CS 251 Statistical Computing

HOP 9: R for statistical project

8/10/2020 Developed by Aya Khalil

7/7/2019 Reviewed by

School of Technology and Computing @City University of Seattle (CityU)

**Before You Start**

* If you already finished this module through any CityU Technology Institute (TI) courses,  
  just skim this module and skip it.
* Version numbers may not match with the guide. But that should be fine.  
  If given the option to choose between stable release (long-term support) or most recent, please choose the stable release.
* This guide targets Windows OS users. So, MacOS users may have different commands to input in the shell/terminal.
* We cannot explain every step. **This cookbook always needs your own creative judgement.**
* **For your working directory, use your course number.** The hands-on tutorial may use a different course number as an example.

**Learning Outcomes**

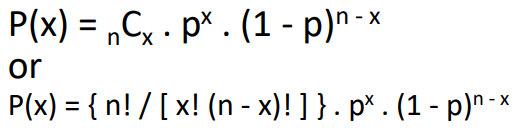
* Binomial Distribution
* Hypothesis testing with the Binomial distribution
* Posterior probability

**Resource**

* Binomial Distribution: <http://www.r-tutor.com/elementary-statistics/probability-distributions/binomial-distribution>
* Binomial Distribution: <https://www.statology.org/dbinom-pbinom-qbinom-rbinom-in-r/>
* Binomial Test: <http://www.instantr.com/2012/11/06/performing-a-binomial-test/>
* Binomial Test: <http://www.endmemo.com/r/binomial.php>
* Hypothesis Testing with the Binomial Distribution: <https://internal.ncl.ac.uk/ask/numeracy-maths-statistics/statistics/hypothesis-testing/hypothesis-testing-with-the-binomial-distribution.html#See%20Also>
* Bayesian models in R: <https://www.r-bloggers.com/bayesian-models-in-r-2/>
* Posterior probability: <https://cran.rproject.org/web/packages/BayesCombo/vignettes/BayesCombo_vignette.html>

**Section1: Binomial Distribution**

The binomial distribution is a discrete probability distribution. It describes the outcome of n independent trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure. If the probability of a successful trial is p, then the probability of having x successful outcomes in an experiment of n independent trials is as follows.



* x: The number of successes that result from the binomial experiment.
* n: The number of trials in the binomial experiment.
* p: The probability of success on an individual trial.
* q: The probability of failure on an individual trial. (This is equal to 1 - p.)
* n!: The factorial of n (also known as n factorial).
* P(x): Binomial probability - the probability that an n-trial binomial experiment results in exactly x successes, when the probability of success on an individual trial is p.
* nCx: The number of combinations of n things, taken x at a time.

**Usage of binomial distribution**

dbinom(x, size, prob, log = FALSE)

pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)

qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)

rbinom(n, size, prob)

**Arguments**

|  |  |
| --- | --- |
| x, q | vector of quantiles. |
| p | vector of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| size | number of trials (zero or more). |
| prob | probability of success on each trial. |
| log, log.p | logical; if TRUE, probabilities p are given as log(p). |
| lower.tail | logical; if TRUE (default), probabilities are *P[X ≤ x]*, otherwise, *P[X > x]*. |

**Setup Working Environment for Module9**

1. Open VS Code.

* **online student:** Open CS251 \_Fall\_2020/**ON**/FirstnameLastname /. ( File > Open )
* **onsite student:** Open CS251 \_ Fall \_2020/**IN**/FirstnameLastname. ( File > Open )

1. Then, create the “**Module9**” directory in the VSCode.

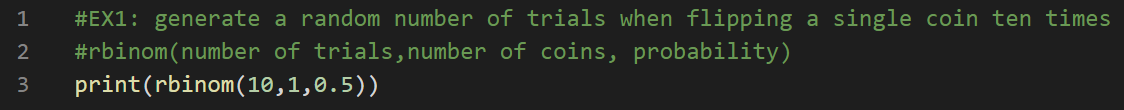
>>>mkdir Module9

1. In module9 project folder, create new file Rbinom.R

**rbinom**

The function rbinom generates a vector of binomial distributed random variables given a vector length n, number of trials (size) and probability of success on each trial (prob). The syntax for using rbinom is as follows: **rbinom(n, size, prob)**

* Type the following code in Rbinom.R file



**Output:** 

The output will not be the same each time because it’s a random generator. My output means that I have 6 heads and 4 tail.

**dbinom**

The function dbinom returns the value of the probability density function (pdf) of the binomial distribution given a certain random variable x, number of trials (size) and probability of success on each trial (prob). The syntax for using dbinom is as follows: **dbinom(x, size, prob)**

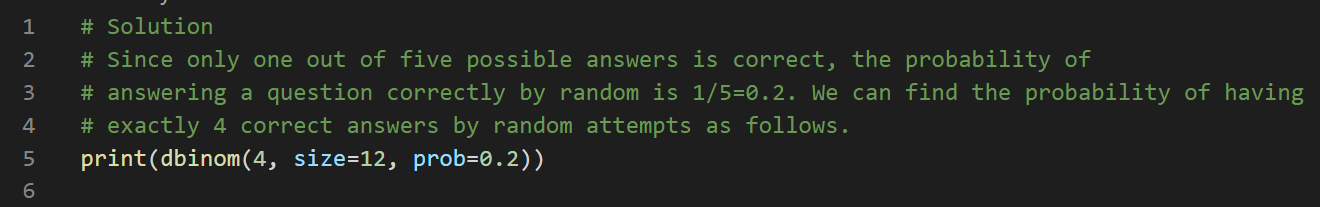
Put simply, dbinom finds the probability of getting a certain number of successes (x) in a certain number of trials (size) where the probability of success on each trial is fixed (prob).

* In module9 project folder, create new file Dbinom.R
* Type the following code in Dbinom.R file

**EX1**

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of **having exactly four** correct answers if a student attempts to answer every question at random.

* Add the following code to Dbinom.R file

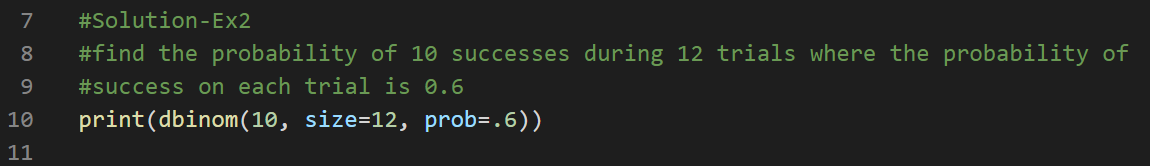


Output: 

**EX2**

Bob makes 60% of his free-throw attempts. If he shoots 12 free throws, what is the probability that he makes exactly 10?

* Add the following code to update Dbinom.R file

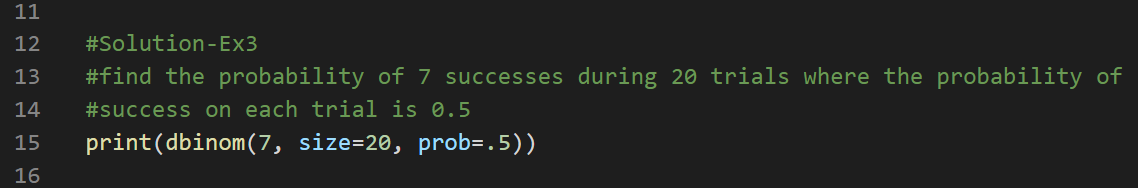


Output: 

**Ex3**

Sasha flips a fair coin 20 times. What is the probability that the coin lands on heads exactly 7 times?

* Add the following code to update Dbinom.R file



Output: 

**pbinom**

The function pbinom returns the value of the cumulative density function (cdf) of the binomial distribution given a certain random variable q, number of trials (size) and probability of success on each trial (prob). The syntax for using pbinom is as follows: **pbinom(q, size, prob)**

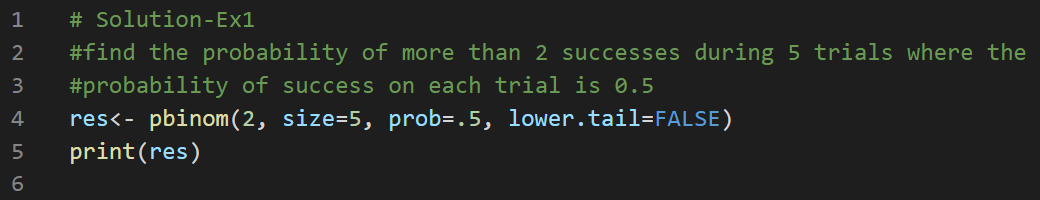
Put simply, pbinom returns the area to the left of a given value q in the binomial distribution. If you’re interested in the area to the right of a given value q, you can simply add the argument lower.tail = FALSE

**pbinom(q, size, prob, lower.tail = FALSE)**

**Ex1**

Ando flips a fair coin 5 times. What is the probability that the coin lands on heads more than 2 times?

* In module9 project folder, create new file Pbinom.R
* Type the following code in Pbinom.R file



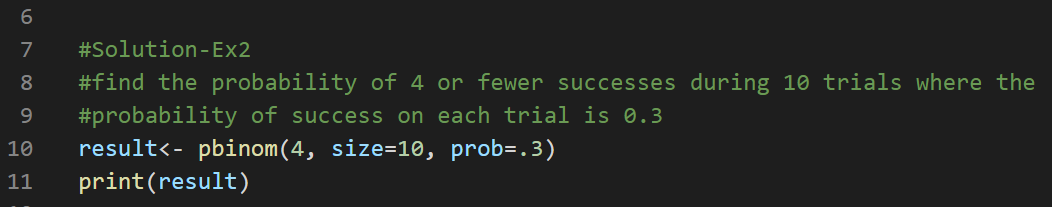
Output: 

The probability that the coin lands on heads more than 2 times is 0.5.

**Ex2:**

Suppose Tyler scores a strike on 30% of his attempts when he bowls. If he bowls 10 times, what is the probability that he scores 4 or fewer strikes?

* Add the following code to update Pbinom.R file

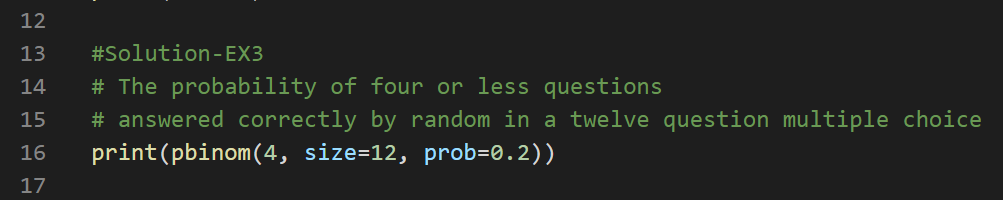


Output: 

**Ex3**:

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.

* Add the following code to update Pbinom.R file

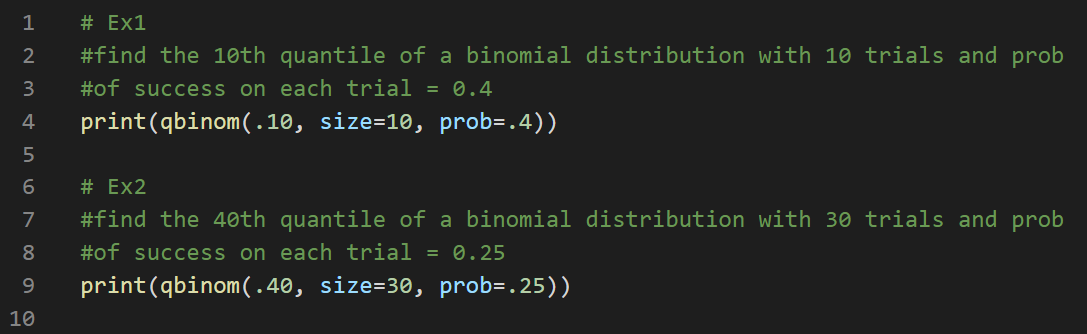


Output: 

**qbinom**

The function qbinom returns the value of the inverse cumulative density function (cdf) of the binomial distribution given a certain random variable q, number of trials (size) and probability of success on each trial (prob). The syntax for using qbinom is as follows: **qbinom(q, size, prob)**

* In module9 project folder, create new file Qbinom.R
* Type the following code in Qbinom.R file



Output: 

**Section 2: Hypothesis testing with Binomial distribution**

To hypothesis test with the binomial distribution, we must calculate the probability, p, of the observed event and any more extreme event happening. We compare this to the level of significance α. If p>α then we do not reject the null hypothesis. If p<α we accept the alternative hypothesis.

We will use **binom.test** {stats}which performs an exact test of a simple null hypothesis about the probability of success. The syntax for using is as follows:

**binom.test(x, n, p = 0.5, alternative = c("two.sided", "less", "greater"), conf.level = 0.95)**

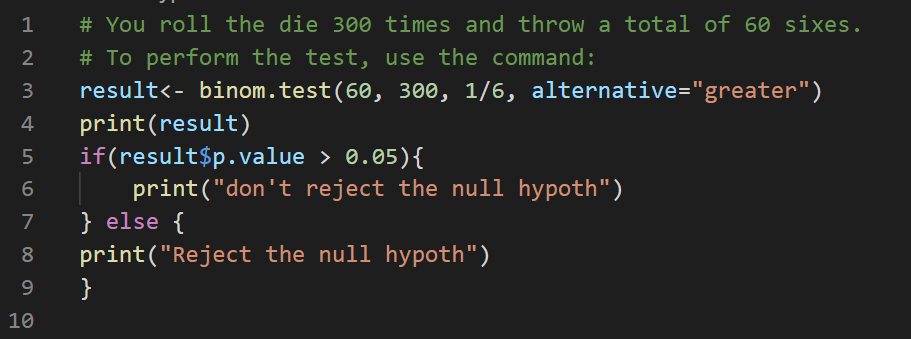
**Arguments**

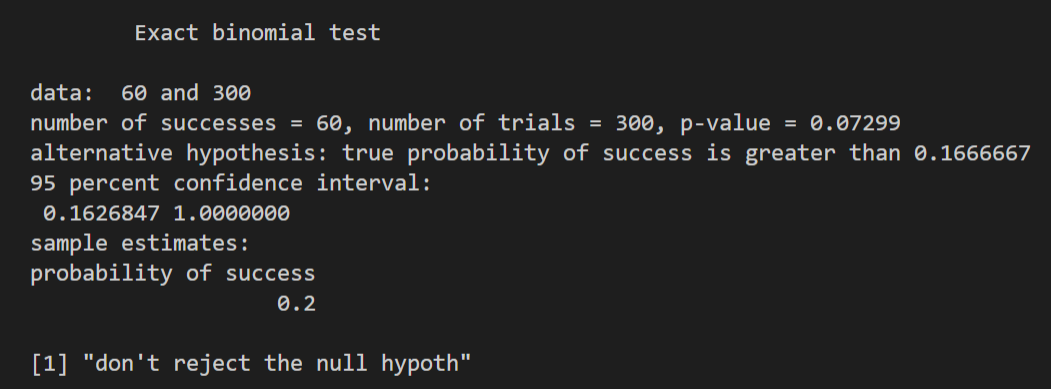
* x: number of successes, or a vector of length 2 giving the numbers of successes and failures, respectively.
* n: number of trials; ignored if x has length 2.
* p: hypothesized probability of success.
* alternative: indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less". You can specify just the initial letter.
* conf.level: confidence level for the returned confidence interval.

**Example**

In a game, you suspect your opponent is using a die which is biased to roll a six greater than 1/6 of the time. Suppose you want to prove this by rolling the die 300 times and using a binomial test to determine whether the probability of rolling a six is equal to 1/6. A one-tailed test with a significance level of 0.05 will be used.

* In module9 project folder, create new file Hypoth-binom.R
* Type the following code in Hypoth-binom.R file

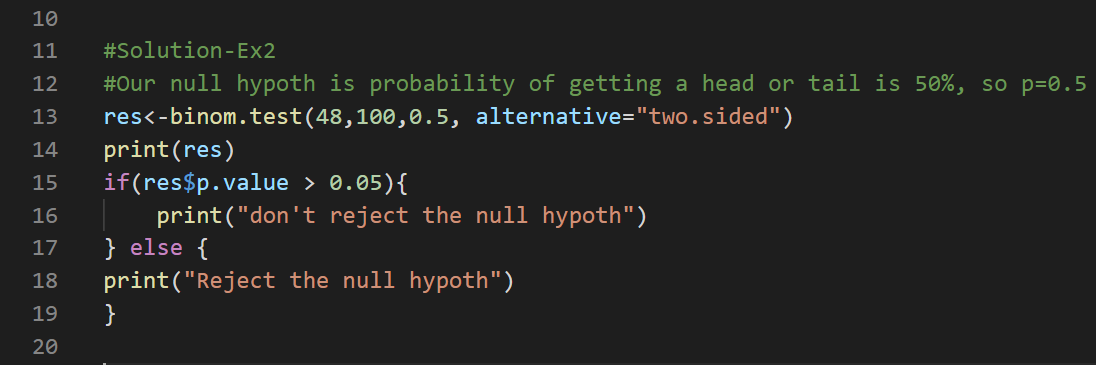


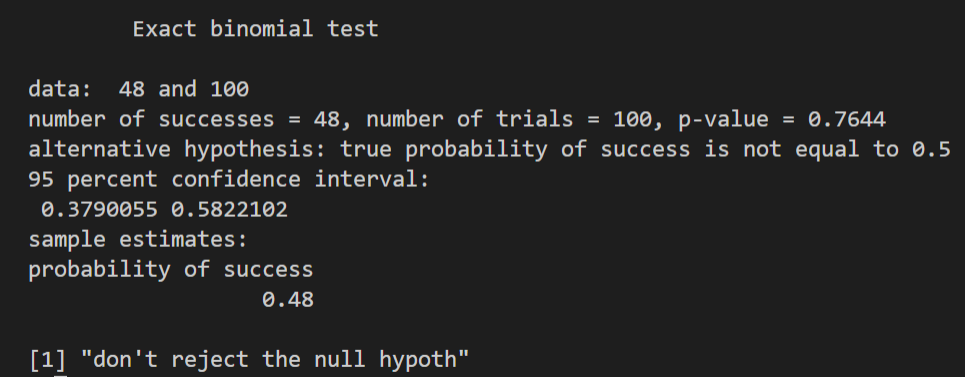
Output: 

From the output you can see that the p-value is 0.07299. As this is not less that the significance level of 0.05, we cannot reject the null hypothesis that the probability of rolling a six is 1/6. This means that there is no evidence to prove that the die is not fair.

Ex2: Suppose in a coin tossing, the chance to get a head or tail is 50%. In a real case, we have 100 coin tossing, and get 48 heads, is our null hypothesis true?

* Add the following code to update Hypoth-binom.R file



Output: 

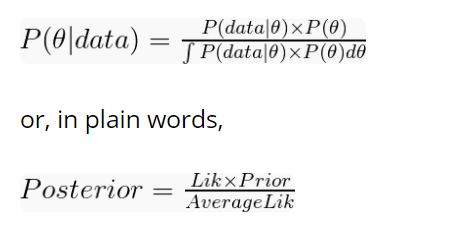
Since the p-value is 0.7644, far greater than 0.05, so do not reject the null hypothesis.

**Section 3: Posterior probability**

## **What Is a Posterior Probability?**

A posterior probability, in Bayesian statistics, is the revised or updated probability of an event occurring after taking into consideration new information. The posterior probability is calculated by updating the [prior probability](https://www.investopedia.com/terms/p/prior_probability.asp) using [Bayes' theorem](https://www.investopedia.com/terms/b/bayes-theorem.asp). In statistical terms, the posterior probability is the probability of event A occurring given that event B has occurred.

The Bayesian perspective is more comprehensive. It produces no single value, but rather a whole probability distribution for the unknown parameter \theta  conditional on your data. This probability distribution, P(\theta | data) , is called posterior. The posterior comes from one of the most celebrated works of [Rev. Thomas Bayes](https://en.wikipedia.org/wiki/Thomas_Bayes) that you have probably met before,



The posterior can be computed from three key ingredients:

* A likelihood distribution, P(data | \theta) ;
* A prior distribution, P(\theta) ;
* The ‘average likelihood’, \int P(data | \theta) \times P(\theta) d\theta = P(data) .

All Bayes theorem does is updating some prior belief by accounting to the observed data and ensuring the resulting probability distribution has density of exactly one.

**Example:**

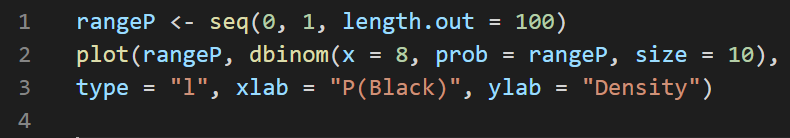
Assume that you and your brother were playing [roulette](https://en.wikipedia.org/wiki/Roulette) in a casino. Among other things, you can bet on hitting either black (B) or red (r) with supposedly equal probability. For simplification, you assumed P(B)=P(r)=0.5  and the ten past draws before you placed a bet:

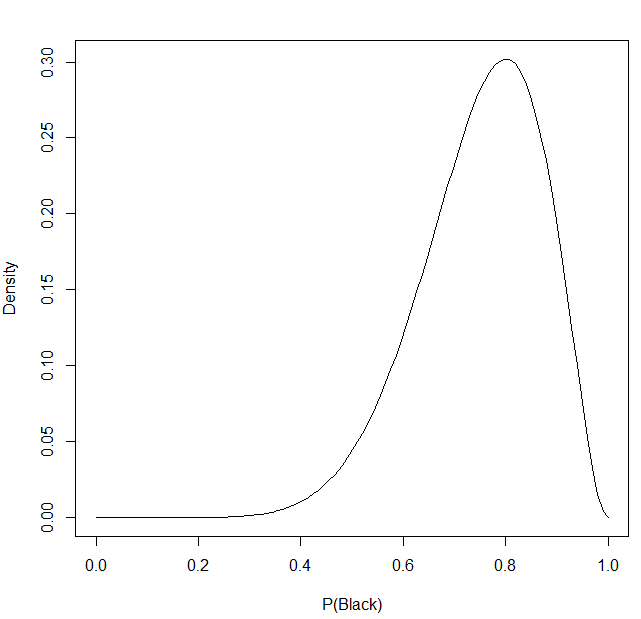


Having f(B) = \frac{8}{10}  based on these ten draws, my brother argued we should go for black. His reasoning was there would be a greater chance of hitting black than red, to which I kind of agreed. We eventually placed a bet on black and won. Knowing nothing of the chances of hitting either colour in this example, f(B) = 0.8  is the MLE of P(B) . This is the frequentist approach. But wouldn’t you assume P(B)=P(r)=0.5 ?

A different way of thinking is to consider the likelihoods obtained using different estimates of P(B) . If we estimate the likelihood P(data | P(B))  from 100 estimates of P(B)  ranging from 0 to 1, we can confidently approximate its distribution. Here, the probability mass function of the binomial distribution \mathcal{B} \left({10, P(B)} \right)  with eight successes, i.e. P(X \sim \mathcal{B} \left({10, P(B)} \right) = 8) , provides the likelihood of all different estimates of P(B) . We can demonstrate it with few lines of R code.

* In module9 project folder, create new file Posterior.R
* Type the following code in Posterior.R file



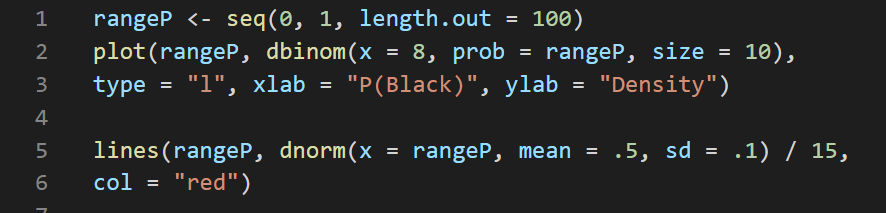
Output: 

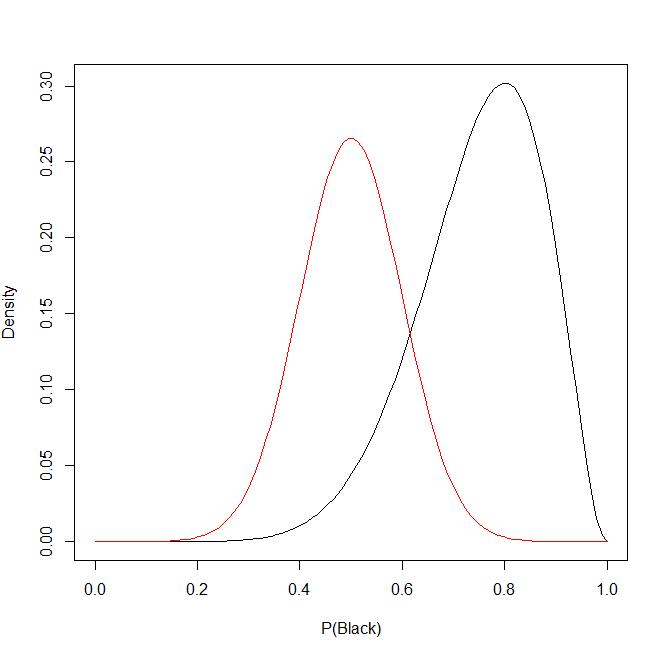
As the name indicates, the MLE in the roulette problem is the peak of the likelihood distribution. However, here we uncover an entire spectrum comprising all possible ways f(B) = 0.8  could have been produced.

**Step 2. Update your belief (prior distribution)**

assume that you intervened and expressed your belief to your brother that P(B)  must be 0.5 or close, e.g. P(B) \sim Normal(0.5, 0.1) . For comparison, overlay this prior distribution with the likelihood from the previous step.

* Type the following to update Posterior.R file

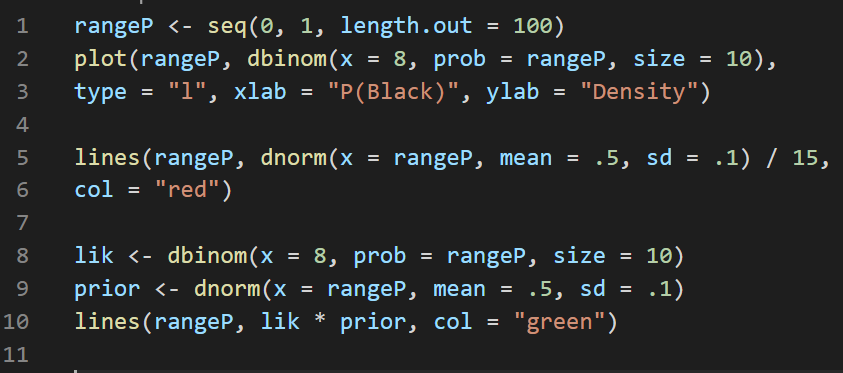


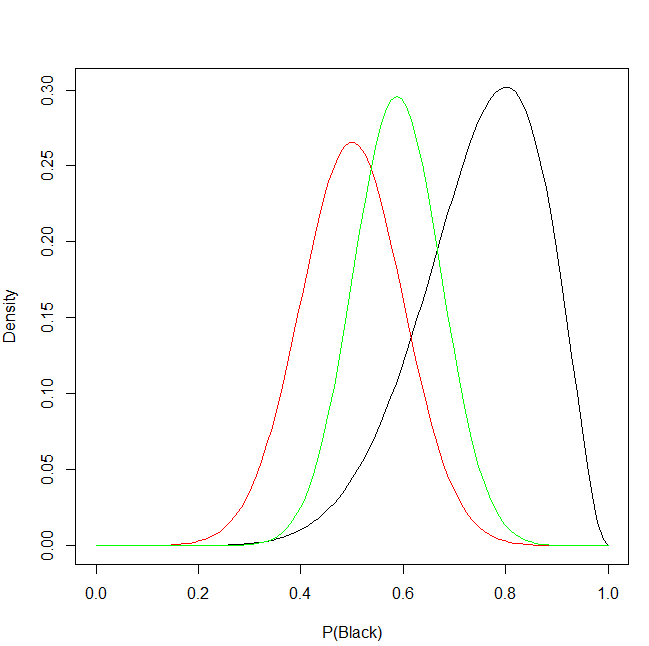
Output: 

The prior is now shown in red. In the code above, I divided the prior by a constant solely for scaling purposes. Keep in mind that distribution density only matters for the posterior.

Computing the product between the likelihood and my prior is straightforward and gives us the numerator from the theorem. The next bit will compute and overlay the unstandardised posterior of P(B) , P(data|\theta) \times P(\theta) . The usage of a sequence of estimates for P(B)  to reconstruct probability distributions is called grid approximation.

* Type the following to update Posterior.R file



Output: 

In short, we have successfully used the ten roulette draws (black) to update our prior (red) into the unstardardised posterior (green). Why we call it ‘unstandardised’? The answer comes with the denominator from the theorem.

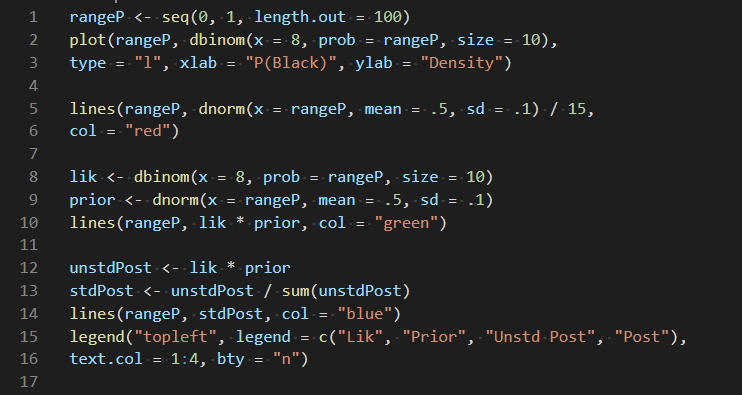
**Step 3. Make it sum up to one (standardizing the posterior)**

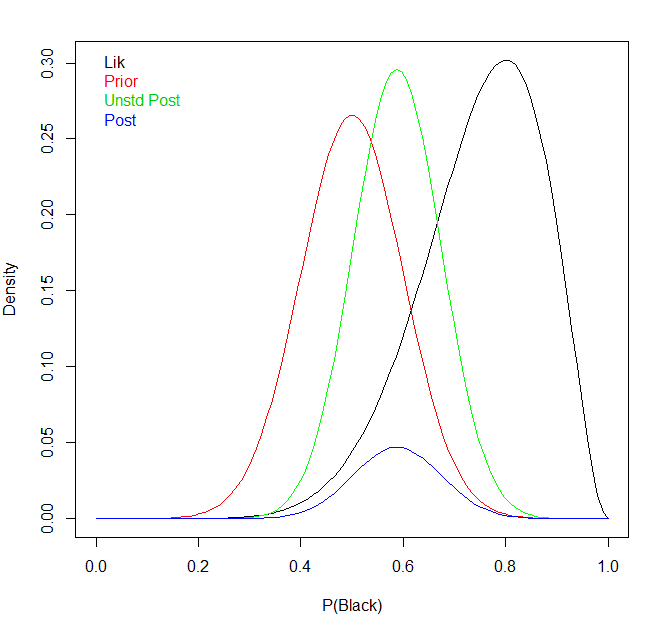
An important property of any probability density or mass function is that it integrates to one. This is the role of  denominator \int P(data | \theta) \times P(\theta) d\theta  we simply called ‘average likelihood’. It standardizes P(data | \theta) \times P(\theta)  into the actual posterior P(\theta | data)  with density of one. the posterior is always proportional to the unstandardized posterior:

P(\theta | data) \propto P(data | \theta) \times P(\theta)  

That symbol \propto  means ‘proportional to’. We will now finalize the roulette example by standardizing the posterior computed above and comparing all pieces of the theorem.

* Type the following to update Posterior.R file



**Output:** 

We have finally reached the final form of the Bayes theorem, P(\theta | data) = \frac{P(data | \theta) \times P(\theta)}{\int P(data | \theta) \times P(\theta) d\theta} . The posterior of P(B)  can now be used to draw probability intervals or simulate new roulette draws.

**Push your work to GitHub**

**Make sure you are in**

Onsite students: CS251\_ Fall \_2020/**IN**/FirstnameLastname

Online students: CS251\_ Fall \_2020/**ON**/FirstnameLastname

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hit the **control + ~** key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 9”

>>> git push origin YOUR\_BRANCH\_NAME

Note: you should change the YOUR\_BRANCH\_NAME to your own branch name. It should be firstname-lastname (e.g. maria-gracia).