiment/sample data only)

Remark 62. Similar results can be derived even for for composite null and alternative hypothesis (but maybe this is less interesting)

3.3.6 Composite null and alternative hypotheses

Important remark 31. In the most general case we have

$$H_0 : \theta \in \Theta_0, \quad \Theta_0 = \{\theta_0, \ldots\}$$
$$H_1 : \theta \in \Theta_1, \quad \Theta_1 = \{\theta_1, \ldots\}$$

$$\Theta_0 \cap \Theta_1 = \emptyset$$
$$\Theta_0 \cup \Theta_1 = \Theta$$

Similarly to what done in the previous case, for the likelihoods of both hypothesis, we need to average for possible values/probabilities of each θ. The posterior odds become:

$$\begin{aligned}\frac{P(H_0|x_1,\ldots,x_n)}{P(H_1|x_1,\ldots,x_n)} &= \frac{P(\theta\in\Theta_0)\cdot P(x_1,\ldots,x_n|\theta\in\Theta_0)}{P(\theta\in\Theta_1)\cdot P(x_1,\ldots,x_n|\theta\in\Theta_1)} \\ &= \frac{P(\theta\in\Theta_0)}{P(\theta\in\Theta_1)}\cdot\frac{\int_{\theta\in\Theta_0} g_0(\theta)\cdot P(x_1,\ldots,x_n|\theta)\,\mathrm{d}\theta}{\int_{\theta\in\Theta_1} g_1(\theta)\cdot P(x_1,\ldots,x_n|\theta)\,\mathrm{d}\theta}\end{aligned}$$

where this time

- g_0, g_1 are again defined to be a proper/normalized density (eg they integrate to 1 in Θ_0, Θ_1 respectively). Thus as:

$$g_0(\theta) = \frac{P(\theta)}{P(\theta \in \Theta_0)}, \quad g_1(\theta) = \frac{P(\theta)}{P(\theta \in \Theta_1)}$$

  where at the numerator we have the prior density of θ and at denominator the total probability that θ falling in the two regions

- again the Bayes factor depend not only on the data but from the prior of θ through $P(\theta)$ and g_0, g_1at previous point (so it's not correct to say that Bayes factor depends on experiment/sample data only)

Chapter 4

Simulation

Remark 63. In Hoff Monte Carlo approximations (the basis of simulation methods) are developed in Chapter 4; posterior approximations via the Gibbs sampler are developed in Chapter 6; nonconjugate priors and Metropolis-Hastings algorithms are developed in Chapter 10.

4.1 Monte Carlo approximation

NB prof: ripreso da hoff

Remark 64 (Rationale). Once we've found the posteriors of our interest we may be interested in summarizing aspects of the posterior other than mean for example:

- calculate $P(\theta \in A|x_1, \ldots, x_n)$ for arbitrary set A
- interested in the distribution of a function/transformation of θ
- interested in the distribution of a function/transformation of more than one parameters: eg comparing two posterior distribution of two parameters, say looking at the distribution $\theta_1 - \theta_2$ or θ_1/θ_2

Important remark 32. Obtaining formula/distribution for these posterior quantities can be difficult/impossible; however if we can generate random sample values of the parameters θs from their posterior distributions involved, then then all these quantities of interest can be aproximated to an arbitrary precision using Monte Carlo method

4.1.1 Method

4.1.1.1 Single parameter

Let θ be a parameter of interest and $x_1, \ldots, x_n$ the sample from a distribution $p(x_1, \ldots, x_n|\theta)$.
Suppose we could sample S independent random values from the *posterior distribution* $p(\theta|x_1, \ldots, x_n)$:

$$\theta^{(1)}, \ldots, \theta^{(S)} \sim \text{iid } p(\theta|x_1, \ldots, x_n)$$

then, as S increases, the empirical distribution of the samples $\left\{\theta^{(1)}, \ldots, \theta^{(S)}\right\}$ would approximate $p(\theta|x_1, \ldots, x_n)$ better and better. For this reason it's called called Monte carlo approximation to $p(\theta|x_1, \ldots, x_n)$.
Furthermore let $g(\theta)$ be any function of the parameter; we know that the expected value of $g(\theta)$ is the integral (g can be identity as well):

$$E[g(\theta)|x_1, \ldots, x_n] = \int p(\theta|x_1, \ldots, x_n) \cdot g(\theta)\, \mathrm{d}\theta$$

The law of large numbers says that if $\left\{\theta^{(1)}, \ldots, \theta^{(S)}\right\}$ are iid samples from $p(\theta|x_1, \ldots, x_n)$ then, as $S \to \infty$ the sample mean of the transformed parameter converge to the expected value, that is to the value of the integral above

$$\frac{1}{S}\sum_{s=1}^{S} g(\theta^{(s)}) \to E[g(\theta)|x_1, \ldots, x_n]$$

This implies that, as $S \to \infty$, any aspect of a posterior distribution we may be interested in can be approximated arbitrarily exaclty with a large enough Monte carlo sample. Precisely:

- the monte carlo mean (where g is just the identity) converges to the expected value of the posterior:

$$\hat{\theta} = \frac{\sum_{s=1}^{S} \theta^{(s)}}{S} \to E[\theta|x_1, \ldots, x_n]$$

- sample variance converges to distribution variance

$$\hat{\sigma}^2 = \frac{\sum_{s=1}^{S} \theta^{(s)} - \hat{\theta}}{S-1} \to Var[\theta|x_1, \ldots, x_n]$$

- the proportion of sample below a treshold converge to the probability distribution

$$\frac{\#(\theta^{(s)} \leq c)}{S} \to P(\theta \leq c|x_1, \ldots, x_n)$$

- the empirical distribution converge to the theorical one

$$\left\{\theta^{(1)}, \ldots, \theta^{(S)}\right\} \to p(\theta|x_1, \ldots, x_n)$$

- α-quantile of $\left\{\theta^{(1)}, \ldots, \theta^{(S)}\right\}$ converge to corresponding θ_α

Example 4.1.1. As a simple start, let's suppose our posterior is a N(0,1) simulation of monte carlo mean (0), cumulative distribution at 0 (0.5), 97.5 quantile (1.96). We plot the monte carlo estimates of these quantities up to $S = 10000$ simulations (fig 4.1)

```
S <- 10000
s <- 1:S
set.seed(452304)
```

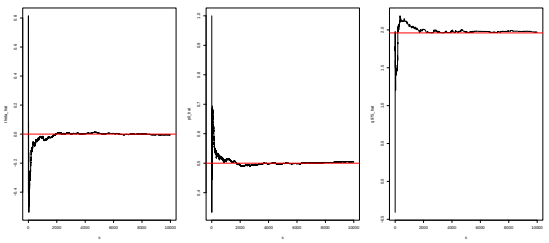


Figure 4.1: MC sim

```r
thetas <- rnorm(S)

cummean <- function(x) cumsum(x)/seq_along(x)

theta_hat <- cummean(thetas) # mean
p0_hat <- cummean(thetas < 0) # cumulative distribution: % below 0

cumf <- function(x, f){
  id <- seq_along(x)
  apply_f_up_to_i <- function(i) f(x[seq_len(i)])
  sapply(id, apply_f_up_to_i)
}

q975_hat  <- cumf(thetas, function(x) quantile(x, probs = 0.975))

## plot
par(mfrow = c(1,3))
plot(s, theta_hat, type = 'l')
abline(h = 0, col = 'red')

plot(s, p0_hat, type = 'l')
abline(h = 0.5, col = 'red')

plot(s, q975_hat, type = 'l')
abline(h = qnorm(.975), col = 'red')
```

4.1.1.2 Mean confidence bar

Furthermore, due to the Central limit theorem, for large S we have that monte carlo sample mean is normally distributed with mean $E[g(\theta)|x_1, \ldots, x_n]$ and variance $\hat{\sigma}^2/S$ thus

$$\frac{\hat{\theta} - E[g(\theta)|x_1, \ldots, x_n]}{\sqrt{\hat{\sigma}^2/S}}$$

is approximately distributed as $N(0, 1)$; this could be helpful to construct

- test regarding the value of the expected value/integral

- confidence bounds on the approximation of the expected value: eg an approximate .95 CI monte carlo confidence interval for posterir mean of θ is $\hat{\theta} \pm 1.96\sqrt{\sigma^2/S}$
- choose the number of simulation S to be large enough so that the monte carlo standard error $\sqrt{\sigma^2/S}$ is less than a specified precision if we want to report a confidence interval for $E(\theta|x_1, \dots, x_n)$.
  EG supposing we've generated a Monte carlo sample of size $S = 100$, for which the estimated $Var(\theta|x_1, \dots, x_n)$ was 0.024; if we want that the width of the confidence interval of the monte carlo mean to be less than 0.01 we would need to increase our sample size so that

$$1.96\sqrt{0.024/S} < 0.01 \implies S > 960$$

Example 4.1.2. plot the monte carlo estimate of mean with asymptotic normal confidence bar in fig 4.2

```
S <- 1000
s <- 1:S
set.seed(452304)
thetas <- rnorm(S)

cummean <- cumsum(thetas)/seq_along(thetas)
cumvar <- cumsum(thetas^2)/seq_along(thetas) - cummean^2
cumse  <- sqrt(cumvar/seq_along(thetas))

z <- qnorm(0.975)
up <- cummean + z * cumse
low <- cummean - z * cumse

plot(s, cummean, type = 'l')
lines(s, up, type = 'l', lty = 'dotted')
lines(s, low, type = 'l', lty = 'dotted')
abline(h = 0, col = 'red')
```

4.1.1.3 Multi-parameter

NB prof: integrazione lambert pag 272

Remark 65 (Multiparameter monte carlo simulation). In this way we see how a sample mean can approximate an integral of interest (expected value); the solution holds for multiple parameters/integrals of interest (not yet seen in practice). For example, let θ_1, θ_2 be two parameters of interest of a distribution: supposing we can extract from the joint posterior distribution $p(\theta_1, \theta_2|x_1, \dots, x_n)$ one way to calculate the bivariate expected value (and avoid double integration) is using Monte carlo method.
The expected value of the distribution $g\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right)$ (with $g : \mathbb{R}^2 \to \mathbb{R}^2$ I guess) is

$$E\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}|x_1, \dots x_n\right) \cdot g\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) \mathrm{d}\theta_2 \, \mathrm{d}\theta_1$$

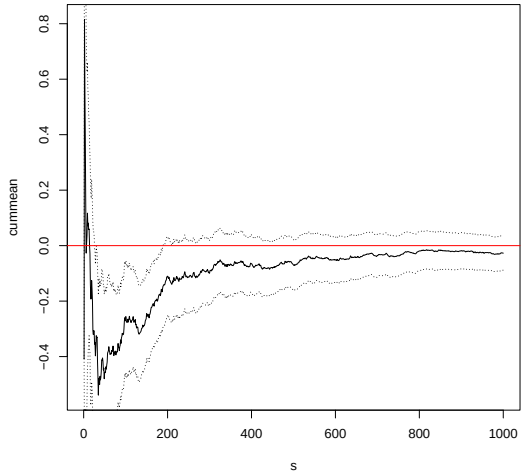


Figure 4.2: MC sim2

It turns out that the monte carlo sample mean converge to the expected value

$$\frac{1}{S}\sum_{s=1}^{S}\left(\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right) \to E\left(g\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right)$$

allowing us to avoid a double integral and simply computing a (vector) mean

4.1.2 Applications

4.1.2.1 Posterior inference for arbitrary functions

Important remark 33 (Function of one parameter). Supposing we're interested in a transformation γ of the parameter coming from the posterior we have to simply transform the estractions $\gamma(\theta)$, obtaining $\left\{\gamma^{(1)} = \gamma(\theta_1), \dots, \gamma^{(S)}\right\}$ and according to the law of large number

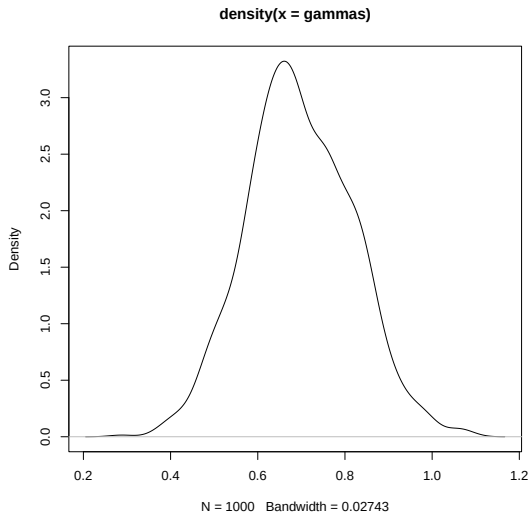
- the empirical distribution $\left\{\gamma(\theta_1), \dots, \gamma^{(S)}\right\} \to p(\gamma|x_1, \dots, x_n)$
- $\hat{\gamma} = \sum_{s=1}^{S} \gamma^{(s)}/S \to E(\gamma|x_1, \dots, x_n)$
- $\sum_{s=1}^{S}(\gamma^{(s)} - \hat{\gamma})^2/(S-1) \to Var(\gamma|x_1, \dots, x_n)$

Example 4.1.3. Suppose we've the posterior $p(\theta|x_1, \dots, x_n)$ of a beta binomial model were θ is a proportion, suppose a $Beta(200, 100)$; if we're interested in the log-odds (instead of the proportion)

$$\gamma(\theta) = \log\left(\frac{\theta}{1-\theta}\right)$$

we procede simply as follows to plot the density of the transformation

```r
thetas <- rbeta(1000, 200, 100)
gammas <- log(thetas/(1-thetas))
# distribution
plot(density(gammas))
```



Important remark 34 (Function of two parameters). We may be interested in comparing two parameters:'asis'

- we may compare them via a function, eg $\theta_1 - \theta_2$ or θ_1/θ_2
- we may be interested in computing $P(\theta_1 > \theta_2 | x_1, \ldots, x_n)$

Both these quantities (distribution, probability) can be obtained with Monte carlo sampling.
The sequence of couples $\left\{(\theta_1^{(1)}, \theta_2^{(1)}), \ldots, (\theta_S^{(1)}, \theta_S^{(1)})\right\}$ and consists of S indepen-det samples from the joint posterior distribution of θ_1 and θ_2

- is an approximation of the bivariate distribution of the two parameters
- can be used to obtain univariate distributions of $\gamma = f(\theta_1, \theta_2)$ with $f : \mathbb{R}^2 \to \mathbb{R}^1$, such as difference or proportions
- can be used to obtain Monte carlo approximations such as

$$\frac{1}{S}\sum_{s=1}^{S} I(\theta_1^{(s)} > \theta_2^{(s)}) \to P(\theta_1 > \theta_2 | x_1, \ldots, x_n)$$

Example 4.1.4. Supposing, in the example of birth rate per woman without (θ_1) and with (θ_2) bachelor degree, that posterior distribution after conducting a study that

$$\theta_1 | x_{1,1}, \ldots, x_{n_1,1} \sim Gamma(219, 112)$$
$$\theta_2 | x_{1,2}, \ldots, x_{n_2,2} \sim Gamma(68, 45)$$

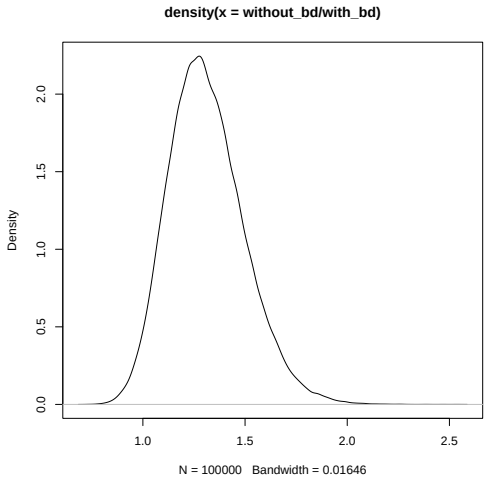
Then, if we want to

- calculate $P(\theta_1 > \theta_2|x)$

```
S <- 1e05
without_bd <- rgamma(S, 219, 112) # theta_1
with_bd <- rgamma(S, 68, 45) # theta_2
mean(without_bd > with_bd) # this is the final result

## [1] 0.9726
```

- if we want to construct the density of the ratio θ_1/θ_2

```
plot(density(without_bd / with_bd))
```



4.1.2.2 Sampling from predictive distributions

In section 3.2.1 we have seen how, under conditional independence (on the parameter of interest) the probability distribution of the new observation given the results of the experiment (posterior predictive distribution) can be written as

$$P(Z = z|x_1, \dots x_n) = \int P(Z = z|\theta) \cdot P(\theta|x_1, \dots, x_n)\, \mathrm{d}\theta$$

To obtain the posterior predictive probability that Z is equal to some specific value z we can apply the Monte carlo method:

- sample S thetas from experiment posterior: $\theta^{(1)}, \dots, \theta^{(S)} \sim$ iid $p(\theta|x_1, \dots, x_n)$
- use the extracted thetas to sample individual observation from the corresponsing distribution involving theta: $z^{(1)}\ p(z|\theta^{(1)}), \dots, z^{(S)}\ p(z|\theta^{(S)})$

- the sequence $\left\{(\theta^{(1)}, z^{(1)}), \ldots, (\theta^{(S)}, z^{(S)})\right\}$ constitutes S independent samples from the joint posterior distribution of (θ, Z) while the sequence $z^{(1)}, \ldots, z^{(S)}$ S independent samples from the *marginal* posterior distribution of Z, which is the posterior predictive distribution

- we can obtain an approximation of $P(Z = z|x_1, \ldots, x_n)$ by calculating the monte carlo mean $\sum_{s=1}^{S} p(z|\theta^{(s)})/S$. [1]

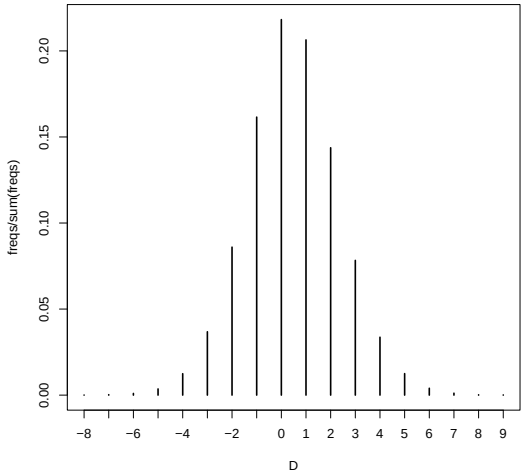
Example 4.1.5. Here we are interested in the predictive probability that an woman without college degree would have more children than one with college degree.
The posterior distribution of mean number of children for woman without and with bachelor degree are Gamma(219,112) and Gamma(68, 45) respectively

```
## create the posterior extraction of theta as before
S <- 1e05
without_bd_lambdas <- rgamma(S, 219, 112) # theta_1
with_bd_lambdas <- rgamma(S, 68, 45) # theta_2

## use all the extraction to extract a new observation, one for each theta
without_children <- rpois(S, without_bd_lambdas)
with_children <- rpois(S, with_bd_lambdas)

mean(without_children > with_children)

## [1] 0.48005
```

However, once we have generated these monte carlo samples from the posterior preictive distribution we can use to calculate other quantities of interest

```
  ## other aspects of interest
  mean(without_children) # mean children for a new obs without degree

## [1] 1.95504

  mean(with_children)

## [1] 1.51472

  ## difference D = (Z_1 - Z_2) in number of children between two individuals,
  ## one sampled from each group
  D <- without_children - with_children
  freqs <- table(D)
  plot(freqs/sum(freqs))
```

[1] This procedure will work well if $p(z|\theta)$ is discrte and we are interested in quantities that are easily computed from $p(y|\theta)$



4.1.2.3 Posterior predictive model checking

we can compare the sample data (ecdf) with posterior predictive density to

Example 4.1.6. Previously (exercise 3.2.4) we derived for two sample of the posterior distribution for women without bd is $\theta_1 \sim Gamma(219, 112)$, and thus predictive probabilities for number of children is $NBinom(219, 112)$.
We can compare this theorical distribution with empirical distribution from the sample (fig 4.3): the two distribution seem to be in conflict (eg the observed data have twice as many women with two children than one).

```
# data from https://www2.stat.duke.edu/~pdh10/FCBS/Replication/gss.RData

## load("/tmp/gss.RData")
## y1<-gss$CHILDS[gss$FEMALE==1 &  gss$YEAR>=1990  & gss$AGE==40 & gss$DEG<3 ]
## y1<-y1[!is.na(y1)]
## dput(y1)

## 111 female
data <- c(2L, 2L, 5L, 0L, 2L, 1L, 2L, 4L, 2L, 0L, 3L, 1L, 0L, 0L, 2L, 1L, 3L,
          3L, 2L, 3L, 1L, 2L, 2L, 2L, 6L, 2L, 3L, 2L, 1L, 2L, 2L, 2L, 0L, 2L,
          4L, 2L, 3L, 4L, 1L, 4L, 1L, 0L, 0L, 0L, 5L, 4L, 3L, 2L, 1L, 0L, 0L,
          3L, 2L, 1L, 0L, 2L, 1L, 4L, 2L, 1L, 2L, 0L, 1L, 3L, 4L, 0L, 0L, 4L,
          3L, 3L, 0L, 1L, 1L, 2L, 0L, 1L, 3L, 2L, 6L, 3L, 3L, 1L, 2L, 1L, 3L,
          1L, 3L, 2L, 2L, 2L, 2L, 2L, 3L, 2L, 3L, 0L, 2L, 2L, 3L, 2L, 1L, 4L,
          4L, 0L, 2L, 2L, 0L, 2L, 2L, 3L, 0L)

(edf <- table(data) / sum(table(data))) ## empirical distribution

## data
##          0          1          2          3          4          5          6
## 0.18018018 0.17117117 0.34234234 0.18018018 0.09009009 0.01801802 0.01801802
```

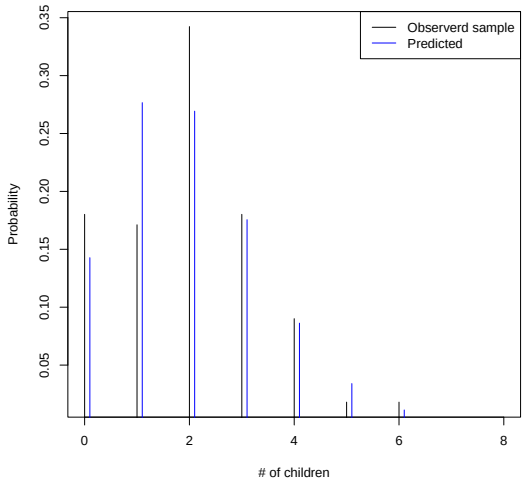


Figure 4.3: Observed vs predicted probabilities of number of children

```
pred_distr <- dnbinom(x = 0:8, size = 219, mu = 219/112) # posterior pred. distributi

## plot1
plot(x = 0:8, y = c(edf, NA, NA),
     type = "h", xlab = '# of children', ylab = 'Probability')
points(x = 0:8 + 0.1, y = pred_distr, col = 'blue', type = 'h')
legend('topright', legend = c("Observerd sample", "Predicted"), col = c("black", "blu
       lty = 1)
```

Remark 66. Some possible explanations for difference between sample and predicted distribution

1. it's the result of the sample variability for which the sample data is very different from the population: this particularly true if sample size is low (not the case with over 100 observations)

2. it's a feature of the population and the observed sample is correctly reflecting it. In contrast, the Gamma-Poisson model unable to represent this feature because there's no Poisson distribution that has such a sharp peak at $x = 2$

These explanation can be assessed numerically with Monte Carlo simulation. We may be interested in the ratio

$$t = \frac{\#\text{ mothers with two children}}{\#\text{ mothers with one children}}$$

which in the sample is $38/19 = 2$

```
table(data)

## data
##  0  1  2  3  4  5  6
## 20 19 38 20 10  2  2
```

Suppose we sample a different set of 111 women what distribution of t we would expect? This can be tackled using monte carlo; we could, for each simulation $s \in \{1, \ldots, S\}$

1. sample a θ from its posterior distribution $\theta^{(s)} \sim p(\theta|x_1, \ldots, x_n)$

2. create a sample of n new units using the posterior predictive density $Z^{(s)} = (z_1^{(s)}, \ldots, z_n^{(s)}) \sim \text{iid} p(z|\theta^{(s)})$, called *posterior predictive datasets* (each of size n)

3. compute or statistics of interest on the generated sample $t^{(s)} = t(Z^{(s)})$

Once the sample of $t^1 \ldots, t^S$ we

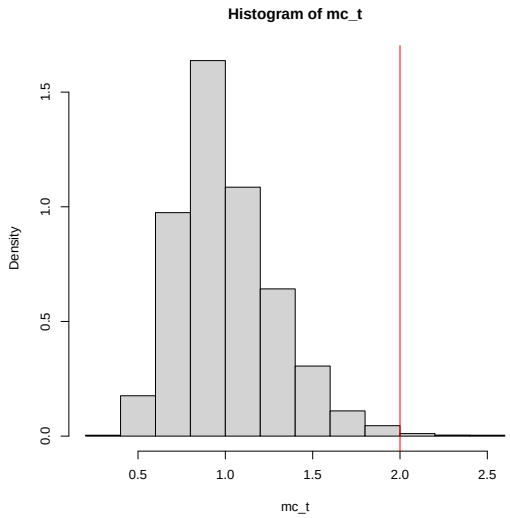
- plot the distribution of $t^1 \ldots, t^S$

- add the value which t assumes in the observed data

- calculate the probability of $t^1 \ldots, t^S$ being more extreme than the sample t

Example 4.1.7. Following the previous steps

```
S <- 10000
thetas <- rgamma(S, 219, 112)
mc_samples <- lapply(thetas, function(t) rpois(n = 111, lambda = t))
mc_t <- sapply(mc_samples, function(x) sum(x==2)/sum(x==1))

# plot ?hist
hist(mc_t, freq = FALSE)
abline(v = 2, col = 'red')
```




```
mean(mc_t >= 2)

## [1] 0.0054
```

So:

- if the gamma poisson model were correct we would obtain a observed value of the ratio of interest 0.51% cases[2]; this suggest our poisson model is flawed. It predicts that we would hardly ever see a dataset that resembled our observed one in terms of t; if we were interested in making prediction we would have to consider a more complicated model (for example, a multinomial sampling model).
- on the other hand, a simple Poisson model may suffice if we are interested only in certain aspects of predictive distribution. We should at least make sure that our model generates predictive datasets $Z^{(s)}$ that resemble the observed dataset in terms of features that are of interest.

4.2 Issues with independent sampling

NB prof: lambert 273

[2] These types of posterior predictive checks have given rise to a notion of posterior predictive p-values, which despite their name, do not generally share the same frequentist properties as p-values based on classical goodness-of-fit tests. This distinction is discussed in Bayarri and Berger (2000), who also consider alternative types of Bayesian goodness of fit probabilities to serve as a replacement for frequentist p-values.

Chapter 5

Priors

NB prof: rivista integrando da lambert/hoff

Remark 67. In previous parts we have seen some special (conjugate) priors: since these are useful to mimick the available knowledge only in subset of cases in this chapter we see other stuff and put it into the historical context of searching objectivity for bayesian analyses.[1] Before start talking about priors some general consideration on them follows.

Remark 68. In Bayesian analysis when we collect more data our conclusion become less affected by priors. The use of a prior allows to make inferences in small sample sizes by using pre-experimental knowledge of a situation, but in larger sampes the *effect of prior choice declines*.

Remark 69. The only exception to this rule (of diminishing importance of the prior) is when the prior has density values equal to zero for some parameter range: choosing a *zero-valued prior* across a parameter range always results in a corresponding zero posterior probability 5.1
For most cases the *step*-type prior should be discouraged, and smoother/ less definitive distribution chosen instead: remember that priors should represent our subjective viewpoints and choosing a zero.valued one ofr such event means that, from our perspective is impossible. In that case the analysis will follow in making it impossible in the posterior as well.

Important remark 35. Finally, we have an obbligation to report whenever the conclusion of an anlysis is sensitive to the form of the prior that is specified. Alternatively, a field called *sensitivity analysis* actually allows a range of priors to be specified and combined to produce a single posterior.

[1] The topics of this part are contained in Chapter 11 (11.4, 11.5) of Lambert (the multivariate normal model is illustrated in Chapter 8.4.11) In hoff:

- Indifference priors: proposed in a dedicated subsection (in Section 5.3) for the Normal distribution
- Jeffreys priors: proposed in some exercises of Chapter 3 (3.12 and 3.13), Chapter 5 (5.4) and Chapter 7 (7.1).
- Indifference priors for the Normal distribution: exposed in Chapter 5 of the textbook. Conjugate priors for the Normal distribution: exposed in Chapter 5 of the textbook. The multivariate Normal model is developed in Chapter 7 of the textbook.

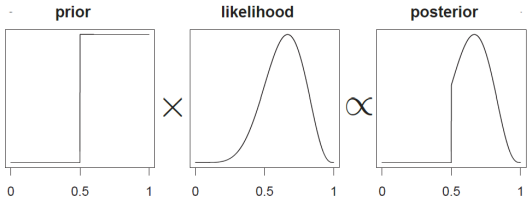


Figure 5.1: Zero prior and its effects

5.1 Informative priors

Remark 70. There are occasione when it is essential to include significant information in the prior: incorporate previously collected data or data from another source, include a former analysis result, expert opinion and so on.

5.1.1 Previous data and moment matching

If *previous data* is available, the construction of a prior can proceed by a method known as *moment matching*: we choose a theoretical distribution looking at shape of data (eg normal) and for parameters we adopt the estimates to the sample one (eg mean/std).
However this is not bayesian in nature[2] and can have some arbitrary choices (eg why mean/std and not skewness kurtosis

5.1.2 Expert opinion and eliciting priors

If *experts* are available we can have them to *eliciting the prior*: there are many methods to create priiors from subjective views, here we see a simple example.

Remark 71. These priors are often used in clinical trials, where clinicians are interviewed before the tral is conducted (also a considerable amount of research in the social sciences use these types of priors

Example 5.1.1. Let's assume a normal prior for the data: we're interested in estimating μ and σ.
Suppose we ask a sample of economists to provide estimates of the 25th and 75th percentiles of the wage benefit that one extra year of college grants on the job market: $wage_{25}$ and $wage_{75}$.
If we assume a normal prior for the data we can, for each economist/expert, relate the obtained quantities to the quantiles of a standardized normal distribution

$$\begin{cases} z_{25} = \frac{wage_{25}-\mu}{\sigma} \\ z_{75} = \frac{wage_{75}-\mu}{\sigma} \end{cases}$$

These two simultaneous equations could be solved for each expert, giving an estimate of μ and σ (which is our interest). These could then be *averaged* to determine the mean and standard deviation across all the experts.

[2] In chapter 17 of lambert a purer bayuesian method that can be used to create priors

Example 5.1.2. A better method instead of averaging relies of linear regression. Sytem of equation of previous example can be rearranged as follows

$$\begin{cases} wage_{25} = \mu + \sigma z_{25} \\ wage_{75} = \mu + \sigma z_{75} \end{cases}$$

we recognize that each equation represent a line $y = mx + c$ in $(z, wage)$ space where in this case $c = \mu$ and $m = \sigma$. If we fit a linear regression line to the data from the whole panel, the values of the y intercept and gradient hence estimate the mean and standard deviation.

5.2 Uninformative/Indifference priors

Remark 72. One criticism of bayesian inference was the dependence of final results on the chosen prior. These yielded to research conducted in order to make inference as objective as possible (eg with the use of "automatic" priors).

Remark 73 (Area of these priors). When there is a strong focus on the objectivity of analysis, as in regulatory areas such as drug development/public policy, then the use of *uninformative priors* is desired.
We aim to make *inference more based solely on the likelihood*, i.e. on what comes from the experiment, with a different interpretation with respect to classical statistics.

Remark 74. Concluding remarks: a non informative prior exists able to reproduce the results of classical statistics. However this peculiarity has been considered as a drawback by non Bayesians.
Following Lambert flat priors are usually not helpful for real-life analysis, and miss the point (no analysis is objective). True Bayesians recognise this and realise that a real benefit of the Bayesian approach is the possibility to include information based on previous experience

Remark 75. One point is what to consider informative; during the years several attempt were made.

Remark 76 (Are neutral priors proper or improper?). They are

- proper if the parameter is in a finite interval. In this case the uniform *a priori* is used, which is proper.
- improper if at least one extreme of the parameter support is not finite. Some researchers studied locally uniform priors and their consequences if transformations are performed.

5.2.1 Improper priors

Remark 77. Neutral, non informative prior distributions are often chosen among improper priors, by extending the idea of uniform prior distributions.

Example 5.2.1 (Uniform improper prior for a parameter allowed to vary in $\mathbb{R}$)**.** In the application of

$$p(\theta|\text{data}) = \frac{p(\theta)p(\text{data}|\theta)}{p(\text{data})}$$

we could set

$$p(\theta) = c$$

with $c \in \mathbb{R} \setminus \{0\}$, for all $\theta \in \mathbb{R}$.
Such priors might be improper (diverging integral) but even in this case the posterior would be a proper density, with shape determined by the likelihood function, since

$$p(\theta|\text{data}) \propto p(\theta)p(\text{data}|\theta) \propto p(\text{data}|\theta)$$

Remark 78 (Other flat priors). In practice these priors are often found in the realm of improper distributions (such as the previous case), which respect the following conditions:

- all possible values of the parameter are equally likely;
- the process starting from it refers to the minimal available information (eg for $U(0,1)$ prior are enough 2 virtual cases, $Beta(1,1)$);
- respects some constraints but gives scarce information on the parameters before data are obtained.
- general/not specific to the opinions of researchers and accepted by most of the observers (effort towards "objectivity")

Example 5.2.2 (An improper prior for a parameter having only positive values)**.** If we set as prior for a parameter $\theta \in R^+$ the function:

$$h(\theta) \propto \frac{1}{\theta}, \quad \theta \in \mathbb{R}^+$$

it's

- improper, still not a density function
- equivalent to assume a flat/uniform improper distribution on the parameter $\log\theta$, so that $f(\log\theta) = c$. If we apply the variable transformation theorem

  $$h(\theta) = f\left(g^{-1}(\theta)\right)\left|\frac{\partial}{\partial\theta}g^{-1}(\theta)\right|$$

  by having:

  $$\begin{aligned} &g^{-1}(\theta) = \log\theta \\ &f(g^{-1}(\theta)) = f(\log\theta) = c \\ &\frac{\partial}{\partial\theta}g^{-1}(\theta) = \frac{\partial}{\partial\theta}\log(\theta) = \frac{1}{\theta} \end{aligned}$$

  we conclude that

  $$h(\theta) = c \cdot \frac{1}{\theta} \propto \frac{1}{\theta}, \quad \theta \in \mathbb{R}^+$$

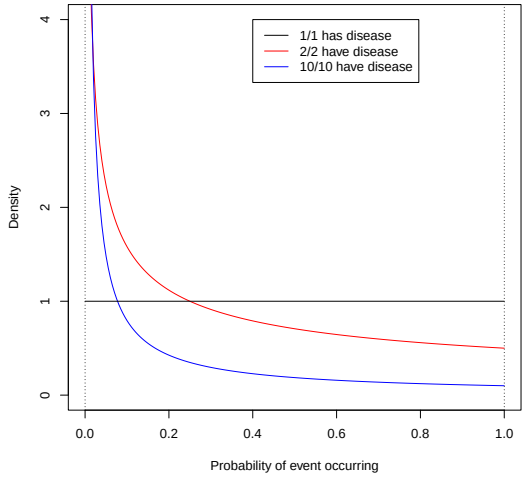


Figure 5.2: Transformation of flat prior

Important remark 36 (Flatness vs informativeness). The flatness of the uniform prior is often termed *uninformative* but this is somewhat misleading: a prior that is flat/uninformative on θ might not correspond to a flat/uninformative prior on a transformation of the parameter $g(\theta)$ (this has lead to the proposal of Jeffreys' priors)

Example 5.2.3. Assuming that the probability that one individual is disease positive is θ and hypothesizing flat uniform prior in the 0, 1 interval, the probability that two randomly chosen individuals have both the disease is θ^2.
The thing is that if we assume a flat prior for θ then this implies a decreasing, no more "flat/uninformative" prior for θ^2, as in figure 5.2, and the same goes on for the probability that in a sample of 10 individuals all are disease positive.
This happens because of density/variable transformation: knowing that $f(\theta) = 1$, $g(\theta) = \theta^2$, $g^{-1}(\theta) = \sqrt{\theta}$, $\frac{\partial}{\partial\theta}g^{-1}(\theta) = \frac{1}{2}\theta^{-1/2} = \frac{1}{2\sqrt{\theta}}$ we have that

$$f_{\theta^2}(\theta) = f_\theta(g^{-1}(\theta)) \cdot \left|\frac{\partial}{\partial\theta}g^{-1}(\theta)\right| = f(\sqrt{\theta}) \cdot \left|\frac{1}{2\sqrt{\theta}}\right| = 1 \cdot \left|\frac{1}{2\sqrt{\theta}}\right| = \frac{1}{2\sqrt{\theta}}$$

as well as, for the θ^{10} case

$$f_{\theta^{10}}(\theta) = \frac{1}{10\sqrt[10]{\theta^9}}$$

Remark 79. So even though a uniform prior for an event appears to convey no information, it actually confers quite confers quite considerable information about other events modelled using g

Important remark 37. However problems attached to choosing flat priors is swept under the carpet for most analyses because we usually care most about the particular event (parameter) for which we create a prior

Important remark 38. All priors contain some information, so we prefer the use of the terms *vague*, *diffuse*, *denoting ignorance*, to represent situations where a premium is placed on drawing conclusion based only on observed data.

5.2.2 Jeffreys priors

We know from that if we define a prior density in terms of one parameter θ we can derive the implied prior density pf a transformation g, $\phi = g(\theta)$ using the formula to compute derived densities from a transformation (Jacobian transform)

$$\begin{aligned} f_Y(y) &= f_X(g^{-1}(y))g^{-1}(y) \\ &= f_X(g^{-1}(y)) \left| \frac{\mathrm{d}x}{\mathrm{d}y} \right| \\ p(\phi) &= p(\theta) \left| \frac{\mathrm{d}x}{\mathrm{d}y} \right| \end{aligned}$$

Jeffreys said that a prior is *uninformative* if we can calculate it using the the above rule or directly through the function p and obtaining the same resul irrespective of the choice of parameter.

Example 5.2.4. For a uniform distribution $p(\theta) = 1$, but as seen in example 5.2.3, if we apply the transformation rule (eg for $\phi = \theta^2$) we obtain

$$p(\phi) = p(\theta^2) = \frac{1}{2\sqrt{\phi}}$$

However, if we approach the problem from the other side and substitute directly in for ϕ we get

$$p(\phi) = 1 \neq \frac{1}{2\sqrt{\phi}}$$

Clearly this distribution does not satisfy the requirements set out by Jeffreys

Jeffrey's suggested a default rule for generating a prior distribution of a parameter θ in a sampling model $p(x|\theta)$ given by:

$$p_J(\theta) \propto \sqrt{I(\theta)} = \sqrt{-\mathbb{E}\left[\frac{\partial^2 \log p(X|\theta)}{\partial \theta^2}\right]} \tag{5.1}$$

where $I(\theta)$ is the Fisher information (matrix), which indicates the curvature of the likelihood surface at a given point. used in maximum likelihood inference. So in this way actually the prior is defined on the base of the likelihood: where the curvature of the likelihood (second derivative) is higher, the prior is higher. The advantage of such prior is that it is invariant to transformations. In other words, it is scale invariant: any arbitrariness in the choice of parameters does not carry differences in results.
We can see that a prior that has the form of 5.1 satisfies his parameter invariance condition. Considering a parameter transformation $\theta \to \phi$

$$p(\phi) = \sqrt{-\mathbb{E}\left[\frac{\partial^2 \log p(x|\phi)}{\partial \phi^2}\right]} = \sqrt{-\mathbb{E}\left[\frac{\partial^2 \log p(x|\theta)}{\partial \theta^2}\right]\left(\frac{\mathrm{d}\theta}{\mathrm{d}\phi}\right)^2} = p(\theta)\left|\frac{\mathrm{d}\theta}{\mathrm{d}\phi}\right|$$

To obtain the third member form the second we used the chain rule of differentiation twice (since we differentiate the expression twice

5.2.2.1 Constructing a Jeffreys prior

Consider the transformation from θ to $\phi = g(\theta)$. Both $p(x|\theta)$ and $p(x|\phi)$ are conceivable, as well as the likelihoods and the Fisher informations $I(\theta; x)$ and $I(\phi; x)$.
Both probability distributions have their scores, but the equality between the two is reached when the suitable derivative is included. The following equality holds:

$$\frac{\partial l(\phi; x)}{\partial \phi} = \frac{\partial l(\theta; x)}{\partial \theta} \frac{\partial \theta}{\partial \phi}$$

Equality holds as well if we square both sides and apply expectation:

$$E\left\{ \left[\frac{\partial l(\phi; x)}{\partial \phi} \right]^2 \right\} = E\left\{ \left[\frac{\partial l(\theta; x)}{\partial \theta} \frac{\partial \theta}{\partial \phi} \right]^2 \right\}$$

Now the Fisher information appears in both sides:

$$I(\phi; x) = I(\theta; x) \left| \frac{\partial \theta}{\partial \phi} \right|^2$$

This result says that the expectation of the square of the score of the transformed variable is proportional to the expectation of the score of the *starting* variable: the quantity $\frac{\partial \theta}{\partial \phi}$ assumes the role of a constant since it does not depend on x.
In this way a reasearcher can firstly decide to propose the following prior

$$p(\theta) \propto \sqrt{I(\theta; x)},$$

but after the transformation one has also that

$$p(\phi) \propto \sqrt{I(\theta; x)},$$

Thus the first proposed prior $p(\theta) \propto \sqrt{I(\theta; x)}$ can be suitably assumed as a prior for ignorance.
Point is: iff two individuals use different parametrisation for a given distribution, but both use Jeffreys priors, then we can get from the posterior of one individual to the other's by applying the Jacobian transformation.
So the invariance property of Jeffreys priors is really about Jeffrey's posterior distribution. We actually find the word *invariance* misleading: posteriors do change under a change of parameters, and so are not invariant in this sense. Jeffreys priors just endsure that the posterior transforms in a way that is nice as determined by the Jacobian trnsformation

5.2.2.2 Examples of Jeffreys' priors

Example 5.2.5 (Bernoulli)**.** In the uniform 0-1 prior of example 5.2.3 we didn't introduced since our discussion was concerned only on priors; but now this is necessary because Jeffreys prior requires that we use it to calculate the information matrix.
A sensible likelihood to describe whether an individual is disesed $X \in \{0, 1\}$ is the bernoulli since we believe that the data are independent. To derive the information matrix (which is actually a scalar quantity here because tehre is only

a single parameter) for this particular case, we first write down the probability mass function:

$$\mathbb{P}(X = x|\theta) = \theta^x(1-\theta)^{1-x}$$

The loglikelihood:

$$\log \mathbb{P}(X = x|\theta) = x\log(\theta) + (1-x)\log(1-\theta)$$

Its first derivative

$$\frac{\partial \log \mathbb{P}(X = x|\theta)}{\partial \theta} = \frac{x}{\theta} - \frac{1-x}{1-\theta}$$

the second derivative

$$\frac{\partial^2 \log \mathbb{P}(X = x|\theta)}{\partial^2 \theta} = -\frac{x}{\theta^2} - \frac{1-x}{(1-\theta)^2}$$

We then take the expectation to yield the information matrix (which in this case is a constant)

$$I(\theta) = -\mathbb{E}\left[\frac{\partial^2 \log \mathbb{P}(X = x|\theta)}{\partial^2 \theta}\right] = \frac{\theta}{\theta^2} + \frac{1-\theta}{(1-\theta)^2} = \frac{1}{\theta} + \frac{1}{1-\theta} = \frac{1}{\theta(1-\theta)}$$

where we used $\mathbb{E}[x] = \theta$ for a bernoulli density. It is now straightforward to find Jeffreys prior using

$$p(\theta) \propto \sqrt{I(\theta)} = \theta^{-\frac{1}{2}}(1-\theta)^{-\frac{1}{2}} \sim \text{Beta}\left(\frac{1}{2}, \frac{1}{2}\right)$$

where we obtained the last line by noticing that the θ dependence is of the same form as a Beta $\left(\frac{1}{2}, \frac{1}{2}\right)$ density, meaning that the prior must be such a distribution.

The Jeffreys prior is plotted in fig 5.3. We note that this prior is inversely proportional to the standard deviation of the Bernoulli, given by $\theta^{1/2}(1-\theta)^{1/2}$: this make sens since the posterior are least affected by data where $\theta = 1/2$ (posterior is not very peaked) comapred to the more extreme values of θ where the likelihood exerts a strong effect (the posterior is more peaked). The likelihood exerts a strong effect at more extreme values of θ because the standard deviation is much lower near these points. The jeffreys prior gives more weight to posterior in this circumstance because it's less likely that these datasets would have arisen by change. The Jeffreys prior here simply tries to align with the likelihood as much as possible.

Important remark 39. Jeffreys prior does not mean that a prior in one frame of reference "looks" the same in any other frame of reference. It means that if two individuals use different parameterisations for a given distribution, but both use Jeffreys priors, then we can get from the posterior of one individual to the other's, by applying the Jacobian transformation. So the invariance property of Jeffreys priors is really about Jeffreys' posterior distribution

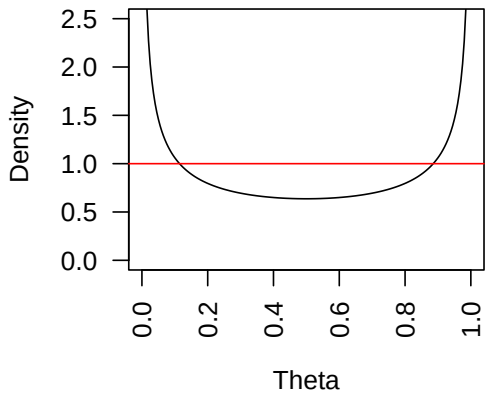


Figure 5.3: Uniform (red) and Jeffreys prior (black) for bernoulli probability parameter

Example 5.2.6 (Normal $\mathrm{N}\left(\theta, \sigma^2\right)$ with known variance)**.** In order to compute the prior for the mean, considering the likelihood

$$L(\theta; x) \propto \frac{1}{\sqrt{\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\theta)^2\right]$$

its logarithm is

$$l(\theta; x) \propto \text{constant} - \frac{1}{2\sigma^2}(x-\theta)^2$$

the first derivative is

$$\frac{\partial l(\theta; x)}{\partial \theta} = \frac{(x-\theta)}{\sigma^2}$$

the second derivative

$$\frac{\partial^2 l(\theta; x)}{\partial^2 \theta} = -\frac{1}{\sigma^2}$$

and taking the expected value with negative sign, one obtains

$$I(\theta) = -\,\mathbb{E}\left[\frac{\partial^2 l(\theta|x)}{\partial \theta^2}\right] = -\,\mathbb{E}\left[-\frac{1}{\sigma^2}\right] = \frac{1}{\sigma^2}$$

So in this case the Jeffrey prior on the mean is

$$p(\theta) \propto \frac{1}{\sqrt{\sigma^2}} = \text{constant}$$

the non informative prior that has already been proposed.

Example 5.2.7. Normal $\mathrm{N}(\mu, \theta)$ with known mean In order to compute the prior for the variance, the likelihood is the same as before:

$$L(\theta; x) \propto \frac{1}{\sqrt{\theta}} \exp\left[-\frac{1}{2}(x-\mu)^2 \frac{1}{\theta}\right]$$

The loglikelihood is:

$$l(\theta; x) \propto -\frac{1}{2}\log\theta - \frac{1}{2\theta}(x-\mu)^2.$$

Its first derivative (score) is:

$$\frac{\partial l(\theta; x)}{\partial \theta} = -\frac{1}{2\theta} - \frac{-2(x-\mu)^2}{4\theta^2} = -\frac{1}{2\theta} + \frac{(x-\mu)^2}{2\theta^2}$$

The second derivative:

$$\frac{\partial^2 l(\theta; x)}{\partial^2 \theta} = \frac{2}{4\theta^2} + \frac{0 - (x-\mu)^2 4\theta}{4\theta^4} = \frac{1}{2\theta^2} - \frac{(x-\mu)^2}{\theta^3}$$

and, taking the expected value with negative sign, one obtains:

$$I(\theta) = -E\left[\frac{\partial^2 l(\theta|x)}{\partial \theta^2}\right] = -\left[\frac{1}{2\theta}^2 - \frac{E[(x-\mu)^2]}{\theta^3}\right] = -\left[\frac{1}{2\theta}^2 - \frac{V(x|\mu)}{\theta^3}\right] = -\left[\frac{1}{2\theta^2} - \frac{\theta}{\theta^3}\right]$$
$$= \frac{1}{2\theta^2}$$

So the Jeffreys' prior on the variance θ is then

$$p(\theta) \propto \sqrt{\frac{1}{2\theta^2}} \propto \frac{1}{\theta}$$

the non informative prior that has already been proposed.

Example 5.2.8. Binomial $\mathrm{Bin}(n, \theta)$ The prior for the success parameter for the success in the single trial is obtained starting from the likelihood

$$L(\theta; x) = \binom{n}{x} \theta^x (1-\theta)^{n-x}$$

log-likelihood is

$$l(\theta; x) = c + x\log\theta + (n-x)\log(1-\theta)$$

its first derivative with respect to the parameter (score) is

$$\frac{\partial l(\theta; x)}{\partial \theta} = \frac{x}{\theta} - \frac{n-x}{(1-\theta)}$$

and the second derivative, again with respect to the parameter

$$\frac{\partial^2 l(\theta|x)}{\partial^2 \theta} = \frac{x}{-\theta^2} - \frac{n-x}{(1-\theta)^2}$$

Computing the expected value with negative sign one obtains

$$I(\theta) = -E\left[\frac{\partial^2 l(\theta|x)}{\partial\theta^2}\right] = \frac{n\theta}{\theta^2} + \frac{n - n\theta}{(1-\theta)^2} = \frac{n}{\theta(1-\theta)}$$

So the Jeffreys' prior for the success in the single trial parameter is found as

$$p(\theta) \propto \sqrt{n\theta^{-1}(1-\theta)^{-1}} \propto \theta^{-\frac{1}{2}}(1-\theta)^{-\frac{1}{2}}$$

The core of a $Beta\left(\frac{1}{2}, \frac{1}{2}\right)$, known as arcsin distribution (one of the reference priors), is recognized in such expression.

Example 5.2.9. Poisson $Po(\theta)$ The prior for the rate parameter starts from the likelihood

$$L(\theta; x) = \frac{\theta^x e^{-\theta}}{x!}$$

Loglikelihood is:

$$l(\theta; x) = x \log\theta - \theta - constant$$

First derivative (score) is:

$$\frac{\partial l(\theta; x)}{\partial\theta} = \frac{x}{\theta} - 1$$

Second derivative:

$$\frac{\partial^2 l(\theta; x)}{\partial^2\pi} = -\frac{x}{\theta^2}.$$

Computing the expected value with opposite sign one obtains

$$I(\theta) = -E\left[\frac{\partial^2 l(\theta; x)}{\partial\theta^2}\right] = -E\left[-\frac{x}{\theta^2}\right] = \frac{1}{\theta}$$

Thus the Jeffreys' prior for the rate parameter is then

$$p(\theta) \propto \sqrt{\theta^{-1}}$$

5.2.2.3 Extension of Jeffreys prior to many parameters

When $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, the Fisher Information for a single observation is a $k \times k$ matrix with generic element

$$I(\boldsymbol{\theta}, x)_{ij} = -\mathbb{E}\left[\frac{\partial^2 l(\boldsymbol{\theta}, x)}{\partial\theta_i \partial\theta_j}\right]$$

Introducing a vector of transformed parameters $\boldsymbol{g} = (g_1, \ldots, g_k)$, one obtains a matrix of derivatives not depending on the data

$$\boldsymbol{J} = [J]_{ij} = \left[\frac{\partial\theta_i}{\partial g_j}\right]_{ij}$$

so that

$$I(\boldsymbol{g};x) = \mathbf{J}I(\boldsymbol{\theta};x))\mathbf{J}'$$

and

$$\det I(\boldsymbol{g};x) = \{\det I(\boldsymbol{\theta};x))\}\,(\det \mathbf{J})^2$$

obtaining

$$p(\boldsymbol{\theta}) \propto \sqrt{\det I(\boldsymbol{\theta}|x)}$$

an invariant prior in the case of many parameters (we can ignore det$\mathbf{J}$ since it does not depend on the data). So, if we are looking for a prior for $\mathbf{g}$, we can use, as before, the prior for $\boldsymbol{\theta}$.

Example 5.2.10. Normal $\boldsymbol{\theta} = (\theta, \phi)$ If

$$L(\theta, \phi; x) \propto \frac{1}{\phi}^{\frac{1}{2}} \exp\left[-\frac{1}{2}(x-\theta)^2 \frac{1}{\phi}\right]$$

whose logarithm is

$$l(\theta, \phi; x) = -\frac{1}{2}\log\phi - \frac{1}{2\phi}(x-\theta)^2$$

the first derivatives are

$$\frac{\partial l(\theta,\phi;x)}{\partial\theta} = \frac{(x-\theta)}{\phi}$$
$$\frac{\partial l(\theta,\phi;x)}{\partial\phi} = -\frac{1}{2\phi} + \frac{(x-\theta)^2}{2\phi^2}$$

The second derivatives are

$$\frac{\partial^2 l(\theta,\phi;x)}{\partial^2\theta} = -\frac{1}{\phi}$$
$$\frac{\partial^2 l(\theta,\phi;x)}{\partial^2\phi} = \frac{1}{2\phi^2} - \frac{(x-\theta)^2}{2\phi^3}.$$

and the mixed partial derivative are

$$\frac{\partial l(\theta,\phi;x)}{\partial\theta\partial\phi} = \frac{\partial}{\partial\theta}\left[-\frac{1}{2\phi} + \frac{(x-\theta)^2}{2\phi^2}\right] = -\frac{x-\theta}{\phi^2}$$
$$\frac{\partial l(\theta,\phi;x)}{\partial\phi\partial\theta} = \frac{\partial}{\partial\phi}\left[\frac{(x-\theta)}{\phi}\right] = -\frac{x-\theta}{\phi^2}.$$

We arrange the derivatives in a matrix and take the expectation with the minus sign:

$$I(\theta,\phi;x) = -E\begin{bmatrix} -\frac{1}{\phi} & -\frac{x-\theta}{\phi^2} \\ -\frac{x-\theta}{\phi^2} & \frac{1}{2\phi^2} - \frac{(x-\theta)^2}{2\phi^3} \end{bmatrix} = \begin{bmatrix} \frac{1}{\phi} & 0 \\ 0 & -\frac{1}{2\phi^2} + \frac{\phi}{\phi^3} \end{bmatrix} = \begin{bmatrix} \frac{1}{\phi} & 0 \\ 0 & \frac{1}{2\phi^2} \end{bmatrix}$$

so that

$$\det I(\theta, \phi; x) = \frac{1}{2\phi^3}$$

the prior is thus

$$p(\theta, \phi) \propto \phi^{-\frac{3}{2}}$$

in this case the prior of the variance is different than in the one-parameter case where the improper prior $p(\theta, \phi) \propto \phi^{-1}$

5.2.2.4 Limitation/drawbacks of Jeffreys prior

Important remark 40 (Limitations of Jeffreys priors (Lambert)). Two issues that limit its practical use:

- It is not simple to extend Jeffreys prior to multi-parameter models.
- Jeffreys priors are improper for many models.

Finally it is important not to obsess about the use of uninformative priors. As we shall see, weakly informative priors and hierarchical models can help guard against some of the issues raised by critics of Bayesian inference. Furthermore, often the difference in posteriors resulting from Jeffreys priors versus vaguely informative priors is minimal

Important remark 41. Finally Jeffreys' prior does not respect the likelihood principle: it depends on the way in which the experiment is organized.

Example 5.2.11. An example showing how the prior for a proportion π in a population is different in the case the experiment is run referring to a binomial or a negative binomial[3] .

[3]Some care is to be deserved to the construction of the negative binomial distribution. The $\text{Bin}(n, \pi)$ is the generalization of n Bernoulli trials $\text{Be}(\pi)$, with support $x = \{0, \dots, n\}$. As is well known, x describes the number of successes in n trials.
Suppose $Z \sim \text{NB}(n, \pi)$, number of independent trials that have to be done in order to obtain n successes, with support $z = \{n, n+1, \dots\}$. Note that n needs to be at least 1. In this case n is no longer the number of trials. The data generation process is described by the following discrete distribution

$$p(z|n, \pi) = \binom{y-1}{n-1} \pi^y (1-\pi)^{y-n},$$

known as negative binomial, with

$$E(Z) = \frac{n}{\pi} \quad \text{and} \quad V(Z) = \frac{n(1-\pi)}{\pi^2}.$$

But it is not the parametrization of the former subsection, where with $Y \sim \text{NB}(n, \pi)$ we denoted the number of failures before observing n successes. Starting from the variable Z above, Y is the traslated variable $Y = Z - n$, with support $y = \{0, 1, \dots\}$, as already stated

$$p(y|n, \pi) = \binom{n+y-1}{y} \pi^n (1-\pi)^y$$

with a different expected value and the same variance:

$$E(Y) = \frac{n(1-\pi)}{\pi} \quad \text{and} \quad V(Y) = \frac{n(1-\pi)}{\pi^2}.$$

In this case Jeffreys prior on the very same parameter π (single success probability) is different in the two cases (even if the probability of success does not change):

- in case of Binomial Likelihood, data generating process models the number of successes in n trials, with possible values running from 0 to n (support of $x = \{0, \dots, n\}$), for a given probability of success θ in each trial

$$p(x|\theta) = \binom{n}{x} \theta^x (1-\theta)^{n-x}.$$

  In this case Jeffreys Prior is Beta $\left(\frac{1}{2}, \frac{1}{2}\right)$, as we've seen

- if the variable, say x, is the number of experiments in iid Bernoullian trials in order to get m successes, and the *probability of success is the same* θ as before, the data generating process is the negative binomial distribution:

$$p(x|\theta) = \binom{m+x-1}{x} \theta^m (1-\theta)^x,$$

  where the former n of the binomial can be retrieved as $n = m+x$ (support of $x = \{0, \dots, ...\}$). To obtain Jeffreys' prior, the loglikelihood is:

$$[p(x;\theta)] \propto m \log(\theta) + x \log(1-\theta)$$

  Its first derivative:

$$\frac{\partial [p(x;\theta)]}{\partial \theta} = \frac{m}{\theta} - \frac{x}{1-\theta}$$

  The second derivative:

$$\frac{\partial^2 [p(x;\theta)]}{\partial^2 \theta} = -\frac{m}{\theta^2} - \frac{x}{(1-\theta)^2}$$

  Then

$$\begin{aligned} I(x;\theta) &= -\mathbb{E}\left[\frac{\partial^2 [p(x;\theta)]}{\partial^2 \theta}\right] = \frac{m}{\theta^2} + \frac{E(x|\theta)}{(1-\theta^2)} \\ &= \frac{m}{\theta^2} + \frac{m}{\theta(1-\theta)} = \frac{m}{\theta^2(1-\theta)} \end{aligned}$$

  and thus the Jeffreys prior is

$$p(\theta) \propto \sqrt{I(x;\theta)} = \left[\theta^2(1-\theta)\right]^{-\frac{1}{2}} = \theta^{-1}(1-\theta)^{-\frac{1}{2}}$$

  which is the kernel of a Beta $\left(0, \frac{1}{2}\right)$. So in case of Negative Binomial (Pascal) Likelihood, here Jeffreys Prior is Beta $\left(0, \frac{1}{2}\right)$

5.2.3 Reference priors

TODO: integra con lambert pag 249

Remark 80. Two interlocutors may not agree on the prior or do not have any *a priori* strong convictions about possible values of the parameter. In these cases is therefore reasonable that they use a reference prior *dominated by the likelihood*, i.e. by the result of the experiment.
We call this prior a *neutral reference* prior dominated by the likelihood.

DO: questo andrebbe o altrove? tipo una generale

Remark 81. Posterior is tipically dominated by the likelihood when

1. the variance of the prior tends to infinity
2. the sample has a very big size.

5.2.3.1 Reference prior for the parameter of a binomial

The ML estimator for the parameter π of the binomial is r/n (r experiment successes out of n trials).
This ratio has different roles in the posterior distribution according to the target that is defined: different priors $Beta(a, b)$ (with a prior virtual successes and b virtual failures) may be required, that induce different posteriors $Beta(a+r, b+n-r)$.
Important syntheses of such posteriors are the expected value and the mode

$$E(\pi|a, b, r, n) = \frac{a+r}{a+b+n}$$
$$Mode(\pi|a, b, r, n) = \frac{a+r-1}{a+b+n-2}.$$

Some different prior for the parameter of a binomial are illustrated in the following points:

1. If a $Beta(1, 1)$ is used, the posterior expectation and mode are

$$E(\pi|1, 1, r, n) = \frac{r+1}{(r+1)+(n-r+1)} = \frac{r+1}{n+2}$$
$$Mode(\pi|1, 1, r, n) = \frac{1+r-1}{1+1+n-2} = \frac{r}{n}$$

   Thus the $Beta(1, 1)$ prior is suitable when *one likes that the ML estimator is the mode (i.e. the maximum) of the posterior distribution.* The underlying idea is to employ the result of the experiment as the maximum of the target function.

2. the $Beta(0, 0)$ (so called Haldane prior, equivalent to the uniform prior for the log-odds $\Lambda = \log[\pi(1-\pi)]$) is suitable when *one wishes that the average of the posterior distribution has the same expression of the ML estimator*, since

$$E(\pi|0, 0, r, n) = \frac{0+r}{0+0+n} = \frac{r}{n}.$$

   The underlying idea is to employ the result of the experiment to stress the unbiasedness of the result.

3. the $Beta(\frac{1}{2}, \frac{1}{2})$ prior,

$$p(\pi) \propto \pi^{-\frac{1}{2}}(1-\pi)^{1-\frac{1}{2}}$$

   also known as the arcsin probability distribution, should be the reference prior to use according to the Jeffreys rule

4. Zellner's prior

$$p(\pi) \propto \pi^{\pi}(1-\pi)^{1-\pi}$$

does not belong to the family of the conjugate priors.

Remark 82. Features shared by the 4 distributions:

- they do not show differences if the reasoning is minimally supported by data;
- they highlight that the formalisation of the concept "not knowing anything" is not elementary.

5.3 Weakly informative priors

Considered that

- as argued by Lambert all analyses are inherently subjective, and so any attempts to make Bayesian inference, or any other analysis, objective are inherently flawed.
  He believe that the fear of subjectivity is inherently misplaced because all statistical analyses require us to choose models from a range of reasonable modelling option
- when we move to computational methods the use of 'uninformative' priors can cause issues for an analysis:
  1. posterior densities that we estimate may have non-zero support in unrealistic areas of parameter space
  2. computational methods may take considerably longer to converge than if we used more reasonable priors

Gelman calls *weakly informative* those priors that sit between those that are maximally uninformative and those that are strongly informative by incorporating:

- more uncertainty than priors constructed using all of our pre-analysis knowledge
- less variability than entirely ignorant priors

Most people automatically use a variety of weakly informative priors and *its usage is encouraged*.

How to set weakly informative priors in practice?

- Allow considerable uncertainty in priors, such that the range of feasible parameter values is wider than we would expect, but not unreasonable.
  Eg for parameters in a regression with standardised variables, use $\beta \sim \mathrm{N}(0, 10)$ rather than $\beta \sim \mathrm{N}(0, 1000)$;
- Use smooth distributions for parameters, rather than discontinuous ones.
  Eg use a long-tailed Cauchy distribution rather than a uniform distribution over some finite scale

- Avoid distributions that give too much weight to areas of unrealistic parameter values
  Eg instead of using an inv-gamma use a half-Cauchy for a prior on a variance parameter.

These rule become much more clear as we apply/practice Bayesian analysis.

5.4 Other stuff

5.4.1 Inference on the variance of a normal distribution when the mean is known

TODO: spostare da qui?

The section illustrates the use of the conjugate prior for a likelihood that belongs to the exponential family.

$$\{X_1, X_2, \dots, X_n \mid \theta, \phi\} \sim i.i.d.\ N(\theta, \phi), \quad \theta \text{ known}$$

Likelihood

$$\begin{aligned} p(x_1, \dots, x_n \mid \phi) &= \prod_{i=1}^{n} p(x_i \mid \phi) \\ &\propto \phi^{-\frac{n}{2}} \exp\left\{-\frac{S}{2\phi}\right\} \qquad \text{with} \quad S = \sum_{i=1}^{n}(x_i - \theta)^2 \end{aligned}$$

5.4.1.1 Conjugate prior

Conjugate prior distribution

$$\phi \sim IG\left(\frac{\nu_0}{2}, \frac{S_0}{2}\right) = S_0 \chi^{-2}_{\nu_0}$$

$$p(\phi) \propto \phi^{-\frac{\nu_0}{2}-1} \exp\left\{-\frac{S_0}{2\phi}\right\}$$

Posterior distribution

$$\begin{aligned} p(\phi \mid x_1, \dots, x_n) &\propto p(\phi)\, p(x_1, \dots, x_n \mid \phi) \\ &\propto \phi^{-\frac{\nu_0}{2}-1} \exp\left\{-\frac{S_0}{2\phi}\right\} \phi^{-\frac{n}{2}} \exp\left\{-\frac{S}{2\phi}\right\} \\ &\propto \phi^{-\frac{\nu_0+n}{2}-1} \exp\left\{-\frac{1}{2\phi}(S_0 + S)\right\} \end{aligned}$$

the last expression contains the parameters of both the prior and the likelihood;

$$\phi \mid x_1, \ldots, x_n \sim IG\left(\frac{\nu_0 + n}{2}, \frac{S_0 + S}{2}\right) = (S_0 + S)\chi^{-2}_{\nu_0+n}$$

Note. The result is obtained with the parameterization on the variance. The transformation into precision can be done: the kernels of χ^2 and Gamma distributions are used in this case.

5.4.1.2 A non-informative prior

Non-informative prior distribution

A $IG(0,0)$ can be proposed, where $\nu_0 = 0$; $S_0 = 0$. It is an alternative stating of $p(\phi) \propto \phi^{-1}$.

Posterior distribution

Here, only the parameters of the likelihood may appear:

$$\begin{aligned} p(\phi \mid x_1, \ldots, x_n) &\propto p(\phi)\, p(x_1, \ldots, x_n \mid \phi) \\ &\propto \phi^{-1}\phi^{-\frac{n}{2}} \exp\left\{-\frac{S}{2\phi}\right\} \\ &\propto \phi^{-\frac{n}{2}-1} \exp\left\{-\frac{S}{2\phi}\right\} \end{aligned}$$

where the kernel is no longer the one of a normal, but of an Inverse-Gamma as follows

$$\phi \mid x_1, \ldots, x_n \sim IG\left(\frac{n}{2}, \frac{S}{2}\right) = S\chi^{-2}_n.$$

5.4.2 Inference on both parameters of a normal

The starting situation is

$$\{X_1, X_2, \ldots, X_n \mid \theta, \phi\} \sim i.i.d.\ N(\theta, \phi)$$

Likelihood

$$
\begin{aligned}
p(x_1, \ldots, x_n \mid \theta, \phi) &= \prod_{i=1}^{n} p(x_i \mid \theta, \phi) \\
&\propto \phi^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2\phi} \left[\sum_{i=1}^{n} (x_i - \bar{x})^2 + n(\theta - \bar{x})^2 \right] \right\} \\
&\propto \phi^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2\phi} \left[(n-1)s^2 + n(\theta - \bar{x})^2 \right] \right\}
\end{aligned}
$$

with $s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$.

Inference:

a) Joint posterior distribution: $p(\theta, \phi \mid x_1, \ldots, x_n)$;

b) Marginal posterior distribution for the mean: $p(\theta \mid x_1, \ldots, x_n)$;

c) Marginal posterior distribution for the variance: $p(\phi \mid x_1, \ldots, x_n)$;

d) Conditional posterior distribution for the mean: $p(\theta \mid \phi, x_1, \ldots, x_n)$.

5.4.2.1 A non informative prior

$$p(\theta, \phi) = p(\theta)\, p(\phi) \propto \phi^{-1}$$

a) Joint posterior distribution

$$
\begin{aligned}
p(\theta, \phi \mid x_1, \ldots, x_n) &\propto p(\theta, \phi)\, p(x_1, \ldots, x_n \mid \theta, \phi) \\
&\propto \phi^{-1} \phi^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2\phi} \left[(n-1)s^2 + n(\theta - \bar{x})^2 \right] \right\} \\
&\propto \phi^{-\frac{n}{2}-1} \exp\left\{ -\frac{1}{2\phi} \left[(n-1)s^2 + n(\theta - \bar{x})^2 \right] \right\}
\end{aligned}
$$

b) Marginal posterior distribution for the mean

$$
\begin{aligned}
p(\theta \mid x_1, \ldots, x_n) &= \int_0^{+\infty} p(\theta, \phi \mid x_1, \ldots, x_n) d\phi \\
&\propto \int_0^{+\infty} \phi^{-\frac{n}{2}-1} \exp\left\{ -\frac{1}{2\phi} \left[(n-1)s^2 + n(\theta - \bar{x})^2 \right] \right\} d\phi \\
&\propto \left[1 + \frac{n(\theta - \bar{x})^2}{(n-1)s^2} \right]^{-\frac{n}{2}}
\end{aligned}
$$

the kernel of a t distribution is acknowledged

$$\theta \mid x_1, \ldots, x_n \sim t_{n-1}(\bar{x}, s^2/n)$$

since $\bar{x}$ and s^2/n are parameters of the distribution, the same kernel holds for the standardised t

$$\frac{\theta - \bar{x}}{\sqrt{s^2/n}} \mid x_1, \ldots, x_n \sim t_{n-1}$$

c) Marginal posterior distribution for the variance

$$\begin{aligned} p(\phi \mid x_1, \ldots, x_n) &= \int_{-\infty}^{+\infty} p(\theta, \phi \mid x_1, \ldots, x_n) d\theta \\ &\propto \int_{-\infty}^{+\infty} \phi^{-\frac{n}{2}-1} \exp\left\{-\frac{1}{2\phi}\left[(n-1)s^2 + n(\theta-\bar{x})^2\right]\right\} d\theta \\ &\propto \phi^{-\frac{n-1}{2}-1} \exp\left\{-\frac{(n-1)s^2}{2\phi}\right\} \end{aligned}$$

$$\phi \mid x_1, \ldots, x_n \sim IG\left(\frac{n-1}{2}, \frac{(n-1)s^2}{2}\right) = (n-1)s^2\chi^{-2}_{n-1}$$

d) Conditional posterior distribution for the mean

$$\begin{aligned} p(\theta \mid \phi, x_1, \ldots, x_n) &= \frac{p(\theta, \phi \mid x_1, \ldots, x_n)}{p(\phi \mid x_1, \ldots, x_n)} \\ &\propto \frac{\phi^{-\frac{n}{2}-1} \exp\left\{-\frac{1}{2\phi}\left[(n-1)s^2 + n(\theta-\bar{x})^2\right]\right\}}{\phi^{-\frac{n-1}{2}-1} \exp\left\{-\frac{(n-1)s^2}{2\phi}\right\}} \\ &\propto \phi^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\phi} n(\theta-\bar{x})^2\right\} \end{aligned}$$

$$\theta \mid \phi, x_1, \ldots, x_n \sim N\left(\bar{x}, \frac{\phi}{n}\right)$$

The result can be compared with the case of known variance, but here the posterior distribution remains conditional on ϕ.

5.4.2.2 Conjugate priors

- $\phi \sim IG\left(\nu_0/2, S_0/2\right) = S_0\chi^{-2}_{\nu_0}$
- $\theta \mid \phi \sim N\left(\theta_0, \phi/n_0\right)$
- $(\theta, \phi) \sim N\text{-}Inv\text{-}\chi^2(\theta_0, S_0/n_0, \nu_0, S_0)$

$$\begin{aligned} p(\theta, \phi) &= p(\theta \mid \phi)\, p(\phi) \\ &\propto \phi^{-\frac{1}{2}} \exp\left\{-\frac{n_0}{2\phi}(\theta-\theta_0)^2\right\} \phi^{-\frac{\nu_0}{2}-1} \exp\left\{-\frac{S_0}{2\phi}\right\} \\ &\propto \phi^{-\left(\frac{\nu_0+1}{2}\right)-1} \exp\left\{-\frac{1}{2\phi} Q_0(\theta)\right\} \end{aligned}$$

with $Q_0(\theta) = n_0\theta^2 - 2(n_0\theta_0)\theta + (n_0\theta_0^2 + S_0)$.

a) Joint posterior distribution

$$\begin{aligned} p(\theta, \phi \mid x_1, \ldots, x_n) &\propto p(\theta, \phi)\, p(x_1, \ldots, x_n \mid \theta, \phi) \\ &\propto \phi^{-\left(\frac{\nu_0+1}{2}\right)-1} \exp\left\{-\frac{1}{2\phi} Q_0(\theta)\right\} \\ &\quad \times \phi^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\phi}\left[(n-1)s^2 + n(\theta - \bar{x})^2\right]\right\} \\ &\propto \phi^{-\left(\frac{\nu_1+1}{2}\right)-1} \exp\left\{-\frac{1}{2\phi} Q_1(\theta)\right\} \end{aligned}$$

with $\nu_1 = \nu_0 + n$ and $Q_1(\theta) = n_1\theta^2 - 2(n_1\theta_1)\theta + (n_1\theta_1^2 + S_1)$.

$$\theta, \phi \mid x_1, \ldots, x_n \sim N\text{-}Inv\text{-}\chi^2(\theta_1, S_1/n_1, \nu_1, S_1)$$

- $\nu_1 = \nu_0 + n$
- $n_1 = n_0 + n$
- $\theta_1 = \frac{n_0\theta_0 + n\bar{x}}{n1} = \frac{n_0}{n_0+n}\theta_0 + \frac{n}{n_0+n}\bar{x}$
- $S_1 = S_0 + S + n_0\theta_0^2 + n\bar{x}^2 - n_1\theta_1^2$

The two central bullets are useful for interpreting d), the conditional posterior distribution for the mean.

b) Marginal posterior distribution for the mean

$$\theta \mid x_1, \ldots, x_n \sim t_{\nu_1}(\theta_1, S_1/n_1\nu_1)$$

c) Marginal posterior distribution for the variance

$$\phi \mid x_1, \ldots, x_n \sim IG(\nu_1/2, S_1/2) = S_1\chi_{\nu_1}^{-2}$$

d) Conditional posterior distribution for the mean

$$\theta \mid \phi, x_1, \ldots, x_n \sim N(\theta_1, \phi/n_1)$$

Note that the above definition of θ_1 has been useful and that the variance of the posterior depends on the variance of the data generating distribution.

Remark 83. Nowadays the use of *weakly informative priors* is recommended by many scholars; these allow to avoid mathematical and computations difficulties of many methodologies we does describe in what follows.

Chapter 6

Linear model and simulation methods

Reference textbook

Regression model in Lambert chapt. 18 Linear regression models[1]

6.1 The linear regression model

Also the classical multiple linear regression model can be faced within the Bayes framework. Here the case of independent and omoschedastic error is presented.

Given a vector of n observations $\mathbf{y}$ and the $n \times k$ matrix $\mathbf{X}$, which contains the information concerning the k covariates, the linear regression model can be defined as

$$y_i = \boldsymbol{\beta}' \mathbf{x}_i + \varepsilon_i \quad i = 1, \ldots, n$$

where

$$E\left[y_i | \mathbf{x}_i . \boldsymbol{\beta}\right] = \beta_1 x_{i1} + \ldots + \beta_k x_{ik} = \boldsymbol{\beta}' \mathbf{x}_i$$

i.e. the conditional expectation has a form that is linear in a set of parameters, and

$$\varepsilon_i \sim \text{iid} N(0, \phi).$$

What above states the normality assumption for the observations. Normality is related to the sampling variability around the mean. In other words

$$\{\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}, \phi\} \sim N_n(\mathbf{X}\boldsymbol{\beta}, \phi \mathbf{I}_n).$$

Inference concerns the k-dimensional vector of the regression coefficients and the error term ϕ.

[1] In hoff the linear model is deepened in Chapter 9 (Linear Regression). Linear and generalized linear mixed effects models are developed as hierarchical linear models with covariates in Chapter 11.

From the normality assumption on the vector of observations the following likelihood function can be computed

$$\begin{aligned} p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta},\phi) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\boldsymbol{\beta},\phi) \\ &= (2\pi\phi)^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\phi}\sum_{i=1}^{n}(y_i-\boldsymbol{\beta}'\mathbf{x}_i)^2\right\} \\ &= (2\pi\phi)^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\phi}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right\} \\ &= (2\pi\phi)^{-\frac{n}{2}} \exp\left\{-\frac{1}{2\phi}\left[(n-k)s^2+(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})'\mathbf{X}'\mathbf{X}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right]\right\} \end{aligned}$$

where

$$(n-k)s^2 = (\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}})'(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}),$$

s^2 is the sampling "unbiased" variance estimator and

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \quad \text{(OLS and ML estimator, as a classical result).}$$

Still in the classical context, an unbiased estimate of ϕ can be obtained from

$$\mathrm{SSR}(\hat{\boldsymbol{\beta}})/(n-k) = \sum_{i=1}^{n}(y_i-\hat{\boldsymbol{\beta}}'\boldsymbol{x}_i)/(n-k).$$

a) Considering a non informative prior for the parameters, seen as independent

$$p(\boldsymbol{\beta},\phi) \propto \frac{1}{\phi}$$

the posterior joint distribution (where the contribution of the prior may be noticed) is

$$p(\boldsymbol{\beta},\phi|\mathbf{y},\mathbf{X}) \propto \phi^{-\frac{n}{2}-1}\exp\left\{-\frac{1}{2\phi}\left[(n-k)s^2+(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})'\mathbf{X}'\mathbf{X}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right]\right\}$$

while the most important conditional and marginals are

- conditional on the variance, the distribution of the coefficients is $\{\boldsymbol{\beta}|\phi,\mathbf{y},\mathbf{X}\} \sim N_k\left(\hat{\boldsymbol{\beta}},\phi(\mathbf{X}'\mathbf{X})^{-1}\right)$;
- marginal for the variance $\{\phi|\mathbf{y},\mathbf{X}\} \sim IG\left((n-k)/2,(n-k)s^2/2\right)$;
- marginal for the coefficients $\{\boldsymbol{\beta}|\mathbf{y},\mathbf{X}\} \sim t_k\left(n-k;\hat{\boldsymbol{\beta}},s^2(\mathbf{X}'\mathbf{X})^{-1}\right)$;

There is to remark that the results are analogous to those obtained following the ML approach.

b) Instead of postulating diffuse priors, it is also possible to specify the conjugate prior starting from:

$$\begin{aligned} \boldsymbol{\beta}|\phi &\sim N_k(\beta_0\mathbf{1}, \phi\mathbf{V}_0); \\ \phi &\sim IG(\nu_0/2, S_0/2). \end{aligned}$$

where the first distribution is a multivariate normal, and $\mathbf{V}_0$ is a matrix of fixed values (more complicated than a diagonal matrix, but that can however be inverted), obtaining:

$$p(\boldsymbol{\beta}, \phi) \propto \phi^{-\frac{k}{2}} \exp\left\{-\frac{1}{2\phi}(\boldsymbol{\beta} - \beta_0\mathbf{1})'\mathbf{V_0}^{-1}(\boldsymbol{\beta} - \beta_0\mathbf{1}))\right\} \phi^{-\frac{\nu_0}{2}-1} \exp\left\{-\frac{S_0}{2\phi}\right\}.$$

From this prior, the joint posterior is

$$p(\boldsymbol{\beta}, \phi|\mathbf{y}, \mathbf{X}) \propto \phi^{-\frac{k+\nu_0+n}{2}-1} \exp\left\{-\frac{1}{2\phi}\left[S_0 + (n-k)s^2 + Q(\boldsymbol{\beta})\right]\right\}$$

where

$$\begin{aligned} Q(\boldsymbol{\beta}) &= (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})'\mathbf{X}'\mathbf{X}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \beta_0\mathbf{1})'\mathbf{V_0}^{-1}(\boldsymbol{\beta} - \beta_0\mathbf{1}) \\ &= (\boldsymbol{\beta} - \boldsymbol{\beta}_*)'\mathbf{V}_*{}^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_*) + (\hat{\boldsymbol{\beta}} - \beta_0\mathbf{1})'(\mathbf{X}'\mathbf{X})\mathbf{V}_*\mathbf{V}_0{}^{-1}(\hat{\boldsymbol{\beta}} - \beta_0\mathbf{1}) \end{aligned}$$

with

$$\mathbf{V}_* = (\mathbf{X}'\mathbf{X} + \mathbf{V}_0{}^{-1})^{-1}$$

and

$$\boldsymbol{\beta}_* = \mathbf{V}_*(\mathbf{X}'\mathbf{y} + \mathbf{V}_0^{-1}\beta_0).$$

After some computations, one obtains

$$p(\boldsymbol{\beta}, \phi|\mathbf{y}, \mathbf{X}) \propto \phi^{-\frac{k}{2}} \exp\left\{-\frac{1}{2\phi}\left[(\boldsymbol{\beta} - \boldsymbol{\beta}_*)'\mathbf{V}_*{}^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_*)\right]\right\} \phi^{-\frac{\nu_0+n}{2}-1} \exp\left\{-\frac{S_*}{2\phi}\right\}$$

where

$$\{\boldsymbol{\beta}, \phi|\mathbf{y}, \mathbf{X}\} \sim \text{N-IG}(\boldsymbol{\beta}_*, \mathbf{V}_*, \nu_*/2, S_*/2)$$

and

$$\begin{aligned} \nu_* &= \nu_0 + n \\ S_* &= S_0 + (n+k)s^2 + (\hat{\boldsymbol{\beta}} - \beta_0\mathbf{1}))'(\mathbf{X}'\mathbf{X})\mathbf{V}_*\mathbf{V_0}{}^{-1}(\hat{\boldsymbol{\beta}} - \beta_0\mathbf{1}) \end{aligned}$$

Chapter 7

Hierarchical Bayesian Models

The topics of this block are contained in Chapters 17, 18 and 19 of Lambert B. (2018) A Student's Guide to Bayesian Statistics, Sage

Chapt. 17 Hierarchical models

Chapt. 18 Linear regression models

Chapt. 19 Generalised linear models and other animals

7.1 Introduction to hierarchical models

The motivation comes from the need of performing analyses on similar data sets, for inference on parameters that denote subgroups.

Immediate examples are (several others can be added):

- Longitudinal analysis
- Small area estimation (SAE)

Conceptually, constructing a Hierarchical Bayesian Model (HBM) is the opposite of using a dummy variable for each subgroup.

The idea is not new, it starts from the request of making inference on several parameters, guessing that such parameters are similar. In the Bayesian framework thia corresponds to proposing:
a hierarchical model,
a multilevel model (discussion),
a random effects model,
a random coefficients model.

Added value of this conceptualization: simultaneous estimate of either a parameter for each group and the variation among groups.

A prior is to be defined common to subgroups parameters. The proposal of a common prior exploits the conceptualization of exchangeability, where the concept of similarity is formalized by the exchangeability assumption of Bayesian statistics.

Exchangeability

A sequence of random variables $(\theta_1, \ldots, \theta_k)$ is said to be *exchangeable* if the joint probability distribution of the sequence is invariant under any finite permutation of its indices. That is, $(\theta_1, \ldots, \theta_k)$ are exchangeable if, for any permutation π,

$$p(\theta_1, \ldots, \theta_k) = p(\theta_{\pi(1)}, \ldots, \theta_{\pi(k)}).$$

Exchangeability is the situation when intermediate parameters $\theta_1, ..., \theta_k$ are used for obtaining conditional independence.

de Finetti's Representation Theorem

If $(\theta_1, \ldots, \theta_k)$ are exchangeable, they can be interpreted as being conditionally independent given a coditioning random variable (denoted by ϕ) that characterizes their common distribution, that is:

$$\theta_i \mid \phi \overset{\text{iid}}{\sim} p(\theta \mid \phi).$$

Saying that $\theta_1, ..., \theta_k$ are exchangeable is therefore equivalent to saying that they are randomly extracted by a common distribution (parameterized by ϕ). THe symbol ϕ denotes a generic parameter.
For the special case of an exchangeable sequence of Bernoulli random variables, de Finetti's theorem states that such a sequence is a "mixture" of sequences of iid Bernoulli random variables.

Exchangeable sequences are foundational for HB models, where group-level parameters introduce dependency among observations.

Main difference of HB models in comparison with simple Bayesian models:
inference is not only on global parameters or **hyperparameters** (like the general mean) but insists on intermediate level parameters conditional on data.

7.1.1 Consequences of stating hierarchical models

There is a bias towards an average: a SHRINKAGE occurs.
Group effects are not requested: the number of intermediate parameters is not necessarily lower than the number of observations, latent entities may occur, even for each observation: unit-specific parameters can be modeled.

Before using computer codes for finding the distribution of any parameter, empirical Bayes methods have been popular: they use (plug-in) ML estimates for

some parameters, in particular the hyperparameters common to the whole population.

The main difficulties of classical methods, overcome by HBMs, are:

- Individual estimates of θ_i are unreliable without a shrinkage
- Such methods ignore that θ_i are related

7.2 Constructing hierarchical models

This occurs via conditional independence models (partitioning a model in simpler components).

Directed Acyclic Models (DAG) models help in hierarchical models architecture: they mimic conditional independence situations.

DAGs are used to visually and mathematically represent the structure of dependencies among variables in a probabilistic model. Each node in the graph corresponds to a random variable, and directed edges indicate conditional dependencies.
In BHMs, parameters are organized into layers, with higher-level parameters (hyperparameters) influencing lower-level parameters. This hierarchical structure can be expressed naturally using a DAG, showing the flow of dependencies from hyperparameters to data. Therefore DAGs can be used to help clarify how priors, hyperpriors, and observed data interact in the hierarchical model.

7.2.1 The definition of different levels

A hierarchical model expresses a marginal model via a sequence of conditional models, where latent entities appear.

A data set $y_1, \ldots, y_n$ is available and a HBM is built.

HBMa are organized in a hierarchy of levels:

- Level 1: a data generation process is proposed, i.e. a data distribution (with parameters)
  $y_i \sim$ a function of a vector of individual $\theta = (\theta_1, \ldots, \theta_n)$ or group $\theta = (\theta_1, \ldots, \theta_k)$ parameters and parameters proper of the whole population.
- Level 2: intermediate level parameters. A prior on the process generating parameters $\theta_i n$ is modeled. Conditional independence assumptions can be formulated. Level 2 expresses the important characteristics of the model; it may have sub-levels that may be specified with more than one equation.
- Level 3: parameters common to the whole population: often called hyperparameters.

Together, levels 2 and 3 model what is unobservable.

Nesting is the way of dealing with replications: it offers a very easy link to multilevel models. See the following Example 2.

The second level parameters do not necessarily have a physical meaning. According to the most genuine idea of exchangeability, inferring on the second level parameters should be useful essentially for predicting future values of the variable under study.

7.2.2 The Measurement Error Model (Example 1)

Level 1 distribution: data distribution
The normal distribution is assumed. There is no nesting and the data variability is due to the measurement error:

$$y_i|\theta_i \overset{\text{iid}}{\sim} N(\theta_i, \sigma^2)$$

An individual unknown expected value is assumed in correspondence of each observation. This model has as many parameters as observations. The object of inference is $\theta_1, \ldots, \theta_n$: a set of unobservable parameters.
Parameter σ^2 is considered as common.

Level 2 distribution: distribution of local parameters

$$\theta_i \overset{\text{iid}}{\sim} N(\mu, \tau^2)$$

Such parameters are unobservable entities. The specification above summarizes very quickly the idea of a common distribution for the θ_is, generated by an underlying process. The amount τ^2 witnesses the variation around a common mean for the process.

Level 3 distribution: distribution of the hyperparameters
A further level is needed for the priors of the common parameters.

The simplest formulation for this level: assuming an improper prior for μ. At their turn, σ^2 and τ^2 are known or estimated via Empirical Bayes (EB) methods). ML estimates, which may be found via iterative methods are in this case plugged in.

Motivating the addition of the third level
If we think to $\tau^2 \to \infty$, remaining only with σ^2, we obtain the traditional Bayesian result, coincident with the classical one and retrieving the single observation as posterior expectation and mode.
On the contrary, if one thinks to a joint prior distribution

$$h(\theta, \mu) = p(\mu) \prod_{i=1}^{n} p(\theta_i|\mu)$$

from which the prior marginal can be computed

$$p(\theta) = \int h(\theta, \mu)\, d\mu = \int p(\mu) \prod_{i=1}^{n} p(\theta_i|\mu)\, d\mu$$

After assuming an improper prior for μ, the desidered posterior for the θ_is will be obtained.

Solution
Unknown parameters ("only" those of level 2 and μ for level 3) are estimated

via the Bayes theorem. Parameter μ can be one of the objects of estimation, but the new quantity of interest is the vector of local parameters, whose posteriors, for the above arguments, are $\theta_i|y \sim N(\widetilde{\theta}_i, v_i)$

where

$$\widetilde{\theta}_i = \frac{\frac{y_i}{\sigma^2} + \frac{\overline{y}}{\tau^2}}{\frac{1}{\sigma^2} + \frac{1}{\tau^2}} = \frac{\tau^2 y_i + \sigma^2 \overline{y}}{\sigma^2 \tau^2} \frac{\sigma^2 \tau^2}{\tau^2 + \sigma^2} = \frac{\tau^2 y_i + \sigma^2 \overline{y}}{\tau^2 + \sigma^2}$$

$$v_i = \frac{\sigma^2 \left(\frac{\sigma^2}{n} + \tau^2 \right)}{\sigma^2 + \tau^2} = \frac{\sigma^2}{n} \cdot \frac{\sigma^2 + n\tau^2}{\sigma^2 + \tau^2}$$

Peculiarities of hierarchical Bayesian models: borrowing information and shrinkage. The whole sample is used for computing estimates concerning each unit.

Each posterior mean is closer to the common sample mean than the corresponding observation y_i (shrinkage). When the measurement error is smaller than the error of the process, greater weight is given to the individual value. When the measurement error is greater than the error of the process, greater weight is given to the general mean (shrinkage is stronger).

7.2.3 General considerations after the example

In the case above, unit specific (or intermediate level) coefficients have an explicit scientific interest, with a direct interpretability, and a parameterization that writes them directly is useful.

The role of shrinkage.
The effective number of parameters may be lower than the number of observations due to the hierarchical structure and the role of the hyperparameters. The contribution of hyperparameters conduces to a posterior that, due to shrinkage, may be not very flexible.

The importance of full conditionals.
A joint distribution of data and all parameters could be computed. Any posterior can be derived conditionally on data. "Intermediate" posteriors can be computed conditional not only on data but also on some parameters (full conditionals). Analytical computation of full conditionals can fasten calculations that are performed via computer codes.

Conjectures on variances.
The most important challenge regards formulating appropriate conjectures on the variance. Such conjectures can be proposed as follows:

- Improper priors for variances
- Setting variances fixed in some way
- Estimating variances via Empirical Bayes

7.3 Marginalization

At the beginning of the exposition of hierarchical models, it was claimed that a hierarchical model expresses a marginal model via conditional models.
If inference regards the hyperparameters, marginalization is important since it offers a useful way of writing, since the parameters of the set of conditional models can be integrated out.

The example of the normal distribution is made.

In the normal case, starting from the two following (marginal and conditional) distributions

$$y_i|\theta_i \sim N(A\mu + a, V)$$

$$\theta_i \overset{\text{iid}}{\sim} N(\mu, \Sigma)$$

the joint marginal model is rewritten as:

$$\begin{pmatrix} y_i \\ \theta_i \end{pmatrix} \sim N\left[\begin{pmatrix} A\mu + a \\ \mu \end{pmatrix}, \begin{pmatrix} V + A\Sigma A' & A\Sigma \\ \Sigma A' & \Sigma \end{pmatrix}\right]$$

Such marginal distribution can be directly considered for inference.

The special case with

$$\begin{array}{ll} A = I & a = 0 \\ \Sigma = \tau^2 & V = 1 \end{array}$$

is a further simplified example:

$$y_i|\theta_i \sim N(\theta_i, 1)$$

$$\theta_i|\tau \overset{\text{iid}}{\sim} N(\mu, \tau^2)$$

where the marginal model is:

$$\begin{pmatrix} y_i \\ \theta_i \end{pmatrix} \sim N\left[\begin{pmatrix} \mu \\ \mu \end{pmatrix}, \begin{pmatrix} \tau^2 + 1 & \tau^2 \\ \tau^2 & \tau^2 \end{pmatrix}\right]$$

The distribution of y_i is $y_i|\mu, \tau^2, 1$, from which the distribution of the parameters given the data $\mu, \tau^2|y_i$ may be derived.

7.4 Inference about future observations

Hierarchical modeling is of help for the inference about future observations. Remember that hyperparameters characterize the whole population, while intermediate level parameters characterize specific units (or groups of elementary units).

In the (already seen) measurement model (that is also called component of variace model):

$$y_i|\theta_i \overset{\text{iid}}{\sim} N(\theta_i, \sigma^2)$$

$$\theta_i|\tau \overset{\text{iid}}{\sim} N(\mu, \tau^2)$$

θ_i can be seen as the expected value of a future observation of any unit i. So, the predictive distribution of a new observation is obtained by the posterior distribution of $(\theta_i|y)$, for instance by integration on the posterior distribution of $(\theta_i, \sigma^2|y)$, if a posterior for σ^2 would also be requested.

7.4.1 A possible conclusion

It is always possible to make inferences conditional on variables at higher or lower values of the model, also remembering the following vocabulary.
Priors: immediate parents.
Likelihood: immediate children.
Basic idea: prediction on level 2 variables is the principal aim, but, as a "sub-product": forecasts on level 1, for non-observed values or for future observations, can be performed.

7.5 Some important Bayesian Hierarchical Models

7.5.1 Model of partial exchangeability, no covariates (Example 2)

Exchangeability may be partial. A model is proposed that shows a completely nested structure, without covariates.
In this case invariance occurs with respect to arbitrary permutations of observations, but restricted within groups (nested exchangeability):

Level 1: It admits subgroups as follows

$$y_{ij} \overset{\text{iid}}{\sim} N(\theta_i, \sigma^2)$$

In this formulation, the j are replications within i's.

Level 2:

$$\theta_i \overset{\text{iid}}{\sim} N(\mu, \tau^2)$$

Level 3:

$$\mu \sim \Pi_\mu, \sigma^2 \sim \Pi_\sigma, \tau^2 \sim \Pi_\tau$$

In this example, priors for variances are expicitely assumed. Indifference priors are assigned to the values of the parameters that are common to the whole population and are indicated with the symbol Π.

In this model the joint distribution of $(y, \theta, \mu, \sigma^2, \tau^2)$ is not invariant wrt any permutation, but only for the permutations of j for fixed i.

7.5.2 Model for a cross classification, no covariates (Example 3)

Level 1:

$$y_{ij} \overset{\text{iid}}{\sim} N(\theta_{ij}, \sigma^2)$$

It may contain replications. In this case the random variable is denoted as y_{ijk} and nesting occurs within cells.

Level 2:
The parameters at the second level are of the kind θ_{ij} (exchangeability holds only within each $i \times j$ cell), for which the simplest model is:

Level 2a: $\theta_{ij} \sim N(\mu + \alpha_i + \gamma_j, \tau_\theta^2)$,

but also the following part is requested, for the priors of parameters that are not common to the whole population:

Level 2b: $\alpha_i \overset{\text{iid}}{\sim} N(\rho, \tau_\alpha^2)$, $\gamma_j \overset{\text{iid}}{\sim} N(\delta, \tau_\gamma^2)$

Level 3:
$\mu \sim \Pi_\mu$, $\rho \sim \Pi_\rho$, $\delta \sim \Pi_\delta$, $\sigma^2 \sim \Pi_\sigma$, $\tau_\theta^2 \sim \Pi_{\tau_\theta}$,$\tau_\alpha^2 \sim \Pi_{\tau_\alpha}$, $\tau_\gamma^2 \sim \Pi_{\tau_\gamma}$

Also in this example, some general means and variances receive indifference priors.

Collapsing levels
Another way to write this model occurs when 2 or more levels are combined in a single distribution. In such case a covariance matrix is needed, built via the component of variance parameters.
Independence and symmetry are no longer explicit in the hierarchical diagram but are reflected in the symmetry of the covariance matrix as follows:
Level 1: $y_{ij} \overset{\text{iid}}{\sim} N(\theta_{ij}, \sigma^2)$
Level 2 in matrix form: $\theta \sim N(\mu\mathbf{1}, \tau_\theta^2\mathbf{J} + \tau_\gamma^2\mathbf{K} + \tau_\alpha^2\mathbf{L})$
where $\mathbf{J}$, $\mathbf{K}$ and $\mathbf{L}$ are block matrices.

7.5.3 Models with Covariates

Inference on parameters ought to be truly Bayesian, i.e. conditional only on data. Inference may be simplified by using the independence relationships. It is an alternative to asymptotic methods, which are difficult to write analytically, since the marginal likelihood cannot always be expressed in closed form (often

Restricted ML is applied instead).

In the following regression models, exchangeability is not expressed directly since there are arbitrary values of covariates for each index or set of indices.

7.5.3.1 The regression model, case 1 (Example 4)

Different proposals can be made. Below, one possibility is illustrated. It is a mixed structure, where at the data level a regression model is formalized, but the intercept is LOCAL and random.

Level 1: $y_{ij} \overset{\text{iid}}{\sim} N(\theta_i + x_{ij}\beta, \sigma^2)$, with $i = 1, \ldots, k$ and $j = 1, \ldots, n_k$
Level 2: $\theta_i \overset{\text{iid}}{\sim} N(\mu, \tau^2)$
Level 3: Priors for the population parameters μ, τ^2, σ^2 have to be assumed. Also β is a population parameter. Usually, indifference priors are assumed, even improper.

For better understanding the model, some explanations follow.
Meaning of i: the groups. A random intercept is proposed for each group.
Meaning of j: an index running within each group.
Covariate x: a variable (possibly multivariate) with a different value available for each individual belonging to each group. This is witnessed by the suffix i, j.
The expected value of the conditional distribution of y_{ij} at the first level depends on the population parameter β (hyperparameter) and on a "process" parameter θ_i.

General considerations coming from the model above.
Importance of a varying intercept: it gives better evaluations of baseline differences among clusters. The introduction of a varying intercept is a way for managing over-dispersion.

7.5.3.2 The regression model, case 2 (Example 5)

In the following example, covariates are available, both for first level and second level units, z are individual covariates for the first level units and x are covariates for second level units. Level 1 model has no intercept.
Here the regression coefficient is random, and is modeled at level 2.
Level 1: $y_{ij} \sim N(z_{ij}\beta_i, \sigma^2)$
Level 2: $\beta_i \overset{\text{iid}}{\sim} N(x_i\lambda, \tau^2)$
Level 3: $\lambda \sim \Pi_\lambda$, $\sigma^2 \sim \Pi_\sigma$, $\tau^2 \sim \Pi_\tau$
are indifference priors for the population parameters.

7.6 HM are already popular in statistics with different names: several are known as General Linear Models GLM

Without covariates

GLM	HBM
Hierarchical linear models	Linear and normal at each stage
Random effect models	All parameters (possibly associated to units) are random
Mixed models	Fixed (not unit specific) and random effects in the same model
Component of variance models	Additive random effects, modeled as independent and normal, for nested or cross classified units

With covariates

GLM	HBM
Generalized Linear Mixed Models	Mixed models where level 1 is a GLM
Models with random coefficients	Random regression coefficients, associated with units at a higher level
Growth curve models	Random coefficient models where the second (and higher) level clusters are groups of observations of the same individual along time

7.6.1 Models without covariates

7.6.1.1 Variance Component Model (Example 2 Continuation)

The model of partial exchangeability without covariates, when proposed with the hierarchical structure, has 3 levels. It considers nested data (no individual parameters).
Written in a different way, it is the well known:

$$y_{ij} = \mu + \theta_i + \epsilon_{ij}.$$

where $\theta_i \overset{\text{iid}}{\sim} N(0, \tau^2)$
and $\varepsilon_{ij} \overset{\text{iid}}{\sim} N(0, \sigma^2)$.

Replications may occur. They will be indicated with the suffix k , for y_{ijk} and ϵ_{ijk}.

7.6.1.2 Random-Effect Models (Example 3 Continuation)

The model of Example 3 may be seen as the sum of random effects:

$y_{ij} = \mu + \alpha_i + \gamma_j + \delta_{ij} + \epsilon_{ij}$

Such random effects can be written as 0-mean errors, each with a different variance:
$\alpha_i \overset{\text{iid}}{\sim} N(0, \tau_\alpha^2)$
$\gamma_j \overset{\text{iid}}{\sim} N(0, \tau_\gamma^2)$
$\delta_{ij} \overset{\text{iid}}{\sim} N(0, \tau_\delta^2)$
components like δ_{ij} denote an interaction between i and j, for instance an oscillation of j for each i
$\varepsilon_{ij} \overset{\text{iid}}{\sim} N(0, \sigma^2)$.

If replications occur, they can be denoted via the further suffix k, obtaining y_{ijk} and ε_{ijk}.

7.6.2 Models with covariates

When covariates are present, the hierarchical model can be written with unit specific, group specific and related to the whole population regression coefficients.
Covariates may be available at the unit level and some others at higher level.

(Example 4 Continuation)

Here the individual covariate relates to the population parameter β.

$y_{ij} = \theta_i + x_{ij}\beta + \varepsilon_{ij}$

$\theta_i \overset{\text{iid}}{\sim} N(0, \psi^2)$, $\varepsilon_{ij} \overset{\text{iid}}{\sim} N(0, \sigma^2)$

(Example 5 Continuation)

One auxiliary variable (covariate) is available at the unit level, and another at the group level .

$y_{ij} = z_{ij}\beta_i + x_i\lambda + \varepsilon_{ij}$

$\beta_i \overset{\text{iid}}{\sim} N(0, \tau^2)$

$\varepsilon_{ij} \overset{\text{iid}}{\sim} N(0, \sigma^2)$

(Example 6: another model)
All auxiliary variables (covariates) are available at the unit level, but some are linked to the second level parameters and some others to the whole population parameters.

Covariates may be available at the unit level and some others at higher level.

$$y_{ij} = z_{ij}\beta_i + w_{ij}\eta + \varepsilon_{ij}$$

$$\beta_i \overset{\text{iid}}{\sim} N(0, \tau^2)$$

$$\varepsilon_{ij} \overset{\text{iid}}{\sim} N(0, \sigma^2)$$

7.6.3 Longitudinal data

Level 1: observations (possibly with covariates) characterizing the same individual at different times.
Level 2: covariates characterize each individual.

7.6.4 Growth curves

Level 1 consists of a regression model that can be associated to a covariance matrix that represents a process for serially correlated residuals (e.g. an AR model or a random walk).
Coefficients β_i describe the trajectories of values at level 2.

7.7 Models for non normal observations

Level 1 model may be non normal. Examples are:
count data (Poisson regression)
binomial data (logistic regression or probit)
binary data (logit model)

In the case of binary data we can write:
Level 1: logit $p(y_{ij} = 1|\theta_i) = \theta_i$

7.7.1 Conjugate non normal models

When observations are not normal, conjugacy can be seen in a seemingly more extended way. A 3 level model can be proposed where:

- at level 1: an element of the exponential family is taken (observations are not normal),
- at level 2: an element belonging to the conjugate family is taken.
It has been shown that the distribution of the parameter at level 2 (conditional on data and level 3 parameters) are members of the conjugate family.
In this case the marginal distribution of the sufficient statistics at level 1 is a member of a known family.

Example: Gamma Poisson regression
If the 3 level model is (group covariates available)
Level 1: $y_{ij} \sim P(\lambda_i)$
Level 2: $\lambda_i \sim \text{Gamma}(a, b_i)$ where $b_i = a \exp(x_i' \beta)$
Level 3: diffuse priors for a and β.
The posterior of level 2 parameters (conditional on data and on level 3 parameters) is
$\lambda_i | y, a, b \sim \text{Gamma}(a + n_i, b_i + y_{i+})$
n_i = number of observations in group i, and
$y_{i+} = \sum_{j=1}^{n_i} y_{ij}$ (total count in group i)
is the sufficient statistic for parameter λ_i at level 1 (under the Poisson likelihood) and follows a negative binomial (Pascal) distribution (it is a result in classical statistics).

In the Bayesian context, if the Gamma prior has expectation $E(\lambda_i | a, b_i)$, the expectation of the posterior for these second level parameters is

$$E(\lambda_i | y, a, b_i) = (a + n_i)(b_i + y_{i+}) = y_{i+}(a + n_i) + b_i(a + n_i)$$

This writing makes even more evident that the posterior expected value with a conjugate prior (conditional on hyperparameters) is a linear function of data.

These conjugate models have rather simple shrinkage interpretation: they are said uniformly shrunk.

7.8 Further considerations on priors in hierarchical models

Hierarchical models are specified via conditional distributions, each of them can lead to posteriors that are conditionally proper but the posterior for the whole model may be improper (degenerate).

As an example, let us consider again the measurement error model (the variance of the measurement error is here set =1):

$$y_i | \theta_i \overset{\text{iid}}{\sim} N(\theta_i, 1)$$

$$\theta_i | \tau \overset{\text{iid}}{\sim} N(0, \tau^2)$$

The marginal model for y_i is $y_i \overset{\text{iid}}{\sim} N(0, \tau^2 + 1)$

The likelihood of the marginal model is

$$L(\tau; y) = (1+\tau^2)^{-\frac{n}{2}} \exp\left[\frac{-ns^2}{1+\tau^2}\right] \text{ where } s^2 = \frac{1}{n}\sum y_i^2$$

It is bounded far from $\tau^2 = 0$, and is large and $\to \infty$ near to $\tau^2 = 0$.
Any prior with infinite mass near zero has a posterior with an infinite mass, i.e. a degenerate distribution: some priors that are adopted in one-level models are to be excluded in the hierarchical case.

In the case above, where $p(y, \theta, \tau) = p(y, \theta|\tau)p(\tau) = p(\tau|y, \theta)p(y, \theta) = p(\tau, \theta|y)p(y)$ the posterior $p(\tau|y, \theta)$ is proper conditionally on θ, while $p(\tau|y)$ is improper.

Another reference textbook

Hoff, P. D. (2009). *A first course in Bayesian statistical methods.* Springer Science & Business Media. ISBN: 978-0-387-92299-7

Content

Group comparisons and hierarchical modeling are dealt in Chapter 8. (The use of covariates is developed in Chapter 11).

Other references will be provided according to students' requests.

Chapter 8

Introduction to `rstanarm`

Remark 84. Exam:

- take your own pc for safety in case of lab misfuctionings
- at end of exam send an e-mail with R script of the exam

8.1 Recap and steps of analysis

8.1.1 Bayesian recap

Suppose we observe data $\mathbf{y} = (y_1, \dots, y_n)$, which is treated as a realization of a random variable $\mathbf{Y} = (Y_1, ..., Y_n)$:

1) Before analyzing the data, we encode our beliefs or knowledge about the parameter θ using a **prior distribution** with pdf $p(\theta)$

2) The likelihood is the probability of observing the data y, given the parameter θ, and is viewed as a **conditional distribution** $f(\mathbf{y}|\theta)$

3) After observing the data, we *update* our knowledge about θ using Bayes theorem, obtaining the **posterior distribution** of θ (the denominator ensures that the posterior distribution integrates to 1, making it a valid pdf)

$$p(\theta|\mathbf{y}) = \frac{f(\mathbf{y}|\theta)p(\theta)}{f(\mathbf{y})} = \frac{f(\mathbf{y}|\theta)p(\theta)}{\int_\Theta f(\mathbf{y}|\theta)p(\theta)d\theta} \propto f(\mathbf{y}|\theta)p(\theta)$$

At the last passage, since the denominator is basically a constant, we can rewrite the posterior ignoring it by using the proportionality sign.
Thus **posterior pdf** $\propto$ **prior pdf** $\times$ **likelihood**

Remark 85 (Impact of the prior). The likelihood is "more fixed" (depends on the data type): what we can mainly change is how we model the prior belief. This has impact on the posterior (fig 8.1); in figure's row

1. we have a non-informative flat prior: the posterior is totally proportional to the likelihood so there's no effect of the prior;

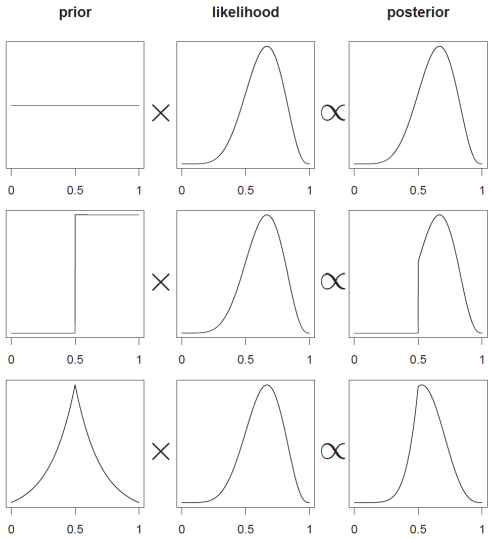


Figure 8.1: Prior choice and its effects

2. a step/truncated prior with 0 probability to all value less than 0.5; the posterior will reflect this with 0 probability less than value, and will be a truncated version of the likelihood

3. a peaked prior: this will make the posterior a skewed and peaked version of the likelihood

8.1.2 Steps of Bayesian Inference

The typical steps to perfom bayesian analyses are:

1) **Identify/Collect the data** (general recommendation: data visualization in order to understand the peculiarity of the data/have an idea how to model them)

2) **Specify a prior distributions** for the model parameters: $p(\theta)$

3) Choose a statistical **model for the data**: $f(\mathbf{y}|\theta)$

4) Obtain the **posterior distributions** for the model parameters as proportional of likelihood and prior product:

$$p(\theta|\mathbf{y}) \propto f(\mathbf{y}|\theta)p(\theta)$$

   4.1) When using computer approximations (eg MCMC for approximating posterior distribution), check the algorithms for convergence (do **Post-run diagnostics**)

5) Conduct a **posterior predictive check** to examine if the fitted model is compatible with the observed data

   5.1) If the model does not fit the data, one should go back to step 2 to specify a different model

6) **Summarize the Posterior Distribution** of the parameter via

- Posterior Mean, Median, and Mode: if posterior distribution is gaussian this is not a problem. But if skewed one has to understand which statistics is better to summarize results
- Uncertainty Estimates
- Credible Intervals

Remark 86. Theoretical lesson discussed first three steps; below

- we focus on steps 4-7.
- for point 4 we use MCMC.

8.2 MCMC

8.2.1 Simulation Methods

Remark 87 (Problem). The first step (step 4) is to generate a random sample from the posterior distribution

$$\theta^{(1)}, ..., \theta^{(S)} \overset{iid}{\sim} p(\theta|\mathbf{y}) = \frac{p(\theta)f(\mathbf{y}|\theta)}{p(\mathbf{y})}$$

This is easy in a *conjugacy setting* where the posterior is known and generators are available, but how to do it elsewhere (in a non-conjugacy setting) where we have no closed form?
In other words, **How is it possible to generate a sample from an unknown distribution?**

Important remark 42 (Solution). The idea is that it is sufficient to generate a random sample from an **unnormalized posterior density**, that is any function $\theta \to q(\theta; \mathbf{y})$ proportional to the posterior density: $p(\theta|\mathbf{y}) \propto q(\theta; \mathbf{y})$.
In particular, as Bayes theorem suggests, a function proportional to posterior is just its numerator, which is called **unnormalized version of the Bayes' theorem**:

$$p(\theta|\mathbf{y}) \propto p(\theta)f(\mathbf{y}|\theta)$$

Remark 88. Thus even if the posterior is unknown/untractable we can generate a sample from the product between the prior and the likelihood. How to do that?

Important remark 43 (Simulation methods). Several methods are available[1]:

- Rejection sampling;
- Importance sampling;
- Analytical approximation: eg one powerful tool is INLA (Integrated Nested Laplace Approximation). It's a flexible, newer and faster method compared to MCMC, and it's used for big dataset/models with lots of parameters;

[1] Reference: *Lambert, 2018; chapter 12 [Gelman, 2020; chapter 10]*

- **Simulation Methods** (our focus)
  - Monte Carlo integration;
  - Monte Carlo Markov Chain (MCMC) methods:
    * the most simple MCMC we can build is the **Gibbs sampler** and we'll see it;
    * Stan (state of the art language for Bayesian inference) is based on Hamiltonian Monte Carlo algorithm (which is just a more complicated Gibbs sampler)
    * to see how a Gibbs sampler works gives us an idea on how Stan works (which is the tool we'll actually use for inference via `rstanarm`)

8.2.2 Monte carlo simulations

Important remark 44. Computing integrals by simulation is known as **Monte Carlo integration** or Monte Carlo method.

8.2.2.1 Why does it work?

Remark 89. MC integration works because it is based on Strong Law of Large Numbers (SLLN) / almost sure convergence.

Theorem 8.2.1. *Strong Law Large Numbers (SLLN) Let $Y_1, Y_2, \dots$ be a sequence of iid random variables with finite mean $\mu := \mathbb{E}[Y_1]$ and $\mathbb{E}[|Y_1|] < \infty$. Then the*

$$P\left(\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} Y_i = \mu\right) = 1$$

Remark 90. So if we have a really long sequence of random variables extraction we can compute the average and get the mean of the unknown common distribution.

Important remark 45. In the same way, if we generate a big enough sample from our posterior distribution, we can get good approximation of it; thanks to the SLLN it is possible to state that the *empirical distribution* of the posterior-generated sample $\{\theta^{(1)}, ..., \theta^{(S)}\}$ (that is its histogram) is an approximation of the true posterior distribution $p(\theta|\mathbf{y})$. It's actually called the *MC approximation* of the real posterior.
Dynamics of that is plotted in figure 8.2, for sample size $S = 10, 100, 1000$.

Important remark 46 (Convergenge of the mean of a transformation g of the extracted sample). In general as $S \to +\infty$

$$\frac{1}{S}\sum_{s=1}^{S} g(\theta^{(s)}) \to \mathbb{E}\left[g(\theta)|\mathbf{y}\right] = \int_{\theta\in\Theta} g(\theta)p(\theta|\mathbf{y})\mathrm{d}\theta.$$

if we apply a function g to the extracted values and calculate the mean, this converge to the expected value of transformation g of the posterior (g can be the identity too as in the case above).

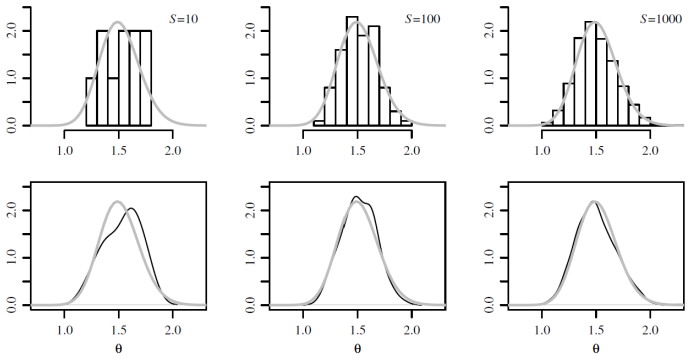


Figure 8.2: Monte carlo approximation as S increase

Remark 91. Being the extracted posterior a better and better approximation of the true posterior, all the empirical evaluations (mean/variance, quantiles, probabilities of the distribution) can be considered as reliable approximation of the true values.

Remark 92. Since we are dealing with approximations, it is possible to provide a measure of the **accuracy**

Definition 8.2.1 (Monte Carlo Standard Error)**.** Defined as

$$SE_{MC} = \sqrt{\frac{\hat{\sigma}^2}{S}},$$

where $\hat{\sigma}^2$ is the sample variance.

Remark 93. The idea here is to have the MC standard error as small as possible (ideally 0.00something) since if this is very small our approximation is basically equal/similar to the one form the true posterior distribution.

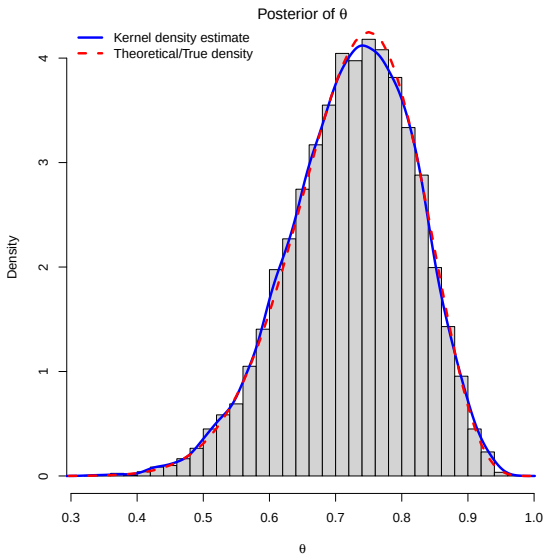
8.2.2.2 Example

Example 8.2.1 (Beta-binomial model)**.** Let assume a Beta prior (with parameters a, b) for the proportion θ of interest, a binomial data model/likelihood function and a trial where we observed r successes among n trial (denoted as $\mathbf{y} = r$ in the following). So posterior is again a beta (with $a + r, b + n - r$ parameters):

$$\begin{aligned}
&\theta|a, b \sim \text{Beta}\,(a, b)\\
&\mathbf{y}|\theta \sim \text{Bin}\,(n, \theta)\\
&\theta|(\mathbf{y} = r) \sim \text{Beta}\,(a + r, b + n - r)
\end{aligned}$$

In this case we could just draw a posterior sample but we see how to do it using MC approximation (eg using the *unnormalized posterior*).

```r
## define a grid for the parameter theta (discretized range of probabilities)
## possible values of theta
theta <- seq(0, 1, length.out = 1000)

## prior parameters and density
a <- 1
b <- 1
prior <- dbeta(theta, a, b)

# sample evidences and likelihood
r <- 15
n <- 20
likelihood <- dbinom(r, size = n, prob = theta)

# unormalized posterior (here coinciding with likelihood)
unnormalized_posterior <- prior * likelihood

# how to draw a random sample from the unnormalized posterior
set.seed(1234)  # for reproducibility
posterior_sample <- sample(theta,
                           size = 10000,
                           replace = TRUE,
                           prob = unnormalized_posterior)

# posterior parameters
a_prime <- a + r
b_prime <- b + n - r

## Posterior density visualization (sampled histogram, kdensity and theoretical are v
par(mar = c(4, 4, 1, 1))
hist(posterior_sample, breaks = 30, probability = T,  # hist of sample drawn
     main = expression(paste("Posterior of ", theta)),
     xlab = expression(theta))
points(density(posterior_sample), type = "l", col = "blue", lwd = 3) # kernel density
curve(expr = dbeta(x, shape1 = a_prime, shape2 = b_prime), # theorical density
      from = 0, to = 1, lty = 2,
      col = "red", lwd = 3, add = T)
legend("topleft", bty = "n",
       legend=c("Kernel density estimate", "Theoretical/True density"),
       col = c("blue", "red"),
       lty = 1:2, lwd = 3)
```




```r
## Summary statistics computation of posterior sampled vs theorical (are very
## similar)

c("mu_sampled" = mean(posterior_sample),
  "mu_theoretical" = a_prime / (a_prime + b_prime))

##     mu_sampled mu_theoretical
##      0.7255638      0.7272727

c("sd_sampled" = sd(posterior_sample),
  "sd_theoretical" = sqrt((a_prime * b_prime) / ((a_prime + b_prime) ^ 2 * (a_prime + b_prime +

##     sd_sampled sd_theoretical
##     0.09377631     0.09286435

p <- c(0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975)
rbind(quantile(posterior_sample, probs = p),  #first row sampled
      qbeta(p = p, shape1 = a_prime, shape2 = b_prime)) # second row theorical

##           2.5%        5%       25%       50%       75%       95%     97.5%
## [1,] 0.5245245 0.5605606 0.6636637 0.7317317 0.7947948 0.8678679 0.8888889
## [2,] 0.5283402 0.5630237 0.6671084 0.7342603 0.7948175 0.8675518 0.8871906

## MCSE, monte carlo standard error
posterior_sd <- sd(posterior_sample)
N <- length(posterior_sample)
(mcse_mean <- posterior_sd / sqrt(N)) # really small

## [1] 0.0009377631
```

8.2.3 Monte Carlo Markov Chains (MCMC)

Important remark 47 (Problems with classic MC). Simple MC simulations works well if we have *low number of parameters* to be estimated; however we can have problems/MC is not enough in case of lot of parameters to be estimated/**high dimensional parameters** problems.

Important remark 48. For the more complex models, sampling is usually done by using **Monte Carlo Markov Chain (MCMC)** methods: these are based by iteratively sampling from a *Markov chain whose stationary distribution is the target distribution*, (and in the case of Bayesian computation is most often the posterior distribution $p(\theta|\mathbf{y})$).

Definition 8.2.2. Markov Chain A discrete-time Markov chain (or Markov process) is a discrete-time stochastic process for which the **Markov property** holds:

$$\mathbb{P}\left[\theta_t^*|\theta_0^*, ..., \theta_{t-1}^*\right] = \mathbb{P}\left[\theta_t^*|\theta_{t-1}^*\right]$$

Remark 94. That is, at any given time, the present state θ_t^* of the chain depends only on the previous state θ_{t-1}^*, and not on the rest of the history.

8.2.3.1 MCMC sampling

Important remark 49 (Autocorrelation issues). Our original aim was to generate an iid sample from the target (posterior) distribution but using a MCMC, the components of the sample $\theta^{(1)}, \ldots, \theta^{(S)}$ has a **very high autocorrelation**, since the next value θ_{t+1} is likely to be somewhere near the current value θ_t of the chain.

Important remark 50 (Autocorrelation fix). To achieve our iid sampling aim, if we generate a large enough sample (and we use the whole sample to approximate our posterior distribution) the autocorrelation of the single values does not matter

Example 8.2.2. With a sample of 10000 values each value is only autocorrelated with another value, not with other 9998, so if we generate a sample which is high enough this correlation doesn't matter and so we can assume we still draw from iid distributions, even though there is this kind of autocorrelation

Remark 95. However, how are we sure that this chain produces values that converge to the posterior distribution?
The Markov chains that are used in MCMC methods are designed so that their stationary distribution (once that our chain hits this stationary distribution, it basically stays there) is the target posterior distribution.

Definition 8.2.3. Stationary Distribution A distribution $\pi(x)$ such that if $P(X_0 = k) = \pi(k)\ \forall k \in \mathcal{S}$, then also $P(X_i = k) = \pi(k)\ \forall i = 1, 2, \ldots$ where $\mathcal{S}$ is the state space of the Markov chain

Remark 96. This means that once the chain hits its stationary distribution it stays there, and thus the value $\pi(k)$ is also a long run proportion of the time the chain stays in a state k.

Important remark 51. Since we defined the chain so that the stationary distribution π is the posterior distribution $p(\theta|\mathbf{y})$, if the chain moves in it stationary distribution long enough, we get a sample from the posterior.

Proposition 8.2.2. *More formally, if a Markov Chain possesses all these three properties:*

- ***Irreducibility**: each set of states can be reached starting from each state with a finite number of steps;*
- ***Positively recurrent (or persistent)**: the probability of returning to the current state in a finite number of steps is 1;*
- ***Aperiodic**: there is no periodic oscillation among the states;*

*then the **ergodic theorem** holds and as* $S \to +\infty$

$$\frac{1}{S}\sum_{s=1}^{S} g(\theta_s^*) \to \mathbb{E}\left[g(\theta)|\mathbf{y}\right]$$

*Therefore these chains converges to the **stationary distribution**, that is unique (and we reach it independently from the initial value* θ_0^**).*

Remark 97. Some initial values θ_0^* will get there faster than others but one way or another we'll reach the convergence.

Important remark 52 (Burn-in period). Since we reach convergence only *after some iterations/a certain period*:

- the **first iterations of MCMC sampling are usually discarded** because the values of the chain before it has converged to the stationary distribution are not representative of the posterior distribution (and are not useful to do inference);
- **how many** sampled points are **discarded** is matter of choice: a very conservative and safe approach is to discard the first half of the iterations. These discarded iterations are called **burn-in period** or warm-up period;
- Stan discards the warm-up period automatically (by default discards the first half of the chain to be sure to extract from the true posterior distribution).

Important remark 53 (Checking convergence of chains). How to be really sure that the chain converged to its stationary distribution? Both

- check the **model diagnostics** (more on this these later);
- **run several chains starting from the different initial values in parallel**: if they all converge to a similar distribution, it is quite likely that this is the stationary distribution (Stan runs four parallel chains as default).

8.2.3.2 Examples of MCMC Algorithms

Some examples are:

- **Metropolis-Hastings algorithm**: general framework which includes
  - **Gibbs Sampler**: special case with acceptance rate uniformly equal to 1
  - **Metropolis algorithm**: special case with symmetric proposal distribution.
- **Hamiltonian Monte Carlo (HMC)**: it allows to sample from the posterior of the target parameters more efficiently than basic MCMC algorithms (what Stan uses).

8.2.3.3 The Gibbs sampler

Remark 98. The **Gibbs sampler** is the easiest MCMC algorithm and it is attractive because it can sample from high-dimensional posteriors.[2]

Important remark 54 (Main idea). Directly sampling from the joint/high-dimensional posterior $p(\boldsymbol{\theta}|\mathbf{y})$ may be challenging thus we decompose it into *conditional distributions*

$$p(\theta_j|\boldsymbol{\theta}_{-j}, y), \quad j = 1, ..., m.$$

where the notation with - at the index means that every parameter θ_j is conditioned on every other parameter excepted θ_j

Important remark 55 (Algorithm). The algorithm is constituted by the following steps:

1. we start by fixing the initial state/guess of all the parameters: $(\theta^*_{1,(0)}, ..., \theta^*_{m,(0)})$. These may be defined randomly (no matter how we choose initialize them, the chain will converge eventually)
2. we iteratively generate/sample each parameter θ_j from its conditional distribution while keeping all the other parameter fixed/as conditioning (at the *most updated version*): eg at step b

$$\begin{aligned}
\theta^*_{1,(b)} &\sim p(\theta_1|\theta^*_{2,(b-1)}, ..., \theta^*_{m,(b-1)}, y),\\
\theta^*_{2,(b)} &\sim p(\theta_2|\theta^*_{1,(b)}, \theta^*_{3,(b-1)}, ..., \theta^*_{m,(b-1)}, y),\\
&\dots\\
\theta^*_{m,(b)} &\sim p(\theta_m|\theta^*_{1,(b)}, ..., \theta^*_{m-1,(b)}, y).
\end{aligned}$$

   so eg we can sample θ_1 while keeping $\theta_2, \dots, \theta_m$ fixed (coming from the previous step $b-1$); the same happens for θ_2 where we use the recently updated θ_1 (from step b) and all the other parameters from previous step $b-1$; finally the last θ_m will be sampled using estimates generated at step b
3. repeat previous step B times/steps ($b = 1, \dots, B$); in this way we sample each theta B times.

[2]Reference: *Lambert, 2018; chapter 14 [Gelman, 2020; chapter 11]*

Remark 99. Some remarks:

- The stepwise updating of theta creates a posterior sample which has some correlation due to the conditioning, but as we said if B is high enough correlation doesn't matter.
  Actually the dependence turns out to be a Markov distribution ($\rightarrow$ MCMC)

- Updates can also be done in blocks (at each step sample the groups of parameters, not just one theta at time)

- Regardless of the initial starting point $\boldsymbol{\theta}^{(0)}$ the distribution of the samples $\boldsymbol{\theta}^{(b)}$ generated by the Gibbs sampler converges to the target distribution $p(\boldsymbol{\theta}|\mathbf{y})$. However some starting points may converge faster.

- As the number of iterations $b \rightarrow \infty$ grows, convergence means that for each subset A the proportion of samples in A

$$\mathbb{P}(\boldsymbol{\theta}^{(b)} \in A|\mathbf{y}) \rightarrow \int_A p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

  where A is a subset of the sample space: that is the proportion of samples in A will approximate the probability given by the integral of the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$.

- More importantly this things work for most functions g of interest of our parameter, as $B \rightarrow \infty$

$$\frac{1}{B}\sum_{b=1}^{B} g(\boldsymbol{\theta}^{(b)}) \rightarrow \mathbb{E}\left[g(\boldsymbol{\theta})\right] = \int g(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

  So we can approximate $\mathbb{E}\left[g(\boldsymbol{\theta})\right]$ with the sample average of $g(\boldsymbol{\theta}^{(1)}), \ldots, g(\boldsymbol{\theta}^{(B)})$ (coming from the step updates done in the algorithm), just as in Monte Carlo approximation.
  This's why we call such approximations Markov Chain Monte Carlo (MCMC) approximations, and the procedure an MCMC algorithm.

8.2.3.4 Gibbs sampler examples

Example 8.2.3 (Conjugate normal model)**.** Let's consider a normal model for data $y_i|\theta, \phi \sim \mathrm{N}(\theta, \phi)$, $\forall i$, with the semi-conjugate prior distributions. We assume normal for mean and inverse gamma for variance:

$$\begin{aligned}\theta|\theta_0, \phi_0 &\sim \mathrm{N}(\theta_0, \phi_0)\\ \phi|\nu_0, S_0 &\sim \mathcal{IG}(\nu_0/2, S_0/2)\end{aligned}$$

If a sample $\mathbf{y}$ is observed, the full conditionals of the model parameters are again normal and inverse gamma:

$$\begin{aligned}\theta|\phi, \mathbf{y} &\sim \mathcal{N}(\theta_1, \phi_1)\\ \phi|\theta, \mathbf{y} &\sim \mathcal{IG}(a_1, b_1)\end{aligned}$$

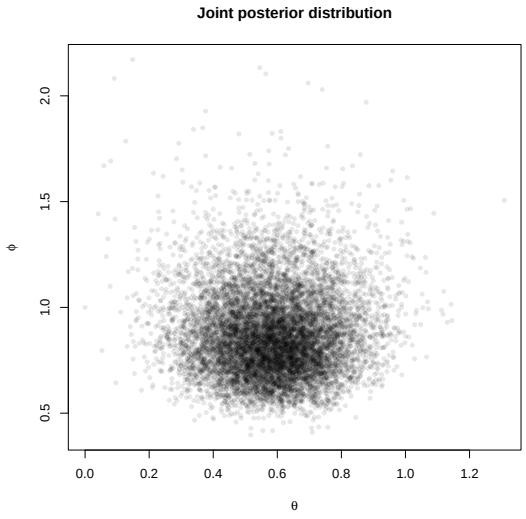
where

$$\theta_1 = \frac{\frac{\theta_0}{\phi_0} + \frac{n\bar{y}}{\phi}}{\frac{1}{\phi_0} + \frac{n}{\phi}}, \quad \phi_1 = \frac{1}{\frac{1}{\phi_0} + \frac{n}{\phi}};$$
$$a_1 = \frac{\nu_0}{2} + \frac{n}{2}, \quad b_1 = \frac{S_0}{2} + \frac{\sum_{i=1}^{n}(y_i - \theta)^2}{2}$$

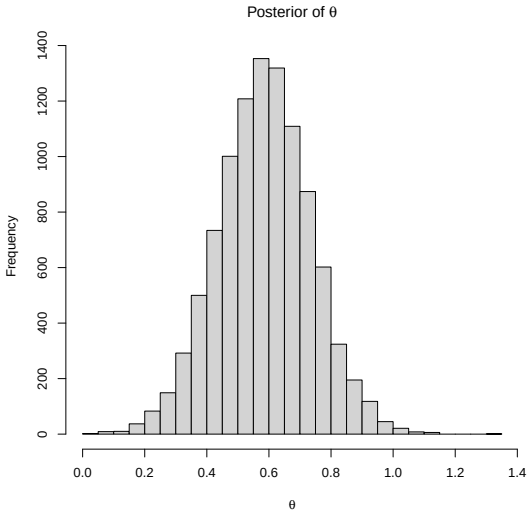
The posterior distributions can be easily obtained by MCMC methods.

```
# Normal prior for the mean: hyperparameters
theta_0 <- 0
phi_0 <- 100

# Inverse gamma prior for variance: hyperparameters
nu_0 <- 1
S_0 <- 1

# sample
n <- 40
set.seed(1234)
y <- rnorm(n = n, mean = 1, sd = 1)
## mean(y)
## var(y)


##########################
###   GIBBS SAMPLER   ###
##########################

## number iterations
B <- 1e4

## Create vector for realizations of theta and phi we'll generate
theta_sample <- numeric(B) #mean
phi_sample <- numeric(B) #variance

## Initialize the loop with arbitrary values: if changed the chaing will
## converge anyway
theta_sample[1] <- 0
phi_sample[1] <- 1

## loop: at each step we update conditional for theta and phi using formulas above
for (i in 2:B) {

## infos (and update) for theta
theta_1_num  <- (theta_0 / phi_0) + (sum(y) / phi_sample[i-1]) # numerator of theta1
phi_1  <- 1/((1/phi_0) +(n/phi_sample[i-1]))
theta_1 <- theta_1_num * phi_1
theta_sample[i] <- rnorm(1, mean = theta_1, sd = sqrt(phi_1))
```

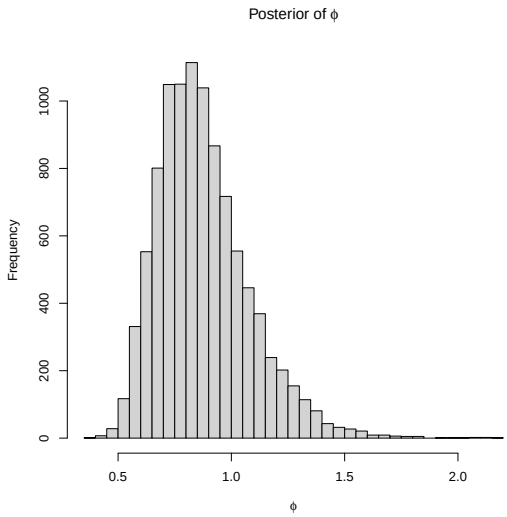
```r
## info and update for phi
a_1 <- nu_0/2 + n/2
b_1  <- S_0/2 + sum((y-theta_sample[i])^2)/2
phi_sample[i] <- 1/rgamma(1, shape = a_1, rate = b_1)

}

## Joint posterior distribution of params: we can see from the cloud there is
## not a pattern so we can conclude they're basically IID
par(mfrow = c(1, 1))
plot(theta_sample,
   phi_sample,
   xlab = expression(theta),
   ylab = expression(phi),
   pch = 20,
   col = lbmisc::col2hex("black", alpha=0.1),
   main = "Joint posterior distribution")
```



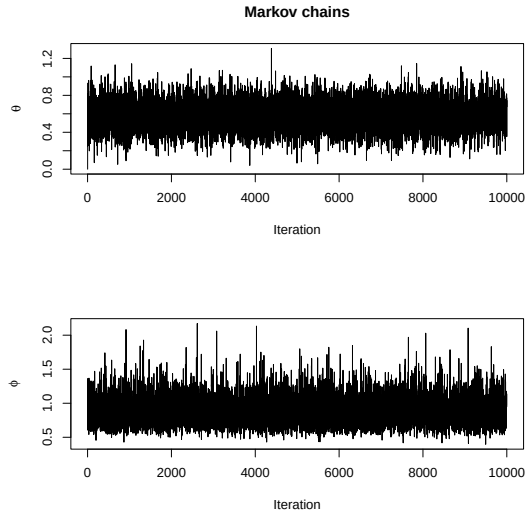

```r
## marginal posteriors for theta (approximately normal)
hist(theta_sample, breaks = 30, xlab = expression(theta),
   main = expression(paste("Posterior of ", theta)))
```




```
mean(theta_sample) ## posterior features

## [1] 0.5849505

sd(theta_sample)

## [1] 0.1500566

quantile(theta_sample, probs = c(0.025, 0.25, 0.5, 0.75, 0.095))

##      2.5%       25%       50%       75%      9.5%
## 0.2908597 0.4842693 0.5855970 0.6837951 0.3881482

## quantiles are useful for credibility intervals (most interesting, for
## working with non statisticians)

## marginal posterior for phi (not gaussian more right skewed)
hist(phi_sample, breaks = 30, xlab = expression(phi),
   main = expression(paste("Posterior of ", phi)))
```




```
mean(phi_sample); sd(phi_sample)

## [1] 0.8766848
## [1] 0.2023807

quantile(phi_sample, probs = c(0.025, 0.25, 0.5, 0.75, 0.095))

##      2.5%       25%       50%       75%      9.5%
## 0.5680013 0.7328650 0.8471696 0.9879092 0.6433984

## In order to check the markov chain has converged to a stationary
## distribution we can plot it (each subsequent parameter against the id
## basically).

## The idea is to idea is to have a plot similar to this two below, that is
## without trends: the parameter are not increasing/decreasing, they're
## stationary and always in the range of posterior distribution.

par(mfrow = c(2,1))
plot(theta_sample, ylab = expression(theta),
   xlab = "Iteration", type = "l",
   main = "Markov chains")
plot(phi_sample, ylab = expression(phi), xlab = "Iteration", type = "l")
```




```
# Markov chains above has converged to stationary distribution and is a good
# plot and so the value we draw for our posterior distribution can be used
# because the algorithm worked properly
```

Example 8.2.4 (Bivariate normal)**.** Assume that we have just one observation $(y_1, y_2) = (0, 0)$ from the two-dimensional normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\mu} = (\mu_1, \mu_2)$ and

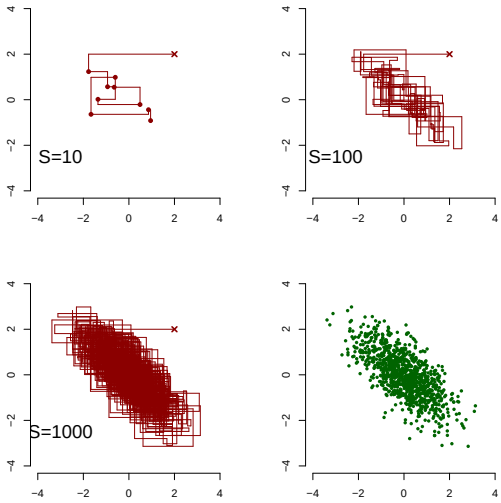
$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

is assumed as a known constant matrix (let's assume that $\rho = -0.7$) and also assume we are using an improper uniform prior for $\boldsymbol{\mu}$ that is $p(\boldsymbol{\mu}) \propto 1$ (so posterior is as well bivariate normal with same parameter as the likelihood).
Now the posterior is a 2-dimensional normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma}_0)$; from theory/properties of the multinormal distribution we get the conditional posterior distributions of $\mu_1|\mu_2$ and $\mu_2|\mu_1$:

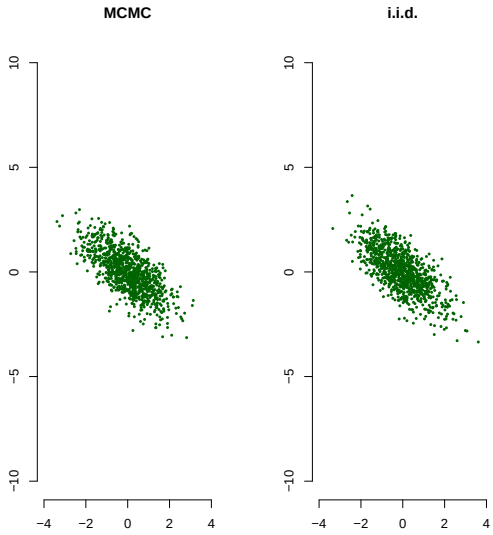
$$\mu_1|\mu_2, \mathbf{y} \sim N(y_1 + \rho(\mu_2 - y_2), 1 - \rho^2)$$
$$\mu_2|\mu_1, \mathbf{y} \sim N(y_2 + \rho(\mu_1 - y_1), 1 - \rho^2)$$

```
#####################################################
## Example: two-dimensional normal distribution  ##
#####################################################

## set the parameter and observation values
y   <- c(0, 0)
rho <- -0.7

## define the functions to update the conditional posterior distributions
```

```r
## in R rnorm use stddev not variance
mu1_update <- function(y, rho, mu2) rnorm(1, y[1] + rho * (mu2-y[2]), sqrt(1-rho^2))
mu2_update <- function(y, rho, mu1) rnorm(1, y[2] + rho * (mu1-y[1]), sqrt(1-rho^2))

# Note that in R the normal distribution is parametrized with standard deviation,
# not variance, so that the parameters are (mu, sigma) instead of the usual
# parameters (mu, sigma^2).

# setup and initialize the results vector and start sampling:
n_sim <- 1000
mu1 <- mu2 <- numeric(n_sim) # vector to store
# set a random initial value (2, 2) for mu
mu1[1] <- 2
mu2[1] <- 2

## iteratively update using conditional distributions
for(i in 2:n_sim) {
  mu1[i] <- mu1_update(y, rho, mu2[i-1])
  mu2[i] <- mu2_update(y, rho, mu1[i])
}

## plot convergence of the chain in two dimensions: we examine the trace of the
## sampler after 10, 100, and 1000 simulation rounds
## function for plotting
draw_gibbs <- function(mu1, mu2, S, points = FALSE) {
  plot(mu1[1], mu2[1], pch = 4, lwd = 2, xlim = c(-4,4), ylim = c(-4,4), asp = 1,
       xlab = expression(mu[1]), ylab = expression(mu[2]), bty = 'n', col = 'darkred')
  for(j in 2:S) {
    lines(c(mu1[j-1], mu1[j]), c(mu2[j-1], mu2[j-1]), type = 'l', col = 'darkred')
    lines(c(mu1[j], mu1[j]), c(mu2[j-1], mu2[j]), type = 'l', col = 'darkred')
    if(points) points(mu1[j], mu2[j], pch = 16, col = 'darkred')
  }
  text(x = -3, y = -2.5, paste0('S=', S), cex = 1.75)
}
draw_sample <- function(mu1, mu2, ...) {
  plot(mu1, mu2, pch = 16, col = 'darkgreen',
       xlim = c(-4,4), ylim = c(-4,4), asp = 1, xlab = expression(mu[1]),
       ylab = expression(mu[2]), bty = 'n', ...)
}
# actual plotting
par(mfrow = c(2,2), mar = c(2,2,4,4))
draw_gibbs(mu1, mu2, 10, points = TRUE) # 2,2 is starting point.
draw_gibbs(mu1, mu2, 100)
draw_gibbs(mu1, mu2, n_sim)
draw_sample(mu1[10:length(mu1)], mu2[10:length(mu2)], cex = 0.7)
```




```
## first three graphs show the dinamics (x is the starting point). We can
## somewhat see that the chains tends to the mass of the distribution (should
## be 0,0 I guess) after few iterations. By increasing the n of iterations, the
## algo converge to the central point mass of our two dimensional distribution.

## The last one in green show the bivariate distribution after throwing the
## first observations


## Now we can compare our posterior distribution with a sample generated using
## iid sampling (from true posterior distribution known from)
Sigma <- matrix(c(1, rho, rho, 1), ncol = 2)
X <- MASS::mvrnorm(n_sim, y, Sigma)
par(mfrow = c(1,2), mar = c(2,2,4,4))
draw_sample(mu1[10:length(mu1)], mu2[10:length(mu2)], cex = 0.5, main = 'MCMC')
draw_sample(X[ ,1], X[ ,2], cex = 0.5, main ='i.i.d.')
```




```
# Here we can see that basically the sample we draw using MCMC and sample
# generate from the true distribution are the same/indistinguishable. So the
# sample we MCMC-generated can be treated as an IID sample and  we can use it
# to approximate our posterior distribution
```

Chapter 9

Introduction to `rstanarm`

9.1 Stan and rstanarm

`Stan`

- is a *generic statistical model specification language* (e.g.: GLM, GLMM) based on the **Hamiltonian Monte Carlo (HMC)** algorithm;
- it implements also *optimization methods*
- is exposed through interfaces into many popular computing environments (`RStanArm` package for R, `PyStan` library for Python)

Remark 100 (Bayesian modelling). We have

- Generalized linear models (GLMs)
- Models including random effects

In bayesian inference the concept of random effect is used to create hierarchical model (eg school for children)
A random effect is an effect not constant but can be different for different observations; while fixed effect can be estimated in a flat way because constant for every unit for the random effect

Important remark 56 (Key steps in basic inferential procedures using `rstanarm`). We have to:

1) **Understand the data**: eg plot, visualize etc
2) **Formulate the model**
3) **Specify the prior**
4) Initialize `rstanarm` model and computations
5) Check *convergence* of the algorithm
6) Check model assumptions (**posterior predictive check**)
   → choice of the better model (WAIC)
7) Analyse the posterior results (**summarize the posterior distribution**)

Remark 101. Every exercise we will do will follow these 7 steps; *all these steps are required for the exam.*

9.2 Example: gaussian linear model

9.2.1 1. Understand the data

In this example we just simulate our data (using 2 normal distributions), and impose a relationship between covariates and the outcome y (so actually here we know everything we don't need to understand)

```
library(rstanarm)
library(rstan)
library(ggplot2)
library(bayesplot)

set.seed(123)
n <- 100
x1 <- rnorm(n = n, mean = 1,   sd = 0.5)
x2 <- rnorm(n = n, mean = 0.5, sd = 0.25)
X <- cbind(x1, x2)
sigma <- 1      # sd of varepsilon
beta <- c(1, 2) # beta coeffs between x1, x2 and y
y <- rnorm(n = n, mean = X %*% beta, sd = sigma)
data_example <- data.frame(y, x1, x2)

## classic estimate
summary(lm(y ~ x1 + x2, data = data_example))

##
## Call:
## lm(formula = y ~ x1 + x2, data = data_example)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8730 -0.6607 -0.1245  0.6214  2.0798
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3538     0.3103   1.140 0.257002
## x1            0.7337     0.2097   3.498 0.000709 ***
## x2            2.0952     0.3960   5.291 7.53e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9513 on 97 degrees of freedom
## Multiple R-squared:  0.2841, Adjusted R-squared:  0.2693
## F-statistic: 19.25 on 2 and 97 DF,  p-value: 9.125e-08
```

9.2.2 2. Model formulation

This is the theoretical part of the analysis where we write down the model (hierarchy) we're fitting: in our case we have a gaussian linear model, with

normal prior. We have to specify:

1. **Likelihood:** we need to express the distributional assumption and the model equation considered. E.g. for a *Gaussian linear model*:

$$y_i|\mu_i, \sigma^2 \sim \mathcal{N}\big(\mu_i, \sigma^2\big), \quad i = 1, ..., n$$
$$\mu_i|\boldsymbol{\beta} = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 \quad i = 1, ..., n.$$

2. **Priors:** we need to state the prior distributions specified for the parameters. In this case we use:

$$\beta_p \sim \mathcal{N}(0, c), \ p = 0, 1, 2;$$
$$\sigma \sim \text{half-Cauchy}.$$

   with c a fixed constant (here we have σ and not σ^2)

3. **Hyperpriors:** In the context of hierarchical models, these are the priors for the *hyperparameters*, which are parameters of the model parameters priors. In this case there are no hyperparameters, since c is a fixed constant
   Eg if otherwise β_p could be distributed according to $\mathrm{N}(0, \alpha)$, with $\alpha \sim$ half-Cauchy or half-T the hyperparameters would be these latter distribution ones and hyperpriors their priors.

9.2.2.1 Implementation

Once the model scheme is clear, we need to implement it with the `rstanarm`. We will use two functions:

- `stan_glm`: allows to make Bayesian inference for GLM;
- `stan_glmer`: allows to make Bayesian inference for GLM with *random effects*.

For each of these two functions we have to specify three arguments:

- `formula`: useful to express the model equation (also group random effects). Syntax is the same as `lm` and `glm` when only *fixed effects* are included in the model (as here).
  When a *random effect* determined by a generic grouping variable `group` is required, the same syntax of `lme4` package is adopted; for example, a **random intercept** can be included in the formula as `(1|group)` and a **random coefficient** for the covariate `x` is declared as `(x|group)`.
- `family`: the distributional assumption. We will consider `"gaussian"`, `"binomial"` and `"poisson"`.
- `data`: the `data.frame` that includes the variables declared in the `formula` statement;

9.2.3 3. Prior specification

Arguments in table 9.1 allows to specify the prior distributions. If ones does not specify anything a non-informative prior is used by default.

Argument	Used in	Specifies priors for ...
prior_intercept	all modelling functions except stan_polr and stan_nlmer	Model intercept, after centering predictors
prior	all modelling functions	regression coefficients. Does *not* include coefficients that vary by group in a multilevel model (see prior_covariance)
prior_aux	stan_glm*, stan_glmer*, stan_gamm4, stan_nlmer	Auxiliary parameter, eg error SD (interpretation depends on the GLM)
prior_covariance	stan_glmer*, stan_gamm4, stan_nlmer	Covariance matrices in multilevel models with varying slopes and intercepts. See the stan_glmer vignette for details on this prior

Table 9.1

9.2.4 4. Initialization and start the computation

Regarding the chain there could be some arguments that one could want to specify, but again one can not specify anything and the defaults are applied:

- usually, the **multichain** approach is used: two or more parallel Monte Carlo Markov Chains are simulated in order to check the convergence to the target distribution by comparison.
- Each chain, for every parameter, needs to be initialized: choosing different starting values the robustness of the algorithm is tested.
  By default, Stan and rstanarm randomly generates appropriate starting values. However in case of particularly complex models the algorithm might requires a more precise initialization.

Some arguments to specify various options for the estimation:

- chains: number of parallel chains to compute (default 4)
- warmup: number of warmup iterations not included in the inference (default 50%)
- iter: number of total iterations/length of the chain (default 2000)
- init: list of lists with the fixed starting values (optional)

To **start the computation**, Stan engine can be invoked in R through the functions stan_glm or stan_glmer, generating a StanFit object

```
## model equation: simple linear model
set.seed(123)
mod_equation <- y ~ x1 + x2
c <- 10
```

```
stan_fit_lr <- stan_glm(
  formula = mod_equation,
  data = data_example,
  family = "gaussian",
  prior = normal(0, c), # priors for betas
  prior_intercept = normal(0, c), # priors for intercept, we assume the same
  prior_aux = cauchy(0, 1)) # prior on our sigma
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054 seconds (Warm-up)
## Chain 1:                0.063 seconds (Sampling)
## Chain 1:                0.117 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.054 seconds (Warm-up)
## Chain 2:                0.063 seconds (Sampling)
## Chain 2:                0.117 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 se
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.053 seconds (Warm-up)
## Chain 3:                0.061 seconds (Sampling)
## Chain 3:                0.114 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 se
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.056 seconds (Warm-up)
## Chain 4:                0.063 seconds (Sampling)
## Chain 4:                0.119 seconds (Total)
## Chain 4:
```

Important remark 57. For `prior_aux` normal, `student_t` or cauchy, result in a half-normal, half-t, or half-Cauchy prior

9.2.5 5. Check the convergence of the algorithm

9.2.5.1 Traceplot

To check the convergence of the algorithm the first step is the visual analysis of the **traceplot**. For each parameter the iterations are plotted as a time series:

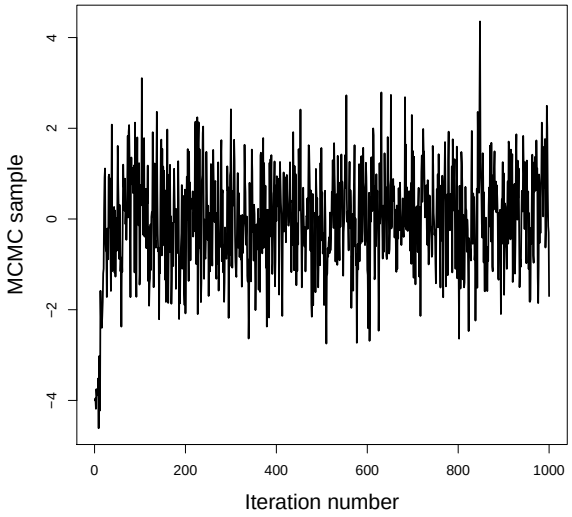
- the warm-up iterations should be included in order to check if the selected number is sufficient (parameter `inc_warmup = TRUE` in `rstanarm`);
- in case of convergence, the series must be regular and stationary and the different chain are overlapped. Ideally it should look like a caterpillar or bar code
- after convergence the samples should not converge to a single point!

To give some examples, in fig 9.1 some traceplot patterns:

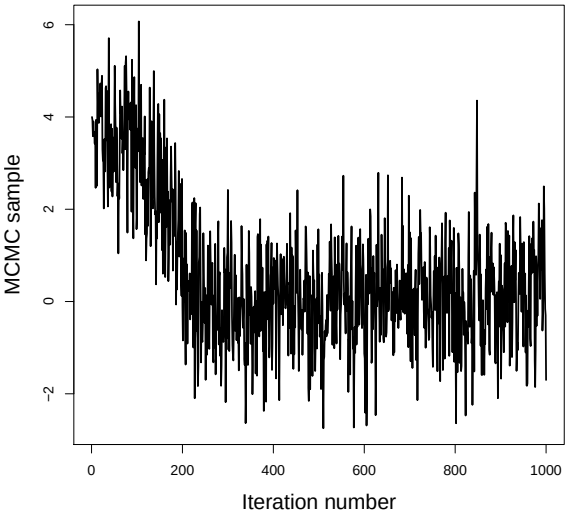
- case (a) is ideal
- case (b) is still ok, it means that it reaches convergence after a few iterations (which are throwed away)
- case (c) is not good: in case like this we need either to specify priors in a different way, change model assumptions, or trying to add more iterations to the chain (eg 2000, 3000, ...) and see if after more points it reaches convergence
- case (d): convergence is questionable and again a solution is to increase number of iterations and hope that the convergence is not questionable at the end and the stationary is more clear.

In our example we obtain figure 9.2: for each parameter we have the 4 chains overlapping, which is good and we get this kind of rectangular/stationary shape.

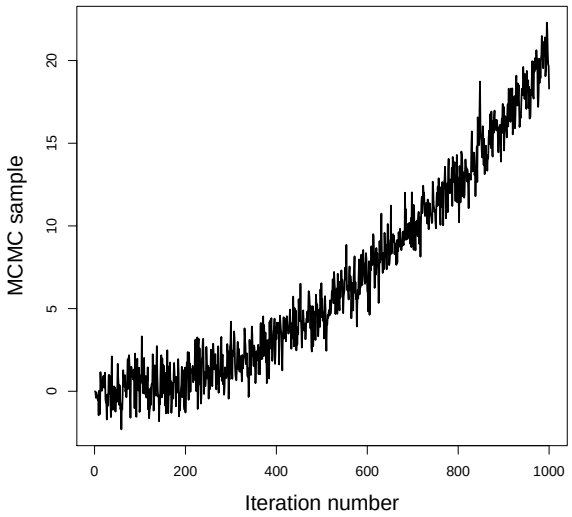
```
# traceplot
stan_trace(stan_fit_lr,
           inc_warmup = TRUE, # remember to specify this to include warmup period
           pars = c("sigma","(Intercept)", "x1", "x2"))
```



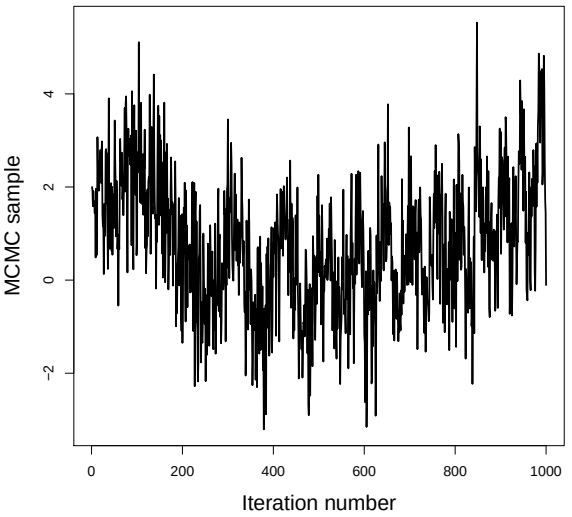
(a) Convergence in a few iterations



(b) Convergence in a few hundred iterations



(c) This one never converged



(d) Convergence is questionable

Figure 9.1: Example of possible traceplots (upper two are ok, lower two no)

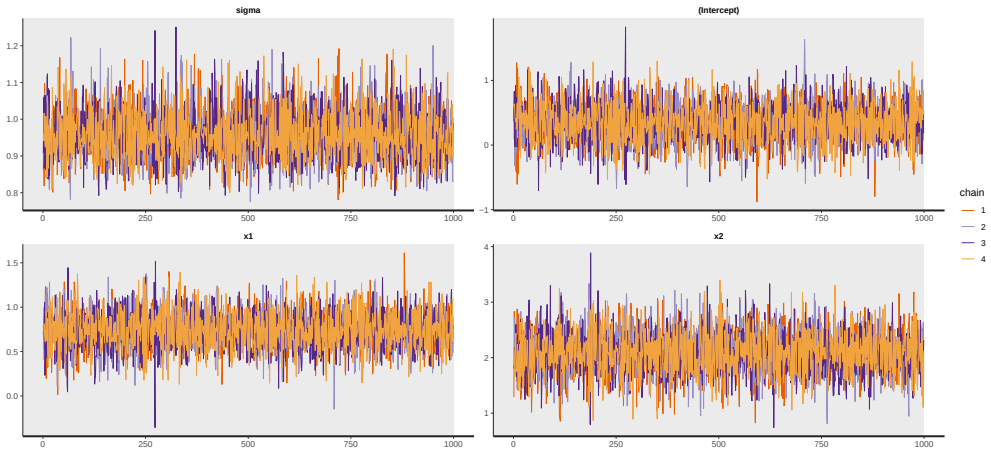


Figure 9.2: Traceplot for our example

9.2.5.2 Results overview

Other important indications (about convergence) can be deduced from the output of the `summary` command

```
# summary
summary(stan_fit_lr, digits = 3)

##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x1 + x2
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 100
##  predictors:   3
##
## Estimates:
##                mean   sd     10%    50%    90%
## (Intercept)  0.348  0.310 -0.037  0.356  0.736
## x1           0.739  0.213  0.470  0.740  1.007
## x2           2.090  0.392  1.605  2.080  2.593
## sigma        0.959  0.069  0.874  0.955  1.051
##
## Fit Diagnostics:
##            mean   sd    10%   50%   90%
## mean_PPD 2.110  0.138 1.934 2.112 2.290
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
```

```
## MCMC diagnostics
##               mcse  Rhat  n_eff
## (Intercept)   0.004 1.001 4806
## x1            0.003 1.000 4796
## x2            0.006 1.001 4830
## sigma         0.001 0.999 4367
## mean_PPD      0.002 1.001 4441
## log-posterior 0.037 1.001 1594
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

In this results we already mentioned the `mcse`, monte carlo standard error (square root of variance of the chain divided by total number of iteration we retain in our chain). What are `Rhat` and `neff`

9.2.5.3 $\hat{R}$ statistic

Remark 102. Adopting a multichain approach, the $\hat{R}$ statistic[1] allows to **check if the chains have reached the** same target distribution.
To define it we need to define a few different quantites first.

Definition 9.2.1 (Between-Sample Variance)**.** Considering M different chains (in stan 4 by default) with B realizations $\theta_m^{(b)}$ each, it is the variability of the means of different chains, defined as:

$$SV_B = \frac{B}{M-1}\sum_{m=1}^{M}\left[\bar{\theta}_m^{(\bullet)} - \bar{\theta}_\bullet^{(\bullet)}\right]^2$$

where $\bar{\theta}_m^{(\bullet)}$ is the mean realization of the m-th chain while $\bar{\theta}_\bullet^{(\bullet)}$ is the overall mean realizzation of all the M chains.

Definition 9.2.2 (Within-sample variance)**.** It is the variability within individual chains, defined as:

$$SV_W = \frac{1}{M(B-1)}\sum_{m=1}^{M}\sum_{b=1}^{B}\left[\theta_m^{(b)} - \bar{\theta}_m^{(\bullet)}\right]^2$$

where $\theta_m^{(b)}$ is a single extraction from the m-th chain, $\bar{\theta}_m^{(\bullet)}$ is again the mean realization of the m-th

Definition 9.2.3 (Pooled variance estimate)**.** It is an estimator of the average variance of samples, defined as:

$$\hat{V}^+[\theta] = \frac{B-1}{B}SV_W + \frac{1}{B}SV_B.$$

Remark 103. Non mi è chiarissimo perché ma comunque if chains have converged, between-chain and within-chain variability should be comparable).
Furthermore $\hat{V}^+[\theta]$ is an unbiased estimate of the variance in case of convergence, otherwise it overestimates it (*Intuition*:

[1] Known also as *Gelman and Rubin statistic* or *Potential Scale Reduction Factor*

Definition 9.2.4 ($\hat{R}$ statistic)**.** It is defined as the ratio between total and within variance

$$\hat{R} = \sqrt{\frac{\hat{V}^+[\theta]}{SV_W}}$$

with values interpreted as follows:

- $\hat{R} \approx 1$: all the chains reached the equilibrium distribution
- $\hat{R} > 1$: at least one chain did not converged (*RStan*: $\hat{R} > 1.05$)

Important remark 58. So **in our summary above** we have to check that for each parameter Rhat is < 1.05.

Important remark 59. The $\hat{R}$ statistic **assumes that the target distribution is approximately normal**; if plotting the posterior sample we see that we're far from the normal, using R $\hat{R}$ to check convergence may not be reliable (we should use other metrics in that case).
Normally 90% of the posterior is gaussian so this indicator is useful.

Remark 104. For models with numerous parameters (e.g., hierarchical models), $\hat{R}$ values for all parameters can be plotted to identify problematic parameters (using tools like `plotfun = "rhat"`).

9.2.5.4 $\hat{N}_{eff}$ (effective sample size))

As we've seen, in MCMC sampling the realizations (samples) are often correlated due to the Markov property (but the autocorrelation reduces if our sample for the posterior is high enough). To check if the sample we generated is effectively iid we can look at the **effective sample size**, which measures how many effectively independent samples are contained within the N total samples (considering the autocorrelation in the chains):

$$N_{eff} = \frac{N}{\sum_{t=-\infty}^{+\infty} \rho_t},$$

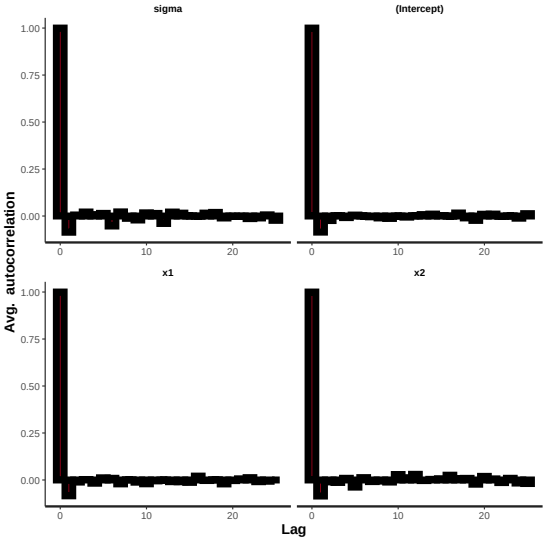
where ρ_t is the autocorrelation at lag t for a chain (at denominator the total correlation across all the lag of the chain).
Idea is that in case of no autocorrelation we have just correlation with the first lag, so the summation at the denominator should be =1. Thus if:

- $N_{eff} \approx N$: **low autocorrelation**, and the samples can be treated as nearly independent
- $N_{eff} \ll N$: **significant autocorrelation**, indicating that more samples may be required to achieve reliable estimates of the posterior distribution. Possible solutions: increase the number of iterations in the MCMC process; reevaluate the choice of priors or the sampling algorithm

9.2.5.5 Autocorrelation

Another way to check for the presence of autocorrelation is to plot the autocorrelation function with `stan_ac`.

```
# acf
stan_ac(stan_fit_lr,
        pars = c("sigma","(Intercept)", "x1", "x2"))
```



Regarding the graph:

- the intercept present an ideal situation
- sigma is still good (we have residual autocorrelation at lag 1 but after that autocorrelation is negligible)
- similar to sigma for x1 and x2

Important remark 60. However if we observe a **slow decay** in the autocorrelation (eg high correlation at lag 2/3 and become 0 say at lag 20 and so on) it is possible to *thin* the simulations (that is keep one simulation every T produced): if the thinning interval of T is chosen, then a simulation every T is considered and the others discarded.

9.2.5.6 Application to the example (comment)

```
# summary
summary(stan_fit_lr, digits = 3)

##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x1 + x2
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 100
```

```
##  predictors:   3
##
## Estimates:
##               mean   sd     10%    50%    90%
## (Intercept)  0.348  0.310 -0.037  0.356  0.736
## x1           0.739  0.213  0.470  0.740  1.007
## x2           2.090  0.392  1.605  2.080  2.593
## sigma        0.959  0.069  0.874  0.955  1.051
##
## Fit Diagnostics:
##            mean   sd    10%   50%   90%
## mean_PPD 2.110  0.138 1.934 2.112 2.290
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse  Rhat  n_eff
## (Intercept)   0.004 1.001 4806
## x1            0.003 1.000 4796
## x2            0.006 1.001 4830
## sigma         0.001 0.999 4367
## mean_PPD      0.002 1.001 4441
## log-posterior 0.037 1.001 1594
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

We can see that

- monte carlo standard error is really really close to 0, so ok
- rhat is almost $=1$ for all our parameters, so ok
- effective sample size must be compared with N which by default is 4000 (default length of the chain is 2000, half is discarded because of the warmup period, so $1000 \cdot 4$ numbero of chains is 4000). Here for all the parameters the effective sample size is close to 4000 except the log posterior (but this is totally normal for the log posterior, we'll see this later)

9.2.6 6. Fit diagnostics

Assured that the MCMC algorithm reached the correct sampling distribution, it is possible to perform the usual checks of the model assumptions (eg **residuals checks** like residuals vs fitted these are **still valid**).
However traditional/frequentist **statistical tests** (e.g.: t-tests, ANOVA) are **not valid**.
In Bayesian inference our *"best friend"* is the **Posterior Predictive Distribution (PPD)**[2] since it allows to evaluate the goodness of fit by comparing the *predicted values from the model* (based on the posterior distribution) *to the actual observed data.*

[2] *[Reference: Chapter 10 - Lambert, 2018]*

The idea is that if our model is able to generate samples close to the data we observe, it means that the model is a good fit. So we generate high number of new data using our posterior distribution, and check if the new data is similar or not to the data we used to fit the model.
Actually the **Bayesian p-value**, based on our simulated new data, assesses the model's ability to generate data that is similar to the observed data.

9.2.6.1 Posterior predictive distribution (PPD)

If $\mathbf{y}_{rep}$ is the replicated data generated from the model (using the posterior distribution) and $\mathbf{y}$ as the observed data, the PPD is defined as:

$$p(\mathbf{y}_{rep}|\mathbf{y}) = \int_{\Theta} p(\mathbf{y}_{rep}|\mathbf{y},\theta)p(\theta|\mathbf{y})\mathrm{d}\theta = \int_{\Theta} p(\mathbf{y}_{rep}|\theta)p(\theta|\mathbf{y})\mathrm{d}\theta.$$

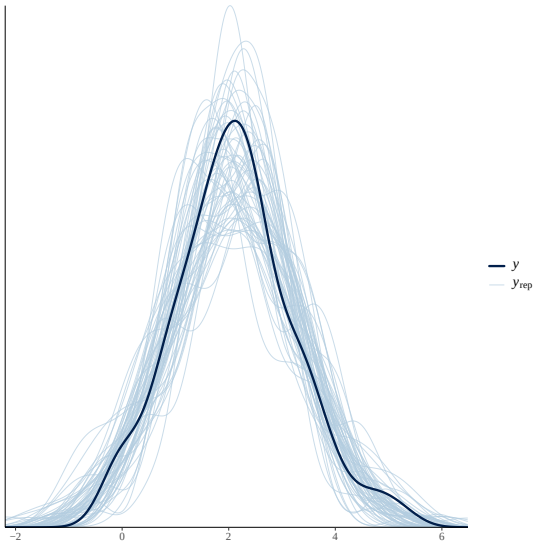
where:

- $p(\theta|\mathbf{y})$ is the posterior probability of a given value for θ (model parameters direi)
- $p(\mathbf{y}_{rep}|\theta)$ is the likelihood of the generated sample using the considered parameters
- The integral expresses the process of averaging the likelihood of the replicated data over all possible values of the parameters, weighted by how likely those parameter values are given the observed data;
- Even though it appears to be a complex integral, the PPD can be approximated easily using MonteCarlo approximation, that is simulating θ from the posterior and using those values to generate replicated data, thus bypassing the need for direct computation of the integral.
  In this way it is possible to **generate a new dataset** that replicates the observed one **at each Monte Carlo iteration**;
- By generating replicated datasets, you can assess how well your model captures the observed data. If the model fits well, the replicated data should be similar to the observed data. If there are large discrepancies between the two, this may suggest that the model is not suitable for the data;
- Essentially, if the model cannot produce data similar to the observed data, it may be necessary to reconsider the model (e.g.: by changing priors, adding or removing variables, or choosing a different model altogether)

Important remark 61. In `rstanarm`, after a model is fitted it is possible to generate new samples from its posterior predictive distribution using `posterior_predict`.
We check visually **empirical density functions** of real data and the empirical distributions of the replicated datasets with the following commands

```
## ----------------------------------------
## generate posterior predictive distribution
## ----------------------------------------
post_pred <- posterior_predict(stan_fit_lr) #?posterior_predict
class(post_pred)
```

```
## [1] "matrix" "array"

dim(post_pred) ## matrix with 4000 generated samples for y,

## [1] 4000  100

               ## each of length 100 (same as y)
head(post_pred[, 1:10]) # first 10 numbers of the first 6 sims

##             1         2        3        4         5         6         7
## [1,] 1.019686 1.8682413 2.727317 1.679591 2.0550877 1.7580900 0.7355346
## [2,] 2.535817 1.8138072 3.827659 1.105827 0.0086396 0.8213476 2.7714320
## [3,] 1.134838 1.8247449 3.533477 2.626967 1.4975982 2.4652220 0.6544159
## [4,] 0.891434 1.2961062 3.888361 3.777786 2.8637904 1.0714002 3.2714705
## [5,] 2.538698 2.2100029 1.711704 1.894188 0.8701231 2.4494499 0.9561317
## [6,] 1.062651 0.4805881 3.129139 1.632252 1.2213187 5.1854567 2.6846128
##               8          9       10
## [1,]  2.6244060  1.9827160 1.812439
## [2,]  0.8503736  1.5343766 2.376168
## [3,]  0.1172433  3.7688753 2.442978
## [4,] -0.5607208  1.5706244 3.198787
## [5,]  2.3389307 -0.4643729 2.901331
## [6,]  0.0279711 -0.5201504 1.998854

## Each row of the matrix is a vector of predictions generated using a single
## draw of the model parameters from the posterior distribution.


## ----------------------------------------
## plot empirical density of generated sample (light blue lines) against the
## observed data (single dark blue line).
## ----------------------------------------

## here we plot just a subset of the 4000 simulations
bayesplot::ppc_dens_overlay(y = y,           #?bayesplot::ppc_dens_overlay
                            yrep = post_pred[40:100,])
```




```
## ## altro test malato mio, it takes forever
## bayesplot::ppc_dens_overlay(y = y,           #?bayesplot::ppc_dens_overlay
##                             yrep = post_pred,
##                             alpha = 0.001)
```

The situation after this should be like the following; the light blue lines simulated from the model are reasonably overlapping with the data (darker blue line); for same sample the simulated densities is not so strictly equal but overall most of the sample we generated are overlapping

9.2.6.2 Posterior predictive checks (Bayesian p-value)

Aside from graphical comparison, we can adopt measures to check consistency between sample and model generated data, that is the **bayesian p-value**.

Definition 9.2.5 (Discrepancy measure)**.** In general it is a function $D(y, \theta)$ that evaluates the "distance" between data and model predictions, conditioned on the parameters (θ).

Remark 105. Some remarks:

- discrepancy measures can be used to assess specific aspects of the model;
- while classical test statistics depend only on the data, discrepancy measures in Bayesian PPCs depend on both y and the model parameters θ;
- as discrepancy measure one can choose any statistic based on data and the parameter.
  One example of discrepancy measure we can use is the expected value given the parameter

Definition 9.2.6 (Bayesian p-value (posterior predictive p-value))**.** Generalizes the classical p-value and is defined as the probability that the discrepancy

measure of the model generated data is higher than the one on the original data (given the orginal data):

$$\begin{aligned} p_B &= \mathbb{P}\left[D(\mathbf{y}_{rep}, \theta) \geq D(\mathbf{y}, \theta)|\mathbf{y}\right] \\ &= \int\int \mathbb{1}_{\{D(\mathbf{y}_{rep},\theta)\geq D(\mathbf{y},\theta)\}} p(\mathbf{y}_{rep}|\theta)p(\theta|\mathbf{y})\mathrm{d}\mathbf{y}_{rep}\mathrm{d}\theta \end{aligned}$$

Remark 106. Thus adopting as distance measure the expected value given the parameter, we can compute the bayesian p-value as probability that the expected value of the model generated sample is $\geq$ to the expected value of original data (given θ).

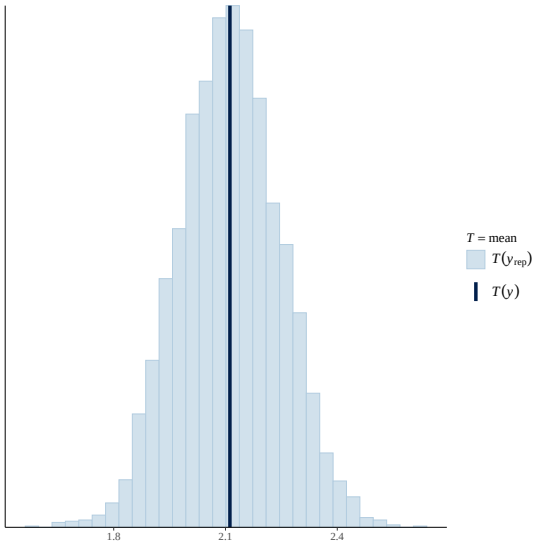
Remark 107. Some remarks:

- the p-value should not be interpreted like the frequentist one:
  - p_B is a probability of model mis-fitness; **values near 0 or 1 indicate a discrepancy between the model and the data**, while $p_B \approx 0.5$ is the ideal.
  - here we do not condition on the parameter value θ and, instead, allow for its variation in accordance with the posterior distribution.
- The integral defining the bayesian p-value cannot be solved analytically but using MCMC sampling. In the k-th **MCMC** step:
  1) A **sample of model parameters** $\theta^*_{(k)}$ is drawn from the posterior $p(\theta|\mathbf{y})$
  2) Using $\theta^*_{(k)}$, **simulate replicated data** $\mathbf{y}_{rep}$ from the PPD $p(y_{rep}|\theta^*_{(k)})$
  3) The **discrepancy measures** $D(\mathbf{y}_{rep,(k)}, \theta^*_{(k)})$ and $D(\mathbf{y}, \theta^*_{(k)})$ are computed and compared.

  Then, the Bayesian p-value is estimated through the proportion of simulations in which $D(\mathbf{y}_{rep,(k)}, \theta^*_{(k)}) \geq D(\mathbf{y}, \theta^*_{(k)})$.

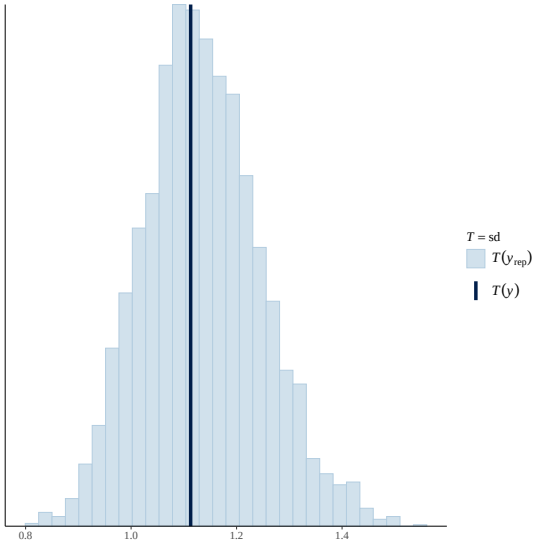
Example 9.2.1. A visual check is usually enough and it could be obtained with the function `ppc_stat`. This plots the distribution of the discrepancy measure in the model generated samples, as well as the discrepancy measure of the original data. Ideally we should see that the value of the discrepancy measure in the sample is equal to the median of the distribution (splits the distribution in two equivalent areas)

```
# we see two statistics
ppc_stat(y = y, yrep = post_pred, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```




```
ppc_stat(y = y, yrep = post_pred, stat = "sd")
```

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



In a similar way we can compare in the `summary`, the mean of the predictive posterior distribution with the mean of our original data the lines `mean_ppd` under `Fit Diagnostics` gives the mean sd and quantiles of mean of model generated data.
Then being mean PPD actually a distribution we can check the monte carlo standard error (Rhat and n_eff) looking at `mean_ppd` line under `MCMC diagnostics`

```
summary(stan_fit_lr)

##
```

```
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x1 + x2
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 100
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) 0.3    0.3  0.0   0.4   0.7
## x1          0.7    0.2  0.5   0.7   1.0
## x2          2.1    0.4  1.6   2.1   2.6
## sigma       1.0    0.1  0.9   1.0   1.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 2.1    0.1  1.9   2.1   2.3
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  4806
## x1            0.0  1.0  4796
## x2            0.0  1.0  4830
## sigma         0.0  1.0  4367
## mean_PPD      0.0  1.0  4441
## log-posterior 0.0  1.0  1594
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv

mean(y) # compare mean_PPD under fit diagnostic with this. They are very similar here

## [1] 2.111895
```

9.2.6.3 Log-posterior

We've seen more or less all the summary components but the log-posterior (`log-posterior` row under `MCMC diagnostics`).

Definition 9.2.7 (Log-posterior)**.** It's just the logarithm of the posterior density evaluated at each MCMC sample of the parameters.

Remark 108. It:

- is a measure on how well the sampled parameter values fit the observed data and align with the prior distributions;

- helps to monitor whether the chain is exploring the parameter space properly; if the log-posterior stabilizes during sampling, it suggests that the chain has converged.
- has typically a *lower effective sample size* than other parameters because it is a summary of all parameters and can exhibit more autocorrelation.
- is not directly interpretable (since it's relative), but **comparing log-posterior** values across models can help identify which model better explains the data (higher values suggest a better fit), however, especially in this case, it:
  - is **sensitive to the priors**: if one uses different priors across models, the comparison may reflect differences in priors rather than the models themselves;
  - **does not penalize for model complexity**, so more complex models may always have a higher value. For penalized comparison, use measures like WAIC.

9.2.6.4 Information criterion: WAIC

Remark 109. In Bayesian model comparison and assessment, traditional information criteria like AIC and BIC are not directly applicable. Deviance based methods are used instead.
The **W**idely **A**pplicable **I**nformation **C**riteria is an estimate of the expected log density of a new dataset and it is composed by two quantities (which resemble the components/ideas of the classical AIC)

Definition 9.2.8 (Log Pointwise Predictive Density)**.** A measure aimed at summarizing the predictive accuracy of the fitted model to the observe data, defined as:

$$\widehat{lppd} = \sum_{i=1}^{n} \log \left(\frac{1}{K} \sum_{k=1}^{K} p(y_i|\theta^*_{(k)}) \right).$$

It averages the likelihood of each data point across all posterior samples, and then sums the log of these average likelihood

Definition 9.2.9. Effective Number of Parameters A penalization of overfitting by estimating the effective number of parameters based on the posterior variance of the predictive density of each data point:

$$p_{WAIC} = \sum_{i=1}^{n} \left(\hat{\mathbb{V}} \left[\log p(y_i|\theta^*_{(\cdot)}) \right] \right).$$

The idea here is if the loglikelihood of a data point varies greatly across the posterior samples it implies more model complexity and so an higher number of parameters

Definition 9.2.10. WAIC Finally, defined as:

$$WAIC = -2\Big(\widehat{lppd} - p_{WAIC}\Big) = -2(\texttt{elpd\_waic})$$

In practice, it is useful to compare two or more models. As with AIC, the model with **the lowest WAIC value is considered the best fit for the data**, balancing accuracy and complexity.
Here `elpd_waic` means "Expected Log Predictive Density for WAIC".

Example 9.2.2. WAIC can be easily computed using the function `loo::waic`

```
# WAIC
loo::waic(stan_fit_lr)

##
## Computed from 4000 by 100 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic   -139.4  6.4
## p_waic         3.9  0.6
## waic         278.9 12.8

# >> 1 (1.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

# This indicates that 1 out of our data points (1.0% of the total)
# has an effective number of parameters (p_WAIC) exceeding the threshold of 0.4.

# High values of p_WAIC (relative to the threshold) suggest that
# certain data points are strongly influencing the model, potentially making
# the WAIC estimate less stable.

# The recommendation suggests using LOO (Leave-One-Out cross-validation)
# instead of WAIC for model comparison or evaluation, which is typically more
# robust than WAIC. However, this warning does not invalidate the model, but
# rather highlights potential issues with using WAIC as a criterion for model
# evaluation.

loo::loo(stan_fit_lr)

##
## Computed from 4000 by 100 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -139.5  6.4
## p_loo         3.9  0.6
## looic       278.9 12.8
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.7, 1.3]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

9.2.7 7. Inference: summarize the posterior distribution

Remark 110. After all the checks, if everything was fine, we have to summarize results; it is possible to derive easily the Bayes estimators (both point and intervals) of the quantity of interest.

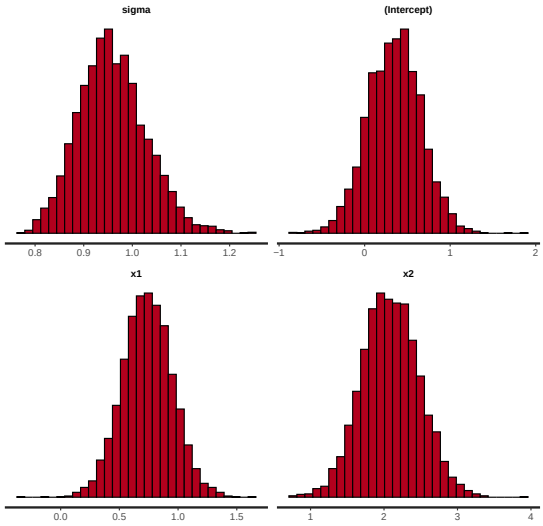
Example 9.2.3. With `summary`

- we have several metrics of the posterior distributions (eg mean, std, qunatiles)
- we can say build an 80% credibility interval for the parameters looking at 10 and 90th quantile

```
params = c("sigma","(Intercept)", "x1", "x2")
summary(stan_fit_lr, pars = params)

##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x1 + x2
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 100
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) 0.3    0.3  0.0   0.4   0.7
## x1          0.7    0.2  0.5   0.7   1.0
## x2          2.1    0.4  1.6   2.1   2.6
## sigma       1.0    0.1  0.9   1.0   1.1
##
## MCMC diagnostics
##             mcse Rhat n_eff
## (Intercept) 0.0  1.0  4806
## x1          0.0  1.0  4796
## x2          0.0  1.0  4830
## sigma       0.0  1.0  4367
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

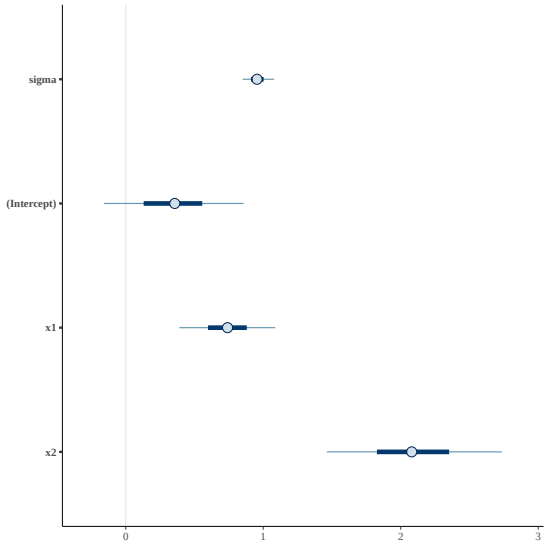
We can plot the distributions themselves with `stan_hist`:

```
# hist: questi dovrebbero essere l'istogramma di estrazioni dalle posterior dei param
stan_hist(stan_fit_lr, bins = 30, pars = params)
```



Finally can plot credibility intervals (darker blue) and range (thin blue), using `plot`

```
# post estimates: stime intervallari derivanti e rappresentazione grafica
plot(stan_fit_lr, pars = params)
```



Remark 111. And this is what one needs to do for a basic bayesian inference analysis.

Chapter 10

Exercise 1: normal linear regression

Remark 112. Starting from here, we see examples one could get at the exam; for the practical part of the exam one is required to perform all the steps seen before.
At the exam **make some short comments about the output** of code (eg MC standard error, Rhat, effective sample size: not required to explain what are but if we check something say it if it's ok or not and why).

```
# load packages
library(rstanarm)
library(loo)
library(bayesplot)
library(rstan)
library(ggplot2)
set.seed(1234)
```

Suppose we have the following situation

- dataset of $n = 25$ vending machines
- response variable: *recharge time*
- explanatory variables: *product amount* and *distance covered by the operator*.
- Normal Linear Regression Model is assumed

The exercise requires to:

1) Write the theoretical form of the regression model.
2) Program the model using the default priors of `rstanarm`.
3) Program a second model specifying a flat prior for the model coefficients.
4) Program a third model specifying a $\mathcal{N}(0, 100)$ prior for the regression coefficients and the intercept, and a $Half - Cauchy(0, 1)$ for σ.

5) Referring to the the model specified in step 3), monitor the convergence of the algorithm. In particular, discuss the interpretation of $\hat{R}$.

NB: In case of item 5 we need to explain what Rhat is, formula included

6) Generate from the posterior predictive distribution and implement the posterior predictive checks.

7) Carry out posterior inference and report the 90% credibility interval of the posterior distribution of observation number 10. In particular, monitor the statistic R^2_B (which is Bayesian version of R^2)

$$R^2_B = 1 - \frac{\sigma^2}{s^2(y)}.$$

10.1 1) Write the theoretical form of the model

The Normal Linear Regression Model is assumed and it has the following **likelihood**:

$$\begin{aligned} &y_i|\mu_i, \sigma^2 \sim \mathcal{N}(\mu_i, \sigma^2), \\ &\mu_i|\boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}, \quad i = 1, \dots, 25. \end{aligned}$$

for all the observation, where:

- $\mathbf{Y}$ = recharge time
- $\mathbf{X}_1$ = product amount
- $\mathbf{X}_2$ = distance covered by the operator

10.2 2) Model with default priors

How to implement this in `rstanarm`; default priors

- means we don't need to specify anything in `stan_glm`;
- with very few exceptions, the **default priors** in `rstanarm` are **not flat priors**, but **weakly informative**, that is are designed to provide moderate regularization and help stabilize computation;
- for many (if not most) applications the defaults will perform well, but this is not guaranteed (no default priors make sense for every possible model specification);
- the way `rstanarm` attempts to make priors weakly informative by default is to internally adjust the scales of the priors.

NB: If during the exam you are required to fit a model with weakly informative prior, you don't have to specify anything (Stan does it by default)

NB: If we want to specify an informative prior we must specify why that one in particular (we should have strong knowledge about the parameter distribution)

```
getwd()

## [1] "/home/l/.sintesi/sintesi_math/bayesian_inference"

(data1 <- read.csv("data/Data_Ex_1.csv"))
```

```
##     time amount distance
## 1  16.68      7      560
## 2  11.50      3      220
## 3  12.03      3      340
## 4  14.88      4       80
## 5  13.75      6      150
## 6  18.11      7      330
## 7   8.00      2      110
## 8  17.83      7      210
## 9  79.24     30     1460
## 10 21.50      5      605
## 11 40.33     16      688
## 12 21.00     10      215
## 13 13.50      4      255
## 14 19.75      6      462
## 15 24.00      9      448
## 16 29.00     10      776
## 17 15.35      6      200
## 18 19.00      7      132
## 19  9.50      3       36
## 20 35.10     17      770
## 21 17.90     10      140
## 22 52.32     26      810
## 23 18.75      9      450
## 24 19.83      8      635
## 25 10.75      4      150
```

```
if (FALSE) data1 <- read.csv("bayesian_inference/data/Data_Ex_1.csv")

#overview
str(data1)
```

```
## 'data.frame': 25 obs. of  3 variables:
##  $ time    : num  16.7 11.5 12 14.9 13.8 ...
##  $ amount  : int  7 3 3 4 6 7 2 7 30 5 ...
##  $ distance: int  560 220 340 80 150 330 110 210 1460 605 ...
```

```
# pairs(data1)

formula_ex1 <- time ~ amount + distance # per comodità

# frequentist equivalent model
summary(lm(formula_ex1, data = data1))
```

```
##
## Call:
## lm(formula = formula_ex1, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -5.7880 -0.6629  0.4364  1.1566  7.4197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.341231   1.096730   2.135 0.044170 *
## amount      1.615907   0.170735   9.464 3.25e-09 ***
## distance    0.014385   0.003613   3.981 0.000631 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 22 degrees of freedom
## Multiple R-squared:  0.9596, Adjusted R-squared:  0.9559
## F-statistic: 261.2 on 2 and 22 DF,  p-value: 4.687e-16
```

```
## default weakly informative priors
set.seed(1234)
mod_ex11 <- stan_glm(formula = formula_ex1,
                     data = data1,
                     family = "gaussian")
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.086 seconds (Warm-up)
## Chain 1:                0.073 seconds (Sampling)
## Chain 1:                0.159 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 se
```

```
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.079 seconds (Warm-up)
## Chain 2:                0.087 seconds (Sampling)
## Chain 2:                0.166 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.084 seconds (Warm-up)
## Chain 3:                0.074 seconds (Sampling)
## Chain 3:                0.158 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
```

```
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.077 seconds (Warm-up)
## Chain 4:                0.077 seconds (Sampling)
## Chain 4:                0.154 seconds (Total)
## Chain 4:
```

```
## command to extract/know the priors of a model
prior_summary(mod_ex11)
```

```
## Priors for model 'mod_ex11'
## ------
## Intercept (after predictors centered)
##   Specified prior:
##     ~ normal(location = 22, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 22, scale = 39)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = [0,0], scale = [2.5,2.5])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [5.64,0.12])
##
## Auxiliary (sigma)
##   Specified prior:
##     ~ exponential(rate = 1)
##   Adjusted prior:
##     ~ exponential(rate = 0.064)
## ------
## See help('prior_summary.stanreg') for more details
```

The last command show the prior used by a model:

- `specified prior` is the starting prior used to fit the model and then

adjusted

- `adjusted prior` is an adjustment that rstanarm does to better fit the data: the actual prior used by the algorithm is the adjusted one.

Eg for the intercept and coefficients we have normal, while for sigma is an exponential.

10.3 3) Model with a flat prior

`rstanarm` will use flat priors if `prior=NULL` is specified. Some remarks about these priors

- when "**non-informative**" or "uninformative" is used in the context of prior distributions, it typically refers to a **flat (uniform) distribution** or a nearly flat distribution. Sometimes it may also be used to refer to the parameterization-invariant Jeffreys priors.
- unless the data is very strong (and all the inference we want to be extracted from them) it is generally wise to avoid them

NB: If we are asked to set non-informative/uninformative/flat priors we set `prior=NULL`

```
set.seed(1234)
mod_ex12 <- stan_glm(formula = formula_ex1,
                     data = data1,
                     family = "gaussian",
                     prior = NULL)  # flat/non-informative prior

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.203 seconds (Warm-up)
## Chain 1:                0.065 seconds (Sampling)
```

```
## Chain 1:                0.268 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.163 seconds (Warm-up)
## Chain 2:                0.07 seconds (Sampling)
## Chain 2:                0.233 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.241 seconds (Warm-up)
```

```
## Chain 3:                0.069 seconds (Sampling)
## Chain 3:                0.31 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.218 seconds (Warm-up)
## Chain 4:                0.07 seconds (Sampling)
## Chain 4:                0.288 seconds (Total)
## Chain 4:
```

```
prior_summary(mod_ex12)
```

```
## Priors for model 'mod_ex12'
## ------
## Intercept (after predictors centered)
##   Specified prior:
##     ~ normal(location = 22, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 22, scale = 39)
##
## Coefficients
##  ~ flat
##
## Auxiliary (sigma)
##   Specified prior:
##     ~ exponential(rate = 1)
##   Adjusted prior:
##     ~ exponential(rate = 0.064)
## ------
## See help('prior_summary.stanreg') for more details
```

Here actually flatness regards only the coefficients.

10.4 4) Model with informative priors

In this case we have to manually specify the priors. The Likelihood is still:

$$y_i|\mu_i, \sigma^2 \sim \mathcal{N}(\mu_i, \sigma^2),$$
$$\mu_i|\boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}, \quad i = 1, \ldots, 25.$$

while for the priors we set:

$$\beta_k \sim \mathcal{N}(0, 100) \quad k = 1, 2$$
$$\beta_0 \sim \mathcal{N}(0, 100)$$
$$\sigma \sim Half - Cauchy(0, 1)$$

```
set.seed(1234)
mod_ex13 <- stan_glm(formula = formula_ex1,
                     data = data1,
                     family = "gaussian",
                     prior = normal(0, 100),
                     prior_intercept = normal(0, 100),
                     prior_aux = cauchy(0, 1))

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 se
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 4.239 seconds (Warm-up)
## Chain 1:                1.37 seconds (Sampling)
## Chain 1:                5.609 seconds (Total)
```

```
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.366 seconds (Warm-up)
## Chain 2:                1.642 seconds (Sampling)
## Chain 2:                4.008 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.689 seconds (Warm-up)
## Chain 3:                1.736 seconds (Sampling)
```

```
## Chain 3:                3.425 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 4.019 seconds (Warm-up)
## Chain 4:                1.508 seconds (Sampling)
## Chain 4:                5.527 seconds (Total)
## Chain 4:
```

```
prior_summary(mod_ex13)
```

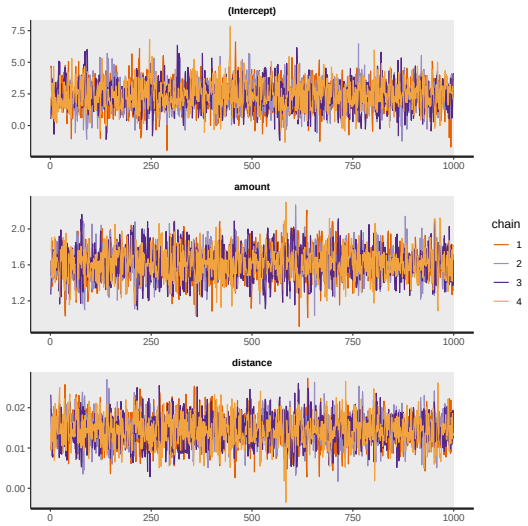
```
## Priors for model 'mod_ex13'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 100)
##
## Coefficients
##  ~ normal(location = [0,0], scale = [100,100])
##
## Auxiliary (sigma)
##  ~ half-cauchy(location = 0, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

Important remark 62. Note that if we specify `cauchy` in rstanarm by default we fit an half-Cauchy; the same happens for `t` distribution (if we specify `t`, rstanarm use an half T).
This because we're setting a prior on σ but the parameter is actually σ^2 so basically a positive or negative σ will give the same σ^2: so in order to be computationally efficient we use half-distributions, eg half-cauchy is a cauchy where one consider only positive values.

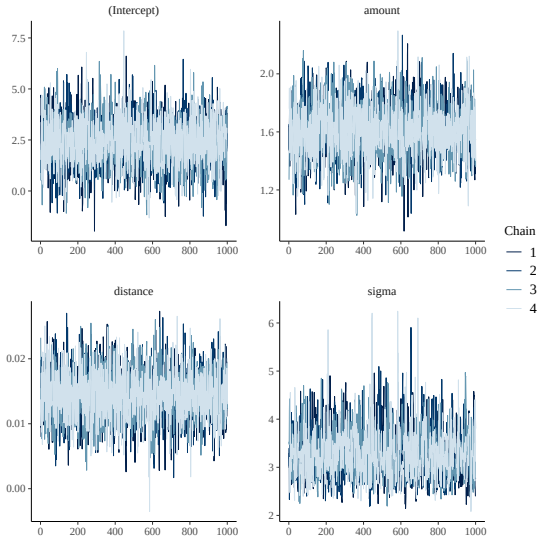
10.5 5) Convergence of the algorithm

Starting with the **graphical stuff**

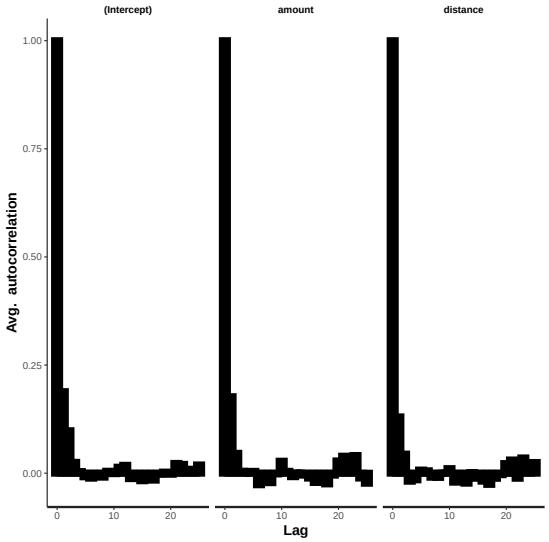
```
##  traceplot: the traceplot are for $\beta_0,\beta_1,\beta_2$ respectively,
## four chains each by default: all chains converged (for all the parameters),
## trends are horizontal with barcode shape
stan_trace(mod_ex13, nrow = 3, ncol = 1, inc_warmup = T)
```



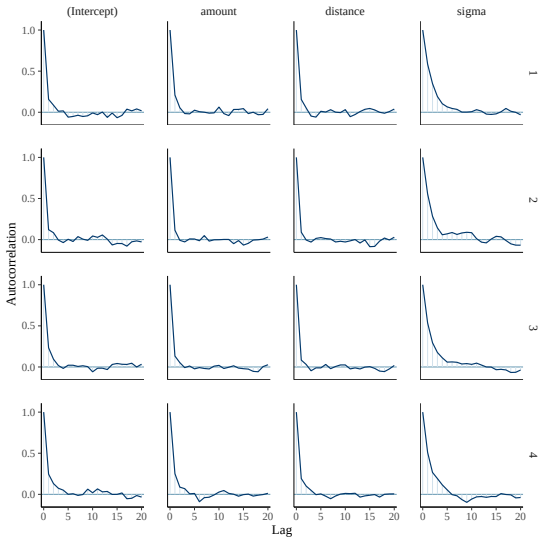

```
## another function to plot the traceplot is the following. Here we have the
## traceplot for sigma as well
plot(mod_ex13, plotfun = "trace")
```



```
## Check autocorrelation functions: here there is some residual autocorrelation
## in the first lags but it eventually goes to 0, so no big problems here
stan_ac(mod_ex13)
```




```
## another autocorrelation plot can be done using. Here we have the
## autocorrelation functions for all the four chains and for all the parameters
## and again there seems not to be big issues
plot(mod_ex13, plotfun = "ac")
```



Going with the **numerical stuff**, remembering that in `summary` `mean_PPD` is the **sample average posterior predictive distribution of the outcome**, a useful heuristic is to check if `mean_PPD` is plausible when compared to `mean(y)`. If it is plausible then this does not mean that the model is good in general (only that it can reproduce the sample mean), however if `mean_PPD` is implausible then

it is a sign that something is wrong (severe model misspecification, problems with the data, computational issues, etc.).

```
## focusing on the fit diagnostic of the following summary and the posterior
## predictive distribution

summary(mod_ex13)

##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      time ~ amount + distance
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 25
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) 2.4    1.1  0.9   2.3   3.8
## amount      1.6    0.2  1.4   1.6   1.8
## distance    0.0    0.0  0.0   0.0   0.0
## sigma       3.3    0.5  2.7   3.2   4.0
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 22.4    0.9 21.2  22.4  23.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  2443
## amount        0.0  1.0  2716
## distance      0.0  1.0  3025
## sigma         0.0  1.0  1106
## mean_PPD      0.0  1.0  1938
## log-posterior 0.0  1.0  1122
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv

mean(data1$time)

## [1] 22.384
```

so here we have 22.4 vs 22.38 which is good: the posterior predictive distribution can generate data data are similar to our original variable/sample.
Then looking at **MCMC diagnostics** above:

- the monte carlo standard error is basically 0 which is very good
- Rhat=1 which is perfect
- effective sample size are **quite low** if compared to the default N fitted by `rstanarm` which is 4000. So in this case, possible solution is to increase the number of iteration of the MCMC algorithm and see if the effective sample size increases

NB: If you are asked to discuss the interpretation of a particular metric (e.g.: $\hat{R}$), it means that you should provide also a brief theoretical explanation of the metric itself.

```
## # a graphical alternative plotting of Rhat which highlights that it should
## be below 1.05
## plot(mod_ex13, plotfun="rhat")
```

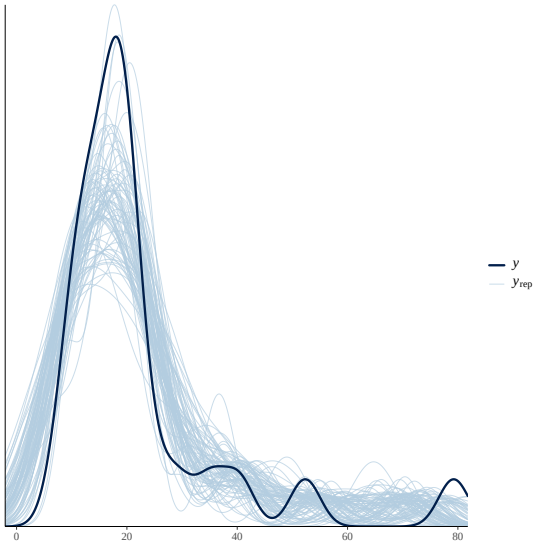
10.6 6) Posterior predictive checks

Then we **generate** 4000 new data sets (one for each row) of size 25 (as the starting dataset) from the **posterior predictive distribution**:

```
y_tilde <- posterior_predict(mod_ex13)
str(y_tilde)

##  num [1:4000, 1:25] 19.9 24.7 24 14.7 23.1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:25] "1" "2" "3" "4" ...
```

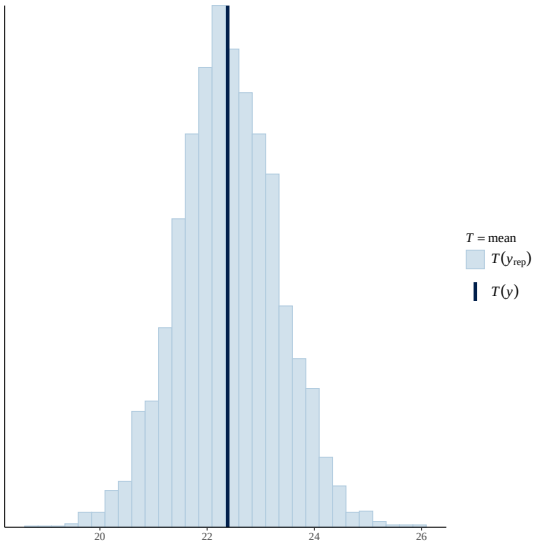
To **compare densities** of generated data and sample one, we see that for some simulation the overlap is not high but overall there seems not a too bad fit (it could be better btw)

```
# Densities comparison
ppc_dens_overlay(y = data1$time, yrep = y_tilde[1000:1080,])
```

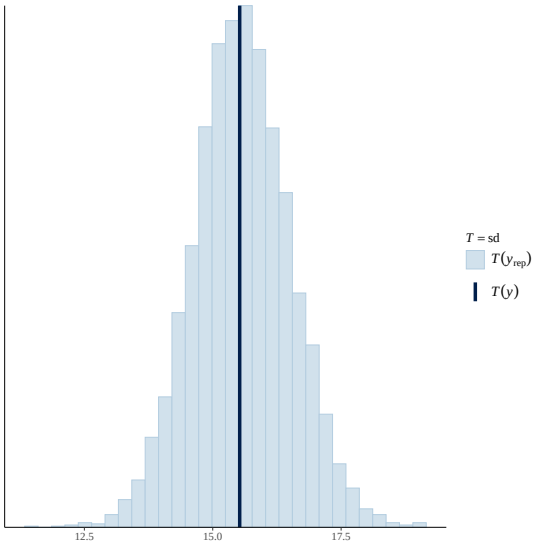


Finally, we perform some **posterior predictive checks**, lets see

```
#Posterior predictive checks
ppc_stat(y = data1$time, yrep = y_tilde, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```

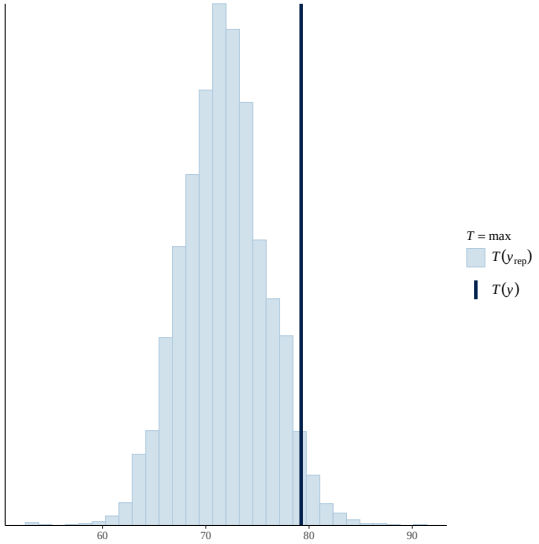



```
ppc_stat(y = data1$time, yrep = y_tilde, stat = "sd")

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



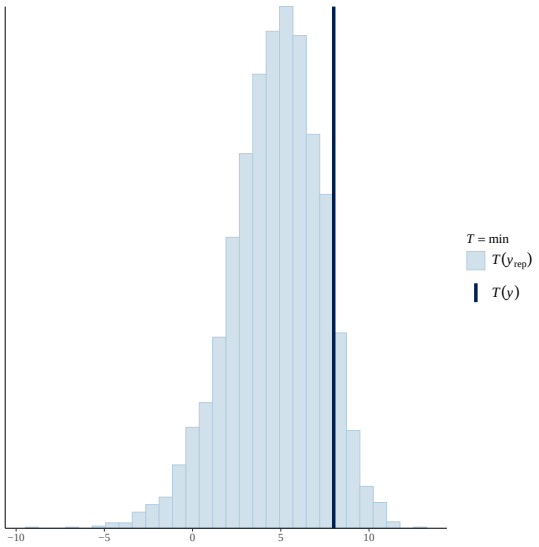
```
ppc_stat(y = data1$time, yrep = y_tilde, stat = "max")
```

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




```
ppc_stat(y = data1$time, yrep = y_tilde, stat = "min")
```

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



we see mean and sd are rightly centered on simulated data, so eg our model is reasonably good to study the mean (or the sd); however if we're interested in the minimum time or the maximum time we see that model generate min and max that are slightly underestimated with respect to the sample one, this model is not the right one. So if we're interested in the min or max time we have to change something in the model, such as update the prior or change totally the model we use.

10.7 7) Posterior inference

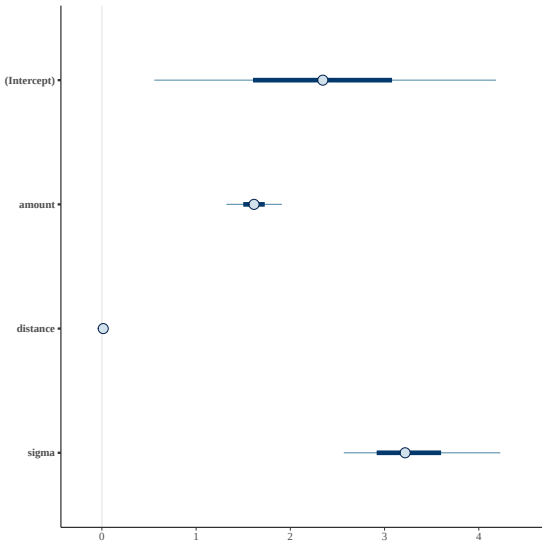
Finally for posterior inference we look at `estimates` table in `summary` (and build 80% credibility interval out of the box using the quantiles)

```
summary(mod_ex13, digits = 4)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      time ~ amount + distance
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 25
##  predictors:   3
##
## Estimates:
##               mean   sd     10%    50%    90%
## (Intercept) 2.3587 1.1302 0.9319 2.3445 3.7877
## amount      1.6148 0.1768 1.3910 1.6148 1.8395
## distance    0.0144 0.0037 0.0097 0.0143 0.0191
## sigma       3.2887 0.5157 2.6817 3.2184 3.9535
##
## Fit Diagnostics:
##            mean    sd      10%     50%     90%
## mean_PPD 22.4042  0.9410 21.2153 22.4135 23.6043
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse   Rhat   n_eff
## (Intercept)   0.0229 1.0013 2443
## amount        0.0034 1.0014 2716
## distance      0.0001 1.0023 3025
## sigma         0.0155 1.0037 1106
## mean_PPD      0.0214 1.0015 1938
## log-posterior 0.0460 1.0002 1122
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

plot of credibility intervals

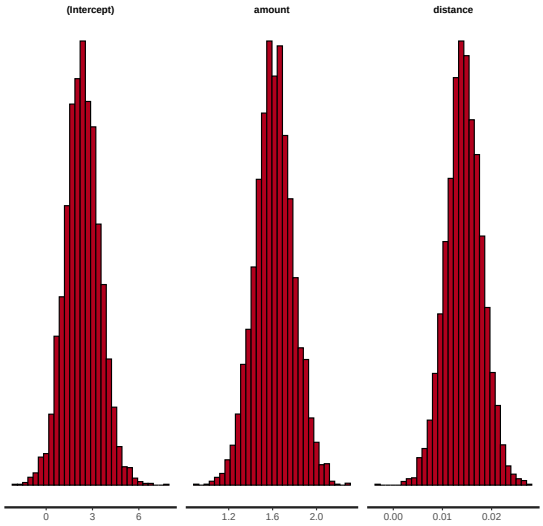
```
plot(mod_ex13)
```



plot of posterior distribution of parameters all more or less gaussian (So in this case we can use Rhat to check if the convergence of the MCMC algorithm)

```
stan_hist(mod_ex13)
```

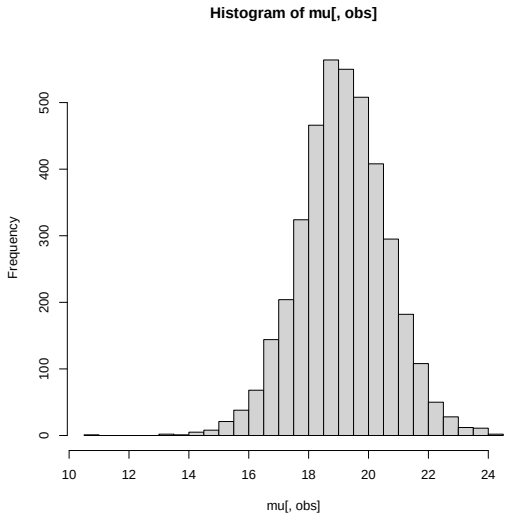
```
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



Finally in the last question we're asked to report the 90% credibility interval for a specific observation; we add some standard inference as well (hist, mean, sd and credibility interval for a single)

```
## we start by obtaining simulations of the linear predictor for all the units
## in the sample
mu <- posterior_linpred(mod_ex13)
dim(mu) # 4000 simulations per 25 units
```

```
## [1] 4000   25

## posterior distribution of the observation 10, we just select the obtained
## matrix and make the stats as usual
obs = 10
hist(mu[, obs], breaks = 30)
```




```
mean(mu[, obs])

## [1] 19.14648

sd(mu[, obs])

## [1] 1.472344

quantile(mu[, obs], probs = c(0.05, 0.5, 0.95))

##       5%      50%      95%
## 16.73160 19.13991 21.52548
```

TODO: spostarlo da qualche parte nella teoria o è gia presente??

Last point we are asked to monitor R^2_B (the bayesian R^2) making posterior inference. For linear regression is defined as

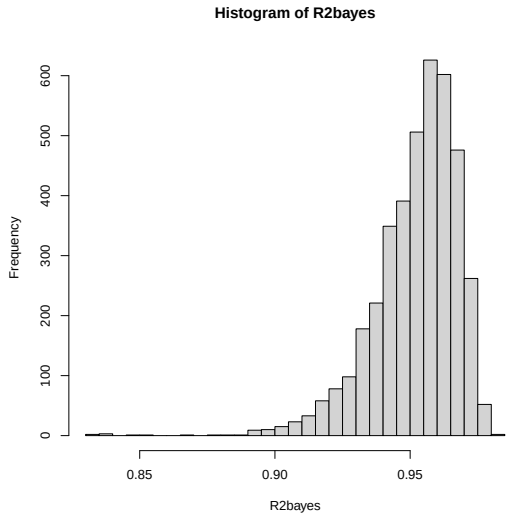
$$R^2_B = 1 - \frac{\sigma^2}{s^2(y)}.$$

and it:

- is the analog to the frequentist R^2 (coefficient of determination) but is adapted for Bayesian analysis to account for uncertainty in the model parameters;
- R^2_B close to 1 indicates that the model explains most of the variation in the data, while values close to 0 indicate poor explanatory power;

- can be used to compare different Bayesian models, where a higher R_B^2 suggests better explanatory power;
- is not a single value but *a distribution*

```
## First we wxtract the posterior sample of interest (sigma parameter)
sigma_post <- as.matrix(mod_ex13, pars = "sigma")
dim(sigma_post)

## [1] 4000    1

## Compute the posterior distribution for R_B^2
n <- nrow(data1)
var_y <- var(data1$time) * ((n-1) / n) # variance of original data
R2bayes <- 1 - sigma_post^2 / var_y # bayesian r-squared

## posterior inference
hist(R2bayes, breaks = 30) ## heavily lefted skewed: good. our model simulate
```




```
                          ## new data which are close to the original one
mean(R2bayes)  # close to 1: good

## [1] 0.9521072

sd(R2bayes)

## [1] 0.01571624

quantile(R2bayes, probs = c(0.025, 0.5, 0.975)) # credibility

##      2.5%       50%     97.5%
## 0.9149402 0.9552330 0.9736798
```

Chapter 11

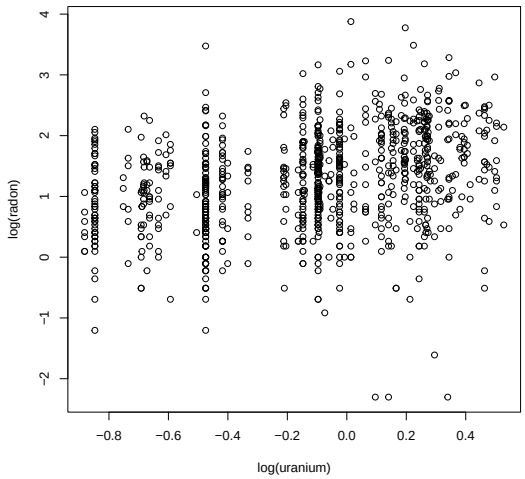
Exercise 2: Multilevel normal linear regression

```
# load packages
library(rstanarm)
library(loo)
library(bayesplot)
library(rstan)
library(ggplot2)
set.seed(1234)
```

In the Radon data we have $n = 919$ observations of the indoor radon concentration in the state of Minnesota (US):

- response variable: *logarithmic transformation of the radon measurement*
- covariates:
  - *log of uranium* concentration (county-level information).
  - *floor* of measurement (dichotomous variable, 0= basement level or 1=first floor)
  - *county*: county of measurements (categorical)
- we estimate a *Bayesian hierarchical gaussian model* with random effects to account for the county of the measurement;
- total number of observations $n = 919$ is partitioned with respect to the county: in each one of the $j = 1, \ldots, 85$ counties there are $i = 1, \ldots, n_j$ measurements
- we see three kind of random effect model using county:
  a) **random intercept model**, where we assume that baseline effect can vary across different countys
  b) **random intercepts and covariates**
  c) **random intercepts, covariates and random slopes**

```
data2 <- read.csv("data/Data_Ex_2.csv")
if (FALSE) data2 <- read.csv("bayesian_inference/data/Data_Ex_2.csv")

# overview
str(data2)

## 'data.frame': 919 obs. of  4 variables:
##  $ floor      : chr  "first" "base" "base" "base" ...
##  $ county     : chr  "AITKIN" "AITKIN" "AITKIN" "AITKIN" ...
##  $ log_radon  : num  0.8329 0.8329 1.0986 0.0953 1.1632 ...
##  $ log_uranium: num  -0.689 -0.689 -0.689 -0.689 -0.847 ...

plot(data2$log_radon ~ data2$log_uranium,
     xlab = "log(uranium)", ylab = "log(radon)")
```



11.1 a) Random intercept model

In this kind of model, remembering j the countries and i the measurement we suppose that measurements are normally distributed with mean μ_j[1] depending on a common level β_0 and a county specific addon $\beta_{0[j]}$ (the random effect specific for county).
Formally, for $j = 1, \ldots, 85$, $i = 1, .., n_j$ the **Likelihood**:

$$y_{ij}|\mu_j, \sigma^2 \sim \mathcal{N}(\mu_j, \sigma^2);$$
$$\mu_j|\beta_{[.]} = \beta_0 + \beta_{0[j]}$$

The second level of the model is given by the **priors**; for the parameters of this model we assume

[1]It's not μ_{ij} because in this case we only have the random intercept and we don't have any variable at the observations level.

- "just" a prior for σ,
- that random effects are normal with mean 0 and variance $\sigma^2_{\beta_0}$, different from σ
- the common intercept to be still normal with zero mean and variance c (to be considered *fixed*)

Formally

$$\begin{aligned}
&\sigma \sim \pi(\sigma) \\
&\beta_{0[j]}|\sigma^2_{\beta_0} \sim \mathcal{N}(0, \sigma^2_{\beta_0}); \\
&\beta_0 \sim \mathcal{N}(0, c)
\end{aligned}$$

The last level of the model regards the prior (**Hyperprior**) for the (not fixed) parameters in the priors above (**Hyperparameter**); the only one above is the variance of random effects $\sigma^2_{\beta_0}$. We assume a prior similar (in shape π) to the one for σ:

$$\sigma_{\beta_0} \sim \pi(\sigma_{\beta_0}).$$

To translate this in `rstanarm` estimation syntax is borrowed from `lme4`; we'll use `stan_lmer`, which is equivalent to `stan_glmer` with `family = gaussian(link = "identity")`

```
# Random intercept model estimation
set.seed(1234)
mod_ex2a <- stan_lmer(log_radon ~ (1 | county), data = data2)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.0001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.689 seconds (Warm-up)
```

```
## Chain 1:                0.8 seconds (Sampling)
## Chain 1:                2.489 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.48 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.904 seconds (Warm-up)
## Chain 2:                0.837 seconds (Sampling)
## Chain 2:                2.741 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.48 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
```

```
## Chain 3:  Elapsed Time: 1.693 seconds (Warm-up)
## Chain 3:                0.807 seconds (Sampling)
## Chain 3:                2.5 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.063 seconds (Warm-up)
## Chain 4:                0.801 seconds (Sampling)
## Chain 4:                2.864 seconds (Total)
## Chain 4:
```

Regarding priors difference with standard model is that

- in the `prior_summary` we have a prior for the covariance which is basically the prior for $\sigma^2_{\beta_0}$.
  `decov` means decomposable covariance prior (prior used in model with group specific effect, like random intercept/random slopes) with four parameters:
  - `shape` and `scale` are the usual parameters of the distributions;
  - we have furthermore `reg` (for regularization) and `conc` (concentration): these two parameters regulates shrinkage of correlation matrix (which in our case is not a matrix but just a single parameter $\sigma^2_{\beta_0}$)

```
# priors
prior_summary(mod_ex2a)

## Priors for model 'mod_ex2a'
## ------
## Intercept (after predictors centered)
```

```
##   Specified prior:
##     ~ normal(location = 1.3, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 1.3, scale = 2)
##
## Auxiliary (sigma)
##   Specified prior:
##     ~ exponential(rate = 1)
##   Adjusted prior:
##     ~ exponential(rate = 1.2)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

Looking at the summary (MCMC diagnostic section) we have

- the intercept, the common β_0
- a value for each of the random effects (the `b` rows for each country)
- `sigma`
- `"Sigma[county:(Intercept),(Intercept)]` is our hyperparameter $\sigma^2_{\beta_0}$

MCSE is basically=0 everywhere, rhat=1, effective sample size seems to be acceptable

```
summary(mod_ex2a)
```

```
##
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      log_radon ~ (1 | county)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 919
##  groups:       county (85)
##
## Estimates:
##                                          mean   sd   10%   50%   90%
## (Intercept)                              1.3    0.0  1.3   1.3   1.4
## b[(Intercept) county:AITKIN]            -0.2    0.2 -0.5  -0.2   0.1
## b[(Intercept) county:ANOKA]             -0.4    0.1 -0.5  -0.4  -0.3
## b[(Intercept) county:BECKER]            -0.1    0.3 -0.4  -0.1   0.2
## b[(Intercept) county:BELTRAMI]          -0.1    0.2 -0.3  -0.1   0.2
## b[(Intercept) county:BENTON]             0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:BIGSTONE]           0.1    0.3 -0.3   0.1   0.4
```

```
## b[(Intercept) county:BLUEEARTH]        0.4    0.2  0.2   0.4   0.6
## b[(Intercept) county:BROWN]             0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) county:CARLTON]          -0.2    0.2 -0.5  -0.2   0.0
## b[(Intercept) county:CARVER]           -0.1    0.2 -0.3  -0.1   0.2
## b[(Intercept) county:CASS]              0.0    0.2 -0.2   0.0   0.3
## b[(Intercept) county:CHIPPEWA]          0.2    0.2 -0.2   0.2   0.5
## b[(Intercept) county:CHISAGO]          -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:CLAY]              0.3    0.2  0.1   0.3   0.5
## b[(Intercept) county:CLEARWATER]       -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:COOK]             -0.2    0.3 -0.5  -0.1   0.2
## b[(Intercept) county:COTTONWOOD]       -0.2    0.2 -0.5  -0.2   0.1
## b[(Intercept) county:CROWWING]         -0.2    0.2 -0.5  -0.2   0.0
## b[(Intercept) county:DAKOTA]            0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:DODGE]             0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) county:DOUGLAS]           0.2    0.2  0.0   0.2   0.4
## b[(Intercept) county:FARIBAULT]        -0.3    0.2 -0.6  -0.3   0.0
## b[(Intercept) county:FILLMORE]         -0.1    0.3 -0.4  -0.1   0.3
## b[(Intercept) county:FREEBORN]          0.4    0.2  0.1   0.4   0.6
## b[(Intercept) county:GOODHUE]           0.3    0.2  0.1   0.3   0.6
## b[(Intercept) county:HENNEPIN]          0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:HOUSTON]           0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) county:HUBBARD]          -0.2    0.2 -0.5  -0.2   0.1
## b[(Intercept) county:ISANTI]           -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:ITASCA]           -0.2    0.2 -0.5  -0.2   0.0
## b[(Intercept) county:JACKSON]           0.3    0.2  0.0   0.3   0.6
## b[(Intercept) county:KANABEC]           0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:KANDIYOHI]         0.3    0.2  0.0   0.3   0.6
## b[(Intercept) county:KITTSON]          -0.1    0.3 -0.4  -0.1   0.3
## b[(Intercept) county:KOOCHICHING]      -0.4    0.2 -0.7  -0.4  -0.2
## b[(Intercept) county:LACQUIPARLE]       0.3    0.3  0.0   0.3   0.7
## b[(Intercept) county:LAKE]             -0.5    0.2 -0.8  -0.5  -0.3
## b[(Intercept) county:LAKEOFTHEWOODS]    0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) county:LESUEUR]           0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) county:LINCOLN]           0.3    0.2  0.0   0.3   0.6
## b[(Intercept) county:LYON]              0.3    0.2  0.0   0.3   0.6
## b[(Intercept) county:MAHNOMEN]          0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:MARSHALL]         -0.1    0.2 -0.3  -0.1   0.2
## b[(Intercept) county:MARTIN]           -0.2    0.2 -0.4  -0.2   0.1
## b[(Intercept) county:MCLEOD]           -0.2    0.2 -0.4  -0.2   0.1
## b[(Intercept) county:MEEKER]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:MILLELACS]        -0.2    0.3 -0.5  -0.2   0.2
## b[(Intercept) county:MORRISON]         -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:MOWER]             0.2    0.2 -0.1   0.2   0.4
## b[(Intercept) county:MURRAY]            0.2    0.3 -0.2   0.2   0.5
## b[(Intercept) county:NICOLLET]          0.3    0.2  0.0   0.3   0.6
## b[(Intercept) county:NOBLES]            0.2    0.3 -0.1   0.2   0.5
## b[(Intercept) county:NORMAN]           -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:OLMSTED]          -0.1    0.1 -0.3  -0.1   0.1
## b[(Intercept) county:OTTERTAIL]         0.0    0.2 -0.2   0.0   0.3
```

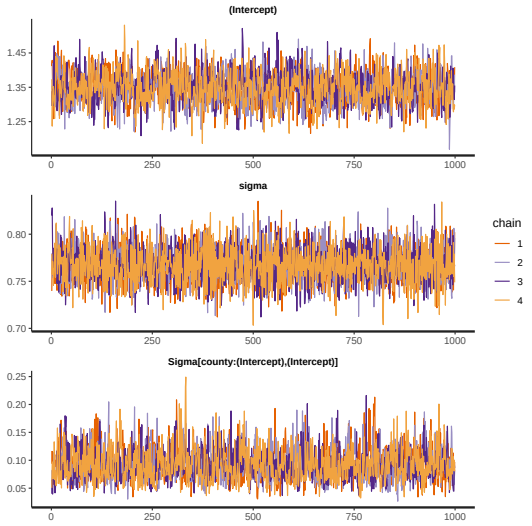
```
## b[(Intercept) county:PENNINGTON]      -0.2    0.3 -0.5  -0.2   0.1
## b[(Intercept) county:PINE]            -0.3    0.2 -0.6  -0.3   0.0
## b[(Intercept) county:PIPESTONE]        0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) county:POLK]             0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:POPE]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:RAMSEY]          -0.2    0.1 -0.3  -0.2   0.0
## b[(Intercept) county:REDWOOD]          0.2    0.2 -0.1   0.2   0.5
## b[(Intercept) county:RENVILLE]         0.0    0.2 -0.3   0.0   0.4
## b[(Intercept) county:RICE]             0.3    0.2  0.0   0.3   0.5
## b[(Intercept) county:ROCK]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ROSEAU]           0.0    0.2 -0.3   0.0   0.2
## b[(Intercept) county:SCOTT]            0.2    0.2 -0.1   0.2   0.4
## b[(Intercept) county:SHERBURNE]       -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:SIBLEY]           0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:STEARNS]          0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:STEELE]           0.2    0.2 -0.1   0.1   0.4
## b[(Intercept) county:STEVENS]          0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) county:STLOUIS]         -0.5    0.1 -0.6  -0.5  -0.4
## b[(Intercept) county:SWIFT]           -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:TODD]             0.0    0.2 -0.3   0.0   0.4
## b[(Intercept) county:TRAVERSE]         0.2    0.2 -0.1   0.2   0.5
## b[(Intercept) county:WABASHA]          0.2    0.2 -0.1   0.2   0.5
## b[(Intercept) county:WADENA]          -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:WASECA]          -0.3    0.2 -0.6  -0.3   0.0
## b[(Intercept) county:WASHINGTON]      -0.1    0.1 -0.2   0.0   0.1
## b[(Intercept) county:WATONWAN]         0.3    0.3  0.0   0.3   0.6
## b[(Intercept) county:WILKIN]           0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) county:WINONA]           0.1    0.2 -0.1   0.1   0.3
## b[(Intercept) county:WRIGHT]           0.2    0.2  0.0   0.2   0.4
## b[(Intercept) county:YELLOWMEDICINE]   0.0    0.3 -0.4   0.0   0.3
## sigma                                  0.8    0.0  0.7   0.8   0.8
## Sigma[county:(Intercept),(Intercept)]  0.1    0.0  0.1   0.1   0.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 1.3    0.0  1.2   1.3   1.3
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                    mcse Rhat n_eff
## (Intercept)                        0.0  1.0  1479
## b[(Intercept) county:AITKIN]       0.0  1.0  5364
## b[(Intercept) county:ANOKA]        0.0  1.0  3903
## b[(Intercept) county:BECKER]       0.0  1.0  5591
## b[(Intercept) county:BELTRAMI]     0.0  1.0  5438
## b[(Intercept) county:BENTON]       0.0  1.0  6225
## b[(Intercept) county:BIGSTONE]     0.0  1.0  5384
## b[(Intercept) county:BLUEEARTH]    0.0  1.0  4625
```

```
## b[(Intercept) county:BROWN]           0.0  1.0  6241
## b[(Intercept) county:CARLTON]         0.0  1.0  5472
## b[(Intercept) county:CARVER]          0.0  1.0  4915
## b[(Intercept) county:CASS]            0.0  1.0  5745
## b[(Intercept) county:CHIPPEWA]        0.0  1.0  5449
## b[(Intercept) county:CHISAGO]         0.0  1.0  5444
## b[(Intercept) county:CLAY]            0.0  1.0  5518
## b[(Intercept) county:CLEARWATER]      0.0  1.0  5411
## b[(Intercept) county:COOK]            0.0  1.0  5632
## b[(Intercept) county:COTTONWOOD]      0.0  1.0  5208
## b[(Intercept) county:CROWWING]        0.0  1.0  5208
## b[(Intercept) county:DAKOTA]          0.0  1.0  3275
## b[(Intercept) county:DODGE]           0.0  1.0  5580
## b[(Intercept) county:DOUGLAS]         0.0  1.0  5695
## b[(Intercept) county:FARIBAULT]       0.0  1.0  4573
## b[(Intercept) county:FILLMORE]        0.0  1.0  5032
## b[(Intercept) county:FREEBORN]        0.0  1.0  4676
## b[(Intercept) county:GOODHUE]         0.0  1.0  4891
## b[(Intercept) county:HENNEPIN]        0.0  1.0  2805
## b[(Intercept) county:HOUSTON]         0.0  1.0  5727
## b[(Intercept) county:HUBBARD]         0.0  1.0  5624
## b[(Intercept) county:ISANTI]          0.0  1.0  4481
## b[(Intercept) county:ITASCA]          0.0  1.0  4747
## b[(Intercept) county:JACKSON]         0.0  1.0  5384
## b[(Intercept) county:KANABEC]         0.0  1.0  5494
## b[(Intercept) county:KANDIYOHI]       0.0  1.0  4591
## b[(Intercept) county:KITTSON]         0.0  1.0  5700
## b[(Intercept) county:KOOCHICHING]     0.0  1.0  4505
## b[(Intercept) county:LACQUIPARLE]     0.0  1.0  3922
## b[(Intercept) county:LAKE]            0.0  1.0  4385
## b[(Intercept) county:LAKEOFTHEWOODS]  0.0  1.0  7320
## b[(Intercept) county:LESUEUR]         0.0  1.0  5662
## b[(Intercept) county:LINCOLN]         0.0  1.0  4545
## b[(Intercept) county:LYON]            0.0  1.0  5303
## b[(Intercept) county:MAHNOMEN]        0.0  1.0  5511
## b[(Intercept) county:MARSHALL]        0.0  1.0  6390
## b[(Intercept) county:MARTIN]          0.0  1.0  5383
## b[(Intercept) county:MCLEOD]          0.0  1.0  4934
## b[(Intercept) county:MEEKER]          0.0  1.0  5826
## b[(Intercept) county:MILLELACS]       0.0  1.0  4533
## b[(Intercept) county:MORRISON]        0.0  1.0  6196
## b[(Intercept) county:MOWER]           0.0  1.0  5501
## b[(Intercept) county:MURRAY]          0.0  1.0  4587
## b[(Intercept) county:NICOLLET]        0.0  1.0  4903
## b[(Intercept) county:NOBLES]          0.0  1.0  4898
## b[(Intercept) county:NORMAN]          0.0  1.0  5959
## b[(Intercept) county:OLMSTED]         0.0  1.0  4999
## b[(Intercept) county:OTTERTAIL]       0.0  1.0  5789
## b[(Intercept) county:PENNINGTON]      0.0  1.0  5093
```

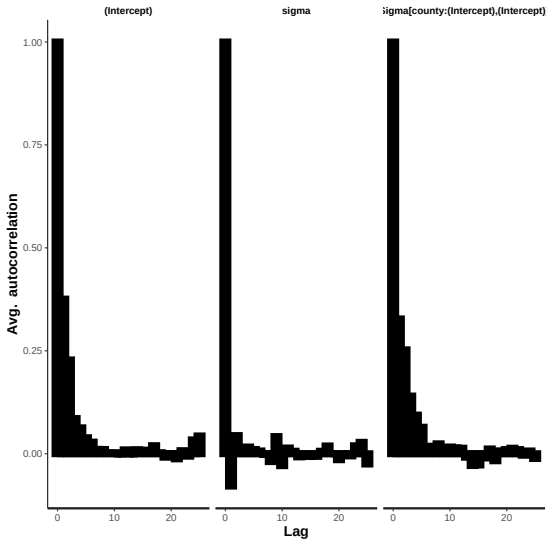
```
## b[(Intercept) county:PINE]             0.0  1.0  4653
## b[(Intercept) county:PIPESTONE]        0.0  1.0  4760
## b[(Intercept) county:POLK]             0.0  1.0  5541
## b[(Intercept) county:POPE]             0.0  1.0  4493
## b[(Intercept) county:RAMSEY]           0.0  1.0  4314
## b[(Intercept) county:REDWOOD]          0.0  1.0  5086
## b[(Intercept) county:RENVILLE]         0.0  1.0  6214
## b[(Intercept) county:RICE]             0.0  1.0  4876
## b[(Intercept) county:ROCK]             0.0  1.0  5943
## b[(Intercept) county:ROSEAU]           0.0  1.0  4606
## b[(Intercept) county:SCOTT]            0.0  1.0  5944
## b[(Intercept) county:SHERBURNE]        0.0  1.0  6253
## b[(Intercept) county:SIBLEY]           0.0  1.0  5629
## b[(Intercept) county:STEARNS]          0.0  1.0  4842
## b[(Intercept) county:STEELE]           0.0  1.0  5138
## b[(Intercept) county:STEVENS]          0.0  1.0  6422
## b[(Intercept) county:STLOUIS]          0.0  1.0  2618
## b[(Intercept) county:SWIFT]            0.0  1.0  4170
## b[(Intercept) county:TODD]             0.0  1.0  5668
## b[(Intercept) county:TRAVERSE]         0.0  1.0  5863
## b[(Intercept) county:WABASHA]          0.0  1.0  4688
## b[(Intercept) county:WADENA]           0.0  1.0  5005
## b[(Intercept) county:WASECA]           0.0  1.0  4342
## b[(Intercept) county:WASHINGTON]       0.0  1.0  3385
## b[(Intercept) county:WATONWAN]         0.0  1.0  5066
## b[(Intercept) county:WILKIN]           0.0  1.0  5718
## b[(Intercept) county:WINONA]           0.0  1.0  4789
## b[(Intercept) county:WRIGHT]           0.0  1.0  4558
## b[(Intercept) county:YELLOWMEDICINE]   0.0  1.0  6250
## sigma                                  0.0  1.0  3956
## Sigma[county:(Intercept),(Intercept)] 0.0  1.0  1328
## mean_PPD                               0.0  1.0  4091
## log-posterior                          0.3  1.0   888
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

Finally for the checks we use `pars` to avoid all the traceplots and autocorrelations to be plotted (especially those from the random effects); this avoids R to crash.

```
params <- c("(Intercept)", "sigma", "Sigma[county:(Intercept),(Intercept)]")
stan_trace(mod_ex2a, pars = params, nrow = 3, ncol = 1)
```




```
stan_ac(mod_ex2a, pars = params)
```



11.2 b) Model with random intercepts and covariates

In this model we also introduce the two measurements levels variables and so the observed variable are assumed to be distributed according to a normal with μ_{ij} (which depends not only on the county j, but also on measurement i due to the covariates).

Formally for the **Likelihood**: always for $j = 1, \ldots, 85$ and $i = 1, .., n_j$

$$y_{ij}|\mu_{ij}, \sigma^2 \sim \mathcal{N}(\mu_{ij}, \sigma^2)$$
$$\mu_{ij}|\boldsymbol{\beta} = \beta_0 + \beta_{0[j]} + \beta_1 \texttt{log\_uranium}_{ij} + \beta_2 \texttt{floor}_{ij}$$

The **Priors** of the parameters above is the same as the previous model with the exception of prior for $k-1$ regression coefficient, here assumed to be normal with mean 0 and c variance (as for the intercept):

$$\sigma \sim \pi(\sigma)$$
$$\beta_{0[j]}|\sigma^2_{\beta_0} \sim \mathcal{N}(0, \sigma^2_{\beta_0});$$
$$\beta_k \sim \mathcal{N}(0, c) \quad k = 0, 1, 2$$

The **Hyperprior** for the parameters above is still the same:

$$\sigma_{\beta_0} \sim \pi(\sigma_{\beta_0}).$$

For the estimation we have the following and in the summary (MCMC diagnostic) we have fixed effects at first and then random effects and variances

```
# Random intercept model with county level covariate and measure level covariate
set.seed(1234)
mod_ex2b <- stan_lmer(log_radon ~ log_uranium + floor + (1 | county), data = data2)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 8.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.84 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.215 seconds (Warm-up)
## Chain 1:                3.041 seconds (Sampling)
## Chain 1:                6.256 seconds (Total)
## Chain 1:
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.54 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.842 seconds (Warm-up)
## Chain 2:                0.891 seconds (Sampling)
## Chain 2:                3.733 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.54 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 4.101 seconds (Warm-up)
## Chain 3:                0.912 seconds (Sampling)
## Chain 3:                5.013 seconds (Total)
```

```
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 5.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.54 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 3.404 seconds (Warm-up)
## Chain 4:                0.996 seconds (Sampling)
## Chain 4:                4.4 seconds (Total)
## Chain 4:
```

```
summary(mod_ex2b)
```

```
##
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      log_radon ~ log_uranium + floor + (1 | county)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 919
##  groups:       county (85)
##
## Estimates:
##                                          mean   sd   10%   50%   90%
## (Intercept)                              1.5    0.0  1.4   1.5   1.5
## log_uranium                              0.7    0.1  0.6   0.7   0.8
## floorfirst                              -0.6    0.1 -0.7  -0.6  -0.6
## b[(Intercept) county:AITKIN]             0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:ANOKA]              0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:BECKER]             0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:BELTRAMI]           0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:BENTON]             0.0    0.1 -0.2   0.0   0.2
```

```
## b[(Intercept) county:BIGSTONE]          0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:BLUEEARTH]         0.1    0.1  0.0   0.1   0.3
## b[(Intercept) county:BROWN]             0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:CARLTON]          -0.1    0.1 -0.2  -0.1   0.1
## b[(Intercept) county:CARVER]            0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:CASS]              0.1    0.1 -0.1   0.0   0.2
## b[(Intercept) county:CHIPPEWA]          0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:CHISAGO]           0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:CLAY]              0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:CLEARWATER]        0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:COOK]              0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:COTTONWOOD]       -0.1    0.1 -0.3  -0.1   0.1
## b[(Intercept) county:CROWWING]          0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:DAKOTA]           -0.1    0.1 -0.2  -0.1   0.0
## b[(Intercept) county:DODGE]             0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:DOUGLAS]           0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:FARIBAULT]        -0.2    0.2 -0.4  -0.2   0.0
## b[(Intercept) county:FILLMORE]          0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:FREEBORN]          0.1    0.1  0.0   0.1   0.3
## b[(Intercept) county:GOODHUE]           0.1    0.1  0.0   0.1   0.3
## b[(Intercept) county:HENNEPIN]          0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:HOUSTON]           0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:HUBBARD]           0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:ISANTI]            0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:ITASCA]            0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:JACKSON]           0.1    0.1 -0.1   0.0   0.2
## b[(Intercept) county:KANABEC]           0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:KANDIYOHI]         0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:KITTSON]           0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:KOOCHICHING]       0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:LACQUIPARLE]       0.1    0.2 -0.1   0.1   0.3
## b[(Intercept) county:LAKE]             -0.1    0.1 -0.3  -0.1   0.0
## b[(Intercept) county:LAKEOFTHEWOODS]    0.1    0.2 -0.1   0.1   0.3
## b[(Intercept) county:LESUEUR]           0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:LINCOLN]           0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:LYON]              0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:MAHNOMEN]          0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:MARSHALL]          0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:MARTIN]           -0.1    0.1 -0.3  -0.1   0.0
## b[(Intercept) county:MCLEOD]           -0.1    0.1 -0.3  -0.1   0.0
## b[(Intercept) county:MEEKER]            0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:MILLELACS]         0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:MORRISON]         -0.1    0.1 -0.2  -0.1   0.1
## b[(Intercept) county:MOWER]             0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:MURRAY]            0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:NICOLLET]          0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:NOBLES]            0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:NORMAN]            0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:OLMSTED]          -0.2    0.1 -0.3  -0.1   0.0
```

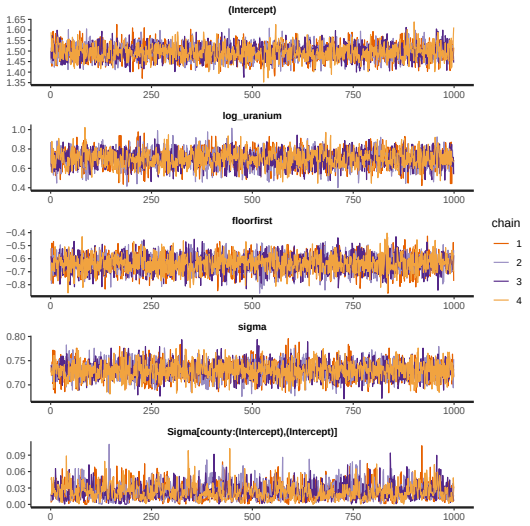
```
## b[(Intercept) county:OTTERTAIL]        0.1    0.1 -0.1   0.1   0.2
## b[(Intercept) county:PENNINGTON]       0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:PINE]            -0.1    0.1 -0.3  -0.1   0.1
## b[(Intercept) county:PIPESTONE]        0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:POLK]             0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:POPE]             0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:RAMSEY]           0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:REDWOOD]          0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:RENVILLE]         0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:RICE]             0.1    0.1 -0.1   0.1   0.2
## b[(Intercept) county:ROCK]             0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:ROSEAU]           0.1    0.1  0.0   0.1   0.3
## b[(Intercept) county:SCOTT]            0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) county:SHERBURNE]        0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:SIBLEY]          -0.1    0.1 -0.2   0.0   0.1
## b[(Intercept) county:STEARNS]          0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:STEELE]           0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) county:STEVENS]          0.0    0.1 -0.2   0.0   0.2
## b[(Intercept) county:STLOUIS]         -0.2    0.1 -0.3  -0.2  -0.1
## b[(Intercept) county:SWIFT]           -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) county:TODD]             0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:TRAVERSE]         0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:WABASHA]          0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:WADENA]           0.0    0.1 -0.1   0.0   0.2
## b[(Intercept) county:WASECA]          -0.1    0.2 -0.4  -0.1   0.0
## b[(Intercept) county:WASHINGTON]       0.0    0.1 -0.1   0.0   0.1
## b[(Intercept) county:WATONWAN]         0.1    0.2 -0.1   0.1   0.3
## b[(Intercept) county:WILKIN]           0.0    0.2 -0.2   0.0   0.2
## b[(Intercept) county:WINONA]          -0.1    0.1 -0.2  -0.1   0.1
## b[(Intercept) county:WRIGHT]           0.1    0.1 -0.1   0.1   0.2
## b[(Intercept) county:YELLOWMEDICINE]   0.0    0.1 -0.2   0.0   0.1
## sigma                                  0.7    0.0  0.7   0.7   0.8
## Sigma[county:(Intercept),(Intercept)]  0.0    0.0  0.0   0.0   0.0
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 1.3    0.0  1.2   1.3   1.3
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                    mcse Rhat n_eff
## (Intercept)                        0.0  1.0  3007
## log_uranium                        0.0  1.0  2948
## floorfirst                         0.0  1.0  5725
## b[(Intercept) county:AITKIN]       0.0  1.0  5058
## b[(Intercept) county:ANOKA]        0.0  1.0  3531
## b[(Intercept) county:BECKER]       0.0  1.0  5937
## b[(Intercept) county:BELTRAMI]     0.0  1.0  3456
```

```
## b[(Intercept) county:BENTON]           0.0  1.0  4899
## b[(Intercept) county:BIGSTONE]         0.0  1.0  5768
## b[(Intercept) county:BLUEEARTH]        0.0  1.0  3716
## b[(Intercept) county:BROWN]            0.0  1.0  5378
## b[(Intercept) county:CARLTON]          0.0  1.0  4538
## b[(Intercept) county:CARVER]           0.0  1.0  5752
## b[(Intercept) county:CASS]             0.0  1.0  4960
## b[(Intercept) county:CHIPPEWA]         0.0  1.0  6653
## b[(Intercept) county:CHISAGO]          0.0  1.0  5271
## b[(Intercept) county:CLAY]             0.0  1.0  4315
## b[(Intercept) county:CLEARWATER]       0.0  1.0  6249
## b[(Intercept) county:COOK]             0.0  1.0  4885
## b[(Intercept) county:COTTONWOOD]       0.0  1.0  4962
## b[(Intercept) county:CROWWING]         0.0  1.0  6274
## b[(Intercept) county:DAKOTA]           0.0  1.0  3379
## b[(Intercept) county:DODGE]            0.0  1.0  5554
## b[(Intercept) county:DOUGLAS]          0.0  1.0  4747
## b[(Intercept) county:FARIBAULT]        0.0  1.0  2471
## b[(Intercept) county:FILLMORE]         0.0  1.0  5323
## b[(Intercept) county:FREEBORN]         0.0  1.0  3211
## b[(Intercept) county:GOODHUE]          0.0  1.0  3864
## b[(Intercept) county:HENNEPIN]         0.0  1.0  4512
## b[(Intercept) county:HOUSTON]          0.0  1.0  4818
## b[(Intercept) county:HUBBARD]          0.0  1.0  4834
## b[(Intercept) county:ISANTI]           0.0  1.0  5032
## b[(Intercept) county:ITASCA]           0.0  1.0  5534
## b[(Intercept) county:JACKSON]          0.0  1.0  4386
## b[(Intercept) county:KANABEC]          0.0  1.0  5733
## b[(Intercept) county:KANDIYOHI]        0.0  1.0  4789
## b[(Intercept) county:KITTSON]          0.0  1.0  5583
## b[(Intercept) county:KOOCHICHING]      0.0  1.0  5269
## b[(Intercept) county:LACQUIPARLE]      0.0  1.0  3823
## b[(Intercept) county:LAKE]             0.0  1.0  2200
## b[(Intercept) county:LAKEOFTHEWOODS]   0.0  1.0  3015
## b[(Intercept) county:LESUEUR]          0.0  1.0  4697
## b[(Intercept) county:LINCOLN]          0.0  1.0  4192
## b[(Intercept) county:LYON]             0.0  1.0  4314
## b[(Intercept) county:MAHNOMEN]         0.0  1.0  5675
## b[(Intercept) county:MARSHALL]         0.0  1.0  5447
## b[(Intercept) county:MARTIN]           0.0  1.0  3159
## b[(Intercept) county:MCLEOD]           0.0  1.0  3352
## b[(Intercept) county:MEEKER]           0.0  1.0  4183
## b[(Intercept) county:MILLELACS]        0.0  1.0  5466
## b[(Intercept) county:MORRISON]         0.0  1.0  4801
## b[(Intercept) county:MOWER]            0.0  1.0  6601
## b[(Intercept) county:MURRAY]           0.0  1.0  5154
## b[(Intercept) county:NICOLLET]         0.0  1.0  4188
## b[(Intercept) county:NOBLES]           0.0  1.0  5810
## b[(Intercept) county:NORMAN]           0.0  1.0  4461
```

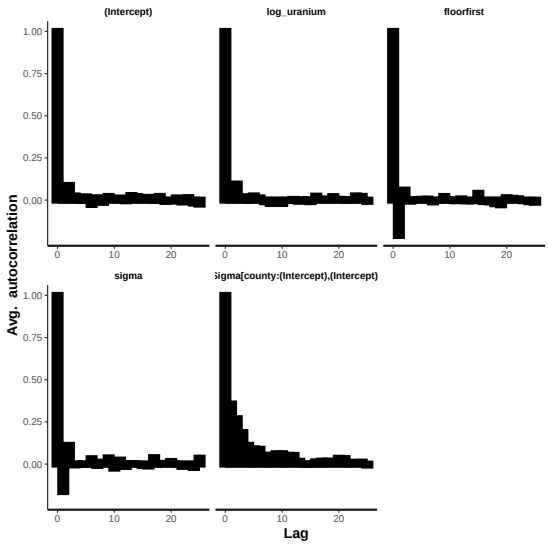
```
## b[(Intercept) county:OLMSTED]          0.0  1.0  2695
## b[(Intercept) county:OTTERTAIL]        0.0  1.0  5229
## b[(Intercept) county:PENNINGTON]       0.0  1.0  3646
## b[(Intercept) county:PINE]             0.0  1.0  3174
## b[(Intercept) county:PIPESTONE]        0.0  1.0  5377
## b[(Intercept) county:POLK]             0.0  1.0  5841
## b[(Intercept) county:POPE]             0.0  1.0  5284
## b[(Intercept) county:RAMSEY]           0.0  1.0  5025
## b[(Intercept) county:REDWOOD]          0.0  1.0  5514
## b[(Intercept) county:RENVILLE]         0.0  1.0  7156
## b[(Intercept) county:RICE]             0.0  1.0  5018
## b[(Intercept) county:ROCK]             0.0  1.0  4730
## b[(Intercept) county:ROSEAU]           0.0  1.0  4179
## b[(Intercept) county:SCOTT]            0.0  1.0  4736
## b[(Intercept) county:SHERBURNE]        0.0  1.0  5677
## b[(Intercept) county:SIBLEY]           0.0  1.0  5178
## b[(Intercept) county:STEARNS]          0.0  1.0  5131
## b[(Intercept) county:STEELE]           0.0  1.0  6165
## b[(Intercept) county:STEVENS]          0.0  1.0  7100
## b[(Intercept) county:STLOUIS]          0.0  1.0  1938
## b[(Intercept) county:SWIFT]            0.0  1.0  3485
## b[(Intercept) county:TODD]             0.0  1.0  4966
## b[(Intercept) county:TRAVERSE]         0.0  1.0  5036
## b[(Intercept) county:WABASHA]          0.0  1.0  4776
## b[(Intercept) county:WADENA]           0.0  1.0  5171
## b[(Intercept) county:WASECA]           0.0  1.0  2945
## b[(Intercept) county:WASHINGTON]       0.0  1.0  4661
## b[(Intercept) county:WATONWAN]         0.0  1.0  2970
## b[(Intercept) county:WILKIN]           0.0  1.0  5165
## b[(Intercept) county:WINONA]           0.0  1.0  5123
## b[(Intercept) county:WRIGHT]           0.0  1.0  5173
## b[(Intercept) county:YELLOWMEDICINE]   0.0  1.0  4881
## sigma                                  0.0  1.0  4585
## Sigma[county:(Intercept),(Intercept)] 0.0  1.0   969
## mean_PPD                               0.0  1.0  4640
## log-posterior                          0.3  1.0   729
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

Same we just look at main diagnostic, not the one for random intercept too: for the traceplot there seems to be no problem, while for the autocorrelation function there seems to be some autocorrelation for the variance of the random effect (but seems to go to 0 after 8 or 9 lag, so not a big issue)

```
params_2b <- c("(Intercept)", "log_uranium", "floorfirst","sigma",
               "Sigma[county:(Intercept),(Intercept)]")
stan_trace(mod_ex2b, pars = params_2b, nrow= 5, ncol = 1)
```




```
stan_ac(mod_ex2b, pars = params_2b)
```



11.3 c) Model random intercepts, covariates and random slopes

In the final model, the structure is similar to the previous one, but we introduce a random effect also for the variable `floor`. In the **Likelihood** we see the dependence on j for β_2 (which becomes a random effect):

$$y_{ij}|\mu_{ij}, \sigma^2 \sim \mathcal{N}(\mu_{ij}, \sigma^2);$$
$$\mu_{ij}|\boldsymbol{\beta} = \beta_0 + \beta_{0[j]} + \beta_1 \texttt{log\_uranium}_{ij} + \beta_{2[j]} \texttt{floor}_{ij};\ j = 1, \ldots, 85;\ i = 1, .., n_j.$$

Regarding **Priors** for the parameters above the only change is for the random effect $\beta_{2[j]}$ which has no more a fixed c variance but $\sigma^2_{\beta_2}$

$$\begin{aligned}
&\sigma \sim \pi(\sigma) \\
&\beta_{0[j]}|\sigma^2_{\beta_0} \sim \mathcal{N}(0, \sigma^2_{\beta_0}); \\
&\beta_k \sim \mathcal{N}(0, c) \quad k = 0, 1 \\
&\beta_{2[j]}|\sigma^2_{\beta_2} \sim \mathcal{N}\big(0, \sigma^2_{\beta_2}\big)
\end{aligned}$$

Now since we have two random effects $(\beta_{0[j]}, \beta_{2[j]})$ the priors for their parameter is actually the prior for a matrix of variance/covariance between the parameters ($\sigma^2_{\beta_0}$ and $\sigma^2_{\beta_2}$, considered that both have mean zero). So **Hyperprior**:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{\beta_0} & \sigma_{\beta_0,\beta_2} \\ \sigma_{\beta_0,\beta_2} & \sigma^2_{\beta_2} \end{bmatrix} \sim \pi(\mathbf{\Sigma})$$

where σ_{β_0,β_2} is the covariance between $\sigma^2_{\beta_0}$ and $\sigma^2_{\beta_2}$.
For the estimation

- the command is below: to have the random effect for floor we add it `+floor` to the left hand side within round parenthesis (eg if we wanted a random effect for log uranium we specified `(1+floor+log_uranium | county)`)
- from `summary` we see that for each county we have the random intercept and random slope
- in the MCMC diagnostic we see that N_eff is quite low for the covariance matrix between random effects `Sigma` (`intercept intercept` is for $\sigma^2_{\beta_0}$, floorfirst floorfirst is for $\sigma^2_{\beta_2}$, while follrfirst intercept is for covariance)
- thus we choose to increment iterations/length of the generated chains by using `update`

```
set.seed(1234)
mod_ex2c <- stan_lmer(log_radon ~ log_uranium + floor + (1 + floor | county), data = 

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000129 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.29 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 8.306 seconds (Warm-up)
## Chain 1:                3.283 seconds (Sampling)
## Chain 1:                11.589 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000116 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 7.583 seconds (Warm-up)
## Chain 2:                3.342 seconds (Sampling)
## Chain 2:                10.925 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000108 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 8.653 seconds (Warm-up)
## Chain 3:                3.409 seconds (Sampling)
## Chain 3:                12.062 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000107 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.07 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 7.817 seconds (Warm-up)
## Chain 4:                3.286 seconds (Sampling)
## Chain 4:                11.103 seconds (Total)
## Chain 4:
```

```
summary(mod_ex2c)
```

```
##
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      log_radon ~ log_uranium + floor + (1 + floor | county)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
```

```
##  observations: 919
##  groups:       county (85)
##
## Estimates:
##                                          mean   sd   10%   50%   90%
## (Intercept)                               1.5    0.0  1.4   1.5   1.5
## log_uranium                               0.7    0.1  0.6   0.7   0.8
## floorfirst                               -0.6    0.1 -0.7  -0.6  -0.5
## b[(Intercept) county:AITKIN]              0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:AITKIN]               0.1    0.3 -0.2   0.0   0.4
## b[(Intercept) county:ANOKA]               0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:ANOKA]                -0.1   0.2 -0.4  -0.1   0.2
## b[(Intercept) county:BECKER]              0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:BECKER]               0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:BELTRAMI]            0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:BELTRAMI]             0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:BENTON]              0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:BENTON]               0.0    0.3 -0.2   0.0   0.4
## b[(Intercept) county:BIGSTONE]            0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:BIGSTONE]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:BLUEEARTH]           0.1    0.1  0.0   0.1   0.3
## b[floorfirst county:BLUEEARTH]            0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:BROWN]               0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:BROWN]                0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:CARLTON]             -0.1   0.1 -0.2   0.0   0.1
## b[floorfirst county:CARLTON]              0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) county:CARVER]              0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:CARVER]               -0.1   0.2 -0.4  -0.1   0.2
## b[(Intercept) county:CASS]                0.1    0.1 -0.1   0.0   0.2
## b[floorfirst county:CASS]                 0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:CHIPPEWA]            0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:CHIPPEWA]             0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:CHISAGO]             0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:CHISAGO]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:CLAY]                0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:CLAY]                 0.0    0.2 -0.3   0.0   0.2
## b[(Intercept) county:CLEARWATER]          0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:CLEARWATER]           0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:COOK]                0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:COOK]                 0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:COTTONWOOD]          -0.1   0.1 -0.2  -0.1   0.1
## b[floorfirst county:COTTONWOOD]           -0.2   0.2 -0.5  -0.2   0.1
## b[(Intercept) county:CROWWING]            0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:CROWWING]             0.2    0.2 -0.1   0.1   0.5
## b[(Intercept) county:DAKOTA]              -0.1   0.1 -0.2  -0.1   0.0
## b[floorfirst county:DAKOTA]               -0.1   0.2 -0.4  -0.1   0.1
## b[(Intercept) county:DODGE]               0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:DODGE]                0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:DOUGLAS]             0.0    0.1 -0.1   0.0   0.2
```

```
## b[floorfirst county:DOUGLAS]            0.1    0.3 -0.2   0.0   0.4
## b[(Intercept) county:FARIBAULT]        -0.2    0.2 -0.4  -0.1   0.0
## b[floorfirst county:FARIBAULT]         -0.1    0.3 -0.5  -0.1   0.2
## b[(Intercept) county:FILLMORE]          0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:FILLMORE]           0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:FREEBORN]          0.1    0.1  0.0   0.1   0.3
## b[floorfirst county:FREEBORN]           0.1    0.2 -0.2   0.0   0.4
## b[(Intercept) county:GOODHUE]           0.1    0.1  0.0   0.1   0.3
## b[floorfirst county:GOODHUE]            0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:HENNEPIN]          0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:HENNEPIN]          -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) county:HOUSTON]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:HOUSTON]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:HUBBARD]           0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:HUBBARD]            0.0    0.2 -0.2   0.0   0.3
## b[(Intercept) county:ISANTI]            0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:ISANTI]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ITASCA]            0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:ITASCA]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:JACKSON]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:JACKSON]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:KANABEC]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:KANABEC]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:KANDIYOHI]         0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:KANDIYOHI]          0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:KITTSON]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:KITTSON]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:KOOCHICHING]       0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:KOOCHICHING]        0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:LACQUIPARLE]       0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:LACQUIPARLE]        0.2    0.3 -0.1   0.1   0.5
## b[(Intercept) county:LAKE]             -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:LAKE]               0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:LAKEOFTHEWOODS]    0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:LAKEOFTHEWOODS]     0.2    0.3 -0.1   0.1   0.5
## b[(Intercept) county:LESUEUR]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:LESUEUR]            0.1    0.3 -0.2   0.0   0.4
## b[(Intercept) county:LINCOLN]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:LINCOLN]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:LYON]              0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:LYON]               0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) county:MAHNOMEN]          0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:MAHNOMEN]           0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:MARSHALL]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MARSHALL]          -0.3    0.3 -0.6  -0.2   0.0
## b[(Intercept) county:MARTIN]           -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:MARTIN]            -0.2    0.3 -0.6  -0.2   0.1
## b[(Intercept) county:MCLEOD]           -0.1    0.1 -0.2  -0.1   0.1
## b[floorfirst county:MCLEOD]            -0.2    0.2 -0.5  -0.1   0.1
```

```
## b[(Intercept) county:MEEKER]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:MEEKER]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:MILLELACS]        0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:MILLELACS]        -0.2    0.3 -0.5  -0.1   0.1
## b[(Intercept) county:MORRISON]        -0.1    0.1 -0.2   0.0   0.1
## b[floorfirst county:MORRISON]          0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:MOWER]            0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MOWER]            -0.2    0.3 -0.5  -0.1   0.1
## b[(Intercept) county:MURRAY]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MURRAY]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:NICOLLET]         0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:NICOLLET]          0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:NOBLES]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:NOBLES]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:NORMAN]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:NORMAN]           -0.1    0.3 -0.4   0.0   0.2
## b[(Intercept) county:OLMSTED]         -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:OLMSTED]          -0.2    0.3 -0.6  -0.2   0.0
## b[(Intercept) county:OTTERTAIL]        0.1    0.1 -0.1   0.0   0.2
## b[floorfirst county:OTTERTAIL]         0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:PENNINGTON]       0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:PENNINGTON]       -0.1    0.2 -0.5  -0.1   0.1
## b[(Intercept) county:PINE]            -0.1    0.1 -0.3  -0.1   0.1
## b[floorfirst county:PINE]             -0.1    0.3 -0.4  -0.1   0.2
## b[(Intercept) county:PIPESTONE]        0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:PIPESTONE]         0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:POLK]             0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:POLK]             -0.1    0.2 -0.4   0.0   0.2
## b[(Intercept) county:POPE]             0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:POPE]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:RAMSEY]           0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:RAMSEY]            0.2    0.2 -0.1   0.1   0.5
## b[(Intercept) county:REDWOOD]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:REDWOOD]           0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:RENVILLE]         0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:RENVILLE]          0.1    0.3 -0.2   0.0   0.4
## b[(Intercept) county:RICE]             0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:RICE]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ROCK]             0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:ROCK]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ROSEAU]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:ROSEAU]            0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:SCOTT]            0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:SCOTT]             0.2    0.2  0.0   0.2   0.6
## b[(Intercept) county:SHERBURNE]        0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:SHERBURNE]         0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:SIBLEY]          -0.1    0.1 -0.2   0.0   0.1
## b[floorfirst county:SIBLEY]            0.0    0.3 -0.4   0.0   0.3
## b[(Intercept) county:STEARNS]          0.0    0.1 -0.2   0.0   0.1
```

```
## b[floorfirst county:STEARNS]          -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:STEELE]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:STEELE]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:STEVENS]          0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:STEVENS]           0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:STLOUIS]         -0.2    0.1 -0.3  -0.2  -0.1
## b[floorfirst county:STLOUIS]           0.0    0.2 -0.2   0.0   0.2
## b[(Intercept) county:SWIFT]           -0.1    0.1 -0.3  -0.1   0.1
## b[floorfirst county:SWIFT]             0.0    0.3 -0.4   0.0   0.3
## b[(Intercept) county:TODD]             0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:TODD]              0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) county:TRAVERSE]         0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:TRAVERSE]          0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:WABASHA]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WABASHA]          -0.1    0.3 -0.4   0.0   0.2
## b[(Intercept) county:WADENA]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WADENA]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:WASECA]          -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:WASECA]           -0.1    0.3 -0.5  -0.1   0.2
## b[(Intercept) county:WASHINGTON]       0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:WASHINGTON]       -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:WATONWAN]         0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:WATONWAN]          0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:WILKIN]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WILKIN]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:WINONA]          -0.1    0.1 -0.2  -0.1   0.1
## b[floorfirst county:WINONA]           -0.3    0.3 -0.7  -0.3   0.0
## b[(Intercept) county:WRIGHT]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:WRIGHT]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:YELLOWMEDICINE]   0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:YELLOWMEDICINE]    0.0    0.3 -0.3   0.0   0.3
## sigma                                  0.7    0.0  0.7   0.7   0.7
## Sigma[county:(Intercept),(Intercept)]  0.0    0.0  0.0   0.0   0.0
## Sigma[county:floorfirst,(Intercept)]   0.0    0.0  0.0   0.0   0.0
## Sigma[county:floorfirst,floorfirst]    0.1    0.1  0.0   0.1   0.2
##
## Fit Diagnostics:
##           mean   sd   10%   50%   90%
## mean_PPD 1.3    0.0  1.2   1.3   1.3
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                    mcse Rhat n_eff
## (Intercept)                        0.0  1.0   4105
## log_uranium                        0.0  1.0   3749
## floorfirst                         0.0  1.0   5343
## b[(Intercept) county:AITKIN]       0.0  1.0   6676
## b[floorfirst county:AITKIN]        0.0  1.0   5140
```

```
## b[(Intercept) county:ANOKA]           0.0  1.0   4857
## b[floorfirst county:ANOKA]            0.0  1.0   4087
## b[(Intercept) county:BECKER]          0.0  1.0   8685
## b[floorfirst county:BECKER]           0.0  1.0   5826
## b[(Intercept) county:BELTRAMI]        0.0  1.0   5214
## b[floorfirst county:BELTRAMI]         0.0  1.0   5166
## b[(Intercept) county:BENTON]          0.0  1.0   8646
## b[floorfirst county:BENTON]           0.0  1.0   5050
## b[(Intercept) county:BIGSTONE]        0.0  1.0   7552
## b[floorfirst county:BIGSTONE]         0.0  1.0   6445
## b[(Intercept) county:BLUEEARTH]       0.0  1.0   3866
## b[floorfirst county:BLUEEARTH]        0.0  1.0   2562
## b[(Intercept) county:BROWN]           0.0  1.0   7180
## b[floorfirst county:BROWN]            0.0  1.0   6720
## b[(Intercept) county:CARLTON]         0.0  1.0   4190
## b[floorfirst county:CARLTON]          0.0  1.0   2976
## b[(Intercept) county:CARVER]          0.0  1.0   7837
## b[floorfirst county:CARVER]           0.0  1.0   4966
## b[(Intercept) county:CASS]            0.0  1.0   6804
## b[floorfirst county:CASS]             0.0  1.0   5230
## b[(Intercept) county:CHIPPEWA]        0.0  1.0   8345
## b[floorfirst county:CHIPPEWA]         0.0  1.0   5303
## b[(Intercept) county:CHISAGO]         0.0  1.0   7443
## b[floorfirst county:CHISAGO]          0.0  1.0   7655
## b[(Intercept) county:CLAY]            0.0  1.0   4349
## b[floorfirst county:CLAY]             0.0  1.0   4545
## b[(Intercept) county:CLEARWATER]      0.0  1.0   7136
## b[floorfirst county:CLEARWATER]       0.0  1.0   7706
## b[(Intercept) county:COOK]            0.0  1.0   8810
## b[floorfirst county:COOK]             0.0  1.0   5037
## b[(Intercept) county:COTTONWOOD]      0.0  1.0   3824
## b[floorfirst county:COTTONWOOD]       0.0  1.0   2757
## b[(Intercept) county:CROWWING]        0.0  1.0   5361
## b[floorfirst county:CROWWING]         0.0  1.0   3034
## b[(Intercept) county:DAKOTA]          0.0  1.0   4686
## b[floorfirst county:DAKOTA]           0.0  1.0   4702
## b[(Intercept) county:DODGE]           0.0  1.0   8499
## b[floorfirst county:DODGE]            0.0  1.0   6390
## b[(Intercept) county:DOUGLAS]         0.0  1.0   9021
## b[floorfirst county:DOUGLAS]          0.0  1.0   8247
## b[(Intercept) county:FARIBAULT]       0.0  1.0   2920
## b[floorfirst county:FARIBAULT]        0.0  1.0   4170
## b[(Intercept) county:FILLMORE]        0.0  1.0   6395
## b[floorfirst county:FILLMORE]         0.0  1.0   6011
## b[(Intercept) county:FREEBORN]        0.0  1.0   3995
## b[floorfirst county:FREEBORN]         0.0  1.0   4672
## b[(Intercept) county:GOODHUE]         0.0  1.0   3515
## b[floorfirst county:GOODHUE]          0.0  1.0   2286
## b[(Intercept) county:HENNEPIN]        0.0  1.0   4755
```

```
## b[floorfirst county:HENNEPIN]          0.0  1.0   4644
## b[(Intercept) county:HOUSTON]          0.0  1.0   7673
## b[floorfirst county:HOUSTON]           0.0  1.0   7806
## b[(Intercept) county:HUBBARD]          0.0  1.0   7395
## b[floorfirst county:HUBBARD]           0.0  1.0   6434
## b[(Intercept) county:ISANTI]           0.0  1.0   8983
## b[floorfirst county:ISANTI]            0.0  1.0   6308
## b[(Intercept) county:ITASCA]           0.0  1.0   9944
## b[floorfirst county:ITASCA]            0.0  1.0   6055
## b[(Intercept) county:JACKSON]          0.0  1.0   7463
## b[floorfirst county:JACKSON]           0.0  1.0   6063
## b[(Intercept) county:KANABEC]          0.0  1.0  10118
## b[floorfirst county:KANABEC]           0.0  1.0   5852
## b[(Intercept) county:KANDIYOHI]        0.0  1.0   6135
## b[floorfirst county:KANDIYOHI]         0.0  1.0   5310
## b[(Intercept) county:KITTSON]          0.0  1.0   6935
## b[floorfirst county:KITTSON]           0.0  1.0   6923
## b[(Intercept) county:KOOCHICHING]      0.0  1.0   4607
## b[floorfirst county:KOOCHICHING]       0.0  1.0   3869
## b[(Intercept) county:LACQUIPARLE]      0.0  1.0   4419
## b[floorfirst county:LACQUIPARLE]       0.0  1.0   3499
## b[(Intercept) county:LAKE]             0.0  1.0   3374
## b[floorfirst county:LAKE]              0.0  1.0   4332
## b[(Intercept) county:LAKEOFTHEWOODS]   0.0  1.0   3493
## b[floorfirst county:LAKEOFTHEWOODS]    0.0  1.0   3302
## b[(Intercept) county:LESUEUR]          0.0  1.0   7454
## b[floorfirst county:LESUEUR]           0.0  1.0   5969
## b[(Intercept) county:LINCOLN]          0.0  1.0   6658
## b[floorfirst county:LINCOLN]           0.0  1.0   5474
## b[(Intercept) county:LYON]             0.0  1.0   5929
## b[floorfirst county:LYON]              0.0  1.0   4976
## b[(Intercept) county:MAHNOMEN]         0.0  1.0   8873
## b[floorfirst county:MAHNOMEN]          0.0  1.0   7505
## b[(Intercept) county:MARSHALL]         0.0  1.0   2731
## b[floorfirst county:MARSHALL]          0.0  1.0   1526
## b[(Intercept) county:MARTIN]           0.0  1.0   3509
## b[floorfirst county:MARTIN]            0.0  1.0   3350
## b[(Intercept) county:MCLEOD]           0.0  1.0   4533
## b[floorfirst county:MCLEOD]            0.0  1.0   3535
## b[(Intercept) county:MEEKER]           0.0  1.0   6376
## b[floorfirst county:MEEKER]            0.0  1.0   7562
## b[(Intercept) county:MILLELACS]        0.0  1.0   6182
## b[floorfirst county:MILLELACS]         0.0  1.0   3634
## b[(Intercept) county:MORRISON]         0.0  1.0   5991
## b[floorfirst county:MORRISON]          0.0  1.0   4896
## b[(Intercept) county:MOWER]            0.0  1.0   5330
## b[floorfirst county:MOWER]             0.0  1.0   2371
## b[(Intercept) county:MURRAY]           0.0  1.0   8237
## b[floorfirst county:MURRAY]            0.0  1.0   5789
```

```
## b[(Intercept) county:NICOLLET]        0.0  1.0   5146
## b[floorfirst county:NICOLLET]         0.0  1.0   6781
## b[(Intercept) county:NOBLES]          0.0  1.0  10166
## b[floorfirst county:NOBLES]           0.0  1.0   8941
## b[(Intercept) county:NORMAN]          0.0  1.0   6749
## b[floorfirst county:NORMAN]           0.0  1.0   6122
## b[(Intercept) county:OLMSTED]         0.0  1.0   3165
## b[floorfirst county:OLMSTED]          0.0  1.0   2711
## b[(Intercept) county:OTTERTAIL]       0.0  1.0   5988
## b[floorfirst county:OTTERTAIL]        0.0  1.0   4623
## b[(Intercept) county:PENNINGTON]      0.0  1.0   6353
## b[floorfirst county:PENNINGTON]       0.0  1.0   3569
## b[(Intercept) county:PINE]            0.0  1.0   4297
## b[floorfirst county:PINE]             0.0  1.0   5456
## b[(Intercept) county:PIPESTONE]       0.0  1.0   8412
## b[floorfirst county:PIPESTONE]        0.0  1.0   7143
## b[(Intercept) county:POLK]            0.0  1.0   6966
## b[floorfirst county:POLK]             0.0  1.0   5294
## b[(Intercept) county:POPE]            0.0  1.0   9189
## b[floorfirst county:POPE]             0.0  1.0   5642
## b[(Intercept) county:RAMSEY]          0.0  1.0   5292
## b[floorfirst county:RAMSEY]           0.0  1.0   2295
## b[(Intercept) county:REDWOOD]         0.0  1.0   4994
## b[floorfirst county:REDWOOD]          0.0  1.0   1847
## b[(Intercept) county:RENVILLE]        0.0  1.0   7668
## b[floorfirst county:RENVILLE]         0.0  1.0   4932
## b[(Intercept) county:RICE]            0.0  1.0   5868
## b[floorfirst county:RICE]             0.0  1.0   6418
## b[(Intercept) county:ROCK]            0.0  1.0   6889
## b[floorfirst county:ROCK]             0.0  1.0   5666
## b[(Intercept) county:ROSEAU]          0.0  1.0   4494
## b[floorfirst county:ROSEAU]           0.0  1.0   4026
## b[(Intercept) county:SCOTT]           0.0  1.0   3506
## b[floorfirst county:SCOTT]            0.0  1.0   2255
## b[(Intercept) county:SHERBURNE]       0.0  1.0   7183
## b[floorfirst county:SHERBURNE]        0.0  1.0   7064
## b[(Intercept) county:SIBLEY]          0.0  1.0   5588
## b[floorfirst county:SIBLEY]           0.0  1.0   6110
## b[(Intercept) county:STEARNS]         0.0  1.0   5562
## b[floorfirst county:STEARNS]          0.0  1.0   3797
## b[(Intercept) county:STEELE]          0.0  1.0   8798
## b[floorfirst county:STEELE]           0.0  1.0   5894
## b[(Intercept) county:STEVENS]         0.0  1.0   7257
## b[floorfirst county:STEVENS]          0.0  1.0   7223
## b[(Intercept) county:STLOUIS]         0.0  1.0   2202
## b[floorfirst county:STLOUIS]          0.0  1.0   2909
## b[(Intercept) county:SWIFT]           0.0  1.0   4525
## b[floorfirst county:SWIFT]            0.0  1.0   5587
## b[(Intercept) county:TODD]            0.0  1.0   5758
```

```
## b[floorfirst county:TODD]              0.0  1.0   4091
## b[(Intercept) county:TRAVERSE]         0.0  1.0   8817
## b[floorfirst county:TRAVERSE]          0.0  1.0   6956
## b[(Intercept) county:WABASHA]          0.0  1.0   5966
## b[floorfirst county:WABASHA]           0.0  1.0   3587
## b[(Intercept) county:WADENA]           0.0  1.0   7410
## b[floorfirst county:WADENA]            0.0  1.0   7459
## b[(Intercept) county:WASECA]           0.0  1.0   3752
## b[floorfirst county:WASECA]            0.0  1.0   4439
## b[(Intercept) county:WASHINGTON]       0.0  1.0   5709
## b[floorfirst county:WASHINGTON]        0.0  1.0   3131
## b[(Intercept) county:WATONWAN]         0.0  1.0   3611
## b[floorfirst county:WATONWAN]          0.0  1.0   2784
## b[(Intercept) county:WILKIN]           0.0  1.0   7810
## b[floorfirst county:WILKIN]            0.0  1.0   7393
## b[(Intercept) county:WINONA]           0.0  1.0   3005
## b[floorfirst county:WINONA]            0.0  1.0   1491
## b[(Intercept) county:WRIGHT]           0.0  1.0   6197
## b[floorfirst county:WRIGHT]            0.0  1.0   6820
## b[(Intercept) county:YELLOWMEDICINE]   0.0  1.0   6087
## b[floorfirst county:YELLOWMEDICINE]    0.0  1.0   5993
## sigma                                  0.0  1.0   3159
## Sigma[county:(Intercept),(Intercept)]  0.0  1.0   1339
## Sigma[county:floorfirst,(Intercept)]   0.0  1.0   1380
## Sigma[county:floorfirst,floorfirst]    0.0  1.0    942
## mean_PPD                               0.0  1.0   4625
## log-posterior                          0.5  1.0    732
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

```
## Low n_eff for covariance matrix between random effect so we choose to
## increment iterations
```

```
mod_ex2c <- update(mod_ex2c, iter = 5000)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000131 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.31 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
```

```
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 13.906 seconds (Warm-up)
## Chain 1:                8.444 seconds (Sampling)
## Chain 1:                22.35 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000111 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.11 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 12.618 seconds (Warm-up)
## Chain 2:                8.208 seconds (Sampling)
## Chain 2:                20.826 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000113 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.13 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
```

```
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 13.126 seconds (Warm-up)
## Chain 3:                8.302 seconds (Sampling)
## Chain 3:                21.428 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000111 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.11 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 12.299 seconds (Warm-up)
## Chain 4:                8.597 seconds (Sampling)
## Chain 4:                20.896 seconds (Total)
## Chain 4:
```

```
summary(mod_ex2c)
```

```
##
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      log_radon ~ log_uranium + floor + (1 + floor | county)
##  algorithm:    sampling
##  sample:       10000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 919
##  groups:       county (85)
##
```

```
## Estimates:
##                                          mean   sd   10%   50%   90%
## (Intercept)                               1.5  0.0   1.4   1.5   1.5
## log_uranium                               0.7  0.1   0.6   0.7   0.8
## floorfirst                               -0.6  0.1  -0.7  -0.6  -0.5
## b[(Intercept) county:AITKIN]              0.0  0.1  -0.2   0.0   0.1
## b[floorfirst county:AITKIN]               0.1  0.3  -0.2   0.0   0.4
## b[(Intercept) county:ANOKA]               0.0  0.1  -0.1   0.0   0.1
## b[floorfirst county:ANOKA]                -0.1  0.2  -0.4  -0.1   0.2
## b[(Intercept) county:BECKER]              0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:BECKER]               0.0  0.2  -0.3   0.0   0.3
## b[(Intercept) county:BELTRAMI]            0.1  0.1  -0.1   0.1   0.3
## b[floorfirst county:BELTRAMI]             0.1  0.2  -0.1   0.1   0.4
## b[(Intercept) county:BENTON]              0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:BENTON]               0.0  0.2  -0.3   0.0   0.3
## b[(Intercept) county:BIGSTONE]            0.0  0.1  -0.2   0.0   0.1
## b[floorfirst county:BIGSTONE]             0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:BLUEEARTH]           0.1  0.1   0.0   0.1   0.3
## b[floorfirst county:BLUEEARTH]            0.2  0.3  -0.1   0.2   0.6
## b[(Intercept) county:BROWN]               0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:BROWN]                0.0  0.2  -0.3   0.0   0.3
## b[(Intercept) county:CARLTON]             0.0  0.1  -0.2   0.0   0.1
## b[floorfirst county:CARLTON]              0.1  0.3  -0.2   0.1   0.5
## b[(Intercept) county:CARVER]              0.0  0.1  -0.2   0.0   0.2
## b[floorfirst county:CARVER]               -0.1  0.2  -0.4  -0.1   0.2
## b[(Intercept) county:CASS]                0.1  0.1  -0.1   0.0   0.2
## b[floorfirst county:CASS]                 0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:CHIPPEWA]            0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:CHIPPEWA]             0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:CHISAGO]             0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:CHISAGO]              0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:CLAY]                0.1  0.1  -0.1   0.1   0.2
## b[floorfirst county:CLAY]                 0.0  0.2  -0.3   0.0   0.2
## b[(Intercept) county:CLEARWATER]          0.0  0.1  -0.2   0.0   0.1
## b[floorfirst county:CLEARWATER]           0.0  0.2  -0.3   0.0   0.3
## b[(Intercept) county:COOK]                0.0  0.1  -0.2   0.0   0.1
## b[floorfirst county:COOK]                 0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:COTTONWOOD]         -0.1  0.1  -0.2  -0.1   0.1
## b[floorfirst county:COTTONWOOD]          -0.2  0.2  -0.5  -0.2   0.1
## b[(Intercept) county:CROWWING]            0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:CROWWING]             0.1  0.2  -0.1   0.1   0.5
## b[(Intercept) county:DAKOTA]             -0.1  0.1  -0.2  -0.1   0.0
## b[floorfirst county:DAKOTA]              -0.1  0.2  -0.4  -0.1   0.1
## b[(Intercept) county:DODGE]               0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:DODGE]                0.0  0.3  -0.3   0.0   0.3
## b[(Intercept) county:DOUGLAS]             0.0  0.1  -0.1   0.0   0.2
## b[floorfirst county:DOUGLAS]              0.0  0.3  -0.2   0.0   0.3
## b[(Intercept) county:FARIBAULT]          -0.2  0.2  -0.4  -0.1   0.0
## b[floorfirst county:FARIBAULT]           -0.1  0.3  -0.5  -0.1   0.2
```

```
## b[(Intercept) county:FILLMORE]          0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:FILLMORE]           0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:FREEBORN]          0.1    0.1  0.0   0.1   0.3
## b[floorfirst county:FREEBORN]           0.0    0.2 -0.2   0.0   0.3
## b[(Intercept) county:GOODHUE]           0.1    0.1  0.0   0.1   0.3
## b[floorfirst county:GOODHUE]            0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:HENNEPIN]          0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:HENNEPIN]          -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) county:HOUSTON]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:HOUSTON]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:HUBBARD]           0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:HUBBARD]            0.0    0.2 -0.2   0.0   0.3
## b[(Intercept) county:ISANTI]            0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:ISANTI]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ITASCA]            0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:ITASCA]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:JACKSON]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:JACKSON]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:KANABEC]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:KANABEC]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:KANDIYOHI]         0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:KANDIYOHI]          0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:KITTSON]           0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:KITTSON]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:KOOCHICHING]       0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:KOOCHICHING]        0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:LACQUIPARLE]       0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:LACQUIPARLE]        0.2    0.3 -0.1   0.1   0.5
## b[(Intercept) county:LAKE]             -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:LAKE]               0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:LAKEOFTHEWOODS]    0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:LAKEOFTHEWOODS]     0.2    0.3 -0.1   0.1   0.5
## b[(Intercept) county:LESUEUR]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:LESUEUR]            0.1    0.2 -0.2   0.0   0.4
## b[(Intercept) county:LINCOLN]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:LINCOLN]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:LYON]              0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:LYON]               0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) county:MAHNOMEN]          0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:MAHNOMEN]           0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:MARSHALL]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MARSHALL]          -0.3    0.3 -0.6  -0.2   0.0
## b[(Intercept) county:MARTIN]           -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:MARTIN]            -0.2    0.3 -0.6  -0.1   0.1
## b[(Intercept) county:MCLEOD]           -0.1    0.1 -0.2  -0.1   0.1
## b[floorfirst county:MCLEOD]            -0.2    0.2 -0.5  -0.1   0.1
## b[(Intercept) county:MEEKER]            0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:MEEKER]             0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:MILLELACS]         0.0    0.1 -0.2   0.0   0.1
```

```
## b[floorfirst county:MILLELACS]        -0.2    0.3 -0.5  -0.1   0.1
## b[(Intercept) county:MORRISON]        -0.1    0.1 -0.2   0.0   0.1
## b[floorfirst county:MORRISON]          0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:MOWER]            0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MOWER]            -0.2    0.3 -0.5  -0.1   0.1
## b[(Intercept) county:MURRAY]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:MURRAY]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:NICOLLET]         0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:NICOLLET]          0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) county:NOBLES]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:NOBLES]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:NORMAN]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:NORMAN]           -0.1    0.2 -0.4   0.0   0.2
## b[(Intercept) county:OLMSTED]         -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:OLMSTED]          -0.2    0.3 -0.6  -0.2   0.0
## b[(Intercept) county:OTTERTAIL]        0.1    0.1 -0.1   0.0   0.2
## b[floorfirst county:OTTERTAIL]         0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:PENNINGTON]       0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:PENNINGTON]       -0.1    0.2 -0.5  -0.1   0.1
## b[(Intercept) county:PINE]            -0.1    0.1 -0.3  -0.1   0.1
## b[floorfirst county:PINE]             -0.1    0.3 -0.4  -0.1   0.2
## b[(Intercept) county:PIPESTONE]        0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:PIPESTONE]         0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:POLK]             0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:POLK]             -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) county:POPE]             0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:POPE]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:RAMSEY]           0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:RAMSEY]            0.2    0.2 -0.1   0.1   0.5
## b[(Intercept) county:REDWOOD]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:REDWOOD]           0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:RENVILLE]         0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:RENVILLE]          0.1    0.3 -0.2   0.0   0.4
## b[(Intercept) county:RICE]             0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:RICE]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ROCK]             0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:ROCK]              0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:ROSEAU]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:ROSEAU]            0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) county:SCOTT]            0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:SCOTT]             0.2    0.2  0.0   0.2   0.5
## b[(Intercept) county:SHERBURNE]        0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:SHERBURNE]         0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:SIBLEY]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:SIBLEY]            0.0    0.3 -0.4   0.0   0.3
## b[(Intercept) county:STEARNS]          0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:STEARNS]          -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:STEELE]           0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:STEELE]            0.0    0.3 -0.3   0.0   0.3
```

```
## b[(Intercept) county:STEVENS]          0.0    0.1 -0.2   0.0   0.2
## b[floorfirst county:STEVENS]           0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:STLOUIS]         -0.2    0.1 -0.3  -0.2  -0.1
## b[floorfirst county:STLOUIS]           0.0    0.2 -0.2   0.0   0.2
## b[(Intercept) county:SWIFT]           -0.1    0.1 -0.3  -0.1   0.1
## b[floorfirst county:SWIFT]             0.0    0.3 -0.4   0.0   0.3
## b[(Intercept) county:TODD]             0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:TODD]              0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) county:TRAVERSE]         0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:TRAVERSE]          0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:WABASHA]          0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WABASHA]          -0.1    0.3 -0.4  -0.1   0.2
## b[(Intercept) county:WADENA]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WADENA]            0.0    0.2 -0.2   0.0   0.3
## b[(Intercept) county:WASECA]          -0.1    0.1 -0.3  -0.1   0.0
## b[floorfirst county:WASECA]           -0.1    0.3 -0.5  -0.1   0.2
## b[(Intercept) county:WASHINGTON]       0.0    0.1 -0.1   0.0   0.1
## b[floorfirst county:WASHINGTON]       -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) county:WATONWAN]         0.1    0.1 -0.1   0.1   0.3
## b[floorfirst county:WATONWAN]          0.2    0.3 -0.1   0.2   0.6
## b[(Intercept) county:WILKIN]           0.0    0.1 -0.1   0.0   0.2
## b[floorfirst county:WILKIN]            0.0    0.3 -0.3   0.0   0.3
## b[(Intercept) county:WINONA]          -0.1    0.1 -0.2  -0.1   0.1
## b[floorfirst county:WINONA]           -0.3    0.3 -0.7  -0.3   0.0
## b[(Intercept) county:WRIGHT]           0.1    0.1 -0.1   0.1   0.2
## b[floorfirst county:WRIGHT]            0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) county:YELLOWMEDICINE]   0.0    0.1 -0.2   0.0   0.1
## b[floorfirst county:YELLOWMEDICINE]    0.0    0.3 -0.3   0.0   0.3
## sigma                                  0.7    0.0  0.7   0.7   0.7
## Sigma[county:(Intercept),(Intercept)]  0.0    0.0  0.0   0.0   0.0
## Sigma[county:floorfirst,(Intercept)]   0.0    0.0  0.0   0.0   0.0
## Sigma[county:floorfirst,floorfirst]    0.1    0.1  0.0   0.1   0.2
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 1.3    0.0  1.2   1.3   1.3
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                    mcse Rhat n_eff
## (Intercept)                        0.0  1.0   9991
## log_uranium                        0.0  1.0   8251
## floorfirst                         0.0  1.0  10302
## b[(Intercept) county:AITKIN]       0.0  1.0  15058
## b[floorfirst county:AITKIN]        0.0  1.0  14408
## b[(Intercept) county:ANOKA]        0.0  1.0   9775
## b[floorfirst county:ANOKA]         0.0  1.0   8812
## b[(Intercept) county:BECKER]       0.0  1.0  17935
```

```
## b[floorfirst county:BECKER]           0.0  1.0  14694
## b[(Intercept) county:BELTRAMI]        0.0  1.0  11409
## b[floorfirst county:BELTRAMI]         0.0  1.0  12767
## b[(Intercept) county:BENTON]          0.0  1.0  17154
## b[floorfirst county:BENTON]           0.0  1.0  14402
## b[(Intercept) county:BIGSTONE]        0.0  1.0  16786
## b[floorfirst county:BIGSTONE]         0.0  1.0  16499
## b[(Intercept) county:BLUEEARTH]       0.0  1.0  10635
## b[floorfirst county:BLUEEARTH]        0.0  1.0   6169
## b[(Intercept) county:BROWN]           0.0  1.0  15942
## b[floorfirst county:BROWN]            0.0  1.0  15891
## b[(Intercept) county:CARLTON]         0.0  1.0  13080
## b[floorfirst county:CARLTON]          0.0  1.0   6524
## b[(Intercept) county:CARVER]          0.0  1.0  15980
## b[floorfirst county:CARVER]           0.0  1.0  12173
## b[(Intercept) county:CASS]            0.0  1.0  14426
## b[floorfirst county:CASS]             0.0  1.0  13084
## b[(Intercept) county:CHIPPEWA]        0.0  1.0  16098
## b[floorfirst county:CHIPPEWA]         0.0  1.0  13881
## b[(Intercept) county:CHISAGO]         0.0  1.0  17379
## b[floorfirst county:CHISAGO]          0.0  1.0  14725
## b[(Intercept) county:CLAY]            0.0  1.0  10369
## b[floorfirst county:CLAY]             0.0  1.0  11031
## b[(Intercept) county:CLEARWATER]      0.0  1.0  16578
## b[floorfirst county:CLEARWATER]       0.0  1.0  15933
## b[(Intercept) county:COOK]            0.0  1.0  17628
## b[floorfirst county:COOK]             0.0  1.0  15905
## b[(Intercept) county:COTTONWOOD]      0.0  1.0  10065
## b[floorfirst county:COTTONWOOD]       0.0  1.0   5136
## b[(Intercept) county:CROWWING]        0.0  1.0  12336
## b[floorfirst county:CROWWING]         0.0  1.0   6895
## b[(Intercept) county:DAKOTA]          0.0  1.0  12302
## b[floorfirst county:DAKOTA]           0.0  1.0  10328
## b[(Intercept) county:DODGE]           0.0  1.0  18035
## b[floorfirst county:DODGE]            0.0  1.0  14532
## b[(Intercept) county:DOUGLAS]         0.0  1.0  16361
## b[floorfirst county:DOUGLAS]          0.0  1.0  17403
## b[(Intercept) county:FARIBAULT]       0.0  1.0   8285
## b[floorfirst county:FARIBAULT]        0.0  1.0   9896
## b[(Intercept) county:FILLMORE]        0.0  1.0  18411
## b[floorfirst county:FILLMORE]         0.0  1.0  13739
## b[(Intercept) county:FREEBORN]        0.0  1.0  10503
## b[floorfirst county:FREEBORN]         0.0  1.0  13352
## b[(Intercept) county:GOODHUE]         0.0  1.0  10373
## b[floorfirst county:GOODHUE]          0.0  1.0   6186
## b[(Intercept) county:HENNEPIN]        0.0  1.0  11454
## b[floorfirst county:HENNEPIN]         0.0  1.0   8131
## b[(Intercept) county:HOUSTON]         0.0  1.0  16865
## b[floorfirst county:HOUSTON]          0.0  1.0  15257
```

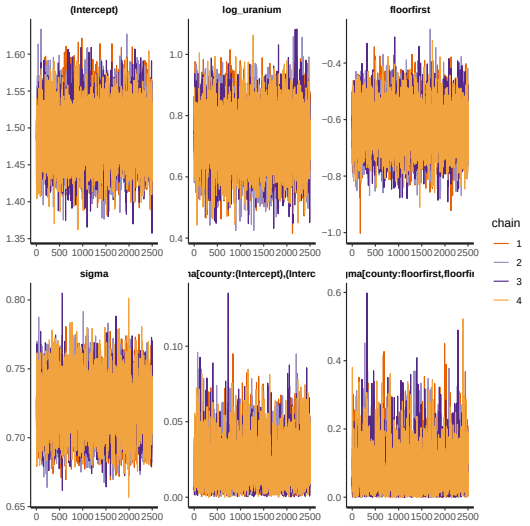
```
## b[(Intercept) county:HUBBARD]          0.0  1.0  14935
## b[floorfirst county:HUBBARD]           0.0  1.0  13784
## b[(Intercept) county:ISANTI]           0.0  1.0  17715
## b[floorfirst county:ISANTI]            0.0  1.0  13061
## b[(Intercept) county:ITASCA]           0.0  1.0  16361
## b[floorfirst county:ITASCA]            0.0  1.0  13166
## b[(Intercept) county:JACKSON]          0.0  1.0  14825
## b[floorfirst county:JACKSON]           0.0  1.0  13730
## b[(Intercept) county:KANABEC]          0.0  1.0  16527
## b[floorfirst county:KANABEC]           0.0  1.0  15890
## b[(Intercept) county:KANDIYOHI]        0.0  1.0  13797
## b[floorfirst county:KANDIYOHI]         0.0  1.0  13476
## b[(Intercept) county:KITTSON]          0.0  1.0  17510
## b[floorfirst county:KITTSON]           0.0  1.0  16795
## b[(Intercept) county:KOOCHICHING]      0.0  1.0  12190
## b[floorfirst county:KOOCHICHING]       0.0  1.0   8427
## b[(Intercept) county:LACQUIPARLE]      0.0  1.0  10512
## b[floorfirst county:LACQUIPARLE]       0.0  1.0   7525
## b[(Intercept) county:LAKE]             0.0  1.0   8741
## b[floorfirst county:LAKE]              0.0  1.0  10224
## b[(Intercept) county:LAKEOFTHEWOODS]   0.0  1.0   9063
## b[floorfirst county:LAKEOFTHEWOODS]    0.0  1.0   7827
## b[(Intercept) county:LESUEUR]          0.0  1.0  18113
## b[floorfirst county:LESUEUR]           0.0  1.0  13828
## b[(Intercept) county:LINCOLN]          0.0  1.0  12051
## b[floorfirst county:LINCOLN]           0.0  1.0  13929
## b[(Intercept) county:LYON]             0.0  1.0  14897
## b[floorfirst county:LYON]              0.0  1.0  11970
## b[(Intercept) county:MAHNOMEN]         0.0  1.0  17247
## b[floorfirst county:MAHNOMEN]          0.0  1.0  12473
## b[(Intercept) county:MARSHALL]         0.0  1.0   5982
## b[floorfirst county:MARSHALL]          0.0  1.0   3132
## b[(Intercept) county:MARTIN]           0.0  1.0   9717
## b[floorfirst county:MARTIN]            0.0  1.0   6725
## b[(Intercept) county:MCLEOD]           0.0  1.0  10630
## b[floorfirst county:MCLEOD]            0.0  1.0   7065
## b[(Intercept) county:MEEKER]           0.0  1.0  15387
## b[floorfirst county:MEEKER]            0.0  1.0  14921
## b[(Intercept) county:MILLELACS]        0.0  1.0  12764
## b[floorfirst county:MILLELACS]         0.0  1.0   8336
## b[(Intercept) county:MORRISON]         0.0  1.0  13092
## b[floorfirst county:MORRISON]          0.0  1.0  13911
## b[(Intercept) county:MOWER]            0.0  1.0  12251
## b[floorfirst county:MOWER]             0.0  1.0   6034
## b[(Intercept) county:MURRAY]           0.0  1.0  18050
## b[floorfirst county:MURRAY]            0.0  1.0  12870
## b[(Intercept) county:NICOLLET]         0.0  1.0  15206
## b[floorfirst county:NICOLLET]          0.0  1.0  13051
## b[(Intercept) county:NOBLES]           0.0  1.0  16213
```

```
## b[floorfirst county:NOBLES]           0.0  1.0  14059
## b[(Intercept) county:NORMAN]          0.0  1.0  16322
## b[floorfirst county:NORMAN]           0.0  1.0  12540
## b[(Intercept) county:OLMSTED]         0.0  1.0   8047
## b[floorfirst county:OLMSTED]          0.0  1.0   6060
## b[(Intercept) county:OTTERTAIL]       0.0  1.0  13432
## b[floorfirst county:OTTERTAIL]        0.0  1.0  10688
## b[(Intercept) county:PENNINGTON]      0.0  1.0  14626
## b[floorfirst county:PENNINGTON]       0.0  1.0   9191
## b[(Intercept) county:PINE]            0.0  1.0  11521
## b[floorfirst county:PINE]             0.0  1.0  12240
## b[(Intercept) county:PIPESTONE]       0.0  1.0  17421
## b[floorfirst county:PIPESTONE]        0.0  1.0  15889
## b[(Intercept) county:POLK]            0.0  1.0  15694
## b[floorfirst county:POLK]             0.0  1.0  10459
## b[(Intercept) county:POPE]            0.0  1.0  17790
## b[floorfirst county:POPE]             0.0  1.0  15116
## b[(Intercept) county:RAMSEY]          0.0  1.0  11119
## b[floorfirst county:RAMSEY]           0.0  1.0   5395
## b[(Intercept) county:REDWOOD]         0.0  1.0  11695
## b[floorfirst county:REDWOOD]          0.0  1.0   4805
## b[(Intercept) county:RENVILLE]        0.0  1.0  18079
## b[floorfirst county:RENVILLE]         0.0  1.0  11363
## b[(Intercept) county:RICE]            0.0  1.0  13343
## b[floorfirst county:RICE]             0.0  1.0  13800
## b[(Intercept) county:ROCK]            0.0  1.0  16780
## b[floorfirst county:ROCK]             0.0  1.0  14276
## b[(Intercept) county:ROSEAU]          0.0  1.0  11341
## b[floorfirst county:ROSEAU]           0.0  1.0  10379
## b[(Intercept) county:SCOTT]           0.0  1.0  10450
## b[floorfirst county:SCOTT]            0.0  1.0   4807
## b[(Intercept) county:SHERBURNE]       0.0  1.0  16293
## b[floorfirst county:SHERBURNE]        0.0  1.0  13660
## b[(Intercept) county:SIBLEY]          0.0  1.0  13565
## b[floorfirst county:SIBLEY]           0.0  1.0  14645
## b[(Intercept) county:STEARNS]         0.0  1.0  14037
## b[floorfirst county:STEARNS]          0.0  1.0   8343
## b[(Intercept) county:STEELE]          0.0  1.0  15223
## b[floorfirst county:STEELE]           0.0  1.0  12926
## b[(Intercept) county:STEVENS]         0.0  1.0  17347
## b[floorfirst county:STEVENS]          0.0  1.0  14881
## b[(Intercept) county:STLOUIS]         0.0  1.0   7071
## b[floorfirst county:STLOUIS]          0.0  1.0   8388
## b[(Intercept) county:SWIFT]           0.0  1.0  11908
## b[floorfirst county:SWIFT]            0.0  1.0  11926
## b[(Intercept) county:TODD]            0.0  1.0  13654
## b[floorfirst county:TODD]             0.0  1.0   8945
## b[(Intercept) county:TRAVERSE]        0.0  1.0  18012
## b[floorfirst county:TRAVERSE]         0.0  1.0  14794
```

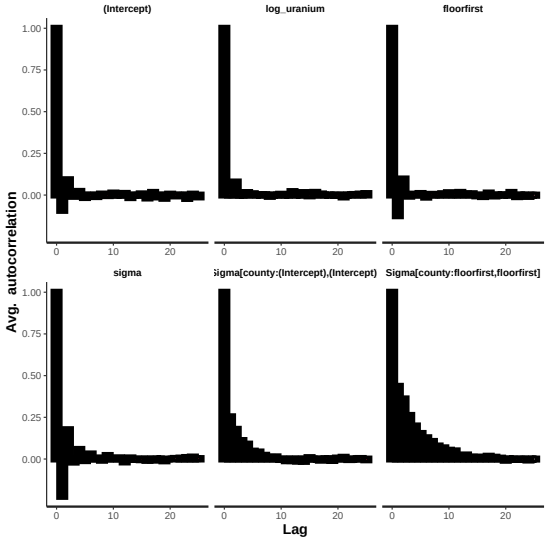
```
## b[(Intercept) county:WABASHA]          0.0  1.0  13297
## b[floorfirst county:WABASHA]           0.0  1.0   8168
## b[(Intercept) county:WADENA]           0.0  1.0  15491
## b[floorfirst county:WADENA]            0.0  1.0  15230
## b[(Intercept) county:WASECA]           0.0  1.0   8991
## b[floorfirst county:WASECA]            0.0  1.0  10052
## b[(Intercept) county:WASHINGTON]       0.0  1.0  12672
## b[floorfirst county:WASHINGTON]        0.0  1.0   7440
## b[(Intercept) county:WATONWAN]         0.0  1.0   8676
## b[floorfirst county:WATONWAN]          0.0  1.0   7173
## b[(Intercept) county:WILKIN]           0.0  1.0  17067
## b[floorfirst county:WILKIN]            0.0  1.0  17331
## b[(Intercept) county:WINONA]           0.0  1.0   7555
## b[floorfirst county:WINONA]            0.0  1.0   3159
## b[(Intercept) county:WRIGHT]           0.0  1.0  13779
## b[floorfirst county:WRIGHT]            0.0  1.0  14542
## b[(Intercept) county:YELLOWMEDICINE]   0.0  1.0  14854
## b[floorfirst county:YELLOWMEDICINE]    0.0  1.0  14237
## sigma                                  0.0  1.0   9103
## Sigma[county:(Intercept),(Intercept)]  0.0  1.0   3934
## Sigma[county:floorfirst,(Intercept)]   0.0  1.0   3109
## Sigma[county:floorfirst,floorfirst]    0.0  1.0   1956
## mean_PPD                               0.0  1.0  12355
## log-posterior                          0.3  1.0   1862
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

Now effective sample size for the critical parameters is increased a bit . . . we procede further with diagnostics (no problems with traceplot and again some autocorrelation for sigmas for random effects, especially variance of random slopes, but seems to be not a big issue)

```
params_2c <- c("(Intercept)", "log_uranium", "floorfirst",
               "sigma",
               "Sigma[county:(Intercept),(Intercept)]",
               "Sigma[county:floorfirst,floorfirst]")
stan_trace(mod_ex2c, pars = params_2c)
```




```
stan_ac(mod_ex2c, pars = params_2c)
```



11.4 Model Choice

Moving to model choice among the three models: we use `waic` and prefer the lowest one

```
waic(mod_ex2a)

## Warning:
## 10 (1.1%) p_waic estimates greater than 0.4.  We recommend trying
loo instead.
```

```
##
## Computed from 4000 by 919 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -1083.7 28.8
## p_waic        44.3  4.0
## waic        2167.4 57.5
##
## 10 (1.1%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(mod_ex2b) # BETTER, waic lower
```

```
## Warning:
## 6 (0.7%) p_waic estimates greater than 0.4.  We recommend trying
loo instead.
```

```
##
## Computed from 4000 by 919 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -1027.5 28.9
## p_waic        25.9  2.3
## waic        2055.1 57.7
##
## 6 (0.7%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
waic(mod_ex2c)
```

```
## Warning:
## 19 (2.1%) p_waic estimates greater than 0.4.  We recommend trying
loo instead.
```

```
##
## Computed from 10000 by 919 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -1028.3 29.5
## p_waic        43.3  5.2
## waic        2056.6 59.0
##
## 19 (2.1%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

11.5 Summary of the better model

Thus we go on studying the best model

```
# summary of the better model
main_pars <- c("(Intercept)", "log_uranium", "floorfirst",
               "sigma", "Sigma[county:(Intercept),(Intercept)]")
```

```
summary(mod_ex2b, pars = main_pars, digits = 3)

##
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      log_radon ~ log_uranium + floor + (1 | county)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 919
##  groups:       county (85)
##
## Estimates:
##                                           mean   sd     10%    50%    90%
## (Intercept)                               1.495  0.037  1.448  1.494  1.541
## log_uranium                               0.700  0.088  0.589  0.700  0.813
## floorfirst                               -0.639  0.067 -0.725 -0.640 -0.552
## sigma                                     0.729  0.018  0.706  0.729  0.753
## Sigma[county:(Intercept),(Intercept)]     0.023  0.015  0.007  0.020  0.044
##
## MCMC diagnostics
##                                          mcse  Rhat  n_eff
## (Intercept)                              0.001 1.001 3007
## log_uranium                              0.002 1.001 2948
## floorfirst                               0.001 1.000 5725
## sigma                                    0.000 1.000 4585
## Sigma[county:(Intercept),(Intercept)] 0.000 1.003  969
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

Focusing on estimates we have all the distribution stuff, with 80% credibility intervals for parameters

11.6 Posterior Intervals

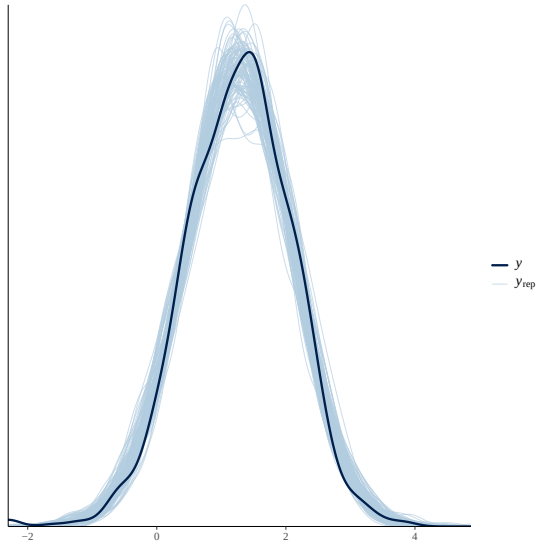
We decide to calculate the 80 and 90% credibility intervals for log_uranium

```
# posterior intervals
posterior_interval(mod_ex2b, prob = 0.8, pars = "log_uranium") # same as summary

##                    10%       90%
## log_uranium 0.5887827 0.8126029

posterior_interval(mod_ex2b, prob = 0.9, pars = "log_uranium") # new

##                     5%       95%
## log_uranium 0.5532326 0.8397326
```

11.7 Posterior predictive checks

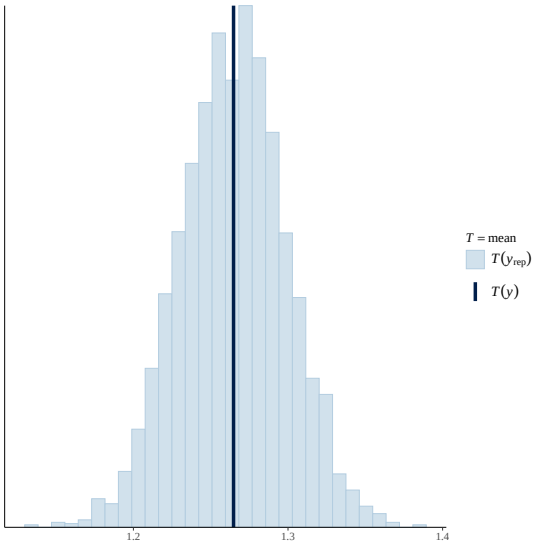
We compare the empirical distribution of original data with model generated data and in this case it's seems very good

```
## Posterior predictive
y_tilde <- posterior_predict(mod_ex2b)
ppc_dens_overlay(y = data2$log_radon, yrep = y_tilde[1100:1200,])
```



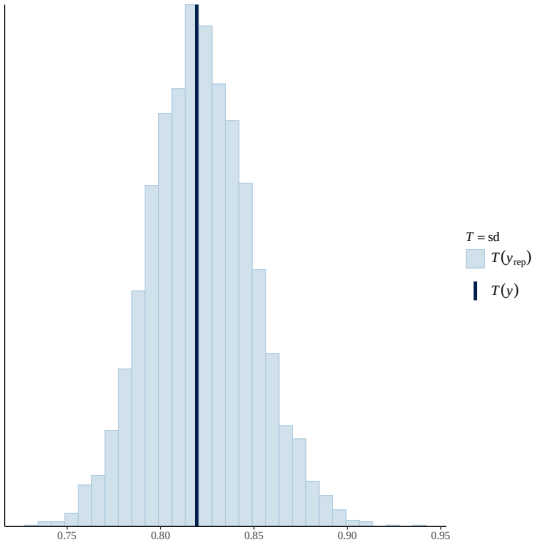
We again check mean, std and so on of generated data to see if the distribution is centered on original sample statistics; again here for mean, sd our model is capable of describing the mean and standard deviation of our data, while the max and especially the min is not good represented. So if we're interested in these latter we should change the model

```
ppc_stat(y = data2$log_radon, yrep = y_tilde, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



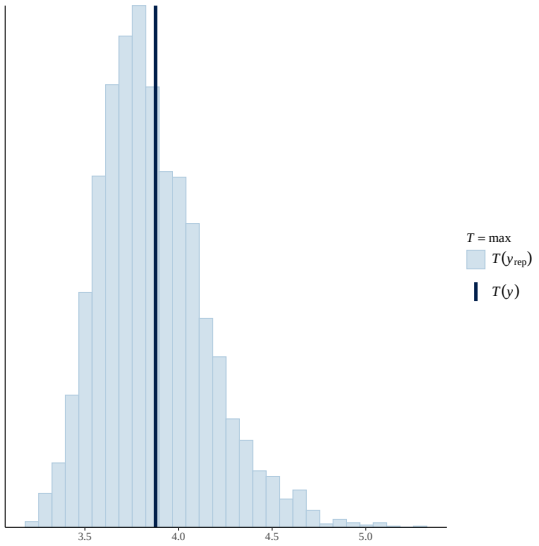

```
ppc_stat(y = data2$log_radon, yrep = y_tilde, stat = "sd")
```

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



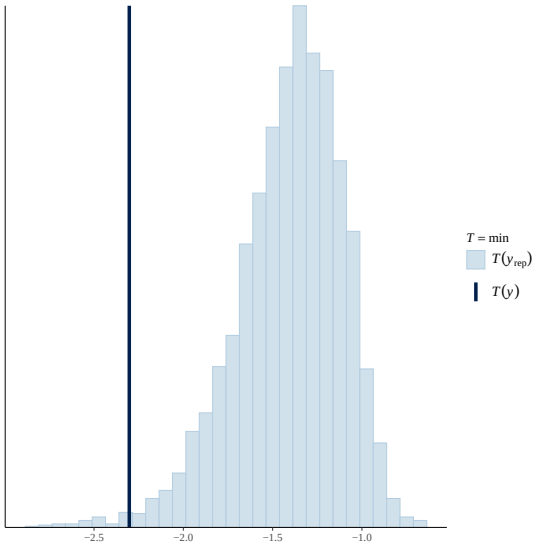

```
ppc_stat(y = data2$log_radon, yrep = y_tilde, stat = "max")
```

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




```
ppc_stat(y = data2$log_radon, yrep = y_tilde, stat = "min")
```

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



Chapter 12

Exercise 3: Logistic Regression

```
# load packages
library(rstanarm)
library(loo)
library(bayesplot)
library(rstan)
library(ggplot2)
set.seed(1234)
```

With a dichotomous outcome variable, the **Logistic Regression Model** is the most common choice:

- it's a GLM with the *Bernoulli (or binomial) distribution* assumed for data;
- the linear predictor (function of the covariate pattern $\mathbf{x}_i$) is specified for the logit of the probability.

The model we look here is

$$y_i|p_i \sim \mathbf{Ber}(p_i),$$
$$\log\left(\frac{p_i}{1-p_i}\right)|\boldsymbol{\beta} = \mathbf{x}_i^T\boldsymbol{\beta}, \quad i = 1, \ldots, n.$$

We use data from a survey about the vote during the 2000 US Presidential elections:

- The response variable = 1 if the subject voted for Bush, 0 otherwise.
- covariates are gender (1=female, 0=male), the race (1=black, 0=other) and the state

```
data3 <- read.csv("data/Data_Ex_3.csv")
if (FALSE) data3 <- read.csv("bayesian_inference/data/Data_Ex_3.csv")
```

```
# data cleaning
data3$state <- factor(data3$state) # state must be a factor
data3$black <- as.integer(data3$black == "yes")    # converting from ..
data3$female <- as.integer(data3$female == "yes") # .. strings to numbers

# overview
str(data3)

## 'data.frame': 2591 obs. of  4 variables:
##  $ bush  : int  1 1 1 0 1 1 0 1 1 1 ...
##  $ black : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ female: int  1 1 1 1 0 1 0 0 0 1 ...
##  $ state : Factor w/ 49 levels "1","3","4","5",..: 40 21 42 42 12 34 16 26 23 24 .
```

We fit two models with different linear predictions:

- **(a) Simple Logistic Regression Model**:

$$\log\left(\frac{p_i}{1-p_i}\right)|\boldsymbol{\beta} = \beta_0 + \beta_1 \texttt{race}_i + \beta_2 \texttt{gender}_i, \quad i = 1, .., n$$

- **(b) Logistic Regression Model with random intercept** considering the $j = 1, \ldots, 49$ states:

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right)|\boldsymbol{\beta} = \beta_0 + \beta_{0[j]} + \beta_1 \texttt{race}_{ij} + \beta_2 \texttt{gender}_{ij}, \quad j = 1, \ldots, 49, \; i = 1, \ldots, n_j.$$

12.1 a) Simple Logistic Regression Model

For the first model we use plain `stan_glm` where we fix number of iterations of MCMC algorithm to 4000 and warmup period (iterations discarded to assure stationary distribution is reached) to 2000, otherwise it would be 2000-1000 for each chain

```
# Simple Logistic model
set.seed(1234)
mod_ex3a <- stan_glm(bush ~ black + female, data = data3, family = "binomial",
                     iter = 4000, warmup = 2000)

##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000129 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.29 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
```

```
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.136 seconds (Warm-up)
## Chain 1:                1.261 seconds (Sampling)
## Chain 1:                2.397 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.8 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.184 seconds (Warm-up)
## Chain 2:                1.311 seconds (Sampling)
## Chain 2:                2.495 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.78 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
```

```
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.137 seconds (Warm-up)
## Chain 3:                1.256 seconds (Sampling)
## Chain 3:                2.393 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7.9e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.79 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.154 seconds (Warm-up)
## Chain 4:                1.253 seconds (Sampling)
## Chain 4:                2.407 seconds (Total)
## Chain 4:
```

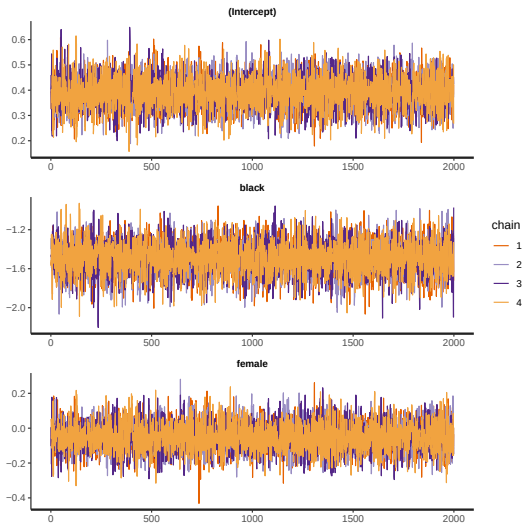
```
summary(mod_ex3a)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      bush ~ black + female
##  algorithm:    sampling
##  sample:       8000 (posterior sample size)
```

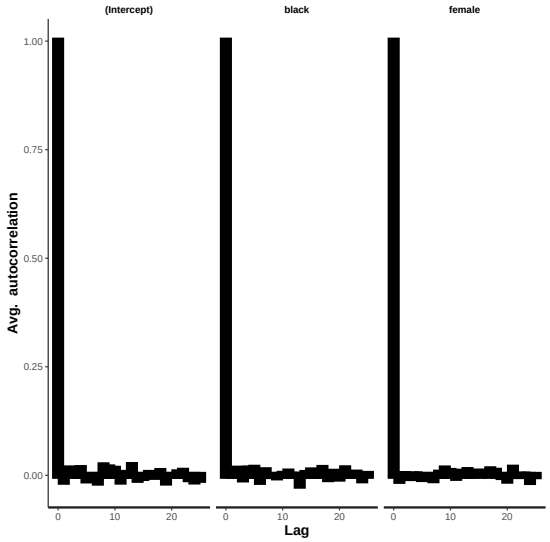
```
##  priors:       see help('prior_summary')
##  observations: 2591
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept)  0.4    0.1  0.3   0.4   0.5
## black       -1.5    0.2 -1.7  -1.5  -1.3
## female       0.0    0.1 -0.2   0.0   0.1
##
## Fit Diagnostics:
##           mean   sd   10%   50%   90%
## mean_PPD 0.6    0.0  0.5   0.6   0.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  7849
## black         0.0  1.0  7639
## female        0.0  1.0  8205
## mean_PPD      0.0  1.0  7427
## log-posterior 0.0  1.0  3156
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

In the MCMC diagnositcs summary there seems to be no problem (n_eff low only for log posterior but as we've said it's no problem). Regarding diagnostics it's all ok

```
stan_trace(mod_ex3a, nrow = 3, ncol=1)
```



```
stan_ac(mod_ex3a)
```



12.2 b) Logistic Regression Model with Random Intercept

We add a random state intercept with `(1|state)` and the rest remains unchanged

```
# Logistic model with random intercept
set.seed(1234)
mod_ex3b <- stan_glmer(bush ~ black + female + (1 | state), data = data3,
                       family = "binomial", iter=4000, warmup = 2000)

##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000329 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.29 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
```

```
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 19.131 seconds (Warm-up)
## Chain 1:                9.786 seconds (Sampling)
## Chain 1:                28.917 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000309 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.09 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 17.933 seconds (Warm-up)
## Chain 2:                9.892 seconds (Sampling)
## Chain 2:                27.825 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000304 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.04 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
```

```
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 17.78 seconds (Warm-up)
## Chain 3:                9.758 seconds (Sampling)
## Chain 3:                27.538 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000298 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.98 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 21.058 seconds (Warm-up)
## Chain 4:                12.303 seconds (Sampling)
## Chain 4:                33.361 seconds (Total)
## Chain 4:
```

```
summary(mod_ex3b)
```

```
##
## Model Info:
##  function:     stan_glmer
##  family:       binomial [logit]
##  formula:      bush ~ black + female + (1 | state)
##  algorithm:    sampling
##  sample:       8000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2591
##  groups:       state (49)
##
## Estimates:
##                                     mean   sd   10%   50%   90%
```

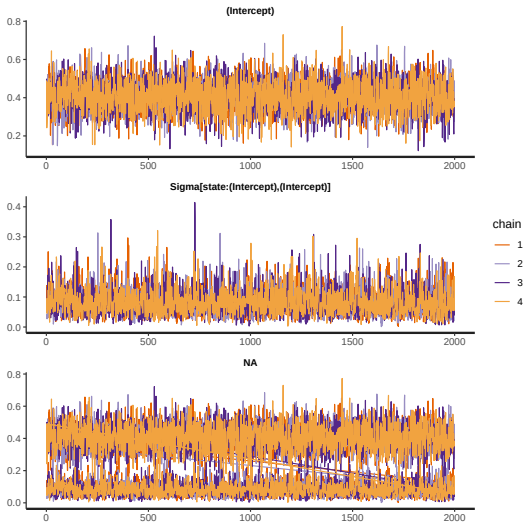
```
## (Intercept)                          0.4    0.1  0.3   0.4   0.5
## black                               -1.5    0.2 -1.8  -1.5  -1.3
## female                               0.0    0.1 -0.2   0.0   0.1
## b[(Intercept) state:1]               0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) state:3]              -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) state:4]               0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) state:5]              -0.1    0.1 -0.2  -0.1   0.1
## b[(Intercept) state:6]              -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) state:7]              -0.3    0.2 -0.6  -0.3   0.0
## b[(Intercept) state:8]               0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) state:9]               0.0    0.3 -0.4   0.0   0.3
## b[(Intercept) state:10]              0.3    0.2  0.0   0.2   0.5
## b[(Intercept) state:11]             -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) state:13]             -0.2    0.3 -0.5  -0.2   0.2
## b[(Intercept) state:14]              0.1    0.2 -0.1   0.1   0.3
## b[(Intercept) state:15]              0.2    0.2 -0.1   0.2   0.4
## b[(Intercept) state:16]             -0.1    0.2 -0.4  -0.1   0.2
## b[(Intercept) state:17]              0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) state:18]              0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) state:19]              0.3    0.2  0.0   0.2   0.6
## b[(Intercept) state:20]              0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) state:21]             -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) state:22]             -0.3    0.2 -0.5  -0.3   0.0
## b[(Intercept) state:23]             -0.1    0.2 -0.3  -0.1   0.1
## b[(Intercept) state:24]             -0.2    0.2 -0.4  -0.2   0.1
## b[(Intercept) state:25]              0.4    0.2  0.1   0.4   0.7
## b[(Intercept) state:26]             -0.1    0.2 -0.4  -0.1   0.1
## b[(Intercept) state:27]             -0.1    0.3 -0.4  -0.1   0.2
## b[(Intercept) state:28]              0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) state:29]              0.0    0.3 -0.3   0.0   0.4
## b[(Intercept) state:30]              0.1    0.3 -0.2   0.1   0.5
## b[(Intercept) state:31]              0.0    0.2 -0.2   0.0   0.2
## b[(Intercept) state:32]              0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) state:33]             -0.3    0.1 -0.5  -0.3  -0.1
## b[(Intercept) state:34]              0.3    0.2  0.0   0.3   0.5
## b[(Intercept) state:35]              0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) state:36]              0.2    0.2  0.0   0.2   0.4
## b[(Intercept) state:37]              0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) state:38]              0.0    0.2 -0.3   0.0   0.3
## b[(Intercept) state:39]             -0.3    0.2 -0.5  -0.3  -0.1
## b[(Intercept) state:40]             -0.2    0.3 -0.5  -0.2   0.1
## b[(Intercept) state:41]              0.3    0.2  0.0   0.3   0.6
## b[(Intercept) state:42]              0.1    0.2 -0.2   0.1   0.4
## b[(Intercept) state:43]              0.1    0.2 -0.1   0.1   0.4
## b[(Intercept) state:44]              0.1    0.1 -0.1   0.1   0.3
## b[(Intercept) state:45]              0.1    0.3 -0.2   0.1   0.4
## b[(Intercept) state:46]             -0.1    0.3 -0.5  -0.1   0.2
## b[(Intercept) state:47]              0.3    0.2  0.1   0.3   0.6
## b[(Intercept) state:48]             -0.1    0.2 -0.4  -0.1   0.1
```

```
## b[(Intercept) state:49]              -0.2    0.2 -0.6  -0.2   0.1
## b[(Intercept) state:50]              -0.2    0.2 -0.4  -0.2   0.1
## b[(Intercept) state:51]              -0.1    0.3 -0.4  -0.1   0.3
## Sigma[state:(Intercept),(Intercept)]  0.1    0.0  0.0   0.1   0.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.6    0.0  0.5   0.6   0.6
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                     mcse Rhat n_eff
## (Intercept)                         0.0  1.0   6357
## black                               0.0  1.0  11397
## female                              0.0  1.0  12590
## b[(Intercept) state:1]              0.0  1.0   9801
## b[(Intercept) state:3]              0.0  1.0  10986
## b[(Intercept) state:4]              0.0  1.0  11225
## b[(Intercept) state:5]              0.0  1.0   7891
## b[(Intercept) state:6]              0.0  1.0  11645
## b[(Intercept) state:7]              0.0  1.0   8139
## b[(Intercept) state:8]              0.0  1.0  11673
## b[(Intercept) state:9]              0.0  1.0  10169
## b[(Intercept) state:10]             0.0  1.0   9390
## b[(Intercept) state:11]             0.0  1.0  11461
## b[(Intercept) state:13]             0.0  1.0  10228
## b[(Intercept) state:14]             0.0  1.0  11156
## b[(Intercept) state:15]             0.0  1.0   9858
## b[(Intercept) state:16]             0.0  1.0  10247
## b[(Intercept) state:17]             0.0  1.0  10597
## b[(Intercept) state:18]             0.0  1.0  11239
## b[(Intercept) state:19]             0.0  1.0   8289
## b[(Intercept) state:20]             0.0  1.0   9964
## b[(Intercept) state:21]             0.0  1.0  10485
## b[(Intercept) state:22]             0.0  1.0   8615
## b[(Intercept) state:23]             0.0  1.0  10865
## b[(Intercept) state:24]             0.0  1.0  10418
## b[(Intercept) state:25]             0.0  1.0   6439
## b[(Intercept) state:26]             0.0  1.0  10454
## b[(Intercept) state:27]             0.0  1.0  10058
## b[(Intercept) state:28]             0.0  1.0  11236
## b[(Intercept) state:29]             0.0  1.0   9286
## b[(Intercept) state:30]             0.0  1.0  10223
## b[(Intercept) state:31]             0.0  1.0  11436
## b[(Intercept) state:32]             0.0  1.0  12524
## b[(Intercept) state:33]             0.0  1.0   8950
## b[(Intercept) state:34]             0.0  1.0   8675
## b[(Intercept) state:35]             0.0  1.0  10669
```

```
## b[(Intercept) state:36]                0.0  1.0  10176
## b[(Intercept) state:37]                0.0  1.0  11860
## b[(Intercept) state:38]                0.0  1.0  10713
## b[(Intercept) state:39]                0.0  1.0   9187
## b[(Intercept) state:40]                0.0  1.0   9179
## b[(Intercept) state:41]                0.0  1.0   7642
## b[(Intercept) state:42]                0.0  1.0  10462
## b[(Intercept) state:43]                0.0  1.0  10220
## b[(Intercept) state:44]                0.0  1.0   9893
## b[(Intercept) state:45]                0.0  1.0  10436
## b[(Intercept) state:46]                0.0  1.0  10708
## b[(Intercept) state:47]                0.0  1.0   8250
## b[(Intercept) state:48]                0.0  1.0   9955
## b[(Intercept) state:49]                0.0  1.0   8809
## b[(Intercept) state:50]                0.0  1.0   9970
## b[(Intercept) state:51]                0.0  1.0  11149
## Sigma[state:(Intercept),(Intercept)] 0.0  1.0   3243
## mean_PPD                               0.0  1.0   9602
## log-posterior                          0.2  1.0   1824
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

In the MCMC diagnostic all is good, as well as traceplot; for the autocorrelation plot we have some residual autocorrelation in the variance for random effect ($\sigma^2_{\beta_0}$) butno big deal (go to zero after 5 or 6 lag)

```
params <- c("(Intercept)", "blackyes", "femaleyes", "Sigma[state:(Intercept),(Intercept)]")
stan_trace(mod_ex3b, nrow = 4, ncol = 1, pars = params)
```



```
stan_ac(mod_ex3b, pars = params)

## Error in factor(rep(1:np, each = nc * nl), labels = levels(pa)):
'labels' non valido; la lunghezza 2 dovrebbe essere 1 o 3
```

12.3 Model Comparison

```
# Comparison
waic(mod_ex3a)

##
## Computed from 8000 by 2591 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -1732.1 11.4
## p_waic         3.1  0.1
## waic        3464.1 22.8

waic(mod_ex3b)

##
## Computed from 8000 by 2591 log-likelihood matrix.
##
##           Estimate   SE
## elpd_waic  -1722.7 12.1
## p_waic        22.5  0.3
## waic        3445.4 24.2
```

second model seems better since WAIC is lower

12.4 Summary of the better model

here we look at the estimates

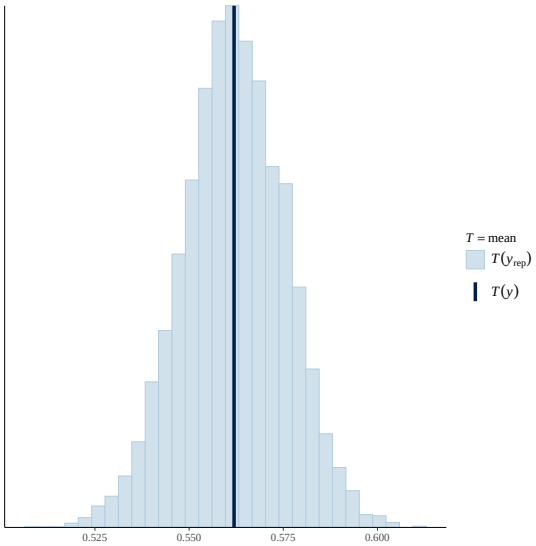
```
main_pars <- c("(Intercept)","blackyes","femaleyes","Sigma[state:(Intercept),(Intercep
summary(mod_ex3b, pars = main_pars, digits = 3)

##
## Model Info:
##  function:     stan_glmer
##  family:       binomial [logit]
##  formula:      bush ~ black + female + (1 | state)
##  algorithm:    sampling
##  sample:       8000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2591
##  groups:       state (49)
```

```
##
## Estimates:
##                                       mean   sd    10%   50%   90%
## (Intercept)                          0.407  0.080 0.305 0.406 0.509
## Sigma[state:(Intercept),(Intercept)] 0.082  0.042 0.035 0.075 0.136
##
## MCMC diagnostics
##                                      mcse  Rhat  n_eff
## (Intercept)                          0.001 1.000 6357
## Sigma[state:(Intercept),(Intercept)] 0.001 1.000 3243
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

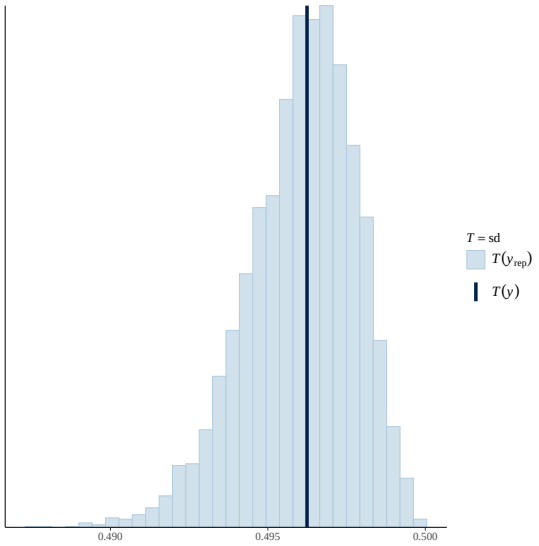
12.5 Posterior predictive checks

here the mean and the sd of the data seems to be quite centered, while the new `sum` (total number of bush voters in the model generated samples) is centered as well

```
y_tilde <- posterior_predict(mod_ex3b)
ppc_stat(y = data3$bush, yrep = y_tilde, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```

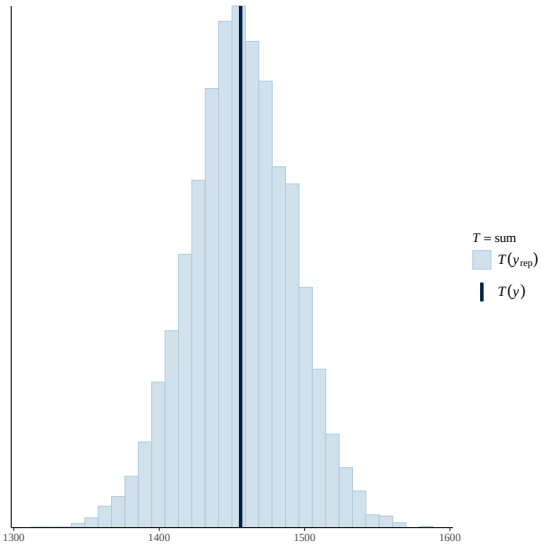



```
ppc_stat(y = data3$bush, yrep = y_tilde, stat = "sd")

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




```
ppc_stat(y = data3$bush, yrep = y_tilde, stat = "sum")
```

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

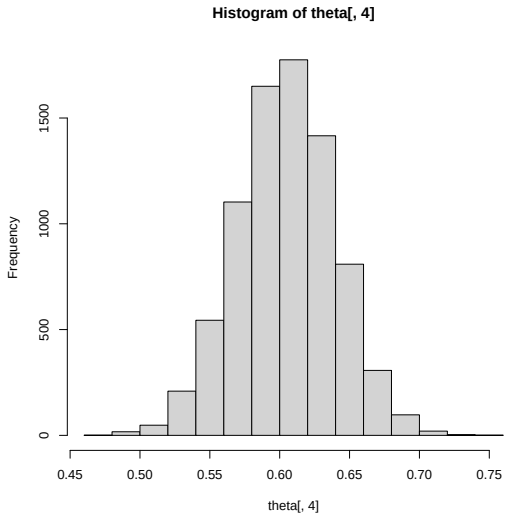


12.6 Posterior Inference

Focusing our attention on subject 4 we can look at its estimated posterior probability: in this case with `posterior_linpred` we have to specify `transform=TRUE` (to get the probability, otherwise we get the logit)

```
# Posterior inference
# Estimated probability for subject 4
```

```
theta <- posterior_linpred(mod_ex3b, transform = TRUE)

## Instead of posterior_linpred(..., transform=TRUE) please call posterior_epred(),
which provides equivalent functionality.

hist(theta[,4])
```




```
mean(theta[,4])

## [1] 0.6044628

sd(theta[,4])

## [1] 0.03493385

## 95% credibility interval
quantile(theta[,4], probs = c(0.025, 0.5, 0.975))

##      2.5%       50%     97.5%
## 0.5355086 0.6045734 0.6725267
```

12.7 [Extra] Additional models

- **(c) Logistic Regression Model with random effect on variable `race`:**

  $$\log\left(\frac{p_{ij}}{1-p_{ij}}\right)|\boldsymbol{\beta} = \beta_0 + \beta_{1[j]}\texttt{race}_{ij} + \beta_2\texttt{gender}_{ij}, \quad j = 1,\ldots,49, \; i = 1,\ldots,n_j.$$

  is estimated through

```
mod_ex3c <- stan_glmer(bush ~ black + female + (black|state),  data = data3,
                       family = "binomial")
```

- **(d) Logistic Regression Model with random effect on intercept and both variables**:

$$\log\left(\frac{p_{ij}}{1-p_{ij}}\right)|\boldsymbol{\beta} = \beta_0 + \beta_{0[j]} + \beta_{1[j]}\texttt{race}_{ij} + \beta_{2[j]}\texttt{gender}_{ij},$$
$$j = 1, \ldots, 49,\ i = 1, \ldots, n_j.$$

  is estimated through

```
mod_ex3d <- stan_glmer(bush ~ black + female + (1+black+female|state),
                       data = data3, family = "binomial")
```

Chapter 13

Exercise 4: Poisson Regression

```
# load packages
library(rstanarm)
library(loo)
library(bayesplot)
library(rstan)
library(ggplot2)
set.seed(1234)
```

A dose-response study on mutagenicity on salmonella bacteria:

- outcome: number of colonies of Salmonella counted y_{ij};
- treatment: quinoline (liquid organic compound) at 6 possible level/doses ($x_i;\ i = 1, \ldots, 6$);
- groups: three plates ($j = 1, 2, 3$), each treated at every treatment level/-dose.

We want to:

- compare a simple Poisson regression model and a Poisson model with random effects in terms of goodness of fit;
- study the effect of `quinoline` both at the measurement scale and at a logarithmic scale [1] (in the same model/simultaneously);
- start from a $\mathcal{N}(0, 10)$ prior for the $\boldsymbol{\beta}$ parameters and define a $\mathcal{N}(0, c)$ weakly informative prior.
  This sentence on priors means we can use the fact that rstanarm adjusts the priors: we start from a normal 0-10 and then using **autoscaling** in order to define an *optimal weakly informative prior*.

Remarks:

[1] here log quinoline is defined as $log(\texttt{quinoline} + 10)$

- Poisson distribution has only one parameter ($\mathbb{E}[Y] = \mathbb{V}[Y] = \lambda$), and in fact the variance increases with the mean.
  So when the mean increases the variance increases at the same level but in practice this results often in overdispersion or underdispersion;
- Poisson model with random effects is usually used *in order to take into account the eventual presence of overdispersion or underdispersion*;

```
(data4 <- read.csv("data/Data_Ex_4.csv"))
```

```
##    colonies quinoline log_quinoline plate
## 1        15         0      2.302585     A
## 2        16        10      2.995732     A
## 3        16        33      3.761200     A
## 4        27       100      4.700480     A
## 5        33       333      5.837730     A
## 6        20      1000      6.917706     A
## 7        21         0      2.302585     B
## 8        18        10      2.995732     B
## 9        26        33      3.761200     B
## 10       41       100      4.700480     B
## 11       38       333      5.837730     B
## 12       27      1000      6.917706     B
## 13       29         0      2.302585     C
## 14       21        10      2.995732     C
## 15       33        33      3.761200     C
## 16       60       100      4.700480     C
## 17       41       333      5.837730     C
## 18       42      1000      6.917706     C
```

```
if (FALSE) data4 <- read.csv("bayesian_inference/data/Data_Ex_4.csv")
data4$plate <- factor(data4$plate)
str(data4)
```

```
## 'data.frame': 18 obs. of  4 variables:
##  $ colonies     : int  15 16 16 27 33 20 21 18 26 41 ...
##  $ quinoline    : int  0 10 33 100 333 1000 0 10 33 100 ...
##  $ log_quinoline: num  2.3 3 3.76 4.7 5.84 ...
##  $ plate        : Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 2 2 2 2 ...
```

13.1 Simple Poisson Model

Bayesian formulation of the simple **Poisson regression model** is:

$$y_i|\mu_i \sim Poisson(\mu_i)$$

$$\log(\mu_i)|\boldsymbol{\beta} = \beta_0 + \beta_1 \texttt{log\_quinoline}_i + \beta_2 \texttt{quinoline}_i, \quad i = 1, \ldots, n.$$

$$\beta_k \sim \mathcal{N}(0, c), \quad k = 0, 1, 2;$$

The estimation with auto-adjusting priors (`autoscale=TRUE`) follows; rstanarm starts from a normal 0-10 and then it adjust the prior to obtain an optimal weakly informative prior

```
## First model estimation and priors check
set.seed(1234)
mod_ex4a <- stan_glm(
  colonies ~ quinoline + log_quinoline,
  data = data4,
  family = "poisson", #new
  prior = normal(0, 10, autoscale = TRUE),
  prior_intercept = normal(0, 10, autoscale = TRUE))

##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.07 seconds (Warm-up)
## Chain 1:                0.074 seconds (Sampling)
## Chain 1:                0.144 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.061 seconds (Warm-up)
## Chain 2:                0.067 seconds (Sampling)
## Chain 2:                0.128 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.066 seconds (Warm-up)
## Chain 3:                0.068 seconds (Sampling)
## Chain 3:                0.134 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.07 seconds (Warm-up)
## Chain 4:                0.068 seconds (Sampling)
## Chain 4:                0.138 seconds (Total)
## Chain 4:
```

```
prior_summary(mod_ex4a)
```

```
## Priors for model 'mod_ex4a'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = [0,0], scale = [10,10])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [0.027,6.091])
## ------
## See help('prior_summary.stanreg') for more details
```

The adjusted prior in the latter command is the resulting prior after optimization (always weakly informative).
Looking at MCMC diagnostics we can see that effective sample size is a bit lower so we decide to increase iterations

```
# Convergence
summary(mod_ex4a)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      colonies ~ quinoline + log_quinoline
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 18
##  predictors:   3
##
## Estimates:
```

```
##                 mean   sd   10%   50%   90%
## (Intercept)   2.2    0.2  1.9   2.2   2.4
## quinoline     0.0    0.0  0.0   0.0   0.0
## log_quinoline 0.3    0.1  0.3   0.3   0.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 29.1    1.8 26.8  29.2  31.4
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  1619
## quinoline     0.0  1.0  1528
## log_quinoline 0.0  1.0  1558
## mean_PPD      0.0  1.0  2658
## log-posterior 0.0  1.0  1445
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

```
# increasing iterations
mod_ex4a <- update(mod_ex4a, iter=4000)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.17 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.127 seconds (Warm-up)
## Chain 1:                0.138 seconds (Sampling)
## Chain 1:                0.265 seconds (Total)
## Chain 1:
##
```

```
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.13 seconds (Warm-up)
## Chain 2:                0.144 seconds (Sampling)
## Chain 2:                0.274 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.127 seconds (Warm-up)
## Chain 3:                0.133 seconds (Sampling)
## Chain 3:                0.26 seconds (Total)
## Chain 3:
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.129 seconds (Warm-up)
## Chain 4:                0.142 seconds (Sampling)
## Chain 4:                0.271 seconds (Total)
## Chain 4:
```

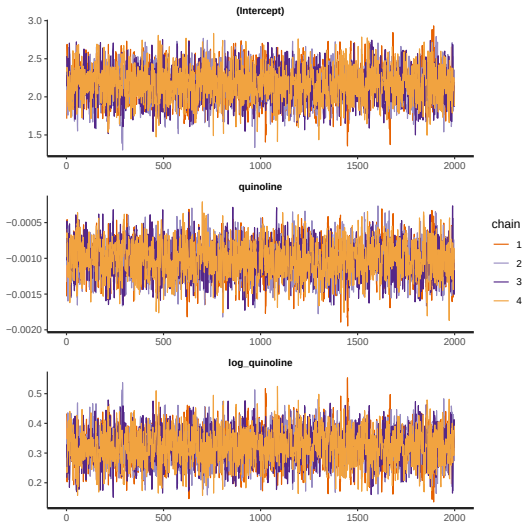
```
summary(mod_ex4a) # n_eff is increased
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      colonies ~ quinoline + log_quinoline
##  algorithm:    sampling
##  sample:       8000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 18
##  predictors:   3
##
## Estimates:
##                 mean   sd   10%   50%   90%
## (Intercept)   2.2    0.2  1.9   2.2   2.4
## quinoline     0.0    0.0  0.0   0.0   0.0
## log_quinoline 0.3    0.1  0.2   0.3   0.4
##
## Fit Diagnostics:
##           mean   sd   10%   50%   90%
## mean_PPD 29.1    1.8 26.8  29.1  31.4
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
```

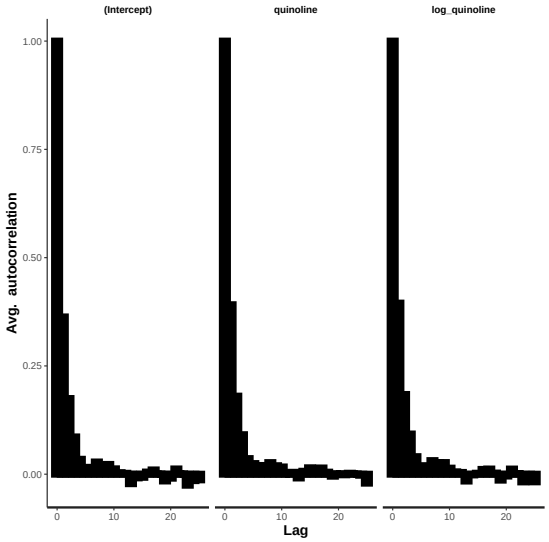
```
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  3153
## quinoline     0.0  1.0  3058
## log_quinoline 0.0  1.0  2969
## mean_PPD      0.0  1.0  5259
## log-posterior 0.0  1.0  2965
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

so effective sample size is iscreased, still not optimal but better than before. Looking at convergence traceplots seems to be ok, there's some residual autocorrelation in the first lags but nothing serious (we can say this model is acceptable)

```
# Convergence
stan_trace(mod_ex4a, nrow = 3, ncol = 1)
```

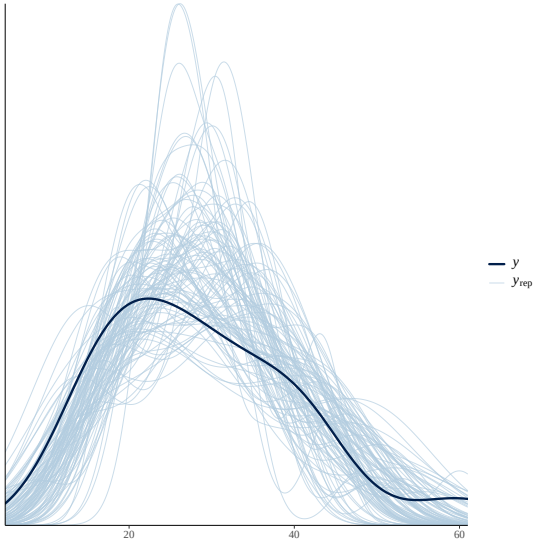



```
stan_ac(mod_ex4a)
```



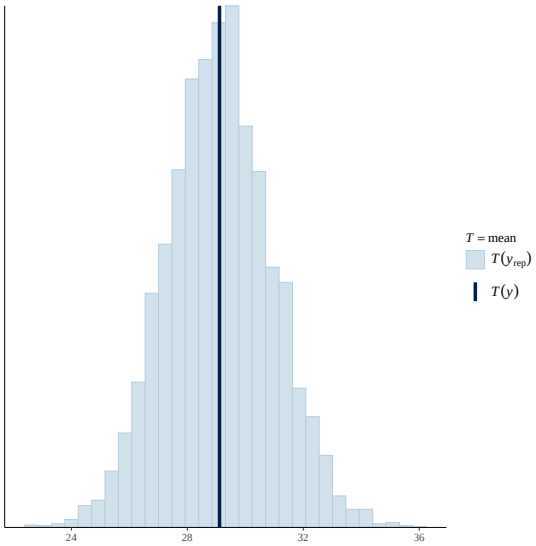
Finally, looking at posterior predictive checks on the left side of density our replicates have higher probability, while in the middle less probability than the actual sample data; so model constantly underestimate original data. This is due to the fact that poisson regression has this problem with over/underdispersion most of the times

```
## model generated data
y_tilde4a <- posterior_predict(mod_ex4a)
ppc_dens_overlay(y = data4$colonies, yrep = y_tilde4a[1100:1200,])
```

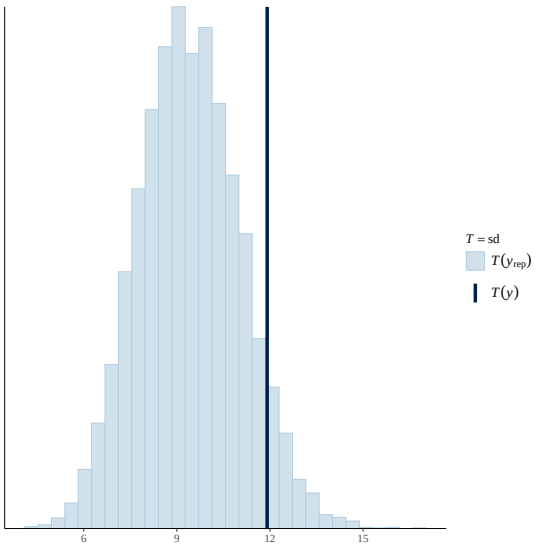


Overdispersion is quite clear looking at positive predictive checks for mean (which is ok) and sd (where the model produce less variability than the actual data). Finally if we look at minimum is acceptable but the maximum is totally underestimated by the model

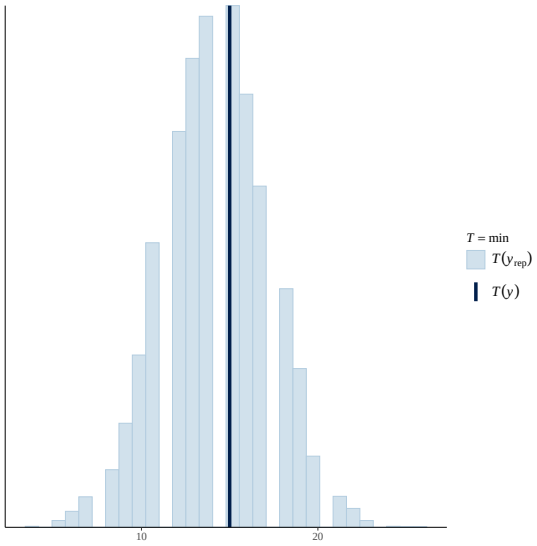
```
ppc_stat(y = data4$colonies, yrep = y_tilde4a, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```




```
ppc_stat(y = data4$colonies, yrep = y_tilde4a, stat = "sd")

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

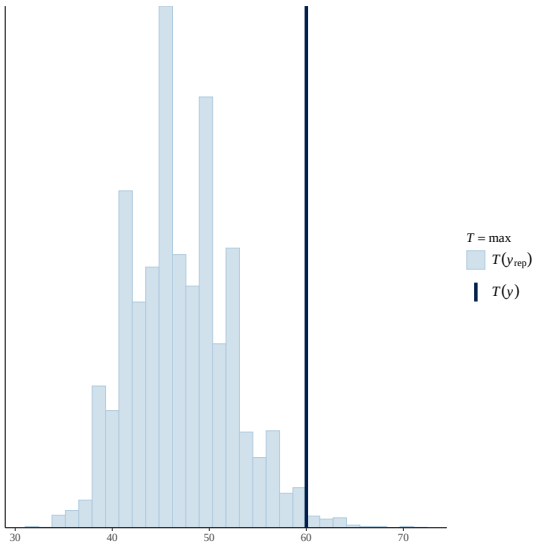



```
ppc_stat(y = data4$colonies, yrep = y_tilde4a, stat = "min")

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




```
ppc_stat(y = data4$colonies, yrep = y_tilde4a, stat = "max")
```

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



13.2 Poisson Regression Model with random effects

To take into account the eventual presence of overdispersion, a random effect λ_j which is *plate-specific* (it has the same structure of a random intercept seen before) is included in the linear predictor.

Thus the **Likelihood** becomes:

$$y_{ij}|\mu_{ij} \sim Poisson(\mu_{ij})$$
$$\log(\mu_{ij})|\boldsymbol{\beta}, \lambda_j = \beta_0 + \beta_1 \texttt{log\_quinoline}_{ij} + \beta_2 \texttt{quinoline}_{ij} + \lambda_j$$

The introduced random effects has its own prior which is normal (depending on a variance σ^2_λ) and thus, overall, **Priors** are:

$$\lambda_j|\sigma^2_\lambda \sim \mathcal{N}(0, \sigma^2_\lambda), \quad j = 1, 2, 3;$$
$$\beta_k \sim \mathcal{N}(0, c), \quad k = 0, 1, 2.$$

and **Hyperprior** for variance of newly introduced parameter:

$$\sigma_\lambda \sim \pi(\sigma_\lambda).$$

```
####################
# Poisson Mixed GLM
####################
set.seed(1234)
mod_ex4b <- stan_glmer( # new
  colonies ~ quinoline + log_quinoline + (1|plate), # new
  data = data4,
  family = "poisson",
  prior = normal(0, 10, autoscale = TRUE),
  prior_intercept = normal(0, 10, autoscale = TRUE))

##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.489 seconds (Warm-up)
## Chain 1:                0.299 seconds (Sampling)
## Chain 1:                0.788 seconds (Total)
```

```
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.197 seconds (Warm-up)
## Chain 2:                0.364 seconds (Sampling)
## Chain 2:                1.561 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.402 seconds (Warm-up)
## Chain 3:                0.326 seconds (Sampling)
```

```
## Chain 3:                0.728 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.589 seconds (Warm-up)
## Chain 4:                0.273 seconds (Sampling)
## Chain 4:                0.862 seconds (Total)
## Chain 4:

## Warning:  There were 12 divergent transitions after warmup.  See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
## Warning:  Examine the pairs() plot to diagnose sampling problems

prior_summary(mod_ex4b)

## Priors for model 'mod_ex4b'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = [0,0], scale = [10,10])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [0.027,6.091])
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

in the prior after introducing the random effect we have the covariance prior as well (che direi sia una matrice di varianza e covarianza 1×1 con una sola varianza, quella della random intercept).
Looking at convergence

- effective sample size is low so we increase number of iterations (a lot, to 8000 since the effective sample size is very low)
- furthermore we had **warnings** (`There were 12 divergent transitions after warmup`) that are **classical** when we work with the Poisson regression with random effects.
  It means that after discarding the warmup iterations in the chain, there are still 12 iterations that didn't reached the stationary distribution.
  We have two solutions:
  - we increase the warmup period in order to discard also this divergent transitions
  - we icrease `adapt_delta` which is the target average proposal acceptance probability;
    * while generating values using MCMC, the algorithm accepts these values with certain probability, by default set to 0.95 (thus we still have 0.05 probability to retain a value which is not optimal)
    * in general one should not need to change `adapt_delta` unless a warning message about divergent transitions is raised; in that case increase `adapt_delta` from the default to a value closer to 1 (e.g. to 0.99, or to 0.999, etc);
    * the step size used by the numerical integrator is a function of `adapt_delta` in that increasing `adapt_delta` will result in a smaller step size and fewer divergences;
    * Increasing `adapt_delta` will typically result in a slower sampler, but it will always lead to a more robust sampler.

```
set.seed(1234)
mod_ex4b <- update(mod_ex4b, iter = 8000, adapt_delta=.99)

##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 8000 [  0%]  (Warmup)
## Chain 1: Iteration:  800 / 8000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1600 / 8000 [ 20%]  (Warmup)
## Chain 1: Iteration: 2400 / 8000 [ 30%]  (Warmup)
## Chain 1: Iteration: 3200 / 8000 [ 40%]  (Warmup)
## Chain 1: Iteration: 4000 / 8000 [ 50%]  (Warmup)
```

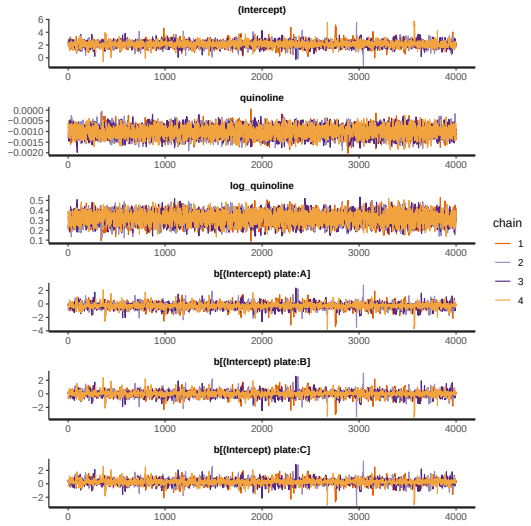
```
## Chain 1: Iteration: 4001 / 8000 [ 50%]  (Sampling)
## Chain 1: Iteration: 4800 / 8000 [ 60%]  (Sampling)
## Chain 1: Iteration: 5600 / 8000 [ 70%]  (Sampling)
## Chain 1: Iteration: 6400 / 8000 [ 80%]  (Sampling)
## Chain 1: Iteration: 7200 / 8000 [ 90%]  (Sampling)
## Chain 1: Iteration: 8000 / 8000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.794 seconds (Warm-up)
## Chain 1:                2.595 seconds (Sampling)
## Chain 1:                5.389 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 8000 [  0%]  (Warmup)
## Chain 2: Iteration:  800 / 8000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1600 / 8000 [ 20%]  (Warmup)
## Chain 2: Iteration: 2400 / 8000 [ 30%]  (Warmup)
## Chain 2: Iteration: 3200 / 8000 [ 40%]  (Warmup)
## Chain 2: Iteration: 4000 / 8000 [ 50%]  (Warmup)
## Chain 2: Iteration: 4001 / 8000 [ 50%]  (Sampling)
## Chain 2: Iteration: 4800 / 8000 [ 60%]  (Sampling)
## Chain 2: Iteration: 5600 / 8000 [ 70%]  (Sampling)
## Chain 2: Iteration: 6400 / 8000 [ 80%]  (Sampling)
## Chain 2: Iteration: 7200 / 8000 [ 90%]  (Sampling)
## Chain 2: Iteration: 8000 / 8000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 4.15 seconds (Warm-up)
## Chain 2:                2.853 seconds (Sampling)
## Chain 2:                7.003 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 8000 [  0%]  (Warmup)
## Chain 3: Iteration:  800 / 8000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1600 / 8000 [ 20%]  (Warmup)
## Chain 3: Iteration: 2400 / 8000 [ 30%]  (Warmup)
## Chain 3: Iteration: 3200 / 8000 [ 40%]  (Warmup)
```

```
## Chain 3: Iteration: 4000 / 8000 [ 50%]  (Warmup)
## Chain 3: Iteration: 4001 / 8000 [ 50%]  (Sampling)
## Chain 3: Iteration: 4800 / 8000 [ 60%]  (Sampling)
## Chain 3: Iteration: 5600 / 8000 [ 70%]  (Sampling)
## Chain 3: Iteration: 6400 / 8000 [ 80%]  (Sampling)
## Chain 3: Iteration: 7200 / 8000 [ 90%]  (Sampling)
## Chain 3: Iteration: 8000 / 8000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.9 seconds (Warm-up)
## Chain 3:                2.683 seconds (Sampling)
## Chain 3:                6.583 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 8000 [  0%]  (Warmup)
## Chain 4: Iteration:  800 / 8000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1600 / 8000 [ 20%]  (Warmup)
## Chain 4: Iteration: 2400 / 8000 [ 30%]  (Warmup)
## Chain 4: Iteration: 3200 / 8000 [ 40%]  (Warmup)
## Chain 4: Iteration: 4000 / 8000 [ 50%]  (Warmup)
## Chain 4: Iteration: 4001 / 8000 [ 50%]  (Sampling)
## Chain 4: Iteration: 4800 / 8000 [ 60%]  (Sampling)
## Chain 4: Iteration: 5600 / 8000 [ 70%]  (Sampling)
## Chain 4: Iteration: 6400 / 8000 [ 80%]  (Sampling)
## Chain 4: Iteration: 7200 / 8000 [ 90%]  (Sampling)
## Chain 4: Iteration: 8000 / 8000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.528 seconds (Warm-up)
## Chain 4:                3.37 seconds (Sampling)
## Chain 4:                5.898 seconds (Total)
## Chain 4:

## Warning:  There were 2 divergent transitions after warmup.  See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
## Warning:  Examine the pairs() plot to diagnose sampling problems

summary(mod_ex4b)

##
## Model Info:
##  function:     stan_glmer
##  family:       poisson [log]
##  formula:      colonies ~ quinoline + log_quinoline + (1 | plate)
```

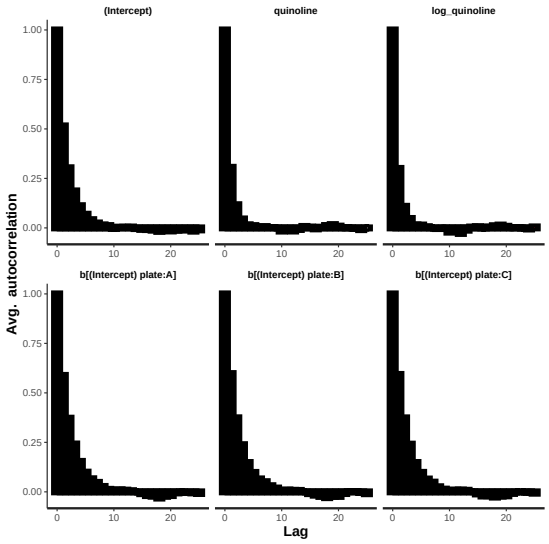
```
##  algorithm:    sampling
##  sample:       16000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 18
##  groups:       plate (3)
##
## Estimates:
##                                        mean   sd   10%   50%   90%
## (Intercept)                            2.1    0.4  1.7   2.1   2.6
## quinoline                              0.0    0.0  0.0   0.0   0.0
## log_quinoline                          0.3    0.1  0.2   0.3   0.4
## b[(Intercept) plate:A]                -0.3    0.4 -0.6  -0.3   0.1
## b[(Intercept) plate:B]                 0.0    0.4 -0.4   0.0   0.4
## b[(Intercept) plate:C]                 0.3    0.4 -0.1   0.3   0.6
## Sigma[plate:(Intercept),(Intercept)]  0.4    0.9  0.0   0.2   0.9
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 29.1    1.8 26.8  29.1  31.4
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##                                       mcse Rhat n_eff
## (Intercept)                           0.0  1.0   4423
## quinoline                             0.0  1.0   7914
## log_quinoline                         0.0  1.0   8044
## b[(Intercept) plate:A]                0.0  1.0   3688
## b[(Intercept) plate:B]                0.0  1.0   3665
## b[(Intercept) plate:C]                0.0  1.0   3694
## Sigma[plate:(Intercept),(Intercept)] 0.0  1.0   4119
## mean_PPD                              0.0  1.0  16654
## log-posterior                         0.0  1.0   4192
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

Only 2 divergent transitions remains while the effective sample size is now fixed.
Doig checks, no problems with the traceplot, while slight autocorrelation with some parameters but not a big problem

```
stan_trace(mod_ex4b, nrow = 6, ncol = 1)
```

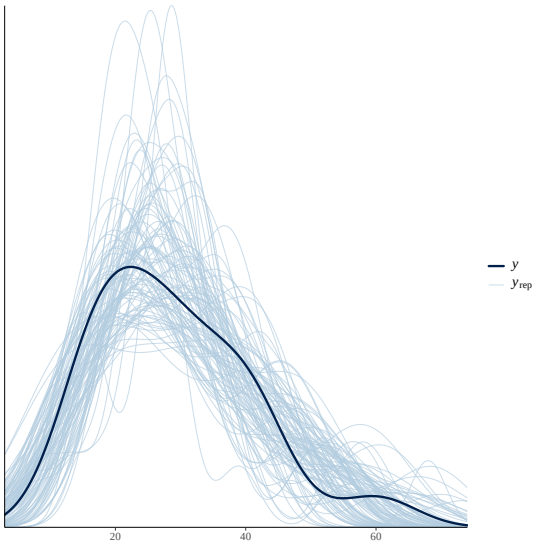



```
stan_ac(mod_ex4b)
```



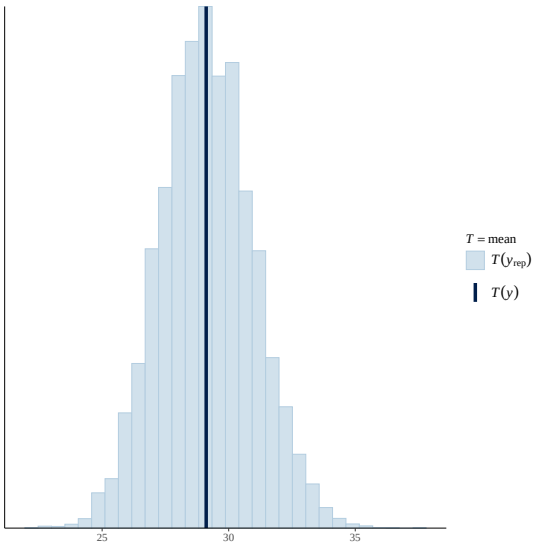
For the posterior checks, now our replicates no longer underestimate our original data; it seems introducing random effect solves the problem of overdispersion seen before

```
# Posterior checks
y_tilde4b <- posterior_predict(mod_ex4b)
ppc_dens_overlay(y = data4$colonies, yrep = y_tilde4b[1100:1200,])
```

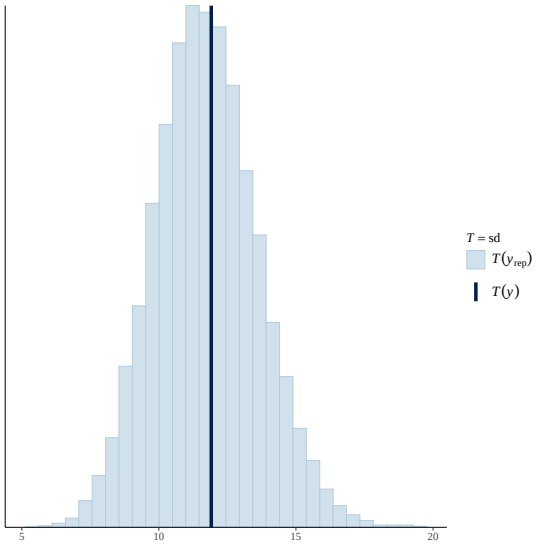


Now even the sd is ok, while if we're interested in the min or the max of our posterior replicates we still have some issues (and in case we're interested in this we should change the model at all)

```
ppc_stat(y = data4$colonies, yrep = y_tilde4b, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```

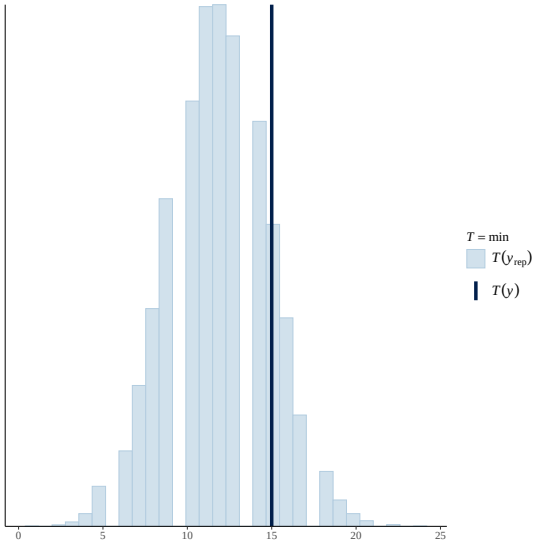



```
ppc_stat(y = data4$colonies, yrep = y_tilde4b, stat = "sd")

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



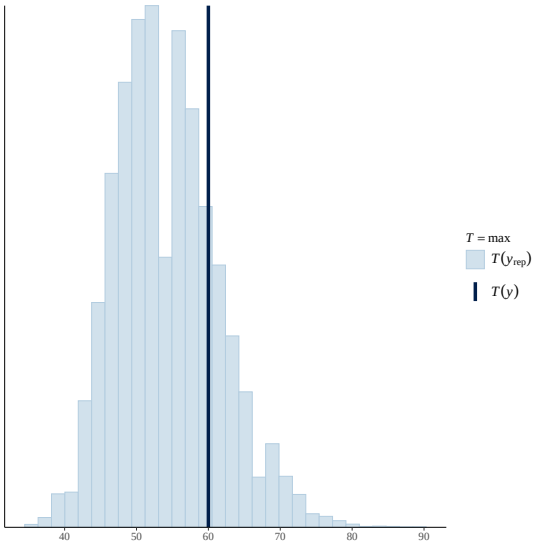

```
ppc_stat(y = data4$colonies, yrep = y_tilde4b, stat = "min")
```

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




```
ppc_stat(y = data4$colonies, yrep = y_tilde4b, stat = "max")
```

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



Suppose now that we want to use our model to perform inference on a new covariate pattern `quinoline = 500` and `plate = "A"`; in the following steps:

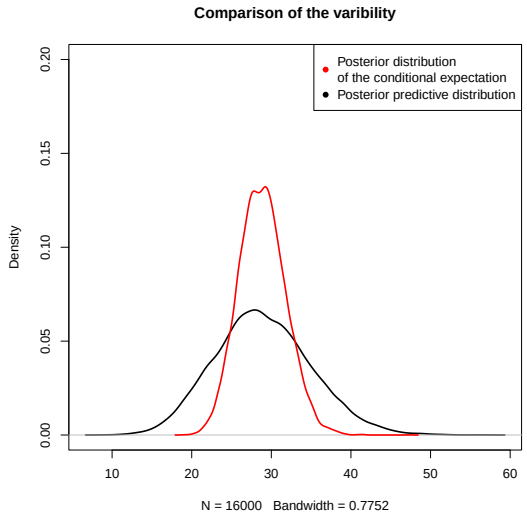
- `posterior_linpred` (with `transform=TRUE`) or `posterior_epred` without (red) compute the expected value of the linear predictor for the new data, based on the posterior distribution of the model `mod_ex4b`, i.e the posterior distribution of the mean response (linear predictor) for the new data.
- `posterior_predict` (black) simulates the posterior predictive distribution, which includes both the uncertainty in the model parameters (from the posterior) and residual uncertainty (noise).

```
# Posterior inference: new covariate pattern quinoline = 500 plate=A

# generate the new dataset
data4_new <- data.frame(quinoline = 500,
                        log_quinoline = log(500 + 10),
                        plate ="A")

# evaluation linear predictor: posterior_linpred is equivalent to
# posterior_epred without transform=TRUE parameter set
mu_new <- posterior_linpred(mod_ex4b,
                            newdata = data4_new,
                            transform = TRUE)

## Instead of posterior_linpred(..., transform=TRUE) please call posterior_epred(),
which provides equivalent functionality.

# posterior predictive distribution
y_tilde_new <- posterior_predict(mod_ex4b, newdata = data4_new)
```

Regarding the comparison below:

- The broader black curve indicates that predictions of the actual observed data are more uncertain than predictions of the conditional mean; when we simulate the posterior predictive distribution we include also the noise due to the simulation (aside the variability of the parameter which is the only considered in the posterior distribution of conditional expectation)[2]
- This highlights the importance of distinguishing between the variability in expected means (red) and full predictive uncertainty (black).

```
# comparison of posterior distribution of conditional expectation and posterior
# predictive distribution
plot(density(y_tilde_new), ylim=c(0,0.2), lwd = 2, main = "Comparison of the varibili
lines(density(mu_new), col="red", lwd=2)
legend("topright",
       legend = c("Posterior distribution \nof the conditional expectation",
                  "Posterior predictive distribution"),
       pch=16,
       col = c("red", "black"))
```



Indeed if we compare the mean they're similar (same center), but sd are different (variaiblity of posterior predict is doubled higher)

```
mean(mu_new)

## [1] 28.85746

sd(mu_new)

## [1] 2.937214

mean(y_tilde_new)
```

[2]The additional variability is due to the nature of the Poisson distribution, where variance increases with the mean (overdispersion is not modeled explicitly in this case).

```
## [1] 28.93825

sd(y_tilde_new)

## [1] 6.067577
```

Chapter 14

Exam simulation

Suppose that we want to make inferences about the efficacy of a certain pest management system at reducing the number of roaches in urban apartments (Gelman and Hill 2007, chapter 8.3, pg. 161). We have data on apartments:

- `y`, the outcome, number of roaches caught in a set of traps;
- `treatment` dummy indicating the type of treatment
- `senior`, dummy indicating whether the apartment is in a building restricted to elderly residents
- `roach1`, covariate, pre-treatment number of roaches

Load the data and rescale the variable `roach1` using the following piece of code.

```
library(loo)
library(bayesplot)
library(rstan)
library(rstanarm)

# data and rescaling
data(roaches)
roaches$roach1 <- roaches$roach1 / 100
head(roaches)

##     y roach1 treatment senior exposure2
## 1 153 3.0800         1      0  0.800000
## 2 127 3.3125         1      0  0.600000
## 3   7 0.0167         1      0  1.000000
## 4   7 0.0300         1      0  1.000000
## 5   0 0.0200         1      0  1.142857
## 6   0 0.0000         1      0  1.000000
```

14.1 Question 1

Assume a simple Poisson regression model without random effects considering `roach1` and `senior` as independent variables (model a). Write the theoretical form of the model assuming weakly-informative prior distributions for the parameters and program the model in `rstanarm`.
For the model

$$
\begin{aligned}
y_i|\mu_i &\sim Poisson(\mu_i) \\
\log(\mu_i)|\boldsymbol{\beta} &= \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} \\
\beta_0 &\sim \mathrm{N}\,(0, 2.5) \\
\beta_1 &\sim \mathrm{N}\,(0, 3.32) \\
\beta_2 &\sim \mathrm{N}\,(0, 5.42)
\end{aligned}
$$

NB: Here, as requested, the value 2.5, 3.32 etc in the β priors comes from the adjusted prior outputted by stan below

where

- Y = post treatment number of roaches
- X_1 = pre-treatment number of roaches
- X_2 = elterly residents indicator

For estimation

```
# 1.
mod_a <- stan_glm(formula = y ~ roach1 + senior, data = roaches, family = "poisson")

##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.26 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.143 seconds (Warm-up)
## Chain 1:                0.146 seconds (Sampling)
## Chain 1:                0.289 seconds (Total)
```

```
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.131 seconds (Warm-up)
## Chain 2:                0.153 seconds (Sampling)
## Chain 2:                0.284 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.143 seconds (Warm-up)
## Chain 3:                0.158 seconds (Sampling)
```

```
## Chain 3:                0.301 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.17 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.138 seconds (Warm-up)
## Chain 4:                0.141 seconds (Sampling)
## Chain 4:                0.279 seconds (Total)
## Chain 4:
```

```
prior_summary(mod_a)
```

```
## Priors for model 'mod_a'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 2.5)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = [0,0], scale = [2.5,2.5])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [3.32,5.42])
## ------
## See help('prior_summary.stanreg') for more details
```

```
summary(mod_a)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      y ~ roach1 + senior
```

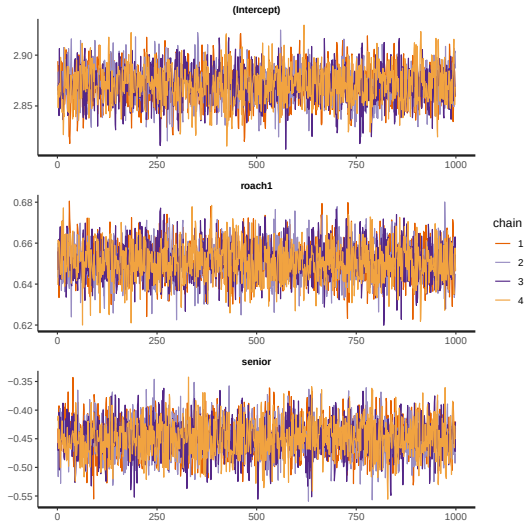
```
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 262
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept)  2.9    0.0  2.8   2.9   2.9
## roach1       0.7    0.0  0.6   0.7   0.7
## senior      -0.4    0.0 -0.5  -0.4  -0.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 25.6    0.4 25.1  25.6  26.2
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  2286
## roach1        0.0  1.0  3191
## senior        0.0  1.0  2987
## mean_PPD      0.0  1.0  3092
## log-posterior 0.0  1.0  1797
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

Here above we took the variance for the prior and reported in the theoretical model; then looking at the MCMC sumary of model, mcse and Rhat are perfect while effective sample size is *acceptable* so there's no need to increase the number of iterations.

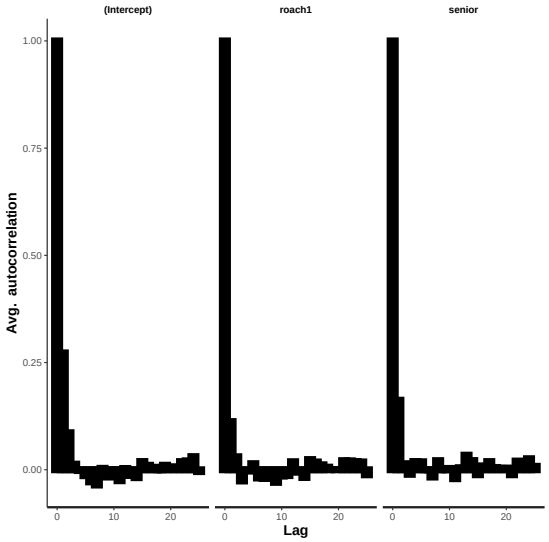
14.2 Question 2

Monitor the convergence of the algorithm (referred to model a).

```
# 2.
stan_trace(mod_a, nrow = 3,ncol = 1)
```




```
library(ggplot2)
stan_ac(mod_a)
```



Regarding:

- the traceplot: the chains are acceptable and it seems that the 4 chains reached the same target distribution;
- autocorrelation: the decay is quite fast and, even if it is not a proper Markov chain, it is acceptable to carry out the posterior analysis

```
summary(mod_a)

##
```

```
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      y ~ roach1 + senior
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 262
##  predictors:   3
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept)  2.9    0.0  2.8   2.9   2.9
## roach1       0.7    0.0  0.6   0.7   0.7
## senior      -0.4    0.0 -0.5  -0.4  -0.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 25.6    0.4 25.1  25.6  26.2
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  2286
## roach1        0.0  1.0  3191
## senior        0.0  1.0  2987
## mean_PPD      0.0  1.0  3092
## log-posterior 0.0  1.0  1797
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

Looking at MCMC diagnostic, values of Rhat near to 1 point out the convergence. `n_eff` is an indicator of autocorrelation of the chains. If the value is too far from the sample size (in this case 4000), then the draws might be too correlated among them.

NB: always gives some short comment to the output, eg mcmc diagnostics; detail if theoretical explanation are requested

14.3 Option 3

Assume now that we are interested in update model a in order to estimate also group-specific effects with respect to the type of treatment (model b). Write the theoretical form of the model assuming a $\mathrm{N}(0, 10)$ prior distribution with automatic scale adjustments for the parameters and program the model in rstanarm.
We have **likelihood** where we introduce the random effect λ_j:

$$y_{ij}|\mu_{ij} \sim Poisson(\mu_{ij})$$
$$\log(\mu_{ij})|\boldsymbol{\beta}, \lambda_j = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + \lambda_j$$

Priors are different from before (variance can be found in the estimates below)

$$\lambda_j|\sigma^2_\lambda \sim \mathrm{N}\left(0, \sigma^2_\lambda\right) \qquad \beta_0 \sim \mathrm{N}\left(0, 10\right)$$
$$\beta_1 \sim \mathrm{N}\left(0, 13.29\right)$$
$$\beta_2 \sim \mathrm{N}\left(0, 21.67\right)$$

Hyperprior

$$\sigma_\lambda \sim \pi(\sigma_\lambda)$$

Going with estimation

```
# 3.

mod_b <- stan_glmer(formula = y ~ roach1 + senior + (1 | treatment),
                    data = roaches,
                    family = "poisson",
                    prior = normal(0, 10, autoscale = TRUE),
                    prior_intercept = normal(0, 10, autoscale = TRUE))

##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.45 s
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 5.964 seconds (Warm-up)
## Chain 1:                5.37 seconds (Sampling)
## Chain 1:                11.334 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.32 s
```

```
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 5.455 seconds (Warm-up)
## Chain 2:                4.461 seconds (Sampling)
## Chain 2:                9.916 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 9.765 seconds (Warm-up)
## Chain 3:                5.945 seconds (Sampling)
## Chain 3:                15.71 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.3e-05 seconds
```

```
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.33 s
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 15.376 seconds (Warm-up)
## Chain 4:                6.488 seconds (Sampling)
## Chain 4:                21.864 seconds (Total)
## Chain 4:

## Warning:  There were 24 divergent transitions after warmup.  See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
## Warning:  Examine the pairs() plot to diagnose sampling problems

prior_summary(mod_b)

## Priors for model 'mod_b'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = [0,0], scale = [10,10])
##   Adjusted prior:
##     ~ normal(location = [0,0], scale = [13.29,21.67])
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details

summary(mod_b)

##
## Model Info:
##  function:     stan_glmer
##  family:       poisson [log]
```

```
##  formula:      y ~ roach1 + senior + (1 | treatment)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 262
##  groups:       treatment (2)
##
## Estimates:
##                                              mean   sd   10%   50%   90%
## (Intercept)                                  2.9    0.5  2.3   2.9   3.5
## roach1                                       0.6    0.0  0.6   0.6   0.7
## senior                                      -0.4    0.0 -0.4  -0.4  -0.3
## b[(Intercept) treatment:0]                   0.3    0.5 -0.3   0.3   0.9
## b[(Intercept) treatment:1]                  -0.3    0.5 -0.8  -0.2   0.4
## Sigma[treatment:(Intercept),(Intercept)]     0.7    1.2  0.1   0.3   1.9
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 25.7    0.4 25.1  25.7  26.2
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable
##
## MCMC diagnostics
##                                             mcse Rhat n_eff
## (Intercept)                                 0.0  1.0   905
## roach1                                      0.0  1.0  4329
## senior                                      0.0  1.0  3874
## b[(Intercept) treatment:0]                  0.0  1.0   901
## b[(Intercept) treatment:1]                  0.0  1.0   905
## Sigma[treatment:(Intercept),(Intercept)] 0.0  1.0  1144
## mean_PPD                                    0.0  1.0  4150
## log-posterior                               0.1  1.0  1228
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

```
## needed to increase the iterations
mod_b <- update(mod_b, iter = 4000, adapt_delta = .99)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
```

```
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 46.682 seconds (Warm-up)
## Chain 1:                26.066 seconds (Sampling)
## Chain 1:                72.748 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.32 s
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 37.95 seconds (Warm-up)
## Chain 2:                19.886 seconds (Sampling)
## Chain 2:                57.836 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.32 s
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
```

```
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 473.347 seconds (Warm-up)
## Chain 3:                20.142 seconds (Sampling)
## Chain 3:                493.489 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 36.905 seconds (Warm-up)
## Chain 4:                67.519 seconds (Sampling)
## Chain 4:                104.424 seconds (Total)
## Chain 4:

## Warning:  There were 1 divergent transitions after warmup.  See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
## Warning:  Examine the pairs() plot to diagnose sampling problems

summary(mod_b)

##
## Model Info:
##  function:     stan_glmer
```

```
##  family:       poisson [log]
##  formula:      y ~ roach1 + senior + (1 | treatment)
##  algorithm:    sampling
##  sample:       8000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 262
##  groups:       treatment (2)
##
## Estimates:
##                                              mean   sd   10%   50%   90%
## (Intercept)                                  2.9    0.7  2.2   2.9   3.5
## roach1                                       0.6    0.0  0.6   0.6   0.7
## senior                                      -0.4    0.0 -0.4  -0.4  -0.3
## b[(Intercept) treatment:0]                   0.3    0.7 -0.4   0.3   1.0
## b[(Intercept) treatment:1]                  -0.2    0.7 -0.9  -0.3   0.4
## Sigma[treatment:(Intercept),(Intercept)]     1.0    2.3  0.1   0.3   2.4
##
## Fit Diagnostics:
##           mean   sd   10%   50%   90%
## mean_PPD 25.7    0.4 25.1  25.7  26.2
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                             mcse Rhat n_eff
## (Intercept)                                 0.0  1.0  1681
## roach1                                      0.0  1.0  7894
## senior                                      0.0  1.0  7003
## b[(Intercept) treatment:0]                  0.0  1.0  1680
## b[(Intercept) treatment:1]                  0.0  1.0  1685
## Sigma[treatment:(Intercept),(Intercept)]    0.0  1.0  2253
## mean_PPD                                    0.0  1.0  8079
## log-posterior                               0.0  1.0  2259
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

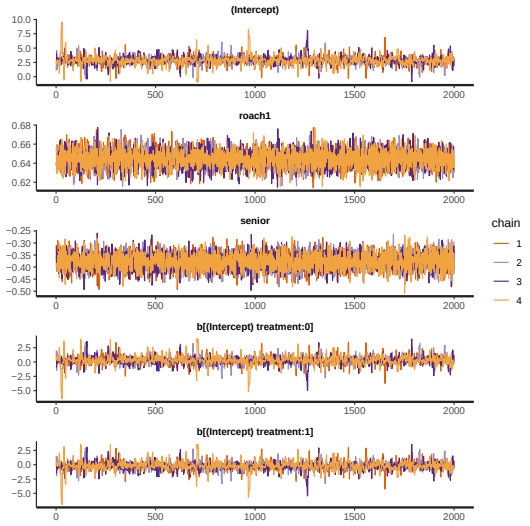
Above we see warnings so we needed to increase the iterations (and `adapt_delta`)

14.4 Question 4

Monitor the convergence of the algorithm (referred to model b) and in particular discuss the interpretation of `n_eff`.

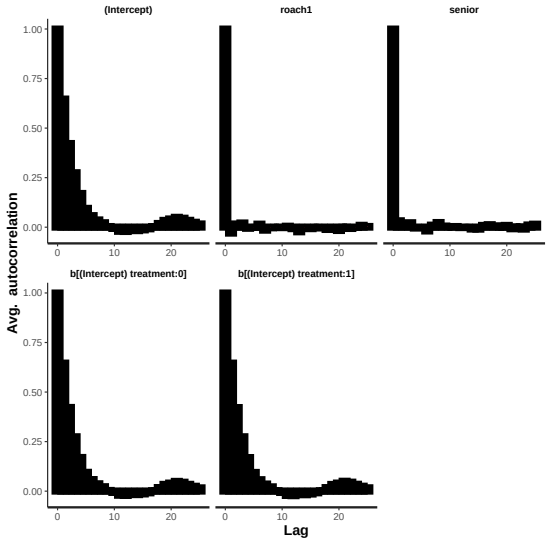
Regarding the traceplot there may be some problems in the chains of the treatment random effects (in the last some variability in the traceplot).

```
# 4.
stan_trace(mod_b, nrow=6, ncol = 1)
```



Looking at autocorrelations there are some evident problems of (decay is really slow and goes to 0 after 20 lag).

```
stan_ac(mod_b)
```



And finally the effective sample size in the MCMC diagnostics is really low.

```
summary(mod_b)

##
## Model Info:
##  function:     stan_glmer
##  family:       poisson [log]
##  formula:      y ~ roach1 + senior + (1 | treatment)
##  algorithm:    sampling
```

```
##  sample:       8000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 262
##  groups:       treatment (2)
##
## Estimates:
##                                              mean   sd   10%   50%   90%
## (Intercept)                                  2.9    0.7  2.2   2.9   3.5
## roach1                                       0.6    0.0  0.6   0.6   0.7
## senior                                      -0.4    0.0 -0.4  -0.4  -0.3
## b[(Intercept) treatment:0]                   0.3    0.7 -0.4   0.3   1.0
## b[(Intercept) treatment:1]                  -0.2    0.7 -0.9  -0.3   0.4
## Sigma[treatment:(Intercept),(Intercept)]     1.0    2.3  0.1   0.3   2.4
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 25.7    0.4 25.1  25.7  26.2
##
## The mean_ppd is the sample average posterior predictive distribution of the outcom
##
## MCMC diagnostics
##                                             mcse Rhat n_eff
## (Intercept)                                 0.0  1.0  1681
## roach1                                      0.0  1.0  7894
## senior                                      0.0  1.0  7003
## b[(Intercept) treatment:0]                  0.0  1.0  1680
## b[(Intercept) treatment:1]                  0.0  1.0  1685
## Sigma[treatment:(Intercept),(Intercept)] 0.0  1.0  2253
## mean_PPD                                    0.0  1.0  8079
## log-posterior                               0.0  1.0  2259
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure o
```

14.5 Question 5

Compare the two models in terms of goodness of fit. Which one provides a better fit of the original data?

```
# 5.
waic(mod_a)

## Warning:
## 36 (13.7%) p_waic estimates greater than 0.4.  We recommend trying
loo instead.

##
## Computed from 4000 by 262 log-likelihood matrix.
##
```

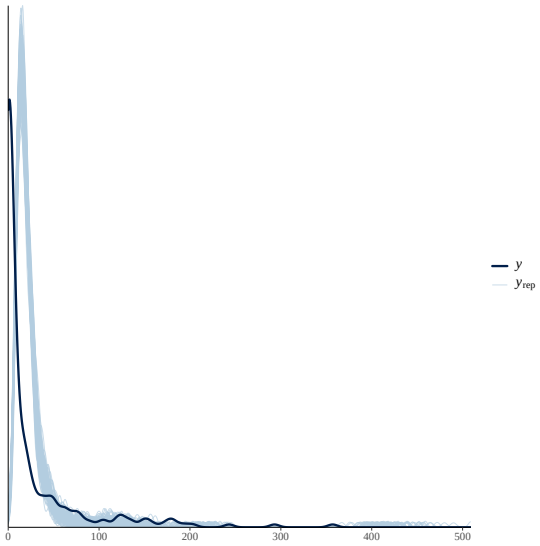
```
##           Estimate     SE
## elpd_waic  -6465.0  812.6
## p_waic       250.1   70.8
## waic       12930.0 1625.3
##
## 36 (13.7%) p_waic estimates greater than 0.4. We recommend trying loo instead.

waic(mod_b)

## Warning:
## 41 (15.6%) p_waic estimates greater than 0.4.  We recommend trying
loo instead.

##
## Computed from 8000 by 262 log-likelihood matrix.
##
##           Estimate     SE
## elpd_waic  -6322.1  758.3
## p_waic       354.8  118.2
## waic       12644.2 1516.6
##
## 41 (15.6%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

Looking at waic the second model is smaller compared to the first one; however second model provide worst convergences and big issue with autocorrelation.
In this case even if waic is lower we should choose model a

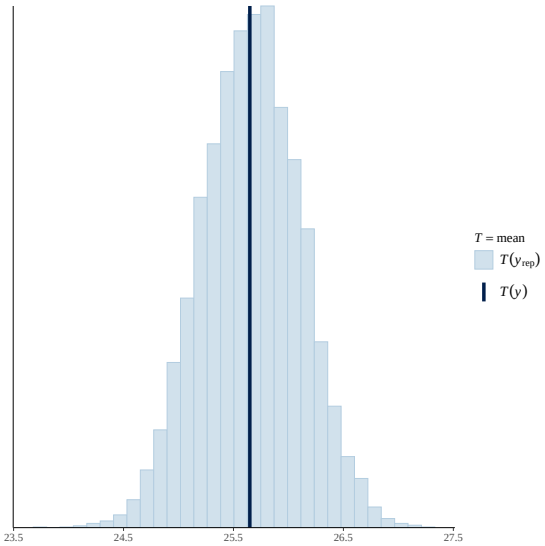
14.6 Question 6

Suppose we trust the WAIC method and go with the second; we are interested in verifying that the chosen model is able to reproduce the mean number of post-treatment roaches. Check this assumption. How do the posterior predictive checks work? Provide a brief explanation.

```
# 6.
y_tildeb <- posterior_predict(mod_b)
ppc_dens_overlay(y = roaches$y, yrep = y_tildeb[1100:1200,])
```




```
ppc_stat(y = roaches$y, yrep = y_tildeb, stat = "mean")

## Note:  in most cases the default test statistic 'mean' is too weak
to detect anything of interest.
## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



we see that that the two distributions are quite far and not too much overlapping (these should give an alert about the model we choosed to use), but if we're interested in the mean this is ok (mean of original data is centered on distribution of mean of replicates).

14.7 Question 7

Report the 90% credible interval of the predicted parameters.

```r
#7.
mu <- posterior_epred(mod_b)
quantile(mu, probs = c(0.05, 0.5, 0.95))
```

```
##        5%       50%       95%
##  9.519158 16.473498 58.674835
```

14.8 Question 8

Provide an overall comment of the performed analysis. How could you improve, if necessary, the fitted models in order to better fit the initial dataset?