Topics in Multivariate Analysis

APSTA GE-2004

Lecture 1 - Introduction

1/25/2022

Outline

Welcome! Today, we'll cover the following:

- Review of linear regression
  - Fitting model
  - Plotting data and fitted model
  - Posterior predictive checking
  - Interpreting results
- Examples
- Course logistics

Reading:

RAOS 1.5-1.6; 4.2, 4.4-4.5; Ch. 6-7; Ch 10.1-10.4; Ch 11.1-11.4
Bayesian data analysis for newcomers by Kruschke and Liddell
RAOS Appendix A and B

Introduction to Regression

What is regression?

Regression is a framework for learning about something using quantitative measurements *and* assessing “uncertainty” in what we’ve learned.

Regression focuses on how a quantity of interest depends on another quantity.

Regression is a statistical technique that summarizes how average values of an **outcome variable** vary across units defined by a **predictor variable**.
[or **predictor variables**]

Why use regression?

We can use regression in a variety of ways:

- For *prediction*: forecasting an election, predicting future sales, or making a medical diagnosis
- To understand *associations*: identifying risk factors for a disease or attitudes that predict voting
- To *extrapolate* from a sample to a population: using data from a non-representative poll to extrapolate to the general population
- To estimate *treatment effects*: the effect of a new teaching method on standardized test scores, or of exposure to a pollutant on incidence of asthma

Linear regression

A method to summarize how the average values of a numerical *outcome* variable vary over subpopulations defined by linear functions of *predictors*

A simple regression model is linear with a single predictor:

$$y = a + bx + \text{error}$$

The quantities a and b are called *coefficients* or, more generally, *parameters*

Example: kid_score = a + b*mom_iq + error

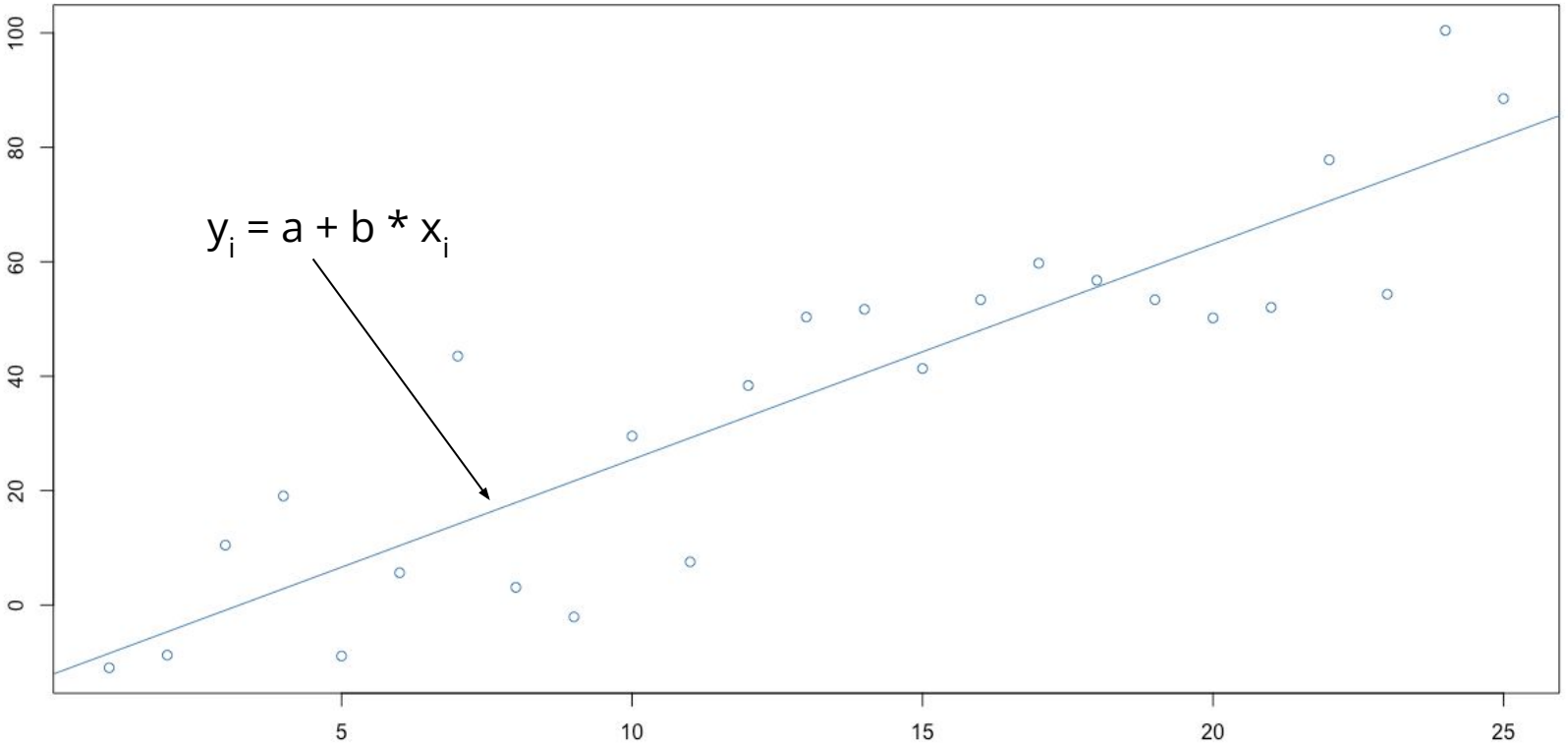
Linear regression: Population parameters vs Sample Statistics

Saying it differently, the expected value (population mean) of Y, rather than the exact individual values, changes linearly with X:

$E[Y_i] = \beta_0 + \beta_1 X_i$

Y-hat (predicted value): $y_i = a + bX_i$ (sample counterpart)

Linear regression: Predicted/Fitted values



Linear regression: Population parameters vs Sample Statistics

Saying it differently, the expected value (population mean) of Y, rather than the exact individual values, changes linearly with X:

$E[Y_i] = \beta_0 + \beta_1 X_i$

Y-hat (predicted value): $y_i = a + bX_i$ (sample counterpart)

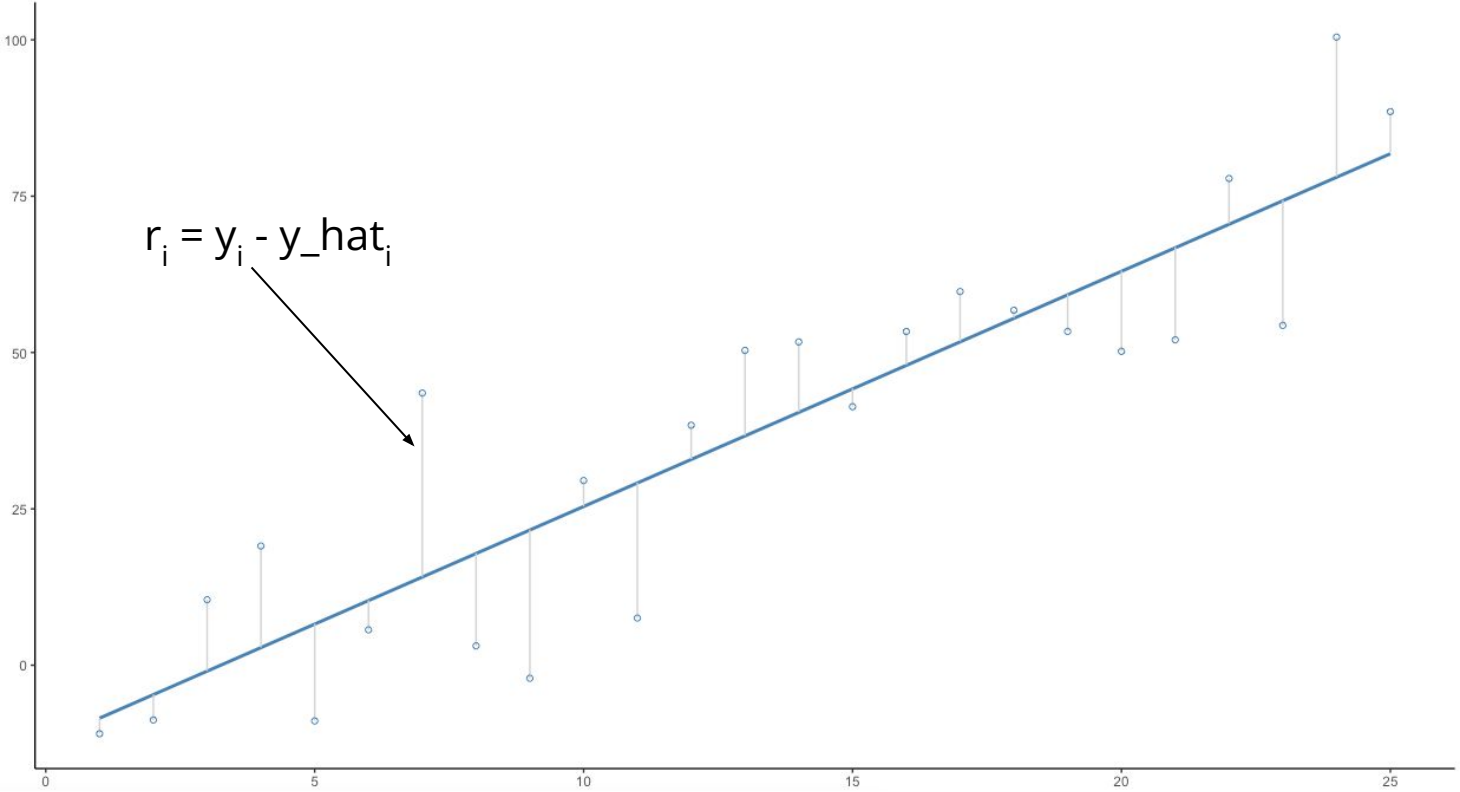
Other things besides X cause individual Y_i to vary around $E[Y_i]$. We represent these "other things" with an error term, ε_i (epsilon):

$\varepsilon_i = Y_i - (\beta_0 + \beta_1 X_i)$

$= Y_i - E[Y_i]$

Residuals (difference between actual and predicted): $r_i = y_i - y\_hat_i$ (sample counterpart)

Linear regression: Residuals



Linear regression: Population parameters vs Sample Statistics

Saying it differently, the expected value (population mean) of Y, rather than the exact individual values, changes linearly with X:

$E[Y_i] = \beta_0 + \beta_1 X_i$

Y-hat (predicted value): $y_i = a + bX_i$ (sample counterpart)

Other things besides X cause individual Y_i to vary around $E[Y_i]$. We represent these "other things" with an error term, ε_i (epsilon):

$\varepsilon_i = Y_i - (\beta_0 + \beta_1 X_i)$

$= Y_i - E[Y_i]$

Residuals (difference between actual and predicted): $r_i = y_i - y\_hat_i$ (sample counterpart)

Actual Y_i equals expectation plus error:

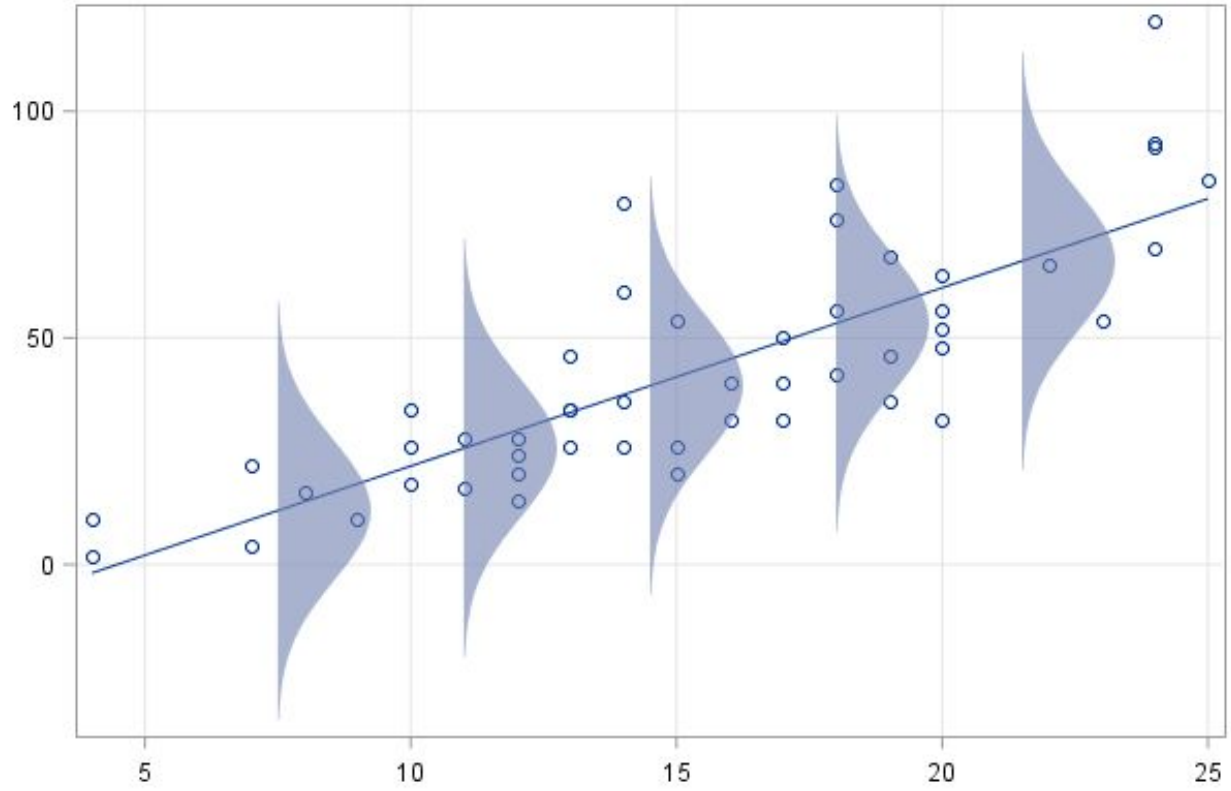
$Y_i = E[Y_i] + \varepsilon_i$

$= \beta_0 + \beta_1 X_i + \varepsilon_i$

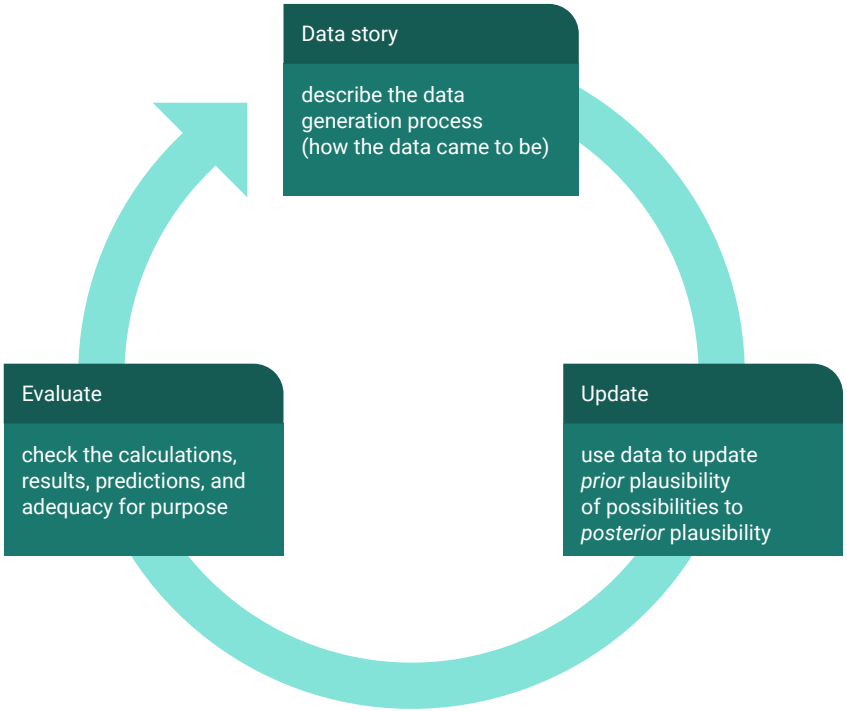
Common assumptions about the errors are that they are normal i.i.d.:

1. Errors have *identical distributions*, with zero mean and the same variance, for every value of X
2. Errors are *independent*: unrelated to X variables or to the errors of other units
3. Errors are *normally distributed*

Linear regression: Errors are normal i.i.d.



How do we design the model? Basic model design loop.



Classical Linear Regression (Binary predictor)

Linear model

We can describe our linear model (kid_score = a + b*mom_hs + error) with the following model components:

$y_i \sim \text{Normal}(\mu_i, \sigma)$ [likelihood]

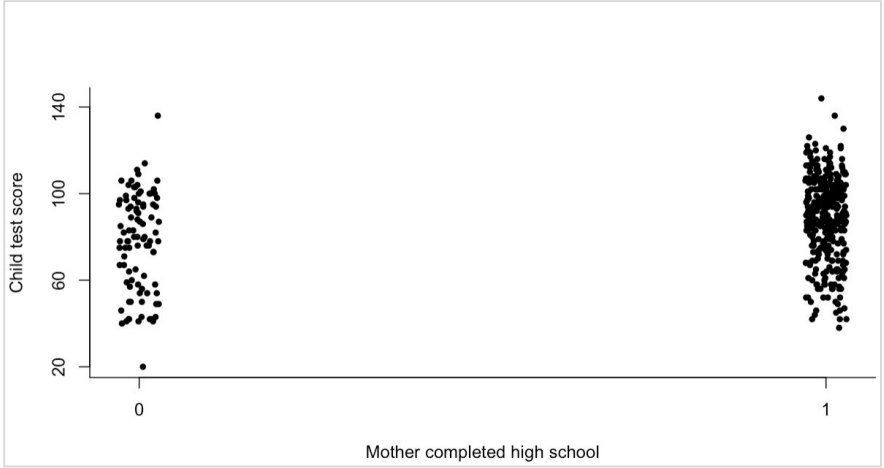
$\mu_i = a + b*x_i$ [linear model]

Linear regression: OLS

Now let's see how to specify this model as a classical regression in R:

```
# Fit a classical linear regression
options(show.signif.stars = FALSE,
        show.coef.Pvalues = FALSE)
fit_ols <- lm(kid_score ~ mom_hs, data=kidiq)
summary( fit_ols )
arm::display( fit_ols )
plot( fit_ols )
```

```
# Display the data and fitted regression
jitt <- runif(nrow(kidiq), -.03, .03)
plot(kidiq$mom_hs + jitt, kidiq$kid_score,
     xlab="Mother completed high school", ylab="Child test score",
     bty="l", pch=20, xaxt="n", yaxt="n")
axis(1, c(0,1))
axis(2, seq(20, 140, 40))
abline(coef(fit_ols), col="steelblue", lwd=2)
```



Interpreting OLS output: Coefficients

The fitted model is:

kid_score = 78 + 12 * mom_hs + error

Interpretation:

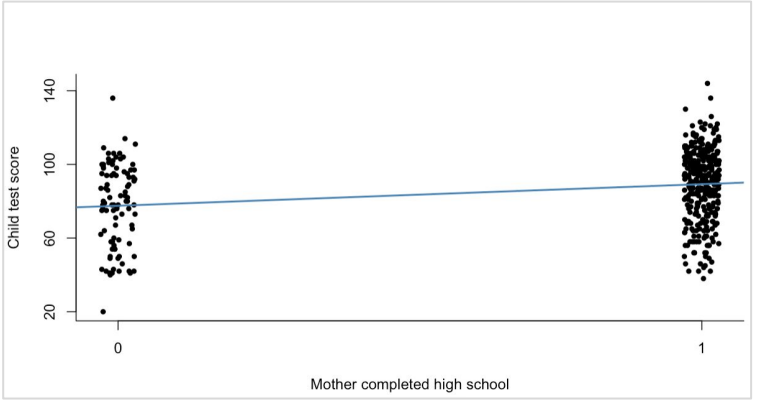
For a binary predictor, the regression coefficient (*here, mom_hs*) is the difference between the averages of the two groups. This model summarizes the difference in average test scores between the children of mothers who complete high school and those with mothers who did not

Intercept:

The intercept, 78, is the average (or predicted) score for children whose mothers did not complete high school

Coefficient of maternal HS completion:

The difference between these two subpopulations means is equal to the coefficient on mom_hs. This coefficient tells us that children of mother who have competed high school score 12 points higher on average than children of mothers who have not completed high school




```
Call:
lm(formula = kid_score ~ mom_hs, data = kidiq)

Residuals:
    Min      1Q  Median      3Q     Max
-57.55 -13.32   2.68  14.68  58.45

Coefficients:
            Estimate Std. Error t value
(Intercept)   77.548      2.059  37.670
mom_hs        11.771      2.322   5.069

Residual standard error: 19.85 on 432 degrees of freedom
Multiple R-squared:  0.05613,   Adjusted R-squared:  0.05394
F-statistic: 25.69 on 1 and 432 DF,  p-value: 5.957e-07
```

Interpreting OLS output: Coefficient Standard Errors

Standard errors:
The standard error is the estimated standard deviation of an estimate, which gets smaller as sample size gets larger, converging on zero as the sample increases in size.

Slope (bivariate)**:**

$SE(b_1) = s_e / \sqrt{[TSS_x]}$

$SE(b_1) = \text{sqrt}\ [\ \Sigma(y_i - \hat{y}_i)^2 / (n - K)\ ] / \text{sqrt}\ [\ \Sigma(x_i - x\_bar)^2\ ]$

Intercept (bivariate)**:**

$SE(b_0) = s_e * \sqrt{[(1/n) + (x\_bar^2 / TSS_x)]}$

$SE(b_0) = \text{sqrt}\ [\ \Sigma(y_i - \hat{y}_i)^2 / (n - K)\ ] *$

$\text{sqrt}\ [\ (1/n) + (x\_bar^2 / \Sigma(x_i - x\_bar)^2)\ ]$

Confidence intervals:
The *confidence interval* represents a range of values of a parameter that are roughly consistent with the data, given the assumed sampling distribution. If the model is correct, then in repeated applications the 50% and 95% confidence intervals will include the true value 50% and 95% of the time

The usual 95% confidence interval for large samples, based on an assumption that the sampling distribution follows the normal distribution, is to take an estimate ±2 standard errors

```
Call:
lm(formula = kid_score ~ mom_hs, data = kidiq)

Residuals:
    Min      1Q  Median      3Q     Max
-57.55 -13.32   2.68  14.68  58.45

Coefficients:
            Estimate Std. Error t value
(Intercept)   77.548      2.059  37.670
mom_hs        11.771      2.322   5.069

Residual standard error: 19.85 on 432 degrees of freedom
Multiple R-squared:  0.05613,   Adjusted R-squared:  0.05394
F-statistic: 25.69 on 1 and 432 DF,  p-value: 5.957e-07
```

95% confidence intervals

```
              2.5 % 97.5 %
(Intercept) 73.50  81.59
mom_hs       7.21  16.34
```

```
# Display confidence intervals for the estimates
confint( fit_ols , level = .95)

# CIs "by-hand"
res <- summary( fit_ols )
names(res)

n <- length(kidiq$kid_score)
# CI for intercept
res$coefficients[1, "Estimate"] +
  qt(c(0.025,0.975), n-1) * res$coefficients[1, "Std. Error"]
# CI for slope
res$coefficients[2, "Estimate"] +
  qt(c(0.025,0.975), n-1) * res$coefficients[2, "Std. Error"]
```

Bayesian Linear Regression (Binary predictor)

Linear model

We can describe our linear model (kid_score = a + b*mom_hs + error) with the following model components:

$y_i \sim \text{Normal}(\mu_i, \sigma)$ [likelihood]

$\mu_i = a + b*x_i$ [linear model]

$a \sim \text{Normal}(87, 2.5)$ [a prior]

$b \sim \text{Normal}(0, 2.5)$ [b prior]

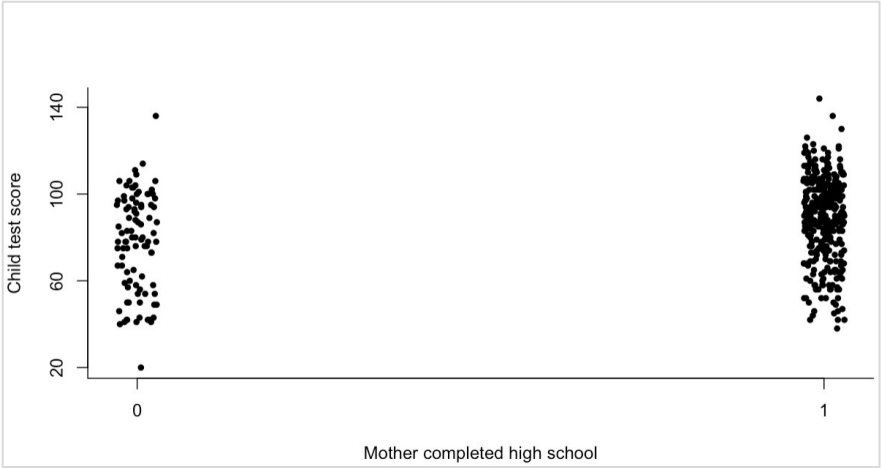
$\sigma \sim \text{Exponential}(1)$ [σ prior]

Priors on parameters are necessary in Bayesian regression, but we'll let `rstanarm` compute sensible priors for us

Linear regression: Bayesian

Now let's see how to specify this model as a Bayesian regression in R:

```
# Fit a Bayesian linear regression
fit_1 <- stan_glm(kid_score ~ mom_hs, data=kidiq)
print( fit_1 )
summary( fit_1 )
plot( fit_1 )


# Display the data and fitted regression
jitt <- runif(nrow(kidiq), -.03, .03)
plot(kidiq$mom_hs + jitt, kidiq$kid_score,
     xlab="Mother completed high school", ylab="Child test score",
     bty="l", pch=20, xaxt="n", yaxt="n")
axis(1, c(0,1))
axis(2, seq(20, 140, 40))
abline(coef(fit_1), col="steelblue", lwd=2)

# Display uncertainty in the fitted regression
sims_1 <- as.matrix(fit_1)
n_sims_1 <- nrow(sims_1)
subset <- sample(n_sims_1, 10)
for (i in subset){
  abline(sims_1[i,1], sims_1[i,2], col="gray")
}
abline(coef(fit_1)[1], coef(fit_1)[2], col="black")
```



Interpreting Bayesian output: Model Info

Function / Family / Formula

Generalized linear model with optional prior distributions for the coefficients, intercept, and auxiliary parameters.

Algorithm

- "sampling" for MCMC (the default)
- "optimizing" for optimization
- "meanfield" or "fullrank" for variational inference

Sample

2000 iterations ; 4 chains ; 3 parameters (a, b, sigma)

Priors

e.g.

Family	Functions
Student t family	`normal`, `student_t`, `cauchy`
Hierarchical shrinkage family	`hs`, `hs_plus`
Laplace family	`laplace`, `lasso`
Product normal family	`product_normal`

Observations / Predictors

Double-check the number of observations and predictors

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
              mean   sd   10%   50%   90%
(Intercept) 77.5    2.1 74.8  77.5  80.1
mom_hs      11.8    2.4  8.8  11.8  14.9
sigma       19.9    0.7 19.0  19.9  20.8

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 86.8    1.4 85.0  86.8  88.6

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.0  1.0  3889
mom_hs        0.0  1.0  4031
sigma         0.0  1.0  3833
mean_PPD      0.0  1.0  4047
log-posterior 0.0  1.0  1686

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: MCMC Diagnostics

Monte Carlo standard error (mcse):
the uncertainty about a statistic in the sample due to sampling error

Potential Scale Reduction (Rhat):
In equilibrium, the distribution of samples from chains should be the same regardless of the initial starting values of the chains. One way to check this is to compare the distributions of multiple chains — in equilibrium they should all have the same mean.

Rule of Thumb: the R-hat values for all parameters are less than 1.1

Effective sample size (ESS, n_eff):
measures the amount by which autocorrelation in samples increases uncertainty (standard errors) relative to an independent sample

Rule of Thumb: the effective sample sizes (n_eff) for all parameters are greater than 2000

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
             mean   sd   10%   50%   90%
(Intercept) 77.5    2.1 74.8  77.5  80.1
mom_hs      11.8    2.4  8.8  11.8  14.9
sigma       19.9    0.7 19.0  19.9  20.8

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.4 85.0  86.8  88.6

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
             mcse Rhat n_eff
(Intercept)  0.0  1.0  3889
mom_hs       0.0  1.0  4031
sigma        0.0  1.0  3833
mean_PPD     0.0  1.0  4047
log-posterior 0.0  1.0  1686

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Coefficients

The fitted model is:

kid_score = 78 + 12 * mom_hs + error

Interpretation:

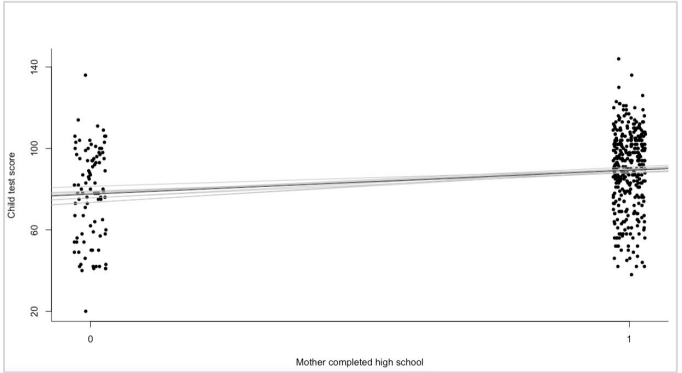
For a binary predictor, the regression coefficient (*here, mom_hs*) is the difference between the averages of the two groups. This model summarizes the difference in average test scores between the children of mothers who complete high school and those with mothers who did not

Intercept:

The intercept, 78, is the average (or predicted) score for children whose mothers did not complete high school

Coefficient of maternal HS completion:

The difference between these two subpopulations means is equal to the coefficient on mom_hs. This coefficient tells us that children of mother who have competed high school score 12 points higher on average than children of mothers who have not completed high school




```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
              mean   sd   10%   50%   90%
(Intercept) 77.5    2.1 74.8  77.5  80.1
mom_hs      11.8    2.4  8.8  11.8  14.9
sigma       19.9    0.7 19.0  19.9  20.8

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 86.8    1.4 85.0  86.8  88.6

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.0  1.0  3889
mom_hs        0.0  1.0  4031
sigma         0.0  1.0  3833
mean_PPD      0.0  1.0  4047
log-posterior 0.0  1.0  1686

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Standard Errors

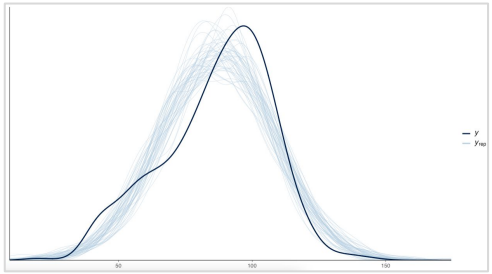
Standard errors:
The standard error is the estimated standard deviation of an estimate, which gets smaller as sample size gets larger, converging on zero as the sample increases in size.

```
# Extract posterior draws from the fitted model object
posterior <- as.array(fit_1)
dim(posterior)
dimnames(posterior)
head(posterior)

# Means and Standard Deviations
# Medians and Median Absolute Deviations (MAD)
# Intercept
mean(posterior[ , , "(Intercept)"])       median(posterior[ , , "(Intercept)"])
sd(posterior[ , , "(Intercept)"])         mad(posterior[ , , "(Intercept)"])

# Slope
mean(posterior[ , , "mom_iq_centered"])   median(posterior[ , , "mom_hs"])
sd(posterior[ , , "mom_iq_centered"])     mad(posterior[ , , "mom_hs"])

# Sigma
mean(posterior[ , , "sigma"])             median(posterior[ , , "sigma"])
sd(posterior[ , , "sigma"])               mad(posterior[ , , "sigma"])

# rstanarm: Bayesian compatibility intervals
ci95 <- posterior_interval(fit_1, prob = 0.95)
round(ci95, 2)
```

Compatibility intervals:
The *compatibility interval* represents a range of values of a parameter that are roughly consistent with the data and model

These posterior intervals report two parameter values that contain between them a specified amount of posterior probability, a probability mass

Posterior predictive check (ppc)




```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
              mean   sd   10%   50%   90%
(Intercept) 77.5    2.1 74.8  77.5  80.1
mom_hs      11.8    2.4  8.8  11.8  14.9
sigma       19.9    0.7 19.0  19.9  20.8

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.4 85.0  86.8  88.6

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.0  1.0  3889
mom_hs        0.0  1.0  4031
sigma         0.0  1.0  3833
mean_PPD      0.0  1.0  4047
log-posterior 0.0  1.0  1686

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

95% compatibility intervals

```
             2.5% 97.5%
(Intercept) 73.31 81.62
mom_hs       7.23 16.34
```

Continuous predictor

Linear model

We can describe our linear model (kid_score = a + b*mom_iq_centered + error) with the following model components:

$$y_i \sim \text{Normal}(\mu_i, \sigma) \quad [\text{ likelihood }]$$

$$\mu_i = a + b(x_i - \text{mean}(x)) \quad [\text{ linear model }]$$

$$a \sim \text{Normal}(87, 2.5) \quad [\text{ a prior }]$$

$$b \sim \text{Normal}(0, 2.5) \quad [\text{ b prior }]$$

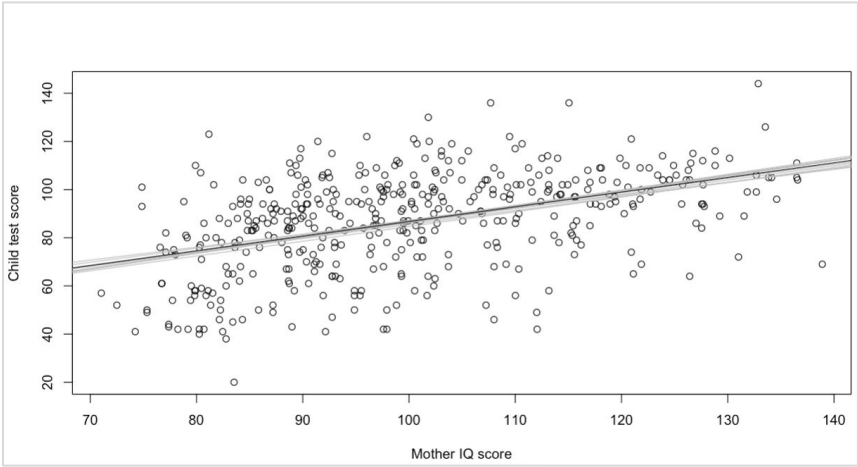
$$\sigma \sim \text{Exponential}(1) \quad [\text{ } \sigma \text{ prior }]$$

Priors on parameters are necessary in Bayesian regression, but we'll let `rstanarm` compute sensible priors for us

Linear regression

Now let's see how to specify this model as a regression in R:

```
# Fit a Bayesian linear regression
mean_mom_iq <- mean(kidiq$mom_iq)
mom_iq_centered <- kidiq$mom_iq - mean_mom_iq
fit_2 <- stan_glm(kid_score ~ mom_iq_centered, data=kidiq)
print(fit_2)
summary(fit_2)
plot(fit_2)

# Display the data and fitted regression
# Plot points and regression line (x-axis original units)
labels <-
  c(-40, -20, 0, 20, 40) + mean(kidiq$mom_iq) %>%
  round(digits = 0)

kidiq %>%
  ggplot(aes(x = mom_iq_centered, y = kid_score)) +
  geom_abline(intercept = coef(fit_2)[1],
              slope     = coef(fit_2)[2]) +
  geom_point(shape = 1, size = 2) +
  scale_x_continuous("Mother IQ score",
                     breaks = c(-40, -20, 0, 20, 40),
                     labels = labels) +
  labs(y = "Child test score", title = "") +
  theme_classic() +
  theme(plot.title=element_text(hjust=0.5))
```



Interpreting Coefficients

The fitted model is:

kid_score = 87 + 0.6 * mom_iq_centered + error

Interpreting points on fitted line:

Either as predicted test scores for kids at each of several maternal IQ levels, or as average test scores for subpopulations defined by these scores

Intercept:

Since we centered mom IQ, the intercept reflects the predicted test scores of kids whose mothers have average IQ (100)

Coefficient of maternal IQ score:

If we compare average kid scores for subpopulations that differ in maternal IQ by 1 point, we expect to see that the group with higher maternal IQ achieves 0.6 points more on average
(if IQs differ by 10 points, scores differ by 6 points on average)

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Binary & Continuous predictors (without interaction)

Linear model

We can describe our linear model (kid_score = a + b_1*mom_hs + b_2*mom_iq_centered + error) with the following model components:

$$y_i \sim \text{Normal}(\mu_i, \sigma) \quad \text{[likelihood]}$$

$$\mu_i = a + b_1 x_{1i} + b_2(x_{2i} - \text{mean}(x_2)) \quad \text{[linear model]}$$

$$a \sim \text{Normal}(87, 2.5) \quad \text{[a prior]}$$

$$b_1 \sim \text{Normal}(0, 2.5) \quad \text{[} b_1 \text{ prior]}$$

$$b_2 \sim \text{Normal}(0, 2.5) \quad \text{[} b_2 \text{ prior]}$$

$$\sigma \sim \text{Exponential}(1) \quad \text{[} \sigma \text{ prior]}$$

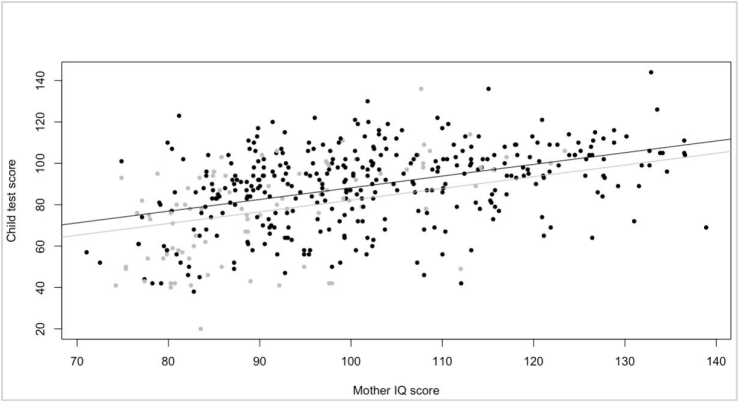
Priors on parameters are necessary in Bayesian regression, but we'll let `rstanarm` compute sensible priors for us

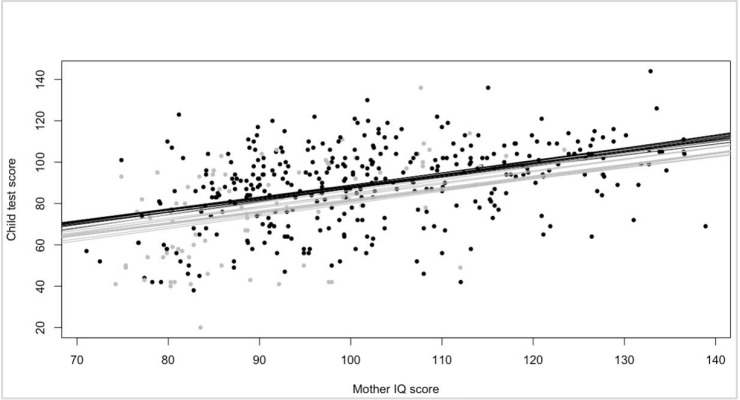
Linear regression

Now let's see how to specify this model as a regression in R:

```
# Fit a Bayesian linear regression
fit_3 <- stan_glm(kid_score ~ mom_hs + mom_iq_centered, data=kidiq)
print(fit_3)
summary(fit_3)
plot(fit_3)
```

```
# Display uncertainty in the fitted regression
# Lines on same plot
sims_3 <- as.matrix(fit_3)
n_sims_3 <- nrow(sims_3)
subset <- sample(n_sims_3, 10)
colors <- ifelse(kidiq$mom_hs==1, "black", "gray")
plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score", col=colors, pch=20)
for (i in subset){
  abline(sims_3[i,1], sims_3[i,3], col="gray")
  abline(sims_3[i,1] + sims_3[i,2], sims_3[i,3], col="black")
}
abline(coef(fit_3)[1], coef(fit_3)[3], col="gray")
abline(coef(fit_3)[1] + coef(fit_3)[2], coef(fit_3)[3], col="black")
```





Interpreting Coefficients

The fitted model is:

kid_score = 82 + 6 * mom_hs + 0.6 * mom_iq_centered + error

Intercept:
If a child had a mother with average IQ who did not complete HS, the model predicts the child's test score to be 82

Coefficient of maternal HS completion:
Comparing children whose mothers have the same IQ, but who differed in whether they completed HS, the model predicts an expected difference of 6 in the test scores

Coefficient of maternal IQ score:
Comparing children with the same value of mom_hs, but whose mothers differ by 1 point in IQ, we expect a difference of 0.6 points in the child's test score

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs + mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   3

Estimates:
                mean   sd   10%   50%   90%
(Intercept)    82.1    2.0 79.6  82.1  84.7
mom_hs          6.0    2.2  3.0   6.0   8.8
mom_iq_centered 0.6    0.1  0.5   0.6   0.6
sigma          18.2    0.6 17.4  18.2  18.9

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4666
mom_hs          0.0  1.0  4495
mom_iq_centered 0.0  1.0  4418
sigma           0.0  1.0  4431
mean_PPD        0.0  1.0  4389
log-posterior   0.0  1.0  1772

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Binary & Continuous predictors (with interaction)

Linear model

We can describe our linear model (kid_score = a + b_1*mom_hs + b_2*mom_iq_centered + b_3*mom_hs*mom_iq_centered + error) with the following model components:

$y_i \sim \text{Normal}(\mu_i, \sigma)$ [likelihood]

$\mu_i = a + b_1 x_{1i} + b_2 x_{2i} + b_3 x_{1i} * x_{2i}$ [linear model (x_2 centered)]

$a \sim \text{Normal}(87, 2.5)$ [a prior]

$b_1 \sim \text{Normal}(0, 2.5)$ [b_1 prior]

$b_2 \sim \text{Normal}(0, 2.5)$ [b_2 prior]

$b_3 \sim \text{Normal}(0, 2.5)$ [b_3 prior]

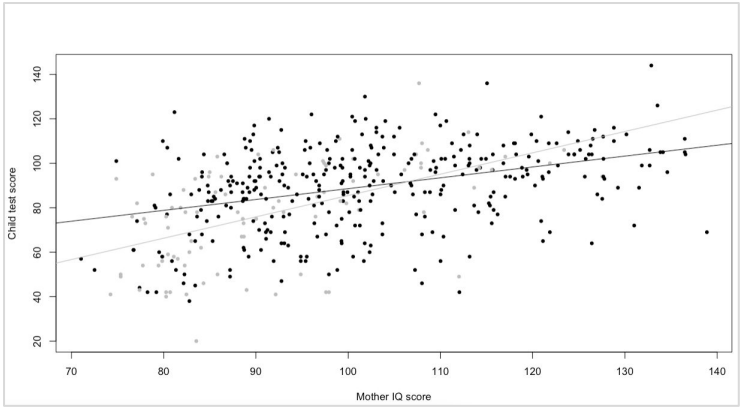
$\sigma \sim \text{Exponential}(1)$ [σ prior]

Priors on parameters are necessary in Bayesian regression, but we'll let `rstanarm` compute sensible priors for us

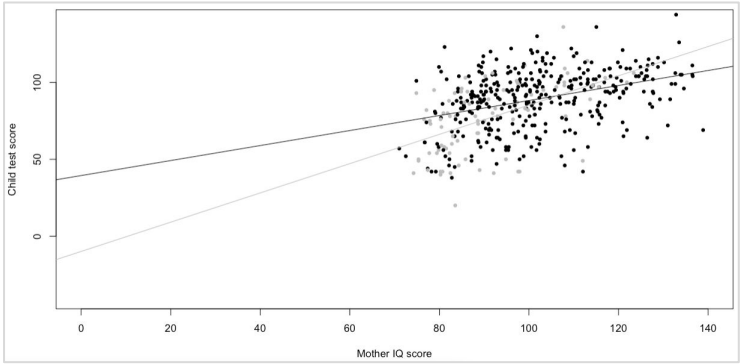
Linear regression

Now let's see how to specify this model as a regression in R:

```
# Fit a Bayesian linear regression
fit_4 <- stan_glm(kid_score ~ mom_hs + mom_iq_centered + mom_hs:mom_iq_centered,
                  data=kidiq)
print(fit_4)
summary(fit_4)
plot(fit_4)
```




```
# Display uncertainty in the fitted regression
# Lines on same plot
sims_4 <- as.matrix(fit_4)
n_sims_4 <- nrow(sims_4)
subset <- sample(n_sims_4, 10)
colors <- ifelse(kidiq$mom_hs==1, "black", "gray")
plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score", col=colors, pch=20)
for (i in subset){
  abline(sims_4[i,1], sims_4[i,3], col="gray")
  abline(sims_4[i,1] + sims_4[i,2], sims_4[i,3] + sims_4[i,4], col="black")
}
abline(coef(fit_4)[1], coef(fit_4)[3], col="gray")
abline(coef(fit_4)[1] + coef(fit_4)[2], coef(fit_4)[3] + coef(fit_4)[4], col="black")
```



Interpreting Coefficients

The fitted model is:

kid_score = 85 + 3 * mom_hs + 1 * mom_iq_centered
– 0.5 * mom_hs * mom_iq_centered + error

Intercept:
If a child had a mother with average IQ who did not complete HS, the model predicts the child's test score to be 85

Coefficient of maternal HS completion:
Comparing children whose mothers have the same average IQ, but who differ in whether they completed HS, the model predicts an expected difference of 3 in the test scores

Coefficient of maternal IQ score:
Comparing children whose mothers did not complete HS, but whose mothers differ by 1 point in IQ, the model predicts an expected difference of 1 point in the test scores

Coefficient of the interaction term:
Represents the difference in the slope for mom_iq_centered, comparing children with mothers who did and did not complete HS – that is, the difference between the slopes of the two regression lines

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs + mom_iq_centered + mom_hs:mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   4

Estimates:
                           mean   sd   10%   50%   90%
(Intercept)              85.3    2.2 82.6  85.3  88.1
mom_hs                    2.9    2.4 -0.1   2.9   5.9
mom_iq_centered           1.0    0.1  0.8   1.0   1.2
mom_hs:mom_iq_centered   -0.5    0.2 -0.7  -0.5  -0.3
sigma                    18.0    0.6 17.2  18.0  18.8

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                         mcse Rhat n_eff
(Intercept)              0.0  1.0  2343
mom_hs                   0.0  1.0  2461
mom_iq_centered          0.0  1.0  1770
mom_hs:mom_iq_centered   0.0  1.0  1820
sigma                    0.0  1.0  2668
mean_PPD                 0.0  1.0  3716
log-posterior            0.0  1.0  1399

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

mom_hs = 0: kid_score = 85 + 3***0*** + 1*mom_iq_centered – 0.5***0***mom_iq_centered
= 85 + 1*mom_iq_centered

mom_hs = 1: kid_score = 85 + 3***1*** + 1*mom_iq_centered – 0.5***1***mom_iq_centered
= 88 + 0.5*mom_iq_centered

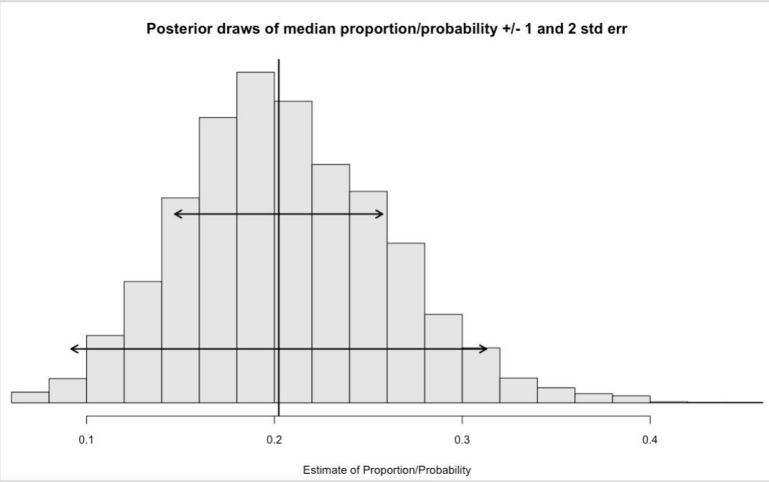
A Few Special Cases

Estimating a Proportion

Logistic regression *with just an intercept* is equivalent to the estimate of a proportion

Here is an example: a random sample of 50 people are tested, and 10 have a particular disease. The proportion is **0.20** with standard error √0.2*0.8/50 = **0.06**.

Alternatively we can set this up as logistic regression:




```
# Estimating a proportion
y <- rep(c(0,1), c(40,10))
simple <- data.frame(y)
fit_prop <- stan_glm(y ~ 1, family = binomial(link = "logit"), data = simple)
print(fit_prop, digits=2)

# Predicted probability
round( plogis(fit_prop$coefficients) , 3)

# Plus/Minus 1 standard error bounds on predicted probability
round( plogis(fit_prop$coefficients + (c(-1,1)*fit_prop$ses)), 3)
```

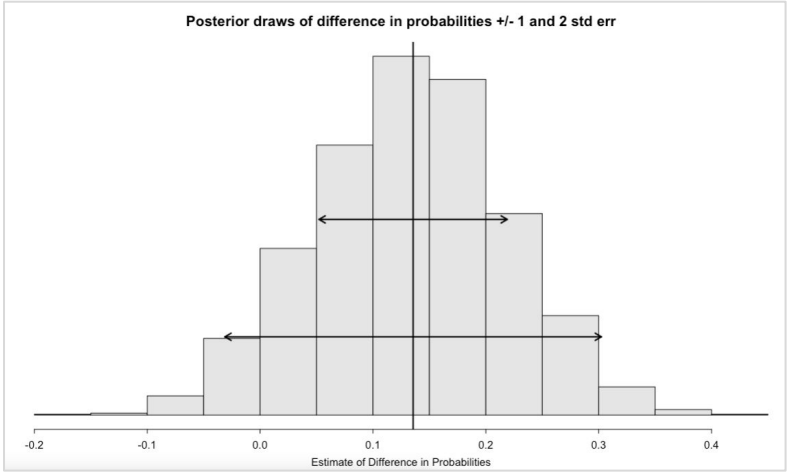
```
> # Inference on the probability scale
> new <- data.frame(x=0)
> epred <- posterior_epred(fit_prop, newdata = new)
> round(mean(epred),2)
[1] 0.2
> round(sd(epred),2)
[1] 0.06
```

```
stan_glm
 family:       binomial [logit]
 formula:      y ~ 1
 observations: 50
 predictors:   1
------
            Median MAD_SD
(Intercept) -1.38   0.36
```

Comparing Two Proportions

Logistic regression *on an indicator variable* is equivalent to a comparison of proportions

Here is an example: consider tests for a disease on samples from two populations, where 10 out of 50 from population A test positive, as compared to 20 out of 60 from population B. The classical estimate is **0.13** with standard error **0.08**.



Alternatively we can set this up as logistic regression:

```
# Comparing two proportions
x <- rep(c(0, 1), c(50, 60))
y <- rep(c(0, 1, 0, 1), c(40, 10, 40, 20))
simple <- data.frame(x, y)
fit_props <- stan_glm(y ~ x, family = binomial(link = "logit"), data = simple,
                      prior_intercept = NULL, prior = NULL, prior_aux = NULL)
print(fit_props, digits=3)
summary(fit_props, digits=3)

# Difference in probabilities
new <- data.frame(x = c(0, 1))
epred <- posterior_epred(fit_props, newdata = new)
diff <- epred[,2] - epred[,1]
print( c( round(mean(diff),3) , round(sd(diff),3) ) )
```

```
> # Difference in probabilities
> new <- data.frame(x = c(0, 1))
> epred <- posterior_epred(fit_props, newdata = new)
> diff <- epred[,2] - epred[,1]
> round(mean(diff),3)
[1] 0.133
> round(sd(diff),3)
[1] 0.084
```

```
stan_glm
 family:       binomial [logit]
 formula:      y ~ x
 observations: 110
 predictors:   2
------
            Median MAD_SD
(Intercept) -1.403  0.348
x            0.714  0.457
```

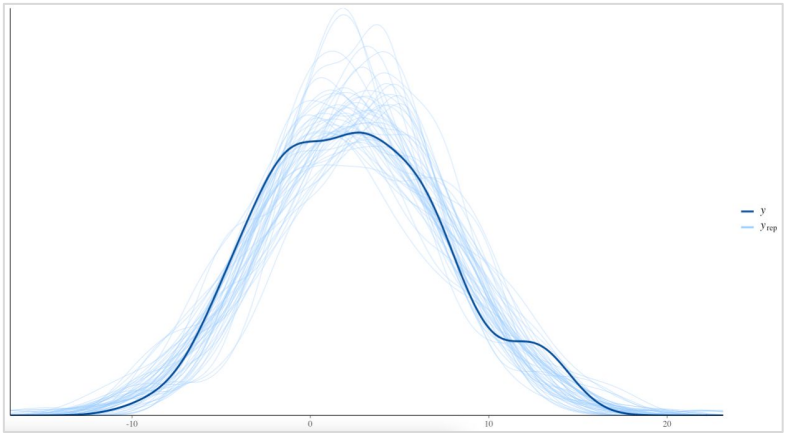
Estimating a Mean

Linear regression *with just an intercept* is equivalent to the estimate of a mean

Here is an example: simulate 200 (n_0) observations (y_0) from a population with mean = **2.0** and standard deviation = **5.0**.

Considering these as a random sample, we can estimate the mean of the population as mean(y_0) with standard error sd(y_0) / sqrt(n_0). The result: an estimate of **2.4** with standard error **0.36**.

Alternatively we can set this up as linear regression:




```
# Estimating a mean
n_0 <- 200
set.seed(2141)
y_0 <- rnorm(n_0, 2.0, 5.0)
fake_0 <- data.frame(y_0)

fit_m <- stan_glm(y_0 ~ 1, data = fake_0, seed=2141,
                  prior_intercept = NULL, prior = NULL, prior_aux = NULL)
print(fit_m, digits=2)
summary(fit_m, digits=2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y_0 ~ 1
 observations: 200
 predictors:   1
------
            Median MAD_SD
(Intercept) 2.43   0.37

Auxiliary parameter(s):
      Median MAD_SD
sigma 5.12   0.27
```

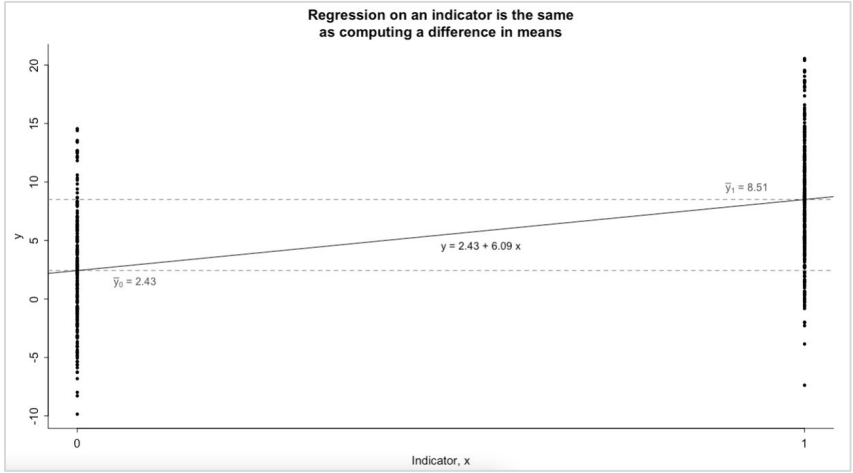
Comparing Two Means

Linear regression *on an indicator variable* is equivalent to a comparison of means

Here is an example: simulate 200 (n_0) observations (y_0) from a population with mean = **2.0** and standard deviation = **5.0**. Next, simulate 300 (n_1) observations (y_1) from a population with mean = **8.0** and standard deviation = **5.0**.

Considering these as random samples from two populations, we can estimate the difference in means with the corresponding standard error. The result: an estimate of **6.1** for the difference with standard error **0.46**.

Alternatively we can set this up as linear regression:




```
# Comparing two means
n_1 <- 300
set.seed(2141)
y_1 <- rnorm(n_1, 8.0, 5.0)

diff <- mean(y_1) - mean(y_0)
se_0 <- sd(y_0)/sqrt(n_0)
se_1 <- sd(y_1)/sqrt(n_1)
se <- sqrt(se_0^2 + se_1^2)

# difference in means
print(round(diff, 2))

# standard error of difference in means
print(round(se, 2))
```

```
> # difference in means
> print(round(diff, 2))
[1] 6.08
> # standard error of difference in means
> print(round(se, 2))
[1] 0.46
```

```
# Difference in means as a regression on an indicator variable
n <- n_0 + n_1
y <- c(y_0, y_1)
x <- c(rep(0, n_0), rep(1, n_1))
fake <- data.frame(y, x)
fit_dim <- stan_glm(y ~ x, data = fake, seed=2141,
                    prior_intercept = NULL, prior = NULL, prior_aux = NULL)
print(fit_dim, digits=2)
summary(fit_dim, digits=2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 500
 predictors:   2
------
            Median MAD_SD
(Intercept) 2.43   0.37
x           6.09   0.48
```

Course logistics

Instructor: Clinton Brownley

I’m a data scientist at Meta (Facebook), where I'm responsible for a variety of analytics projects designed to empower employees to do their best work. I also teach a graduate course in interactive data visualization at UC Berkeley.

I've written two books, "Foundations for Analytics with Python" and "Multi-objective Decision Analysis", and I'm a past-president of the San Francisco Bay Area Chapter of the American Statistical Association.

Please feel free to call me Clinton or Professor Brownley.

This is the first time I’m teaching this course, so your feedback is very useful to me!

TA: Martha Moreno

I’m looking forward to helping students with the course. Please feel free to contact me if you have any questions!

TA: Lina Cook

I’m looking forward to helping students with the course. Please feel free to contact me if you have any questions!

Course goals

After completing this course, you should be able to:

- Understand and explain the goals and assumptions of linear regression, including when regression is appropriate for a given situation.
- Estimate regression models from data using R, correctly interpret fitted model parameters, and correctly compare models.
- Clearly communicate your findings, including measures of uncertainty.

Nuts and bolts

Class format: 150 minutes. Complete assigned readings before lecture.

Office hours: Wednesdays (12-1pm PT | 3-4pm ET) or email me to schedule

TAs: Martha Moreno (mm8698@nyu.edu), Lina Cook (lac9209@nyu.edu)

Prerequisites: APSTA-GE 2003 or equivalent.
[Everything in R. No Stata/SPSS/Python/etc.]

Textbooks:
Required - Regression and Other Stories [RAOS]
Helpful - Statistical Rethinking, 2nd Ed [SR]

Other important things

1. Please don’t hesitate to reach out to me or your TA for help with anything.
   [we may not be able to respond immediately]

2. Please participate as much as possible!

3. Use class forum for questions.

Course components

Participation: **10%**

Individual weekly assignments: **90%**

Grading rubric can be found on syllabus.

Course components: participation

You can earn participation points by:

- Asking or answering questions in class or website forum
- Completing in-class group work
- Attending lecture, lab, and office hours
- Attending seminars related to class material

Keep track of your own participation. At the end of the semester, I will ask you to let me know how you participated.

Course components: participation - COVID policy

If you experience symptoms of, test positive for, or know you have come in contact with anyone who tested positive for COVID-19, and have not reported this information via the Daily Screener, submit the COVID-19 Reporting Form immediately. Questions or concerns can be directed to studentlink@nyu.edu.

*It is NYU's policy, for both medical and privacy reasons, that any communication regarding COVID-19 cases come only and directly from NYU's COVID-19 Prevention & Response Team (CPRT). Faculty or schools that become aware of a case are **not** permitted to notify their classes, schools, or anyone else about their knowledge of a case; contact tracing and any necessary notifications are done solely by the CPRT.*

Course components: weekly assignments

You will turn in 6 individual weekly assignments.

Late assignments received:

- **< 24 hours** after due date lose 5 percentage points
- **< 48 hours** after due date lose 10 percentage points
- **> 48 hours** after due date will receive no credit.

You can discuss assignments with each other, but *you must turn in your own work*.

Course outline [subject to modification]

Week 1 (1/25): Introduction: review of linear regression

Week 2 (2/1): Linear regression + global F-tests [HW 1 due]

Week 3 (2/8): Sampling distributions + CLT [HW 2 due]

Week 4 (2/15): ANOVA + linear regression [HW 3 due]

Week 5 (2/22): Info Criteria + model evaluation [HW 4 due]

Week 6 (3/1): Contingency tables + linear regression [HW 5 due]

Week 7 (3/8): Additional topics [HW 6 due]

Recap

- Regression is a statistical technique that summarizes how average values of an **outcome variable** vary across units defined by a **predictor variable**.

- Homework is a core component of this course

- HW 1 is on Brightspace [Due by 2/1/22]

- Download and install R and download and install RStudio before next week's lecture if you haven't done so already.

Appendix

Resources

Regression and Other Stories

Statistical Rethinking

Statistical rethinking with brms, ggplot2, and the tidyverse: Second edition

Bayes Rules!

Tidy Modeling with R

Doing Bayesian Data Analysis, Second edition

Doing Bayesian Data Analysis in brms and the tidyverse

rstanarm vignettes

bayesplot vignettes

R for Data Science

R Graphics Cookbook

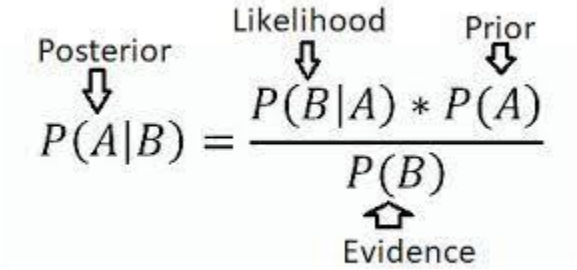
Bayesian Linear Regression

Linear regression: Model components

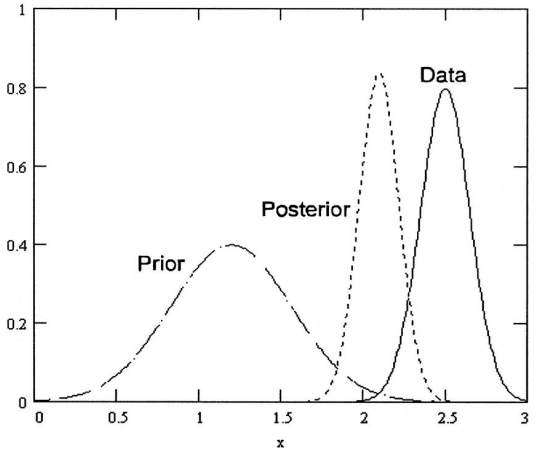
- Variables
  - Observed
  - Unobserved
- Prior
- Probability of the data (likelihood)
- Average probability of the data
- Posterior

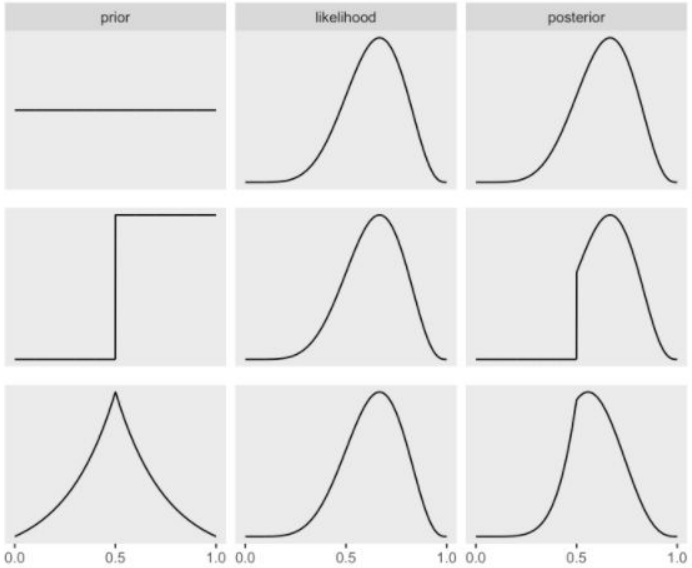
Linear regression: Bayesian updating

$$\text{Posterior} = \frac{\text{Probability of the data} * \text{Prior}}{\text{Average probability of the data}}$$



Linear regression: Bayesian updating





Linear regression

Let's rewrite our linear model (kid_score = a + b*mom_iq_centered + error) using our model components:

$$y_i \sim \text{Normal}(\mu_i, \sigma) \quad [\text{ likelihood }]$$

$$\mu_i = a + b(x_i - \text{mean}(x)) \quad [\text{ linear model }]$$

$$a \sim \text{Normal}(87, 20) \quad [\text{ a prior }]$$

$$b \sim \text{Normal}(0, 10) \quad [\text{ b prior }]$$

$$\sigma \sim \text{Exponential}(1) \quad [\text{ } \sigma \text{ prior }]$$

Linear regression

Now let’s see how to specify this model in R:

Package: rethinking

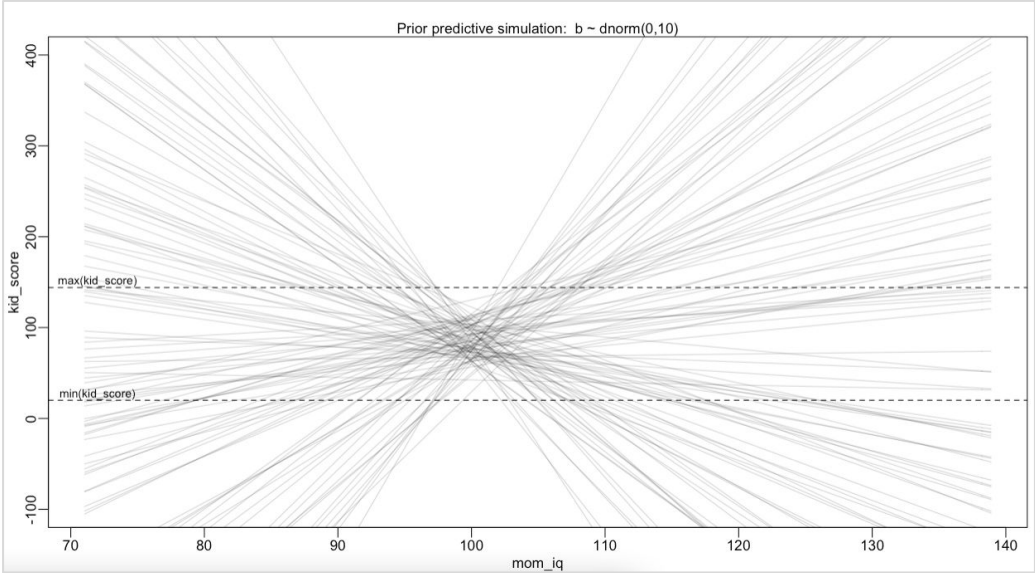
```
fit_2a <- quap(
    alist(
        kid_score ~ dnorm( mu , sigma ) ,
        mu <- a + b*( mom_iq - mean_mom_iq ) ,
        a ~ dnorm( 87 , 20 ) ,
        b ~ dnorm( 0 , 10 ) ,
        sigma ~ dexp( 1 )
    ) , data = kidiq)

precis(fit_2a, prob = 0.89, digits = 2)
```

Package: rstanarm

```
fit_2b <- stan_glm(kid_score ~ mom_iq_minus_mean_iq,
                   data=kidiq, refresh = 0)
print(fit_2b)
```

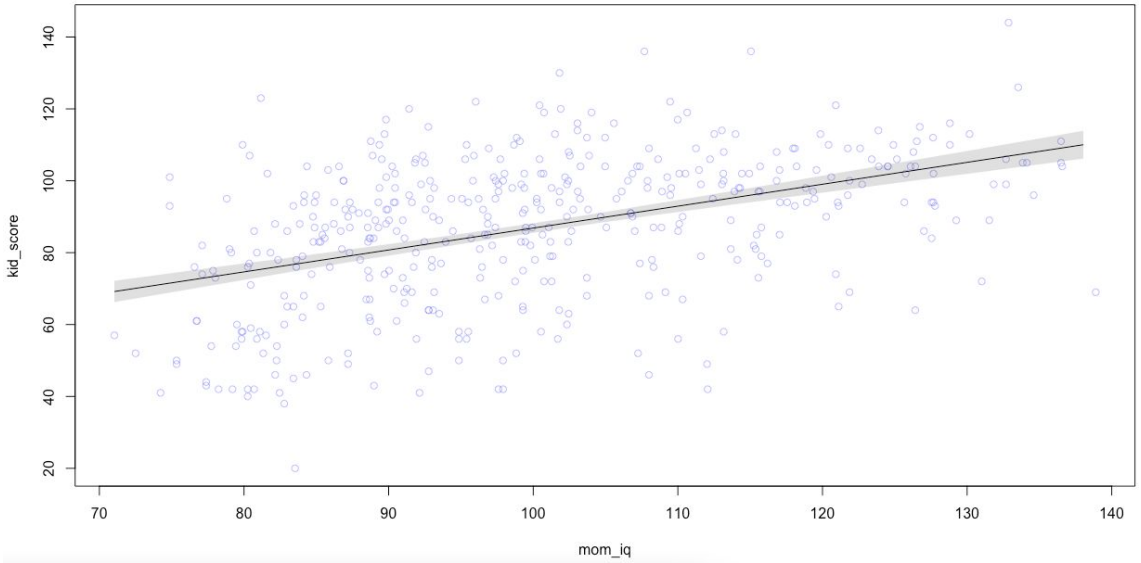
Linear regression: Prior predictive simulation

A Gaussian prior centered on zero places as much probability below zero as above zero, and when b = 0, mom_iq has no relationship to kid_score. This prior is too flexible, but in this case the data will overwhelm it. We'll learn to do better:



Linear regression: Plotting posterior mean and interval

Now that we've computed the joint posterior of the three parameters, let's plot the kidiq data, the regression line, and the 89% compatibility interval of the mean (shaded region):

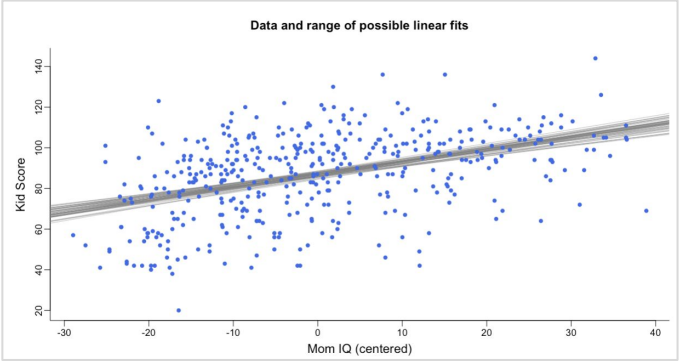


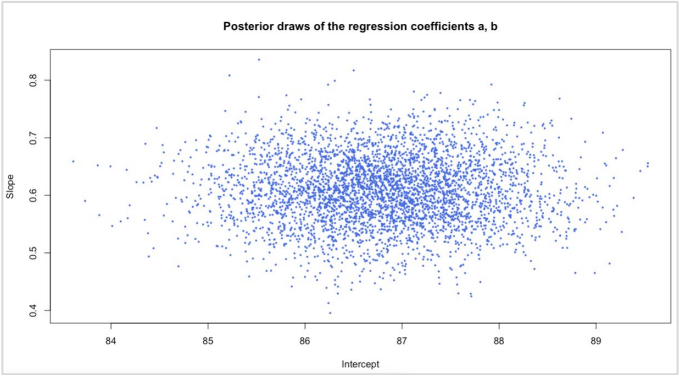
Example

Linear regression: Bayesian

Now let's see how to specify this model as a Bayesian regression in R:

```
# Fit a Bayesian linear regression
fit_bayes <- stan_glm(kid_score ~ mom_iq_centered, data=kidiq)
summary(fit_bayes)
plot(fit_bayes)
```





Interpreting Bayesian output: Model Info

Function / Family / Formula

Generalized linear model with optional prior distributions for the coefficients, intercept, and auxiliary parameters.

Algorithm

- "sampling" for MCMC (the default)
- "optimizing" for optimization
- "meanfield" or "fullrank" for variational inference

Sample

2000 iterations ; 4 chains ; 3 parameters (a, b, sigma)

Priors

e.g.

Family	Functions
Student t family	`normal`, `student_t`, `cauchy`
Hierarchical shrinkage family	`hs`, `hs_plus`
Laplace family	`laplace`, `lasso`
Product normal family	`product_normal`

Observations / Predictors

Double-check the number of observations and predictors

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: MCMC Diagnostics

Monte Carlo standard error (mcse): the uncertainty about a statistic in the sample due to sampling error

Potential Scale Reduction (Rhat):
In equilibrium, the distribution of samples from chains should be the same regardless of the initial starting values of the chains. One way to check this is to compare the distributions of multiple chains — in equilibrium they should all have the same mean.

Rule of Thumb: the R-hat values for all parameters are less than 1.1

Effective sample size (ESS, n_eff):
measures the amount by which autocorrelation in samples increases uncertainty (standard errors) relative to an independent sample

Rule of Thumb: the effective sample sizes (n_eff) for all parameters are greater than 2000

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Coefficients

The fitted model is:

kid_score = 87 + 0.6 * mom_iq_centered + error

Interpreting points on fitted line:
Either as predicted test scores for kids at each of several maternal IQ levels, or as average test scores for subpopulations defined by these scores

Slope:
If we compare average kid scores for subpopulations that differ in maternal IQ by 1 point, we expect to see that the group with higher maternal IQ achieves 0.6 points more on average
(if IQs differ by 10 points, scores differ by 6 points on average)

Intercept:
Since we centered mom IQ, the intercept reflects the predicted test scores of kids whose mothers have average IQ (100)

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Coefficient Estimates

The fitted model is:

kid_score = 87 + 0.6 * mom_iq_centered + error

```
# Extract posterior draws from the fitted model object
posterior <- as.array(fit_bayes)
dim(posterior)
dimnames(posterior)
head(posterior)

# Means and Standard Deviations              # Medians and Median Absolute Deviations (MAD)

# Intercept
mean(posterior[ , , "(Intercept)"])          median(posterior[ , , "(Intercept)"])
sd(posterior[ , , "(Intercept)"])            mad(posterior[ , , "(Intercept)"])

# Slope
mean(posterior[ , , "mom_iq_centered"])      median(posterior[ , , "mom_iq_centered"])
sd(posterior[ , , "mom_iq_centered"])        mad(posterior[ , , "mom_iq_centered"])

# Sigma
mean(posterior[ , , "sigma"])                median(posterior[ , , "sigma"])
sd(posterior[ , , "sigma"])                  mad(posterior[ , , "sigma"])
```

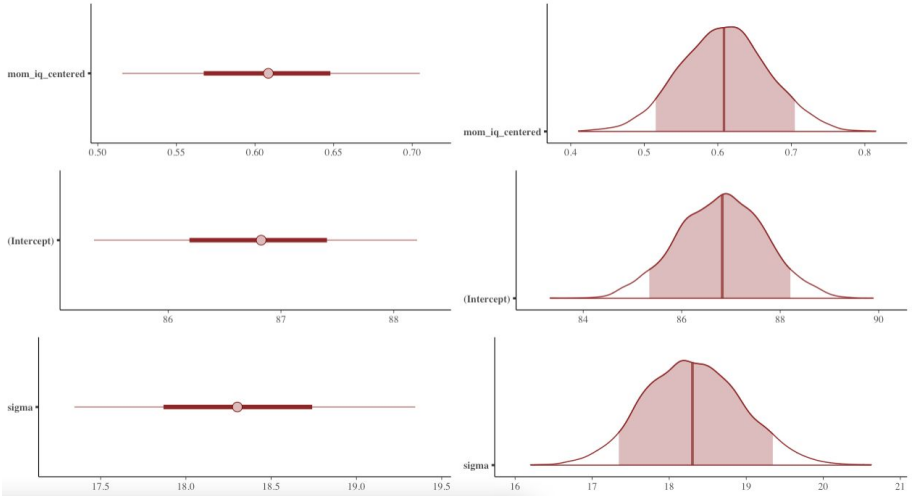
```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Coefficient Distribution Plots

The fitted model is:

kid_score = 87 + 0.6 * mom_iq_centered + error

Central posterior uncertainty intervals (median ± 0.5 and 0.89)




```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
          mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```

Interpreting Bayesian output: Fit Diagnostics

The fitted model is:

kid_score = 87 + 0.6 * mom_iq_centered + error

mean_PPD: the sample average posterior predictive distribution of the outcome. This is useful as a quick diagnostic.

A useful heuristic is to check if mean_PPD is plausible when compared to mean(y):

```
> # check mean_PPD against mean(kid_score)
> mean(kidiq$kid_score)
[1] 86.79724
```

- If it is plausible, then this does not mean that the model is good in general (only that it can reproduce the sample mean)
- If it is implausible, then it is a sign that something is wrong (severe model misspecification, problems with the data, computational issues, etc.)

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 434
 predictors:   2

Estimates:
                mean   sd   10%   50%   90%
(Intercept)     86.8    0.9 85.7  86.8  87.9
mom_iq_centered  0.6    0.1  0.5   0.6   0.7
sigma           18.3    0.6 17.5  18.3  19.1

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 86.8    1.2 85.2  86.8  88.4

The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for details see help('summary.stanreg')).

MCMC diagnostics
                mcse Rhat n_eff
(Intercept)     0.0  1.0  4247
mom_iq_centered 0.0  1.0  3554
sigma           0.0  1.0  3985
mean_PPD        0.0  1.0  4208
log-posterior   0.0  1.0  1880

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample size, and Rhat is the potential scale reduction factor on
 split chains (at convergence Rhat=1).
```