UTSSRP Tutorial: Bayesian Linear Regression

Prof. Gwen Eadie

2025-06-11

Fitting a simple regression line to mock data (frequentist approach)

This exercise follows section 6.2 in Gelman, Hill, and Vehtari (2021).

1. Simulate 20 fake data points from the model

$$y_i = a + bx_i + \epsilon_i$$

   where there is one predictor (covariate) that takes on values between 1 and 20, and where the intercept is 0.2 and the slope is 0.3, and

$$\epsilon_i \sim N(0, \sigma = 0.5)$$

```
# write code and a function to simulate y values here.
n = 20
x = 1:20

# intercept
beta0 = 0.2

# slope
beta1 = 0.3

# sigma for random error
mysigma = rnorm(20, 0, 0.5)
# simulated y values
y = beta0 + beta1*x + mysigma
```

2. Fit a standard linear regression model to these fake data. You will need to create a `data.frame` containing both the predictor and the outcome. Use the function `lm` to fit the simple linear regression.

```
# create data frame with covariate and outcome in columns
fakedata = data.frame(x, y)

# you should use the lm function and assign the results to a new object
lmfit = lm(y ~ x, data = fakedata)
# lmfit
```

3. Now display the results using `print`. What does print show you?

```
print(lmfit)
```

```
##
## Call:
## lm(formula = y ~ x, data = fakedata)
##
## Coefficients:
```

```
## (Intercept)            x
##     -0.1005       0.3167
```

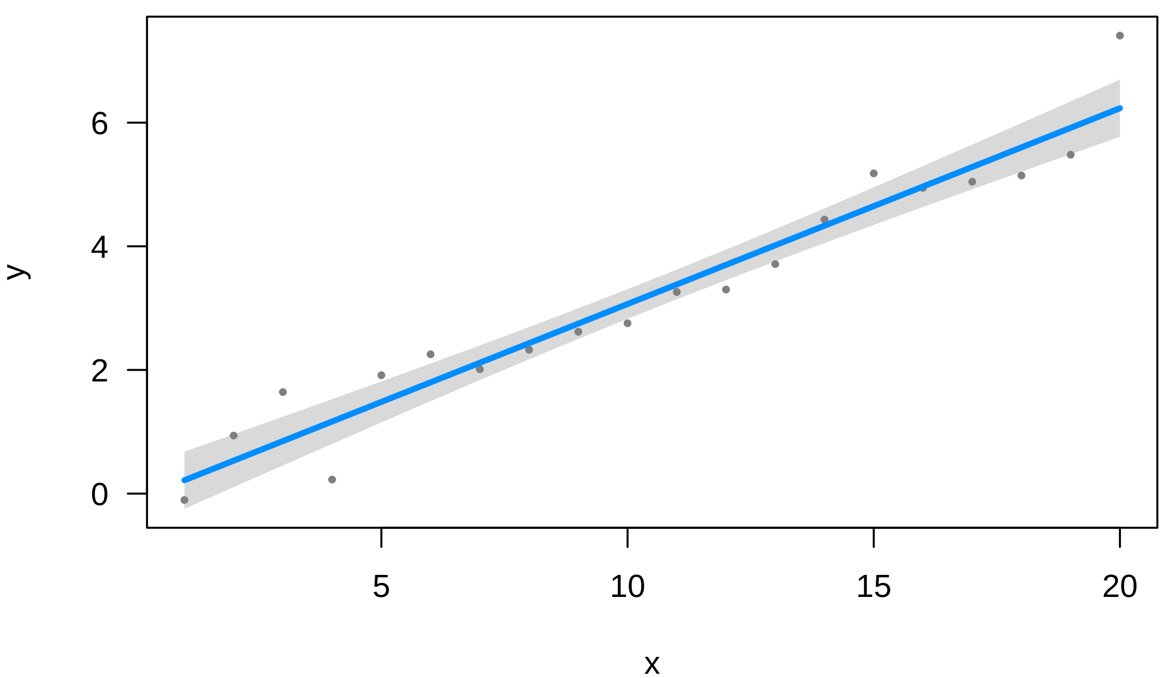
4. Display results using `summary`, which gives a little more information (e.g., the standard error of each estimate)

```r
summary(lmfit)
```

```
##
## Call:
## lm(formula = y ~ x, data = fakedata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9391 -0.3134 -0.1139  0.4123  1.1747
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.10055    0.23729  -0.424    0.677
## x            0.31669    0.01981  15.987 4.42e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5108 on 18 degrees of freedom
## Multiple R-squared:  0.9342, Adjusted R-squared:  0.9306
## F-statistic: 255.6 on 1 and 18 DF,  p-value: 4.417e-12
```

5. Plot the data, regression line, and confidence bands

```r
# simple plot without confidence bands

# can use the visreg package to easily make fit line with confidence bands
# install.packages("visreg")
library(visreg)
visreg(lmfit)
```



Inferring the parameters for a line (Bayesian approach)

Using the same mock data, infer the parameters of the intercept and slope.

```r
# load packages
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.3.0
## v purrr     1.0.4     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
#install.packages("tidybayes")
library(tidybayes)
#install.packages("brms")
library(brms)
```

```
## Loading required package: Rcpp
## Loading 'brms' package (version 2.22.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
##
## Attaching package: 'brms'
##
## The following objects are masked from 'package:tidybayes':
##
##     dstudent_t, pstudent_t, qstudent_t, rstudent_t
##
## The following object is masked from 'package:stats':
##
##     ar
```

1. Run the bayesian analysis using the `brm` function from the `brms` package. We will use the default priors in stan_lm. To read the help, do `?brm`.

```r
# use Bayesian Regression Model (brm) function from brms package
# Fit Bayesian linear model with default priors
bayes_fit <- brm(
  formula = y ~ x,
  data = fakedata)
```

```
## Compiling Stan program...
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: ‘Apple clang version 17.0.0 (clang-1700.0.13.5)’
## using SDK: ‘MacOSX15.5.sdk’
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
```

```
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen/include/Eigen/src/Core
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'anon_model' 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 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: 0.015 seconds (Warm-up)
## Chain 1:                0.014 seconds (Sampling)
## Chain 1:                0.029 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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:
```

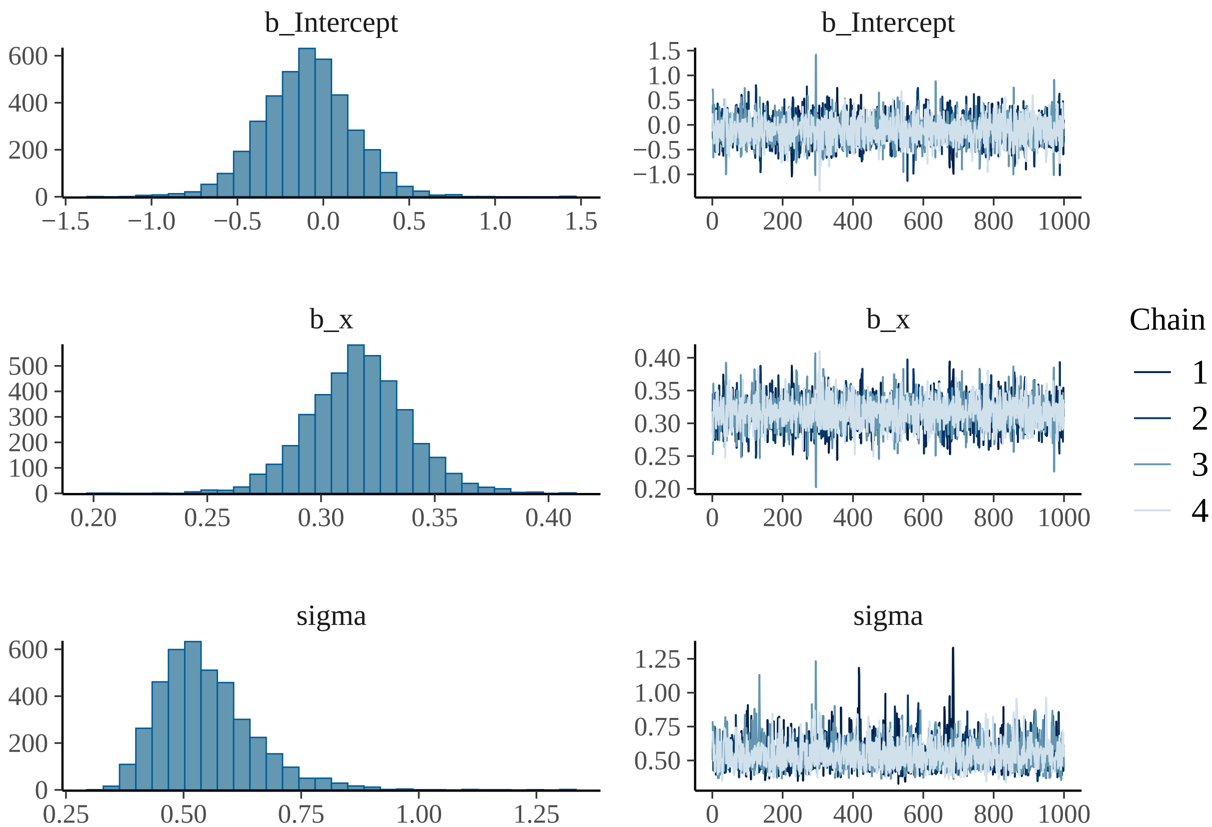
```
## Chain 2:  Elapsed Time: 0.02 seconds (Warm-up)
## Chain 2:                0.016 seconds (Sampling)
## Chain 2:                0.036 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.019 seconds (Warm-up)
## Chain 3:                0.013 seconds (Sampling)
## Chain 3:                0.032 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.017 seconds (Warm-up)
## Chain 4:                0.014 seconds (Sampling)
## Chain 4:                0.031 seconds (Total)
## Chain 4:
```

```r
get_prior(y ~ x, data = fakedata)
```

```
##                 prior     class coef group resp dpar nlpar lb ub       source
##                (flat)         b                                       default
##                (flat)         b    x                             (vectorized)
##  student_t(3, 3, 2.5) Intercept                                       default
##  student_t(3, 0, 2.5)     sigma                              0        default
```

2. Look at the chains and summary of the output of the inferred posterior distribution

```r
# plot the Markov chains and marginal distribution of the draws
plot(bayes_fit)
```




```r
# get the summary output
summary(bayes_fit)
```

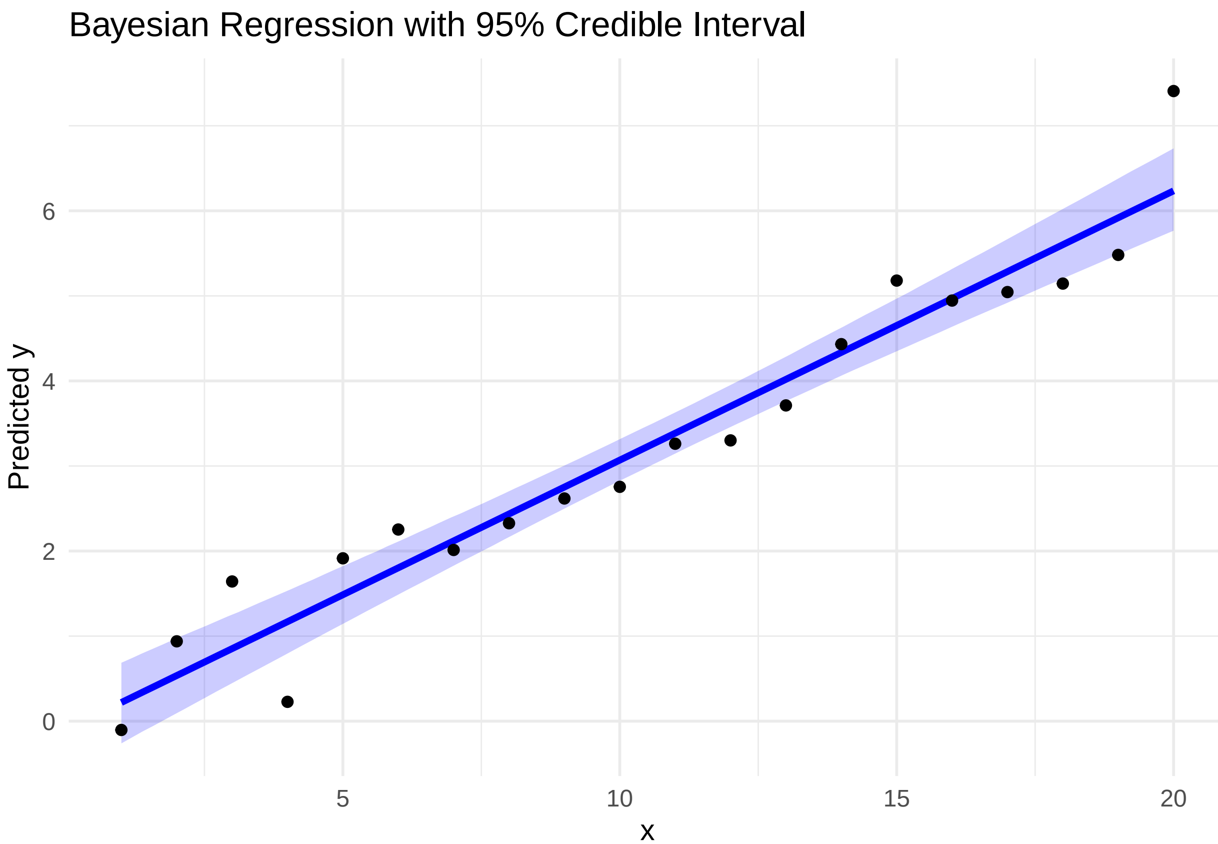
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ x
##    Data: fakedata (Number of observations: 20)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.10      0.26    -0.62     0.41 1.00     3442     2539
## x             0.32      0.02     0.27     0.36 1.00     3704     2535
##
## Further Distributional Parameters:
```

```
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.55      0.10     0.39     0.80 1.00     2511     2083
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

3. Plot the inferred relationship using the posterior distribution

```r
# Create a new data frame that will be used to generate predictions given the (Bayesian) regression pre
new_data <- data.frame(x = seq(1, 20, length.out = 100))

# Get fitted values for the new_data object
fitted_values <- fitted(bayes_fit, newdata = new_data)

# Use the posterior draws and new_data object to calculate credible intervals. Use the `posterior_epred
posterior_draws <- posterior_epred(bayes_fit, newdata = new_data)

# Calculate the credible interval (e.g., 95% CI) from the posterior draws using the quantile function:
credible_interval <- apply(posterior_draws, 2, function(draws_at_x) {
  c(mean = mean(draws_at_x),
    lower = quantile(draws_at_x, 0.025),
    upper = quantile(draws_at_x, 0.975))
})
# Now plot the line and credible intervals (I recommend using the ggplot2 package)

# Convert to data frame and add x values
credible_interval <- as.data.frame(t(credible_interval))
colnames(credible_interval) <- c("mean", "lower", "upper")
credible_interval$x <- new_data$x

library(ggplot2)

ggplot(credible_interval, aes(x = x, y = mean)) +
  geom_line(color = "blue", size = 1.2) +  # posterior mean line
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "blue", alpha = 0.2) +  # 95% credible band
  geom_point(data = fakedata, aes(x = x, y = y), color = "black", size = 1.5) +  # original data points
  labs(
    title = "Bayesian Regression with 95% Credible Interval",
    x = "x",
    y = "Predicted y"
  ) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Bayesian Regression with 95% Credible Interval
Predicted y
0
2
4
6
x
5
10
15
20