Topics in Multivariate Analysis

APSTA GE-2004

Lecture 4 - ANOVA and Regression

2/15/2022

Outline

Welcome! Today, we'll cover the following:

- ANOVA
- Framing ANOVA as regression

Also included:

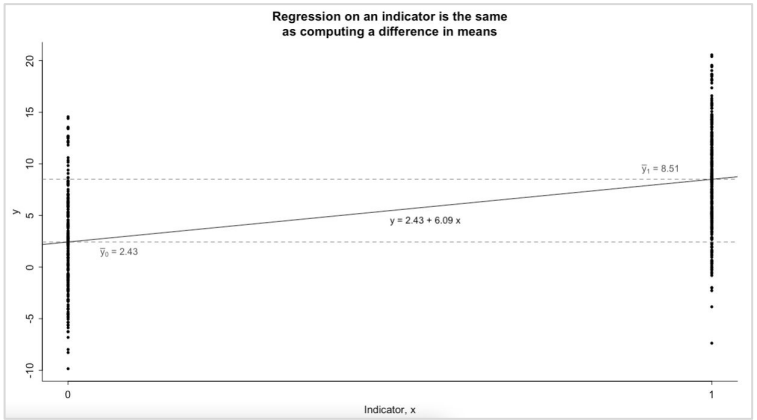
- Curves and Interactions

Reading:
RAOS Ch 7.3; Ch 9.1-9.2; Ch 10.1-10.5; Ch 11.2
Estimating ANOVA models with rstanarm by Gabry and Goodrich
RAOS Appendix A and B

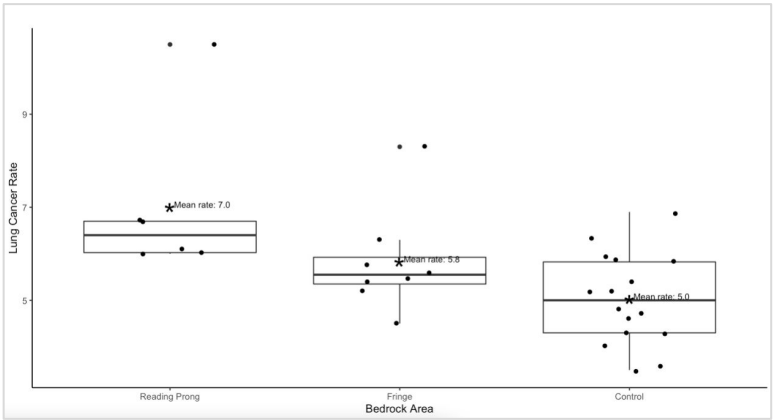
ANOVA

How do we analyze whether ***several*** population means differ?

We know how to analyze whether ***two*** population means differ…



But how do we analyze whether ***several*** population means differ?



One-way ANOVA

To assess whether several populations all have the same mean, we compare the means of samples drawn from each population

We do not expect sample means to be equal even if the population means are all identical. The question is, *could the observed difference among the groups be due to chance variation*?

The purpose of ANOVA is to statistically assess whether the observed differences among sample means could plausibly be due to chance or are evidence for differences among the population means

The question can't be answered from the sample means alone

Because the standard deviation of a sample mean is the population standard deviation σ divided by √n, the answer depends upon both the ***variation within the groups*** and the ***sizes of the groups***

locale	mean_cancer	sd_cancer	n
<chr>	<dbl>	<dbl>	<int>
Reading Prong	7.00	1.75	6
Fringe	5.83	1.12	8
Control	5.03	0.975	16

Quick sidebar: two-sample t-statistic

Two-sample t-statistics compare the means of two populations. If the two populations are assumed to have equal, unknown standard deviations and the sample sizes are both *n*, the t-statistic is:

$t = g1\_bar - g2\_bar / s_p * \sqrt{1/n + 1/n} = \sqrt{n/2} * (g1\_bar - g2\_bar) / s_p$

The square of this t statistic is:

$$\mathbf{t^2 = n/2 * (g1\_bar - g2\_bar)^2 / s_p^2}$$

If we use ANOVA to compare two populations, the ANOVA F-statistic is equal to this t^2.

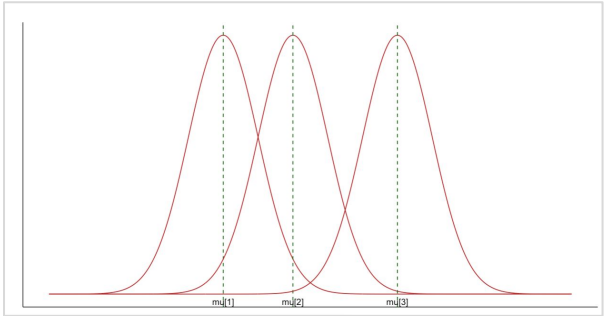
We can learn how ANOVA works by reviewing the statistic in this form:

Numerator:
- Measures the variation ***between*** the groups in terms of difference in sample means
- Includes a factor for the common sample size *n*
- Can be large because of (a) large difference in means or (b) large sample sizes

Denominator:
- Measures the variation ***within*** groups by s_p^2, the pooled estimator of the common variance
- If the within-group variation is small, the same variation between groups produces a larger, more significant statistic

Therefore, to assess whether several populations all have the same mean, we compare the variation ***among*** the means of several groups with the variation ***within*** groups. Because we are comparing variation, the method is called ***analysis of variance*** **(ANOVA)**

ANOVA estimates of population parameters



The unknown parameters in the statistical model for ANOVA are:

1. the population means $\boldsymbol{\mu_i}$
2. the common population standard deviation $\boldsymbol{\sigma}$

1. Means: To estimate μ_i we use the sample mean for the *i*th group:

$$\boldsymbol{\mu_i = 1/n_i \sum x_{ij}}$$

The ANOVA model assumes the population standard deviations are all equal:

2. Pooled Estimator of σ: Suppose we have sample variances s_1^2, s_2^2, ..., s_I^2 from I independent samples of sizes n_1, n_2, ..., n_I from populations with common variance σ^2. The pooled sample variance is:

$$s_p^2 = (n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \ldots + (n_I - 1)s_I^2 \;/\; (n_1 - 1) + (n_2 - 1) + \ldots + (n_I - 1)$$

The pooled sample standard deviation is:

$$\boldsymbol{s_p = \sqrt{s_p^2}}$$

ANOVA estimates of population parameters

The unknown parameters in the statistical model for ANOVA are:

1. the population means μ_i
2. the common population standard deviation σ

1. Means:

locale	mean_cancer	sd_cancer	n
<chr>	<dbl>	<dbl>	<int>
Reading Prong	7.00	1.75	6
Fringe	5.83	1.12	8
Control	5.03	0.975	16

2. Pooled Estimator of σ:

The pooled sample variance is:

$s_p^2 = [(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \ldots + (n_I - 1)s_I^2] / [(n_1 - 1) + (n_2 - 1) + \ldots + (n_I - 1)]$

$= [(6-1)1.75^2 + (8-1)1.12^2 + (16-1)0.975^2] / [(6-1) + (8-1) + (16-1)]$

$= [5*3.06 + 7*1.25 + 15*0.95] / [5 + 7 + 15]$

$= 38.3 / 27 = 1.42$

The pooled sample standard deviation is:

$$\mathbf{s_p = \sqrt{1.42} = 1.19}$$

ANOVA hypotheses

ANOVA tests the null hypothesis that the population means are *all* equal. The alternative is that they are not all equal:

H_0: $\mu_1 = \mu_2 = \ldots = \mu_I$

H_a: not all of the μ_i are equal

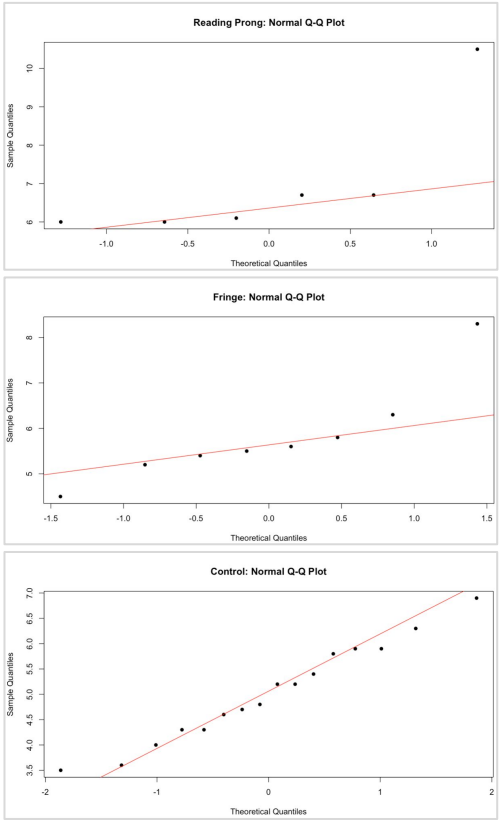
This alternative could be true because all of the means are different, or simply because one of them differs from the rest

If we reject the null hypothesis, we need to perform some further analysis to draw conclusions about which population means differ from which others

Check normality assumption

Before proceeding with ANOVA, review normal quantile plots for the groups to assess whether the assumption that the residuals are normal appears reasonable

If they do not appear normal, you may try transforming the variables, e.g. with $\sqrt{x_{ij}}$ or $\log(x_{ij})$, to make the distributions in each group more nearly normal



The ANOVA table

Sources of ***variation***:

- Groups (“***among*** *group means*”)
- Residuals (“***within*** *groups*”)
- Total

SSG: Variation of the group means around the overall mean
SSR: Variation of each observation around its group mean
SST: Variation of the data around the overall mean

DFG: k group means around overall mean, k – 1
DFR: All N observations around k group means, N – k
DFT: All N observations around overall mean, N – 1

MSG: SSG / DFG
MSR: SSR / DFR

F: MSG / MSR (This ratio is a statistic that is approx. 1 if H_O is true and tends to be larger if H_a is true)

N: Number of observations
k: Number of groups

SST = SSG + SSR

$$\sum_{obs} n_i (x_{ij} - x\_bar)^2 = \sum_{groups} n_i * (x\_bar_i - x\_bar)^2 + \sum_{groups} (n_i - 1) * (x_{ij} - x\_bar_i)^2$$

Source	Degrees of freedom	Sum of squares	Mean square	F
Groups	k – 1	$\sum_{groups} n_i * (x\_bar_i - x\_bar)^2$	MSG = SSG / DFG	MSG / MSR
Residuals	N – k	$\sum_{groups} (n_i - 1) * (x_{ij} - x\_bar_i)^2$	MSR = SSR / DFR	
Total	N – 1	$\sum_{obs} n_i (x_{ij} - x\_bar)^2$	SST / DFT	

The ANOVA F Test

To test the null hypothesis in a one-way ANOVA, calculate the **F statistic**:

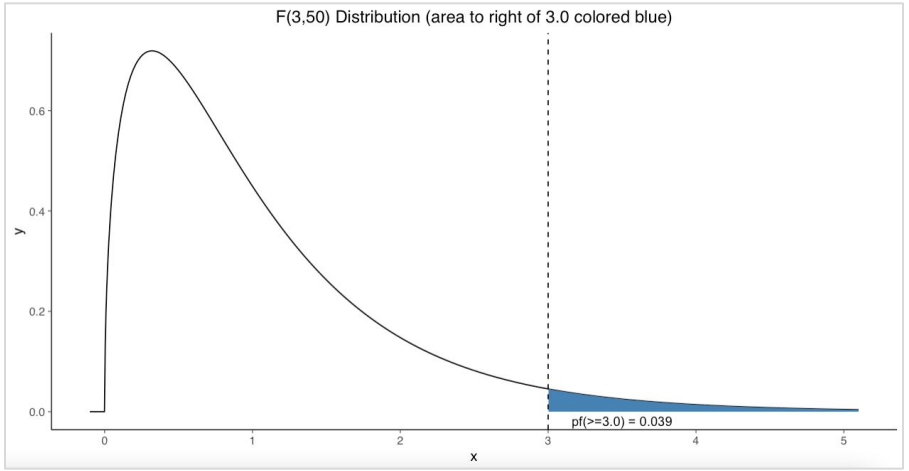
F = MSG / MSR

When H_O is true, the F statistic has the F(k–1, N–k) distribution

When H_a is true, the F statistic tends to be large. We reject H_O in favor of H_a if the F statistic is sufficiently large

The *p*-value of the F test is the probability that a random variable having the F(k–1, N–k) distribution is *greater than or equal to* the calculated value of the F statistic

Example: F distribution and *p*-value of F statistic colored blue



Do the population means for **Reading Prong**, **Fringe**, and **Control** differ?

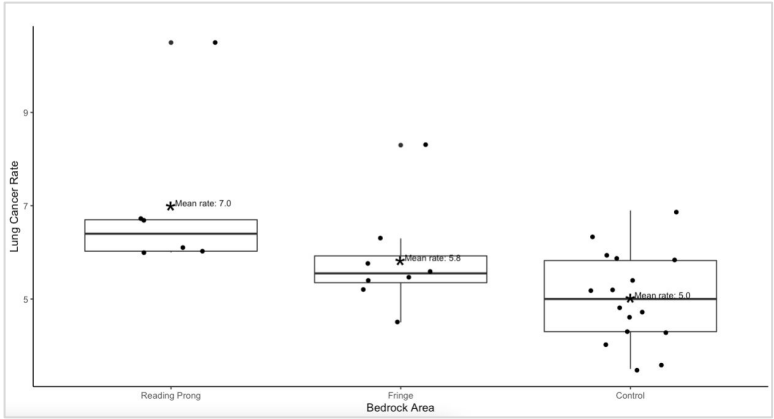
ANOVA tests the null hypothesis that the population means are *all* equal. The alternative is that they are not all equal:

H_0: $\mu_1 = \mu_2 = \ldots = \mu_I$

H_a: not all of the μ_i are equal

This alternative could be true because all of the means are different, or simply because one of them differs from the rest

If we reject the null hypothesis, we need to perform some further analysis to draw conclusions about which population means differ from which others



locale	mean_cancer	sd_cancer	n
<chr>	<dbl>	<dbl>	<int>
Reading Prong	7.00	1.75	6
Fringe	5.83	1.12	8
Control	5.03	0.975	16

ANOVA in R

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

```
# ANOVA
# Step 1: Create an aov object
res_aov <- aov(cancer ~ 1 + factor(locale), data = radon)

# Step 2: Review the summary of the aov object
coef(res_aov)

summary(res_aov)

print(res_aov)
res <- print(res_aov)
names(res)
```

```
> summary(res_aov)
               Df Sum Sq Mean Sq F value Pr(>F)
factor(locale)  2   17.4    8.71    6.13 0.0064
Residuals      27   38.3    1.42
>
>
> print(res_aov)
Call:
   aov(formula = cancer ~ 1 + factor(locale), data = radon)

Terms:
                factor(locale) Residuals
Sum of Squares            17.4      38.3
Deg. of Freedom              2        27

Residual standard error: 1.19
Estimated effects may be unbalanced
```

```
> # Residual standard error
> N <- dim(radon)[1] # number of observations (30)
> k <- 3             # number of predictors (3)
> sqrt( sum( (radon$cancer - res$fitted.values)^2 ) / (N - k) )
[1] 1.19
```

The ANOVA table for this example

SSG:
```
res <- print(res_aov)
cancer_mean <- mean(radon$cancer, na.rm = TRUE)
SSG <- sum( (res$fitted.values - cancer_mean)^2 )
```

SSR:
```
cancer_reading_prong <- radon %>% filter(locale == 1) %>% pull(cancer)
cancer_fringe        <- radon %>% filter(locale == 2) %>% pull(cancer)
cancer_control       <- radon %>% filter(locale == 3) %>% pull(cancer)

cancer_reading_prong_mean <- radon %>% filter(locale == 1) %>% pull(cancer) %>% mean(.)
cancer_fringe_mean        <- radon %>% filter(locale == 2) %>% pull(cancer) %>% mean(.)
cancer_control_mean       <- radon %>% filter(locale == 3) %>% pull(cancer) %>% mean(.)

SSR <- sum( (cancer_reading_prong - cancer_reading_prong_mean)^2 ,
            (cancer_fringe - cancer_fringe_mean)^2 ,
            (cancer_control - cancer_control_mean)^2 )
```

SST:
```
SST <- sum( (radon$cancer - cancer_mean)^2 )
SST
```

DFG: k – 1 = 3 – 1 = 2
DFR: N – k = 30 – 3 = 27

MSG:
```
MSG <- SSG / (k - 1)
MSG
```

MSR:
```
MSR <- SSR / (N - k)
MSR
```

F:
```
F_statistic <- MSG / MSR
F_statistic
```

N: 30
k: 3

```
> summary(res_aov)
                Df Sum Sq Mean Sq F value Pr(>F)
factor(locale)  2   17.4    8.71    6.13 0.0064
Residuals      27   38.3    1.42
```

Source	Degrees of freedom	Sum of squares	Mean square	F

Groups	k – 1 = 3 – 1 = **2**	$\sum_{groups}(x\_bar_i - x\_bar)^2$
		`> SSG <- sum( (res$fitted.values - cancer_mean)^2 )`
		`> SSG`
		`[1] 17.4`
			MSG = SSG / DFG
			`> MSG <- SSG / (k - 1)`
			`> MSG`
			`[1] 8.71`
				MSG / MSR
				`> F_statistic <- MSG / MSR`
				`> F_statistic`
				`[1] 6.13`

Residuals	N – k = 30 – 3 = **27**	$\sum_{groups}(x_{ij} - x\_bar_i)^2$
		`> SSR <- sum( (cancer_reading_prong - cancer_reading_prong_mean)^2 ,`
		`+ (cancer_fringe - cancer_fringe_mean)^2 ,`
		`+ (cancer_control - cancer_control_mean)^2 )`
		`> SSR`
		`[1] 38.3`
			MSR = SSR / DFR
			`> MSR <- SSR / (N - k)`
			`> MSR`
			`[1] 1.42`	

Critical values for this F(2,27) distribution

Result:

F statistic: **6.13** (*p*-value: **0.0064**)

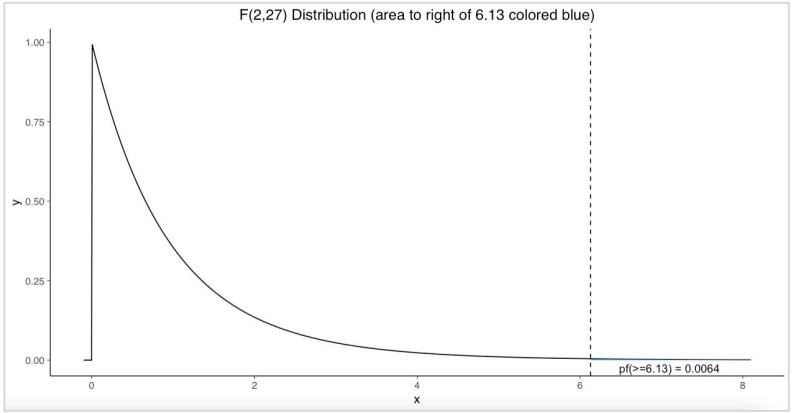
Interpretation:

Because the F statistic, 6.13, is larger than the critical value for the 1% critical value, 5.49, we reject H_O at the 1% level and conclude that the differences in means are statistically significant with P < 0.01

Alternatively, because the p-value, 0.0064, is smaller than 0.01, we reject H_O at the 1% level and conclude that the differences in means are statistically significant with P < 0.01

Critical values:

```
> # Critical values for this F distribution
> ( crit10  <- qf( 0.90  , df1 = (k-1) , df2 = (n-k) ) )
[1] 2.51
> ( crit05  <- qf( 0.95  , df1 = (k-1) , df2 = (n-k) ) )
[1] 3.35
> ( crit025 <- qf( 0.975 , df1 = (k-1) , df2 = (n-k) ) )
[1] 4.24
> ( crit01  <- qf( 0.99  , df1 = (k-1) , df2 = (n-k) ) )
[1] 5.49
> ( crit001 <- qf( 0.999 , df1 = (k-1) , df2 = (n-k) ) )
[1] 9.02
```



So, the means are different...but which ones, and by how much?

We have determined that the means are significantly different, but which ones, and by how much?

Issue:

When comparing the means for the levels of a factor in an ANOVA, a simple comparison using t-tests will inflate the probability of declaring a significant difference when it is not in fact present. This because the intervals are calculated with a given coverage probability for each interval but the interpretation of the coverage is usually with respect to the entire family of intervals.

A solution:

It's time for *a procedure*[*] called ***Tukey honest significant differences***, which allows a comparison of all pairs of levels of a factor whilst maintaining the nominal significance level at its selected value and producing adjusted confidence intervals for mean differences

* Others include Scheffe's and Bonferroni methods

Tukey Honest Significant Differences

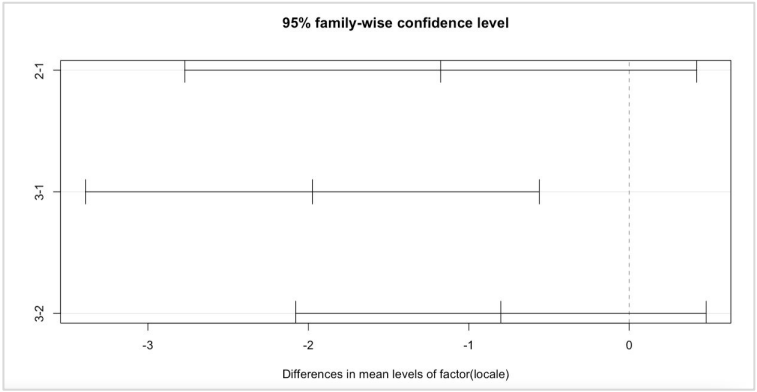
Now let's see how to specify the Tukey HSD method in R:

```
# Compute Tukey Honest Significant Differences (TukeyHSD)
TukeyHSD(res_aov)

# Optional arguments
TukeyHSD(res_aov, ordered = TRUE, conf.level = 0.89)

# Plot the results
plot( TukeyHSD(res_aov) )
```

```
> # Compute Tukey Honest Significant Differences (TukeyHSD)
> TukeyHSD(res_aov)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = cancer ~ 1 + factor(locale), data = radon)

$`factor(locale)`
     diff   lwr    upr p adj
2-1 -1.17 -2.77  0.421 0.180
3-1 -1.97 -3.39 -0.561 0.005
3-2 -0.80 -2.08  0.479 0.284
```

Results:

The simultaneous pairwise comparisons only provide evidence for a significant difference between the means of Reading Prong (1) and Control (3). The other confidence intervals include 0.



ANOVA regression (classical)

One-way ANOVA (ANOVA with one K-category X variable) is equivalent to regression on K−1 indicator variables

Recasting ANOVA as classical regression in R

Now let's see how to recast this ANOVA model as a linear regression model in R:

```r
# Formulate as a linear regression
cancer_ols <- lm(cancer ~ 1 + reading + fringe, data = radon)
options(digits = 3,
        show.signif.stars = FALSE,
        show.coef.Pvalues = FALSE)
summary(cancer_ols)
```

```r
# Formulate as a linear regression
# Add locale as a factor with reordered levels so Control is comparison category
radon$locale <- factor(radon$locale, levels = c(3, 1, 2))
cancer_ols <- lm(cancer ~ 1 + locale, data = radon)
options(digits = 2,
        show.signif.stars = FALSE,
        show.coef.Pvalues = FALSE)
summary(cancer_ols)
```

```
> radon
# A tibble: 30 × 11
   radon cancer number state locale reading fringe Ereading Efringe control county
   <dbl>  <dbl>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>    <dbl>   <dbl>   <dbl> <chr>
 1  1.33   6       179     1      1       1      0        1       0       0 Orange
 2  1.80  10.5     372     1      1       1      0        1       0       0 Putnam
 3  1.70   6.70   1935     2      1       1      0        1       0       0 Sussex
 4  2.61   6       481     2      1       1      0        1       0       0 Warren
 5  1.35   6.10   4493     2      1       1      0        1       0       0 Morris
 6  2.83   6.70   1384     2      1       1      0        1       0       0 Hunterdon
 7  4.30   5.20    281     3      2       0      1        0       1       0 Berks
 8  3.5    5.60    872     3      2       0      1        0       1       0 Lehigh
 9  3.71   5.80    651     3      2       0      1        0       1       0 Northampton
10 NA     8.30     NA     3      2       0      1        0       1       0 Monroe
# … with 20 more rows
```

```
Call:
lm(formula = cancer ~ 1 + reading + fringe, data = radon)

Residuals:
   Min     1Q Median     3Q    Max
-1.525 -0.725 -0.300  0.450  3.500

Coefficients:
            Estimate Std. Error t value
(Intercept)    5.025      0.298   16.87
reading        1.975      0.570    3.46
fringe         0.800      0.516    1.55

Residual standard error: 1.19 on 27 degrees of freedom
Multiple R-squared:  0.312,     Adjusted R-squared:  0.261
F-statistic: 6.13 on 2 and 27 DF,  p-value: 0.00637
```

Estimate significance of pairwise differences in means (aka contrasts) with `emmeans`

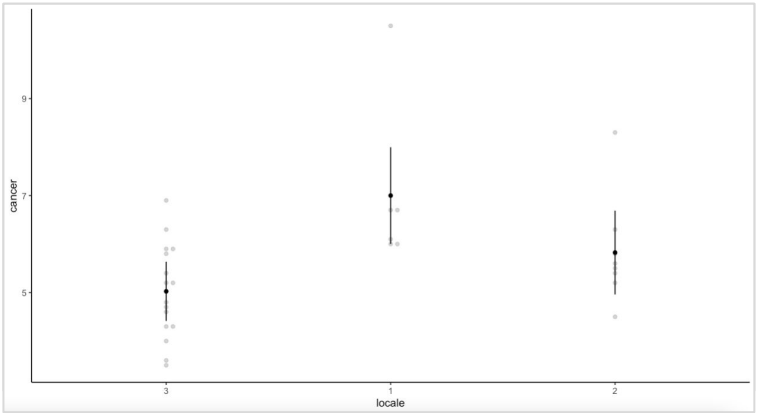
Now let's see how to estimate the contrasts between levels in R:

```
# Contrasts with emmeans
cancer_emm <- emmeans(cancer_ols, pairwise ~ locale)

coef(cancer_emm)

contrast(cancer_emm)

pairs(cancer_emm)
```

```
> emmeans(cancer_ols, pairwise ~ locale)
$emmeans
 locale emmean   SE df lower.CL upper.CL
 3         5.0 0.30 27      4.4      5.6
 1         7.0 0.49 27      6.0      8.0
 2         5.8 0.42 27      5.0      6.7

Confidence level used: 0.95

$contrasts
 contrast estimate   SE df t.ratio p.value
 3 - 1       -1.97 0.57 27  -3.500  0.0050
 3 - 2       -0.80 0.52 27  -1.600  0.2840
 1 - 2        1.17 0.64 27   1.800  0.1800
```

Results:

The simultaneous pairwise comparisons only provide evidence for a significant difference between the means of Reading Prong (1) and Control (3). The other confidence intervals include 0.



ANOVA regression (Bayesian)

One-way ANOVA (ANOVA with one K-category X variable) is equivalent to regression on K−1 indicator variables

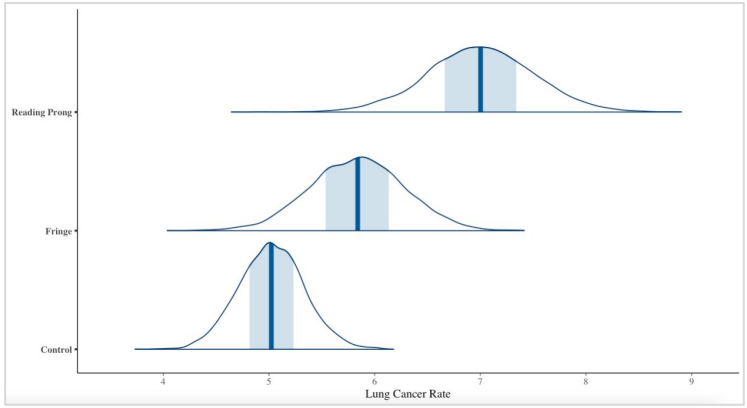
Recasting ANOVA as Bayesian regression in R

Now let's see how to recast this ANOVA model as a linear regression model in R:

```
# Formulate as a linear regression
cancer_bayes <- stan_glm(cancer ~ 1 + reading + fringe, data = radon,
                         prior_intercept=NULL, prior=NULL, prior_aux=NULL)
summary(cancer_bayes)

# Formulate as a linear regression
# Add locale as a factor with reordered levels so Control is comparison category
radon$locale <- factor(radon$locale, levels = c(3, 1, 2))
cancer_bayes <- stan_glm(cancer ~ 1 + locale, data = radon,
                         prior_intercept=NULL, prior=NULL, prior_aux=NULL)
summary(cancer_bayes)
```

```
> summary(cancer_bayes)

Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      cancer ~ 1 + reading + fringe
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 30
 predictors:   3

Estimates:
              mean   sd   10%   50%   90%
(Intercept) 5.0    0.3  4.6   5.0   5.4
reading     2.0    0.6  1.2   2.0   2.7
fringe      0.8    0.6  0.1   0.8   1.5
sigma       1.3    0.2  1.0   1.2   1.5
```

```
as.data.frame(cancer_bayes) %>%
  mutate(Control = `(Intercept)`,
         Fringe  = `(Intercept)` + fringe,
         `Reading Prong` = `(Intercept)` + reading) %>%
  mcmc_areas(pars = c("Reading Prong", "Fringe", "Control")) +
  labs(x="Lung Cancer Rate")
```

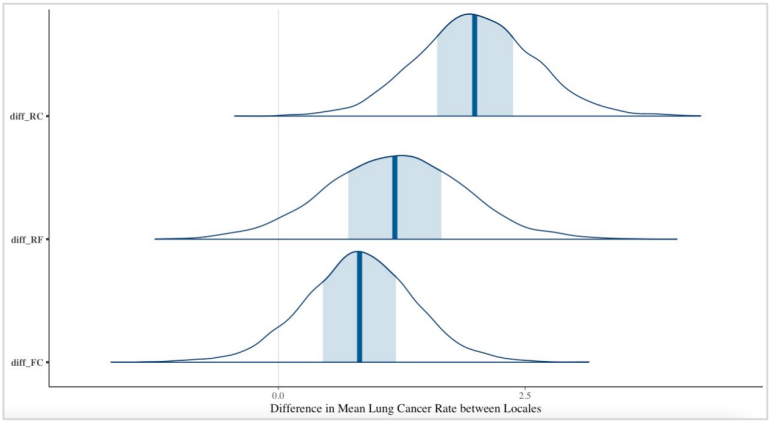


Estimate significance of pairwise differences in means (aka contrasts) with posterior distribution

Now let's see how to estimate the contrasts between levels in R:

```
# Plot the pairwise differences in mean cancer rate between locales
posterior <- as.matrix(cancer_bayes)
posterior[1:5, ]

diff_RC <- posterior[ , "reading"]
diff_RF <- posterior[ , "reading"] - posterior[ , "fringe"]
diff_FC <- posterior[ , "fringe"]
posterior <- cbind(posterior, diff_RC)
posterior <- cbind(posterior, diff_RF)
posterior <- cbind(posterior, diff_FC)
posterior[1:5, ]

as.data.frame(posterior) %>%
  dplyr::select(diff_RC, diff_RF, diff_FC) %>%
  mcmc_areas(pars = c("diff_RC", "diff_RF", "diff_FC")) +
  labs(x="Difference in Mean Lung Cancer Rate between Locales")
```

```
# A tibble: 3 x 5
  contrast  mean    sd `2.5%` `97.5%`
  <chr>     <dbl> <dbl>  <dbl>   <dbl>
1 RC         2.00 0.600  0.845    3.21
2 RF         1.18 0.700 -0.211    2.58
3 FC        0.818 0.571 -0.322    1.95
```



Results:

The simultaneous pairwise comparisons only provide evidence for a significant difference between the means of Reading Prong (1) and Control (3). The other confidence intervals include 0.

ANOVA regression (classical)

One-way ANOVA (ANOVA with one K-category X variable) is equivalent to regression on K−1 indicator variables

One-way ANOVA

The data for this example (in DBDA2E) arise from an experiment to study the lifespan of male fruit flies as a function of the their sexual activity

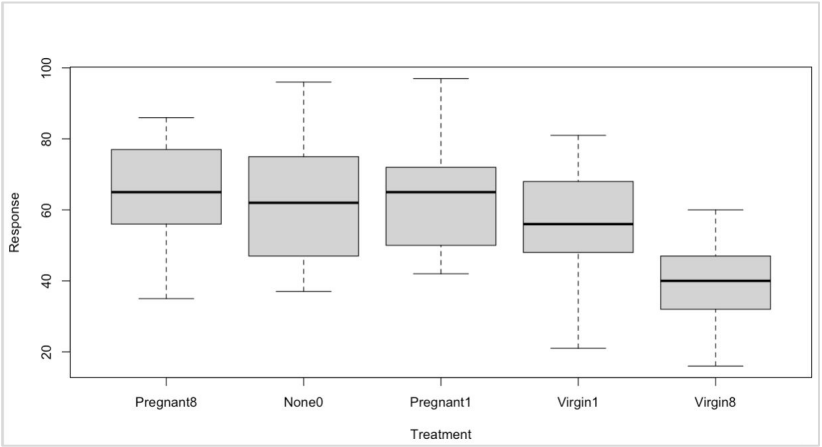
Twenty-five male fruit flies were randomised to each of five treatments:

- 8 virgins
- 1 virgin
- 8 pregnants
- 1 pregnant
- 0 females

and their lifespan was recorded

The question of interest is whether male sexual activity reduced lifespan, as this was already established for females

```
# A tibble: 5 × 4
  CompanionNumber  mean    sd     n
  <fct>           <dbl> <dbl> <int>
1 Pregnant8        63.4  14.5    25
2 None0            63.6  16.5    25
3 Pregnant1        64.8  15.7    25
4 Virgin1          56.8  14.9    25
5 Virgin8          38.7  12.1    25
```



Recasting ANOVA as classical regression in R

Now let's see how to recast this ANOVA model as a linear regression model in R:

```
# Formulate as a linear regression
fruit_fly_fit <- lm(Longevity ~ 0 + CompanionNumber, data = fruit_fly)

summary(fruit_fly_fit)

anova(fruit_fly_fit)
```

```
> fruit_fly
# A tibble: 125 × 3
   Longevity CompanionNumber
       <dbl> <fct>
 1        35 Pregnant8
 2        37 Pregnant8
 3        49 Pregnant8
 4        46 Pregnant8
 5        63 Pregnant8
 6        39 Pregnant8
 7        46 Pregnant8
 8        56 Pregnant8
 9        63 Pregnant8
10        65 Pregnant8
# … with 115 more rows
```

```
> summary(fruit_fly_fit)

Call:
lm(formula = Longevity ~ 0 + CompanionNumber, data = fruit_fly)

Residuals:
   Min     1Q Median     3Q    Max
-35.76  -8.76   0.20  11.20  32.44

Coefficients:
                          Estimate Std. Error t value
CompanionNumberPregnant8     63.36       2.96    21.4
CompanionNumberNone0         63.56       2.96    21.5
CompanionNumberPregnant1     64.80       2.96    21.9
CompanionNumberVirgin1       56.76       2.96    19.2
CompanionNumberVirgin8       38.72       2.96    13.1

Residual standard error: 15 on 120 degrees of freedom
Multiple R-squared:  0.942,     Adjusted R-squared:  0.939
F-statistic:  387 on 5 and 120 DF,  p-value: <2e-16
```

```
> anova(fruit_fly_fit)
Analysis of Variance Table

Response: Longevity
                 Df Sum Sq Mean Sq F value Pr(>F)
CompanionNumber   5 424358   84872     387 <2e-16
Residuals       120  26314     219
```

Differences among means is significant. Which ones, and by how much?

Estimate significance of pairwise differences in means (aka contrasts) with `emmeans`

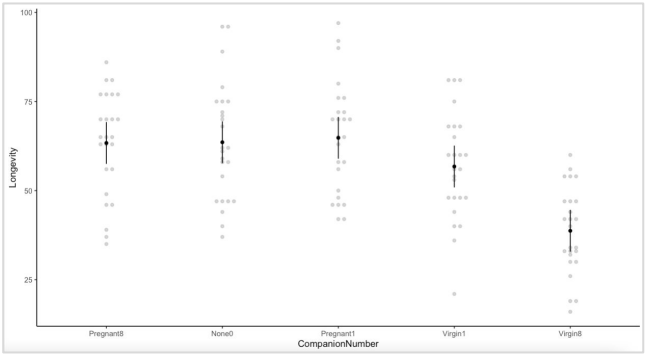
Now let's see how to estimate the contrasts between levels in R:

```
# Print contrast estimates and p-values
fruit_fly_emm <- emmeans(fruit_fly_fit, pairwise ~ CompanionNumber)

coef(fruit_fly_emm)

contrast(fruit_fly_emm)

pairs(fruit_fly_emm)
```

Results:

The simultaneous pairwise comparisons provide evidence for a significant difference between the means of P8 – V8, None – V8, P1 – V8, and V1 – V8. The other *p*-values are greater than 0.05.

```
> emmeans(fruit_fly_fit, pairwise ~ CompanionNumber)
$emmeans
 CompanionNumber emmean   SE  df lower.CL upper.CL
 Pregnant8           63 2.96 120       57       69
 None0               64 2.96 120       58       69
 Pregnant1           65 2.96 120       59       71
 Virgin1             57 2.96 120       51       63
 Virgin8             39 2.96 120       33       45

Confidence level used: 0.95

$contrasts
 contrast              estimate  SE  df t.ratio p.value
 Pregnant8 - None0         -0.2 4.2 120   0.000  1.0000
 Pregnant8 - Pregnant1     -1.4 4.2 120  -0.300  1.0000
 Pregnant8 - Virgin1        6.6 4.2 120   1.600  0.5200
 Pregnant8 - Virgin8       24.6 4.2 120   5.900  <.0001
 None0 - Pregnant1         -1.2 4.2 120  -0.300  1.0000
 None0 - Virgin1            6.8 4.2 120   1.600  0.4900
 None0 - Virgin8           24.8 4.2 120   5.900  <.0001
 Pregnant1 - Virgin1        8.0 4.2 120   1.900  0.3100
 Pregnant1 - Virgin8       26.1 4.2 120   6.200  <.0001
 Virgin1 - Virgin8         18.0 4.2 120   4.300  <.0001

P value adjustment: tukey method for comparing a family of 5 estimates
```



ANOVA regression (classical)

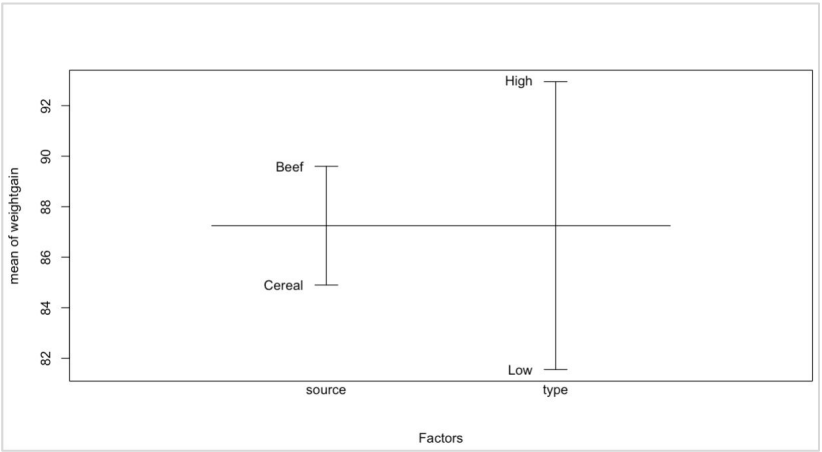
Two-way ANOVA

The data for this example (in HSAUR) arise from an experiment to study the gain in weight of rats fed on four different diets, distinguished by amount of protein (low and high) and by source of protein (beef and cereal)

Ten rats are randomised to each of the four treatments and the weight gain in grams recorded

The question of interest is how diet affects weight gain

```
# A tibble: 4 x 5
# Groups:   source [2]
  source type   mean    sd     n
  <fct>  <fct> <dbl> <dbl> <int>
1 Beef   High  100    15.1    10
2 Beef   Low    79.2  13.9    10
3 Cereal High   85.9  15.0    10
4 Cereal Low    83.9  15.7    10
```



Recasting ANOVA as classical regression in R

Now let's see how to recast this ANOVA model as a linear regression model in R:

```
# Formulate as a linear regression
weightgain <- weightgain %>%
  mutate(source = factor(source),
         type   = factor(type))

wg_fit <- lm(weightgain ~ 1 + source + type + source:type, data = weightgain)

summary(wg_fit)

anova(wg_fit)
```

```
> weightgain
   source type weightgain
1    Beef  Low         90
2    Beef  Low         76
3    Beef  Low         90
4    Beef  Low         64
5    Beef  Low         86
6    Beef  Low         51
7    Beef  Low         72
8    Beef  Low         90
9    Beef  Low         95
10   Beef  Low         78
11   Beef High         73
12   Beef High        102
13   Beef High        118
14   Beef High        104
15   Beef High         81
16   Beef High        107
17   Beef High        100
18   Beef High         87
19   Beef High        117
20   Beef High        111
21 Cereal  Low        107
22 Cereal  Low         95
```

```
> summary(wg_fit)

Call:
lm(formula = weightgain ~ 1 + source + type + source:type, data = weightgain)

Residuals:
   Min     1Q Median     3Q    Max
-29.90  -9.90   2.05  10.85  25.10

Coefficients:
                  Estimate Std. Error t value
(Intercept)         100.00       4.73   21.15
sourceCereal        -14.10       6.69   -2.11
typeLow             -20.80       6.69   -3.11
sourceCereal:typeLow  18.80       9.46    1.99

Residual standard error: 15 on 36 degrees of freedom
Multiple R-squared:  0.23,      Adjusted R-squared:  0.166
F-statistic: 3.58 on 3 and 36 DF,  p-value: 0.023
```

```
> anova(wg_fit)
Analysis of Variance Table

Response: weightgain
            Df Sum Sq Mean Sq F value Pr(>F)
source       1    221     221    0.99  0.327
type         1   1300    1300    5.81  0.021
source:type  1    884     884    3.95  0.054
Residuals   36   8049     224
```

The main effect of **type** is significant

The main effect of **source** is not significant

Interpretation of both these main effects is complicated by the **type ✕ source** interaction which approaches significance at the 5% level

Estimate significance of pairwise differences in means (aka contrasts) with `emmeans`

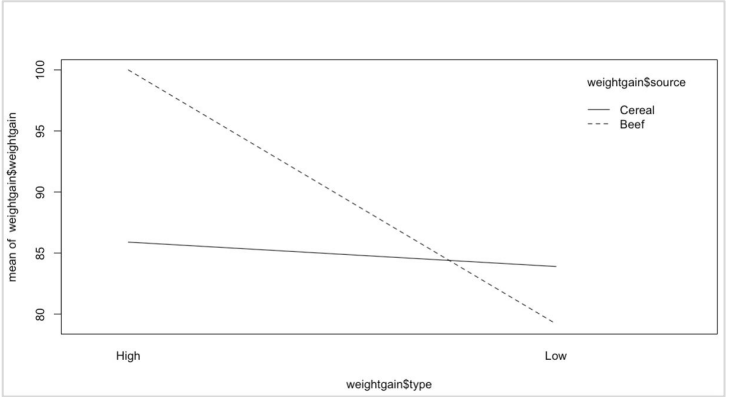
Now let's see how to estimate the contrasts between levels in R:

```
# Plot contrasts between types, separately by source
emmip(wg_fit, ~ type | source)

# Plot contrasts between types with different lines for sources
interaction.plot(weightgain$type, weightgain$source, weightgain$weightgain)

# Print contrast estimates and p-values
emmeans(wg_fit, pairwise ~ type | source)
```

```
> emmeans(wg_fit, pairwise ~ type | source)
$emmeans
source = Beef:
 type emmean  SE df lower.CL upper.CL
 High    100 4.7 36       90      110
 Low      79 4.7 36       70       89

source = Cereal:
 type emmean  SE df lower.CL upper.CL
 High     86 4.7 36       76       95
 Low      84 4.7 36       74       93

Confidence level used: 0.95

$contrasts
source = Beef:
 contrast   estimate  SE df t.ratio p.value
 High - Low     20.8 6.7 36   3.110  <.0001

source = Cereal:
 contrast   estimate  SE df t.ratio p.value
 High - Low      2.0 6.7 36   0.300  0.7700
```

Results:

- For *low-protein* diets, the use of cereal as the source of the protein leads to a greater weight gain than using beef
- For *high-protein* diets the reverse is the case with the beef/high diet leading to the highest weight gain



Curves and Interactions

Curves from lines

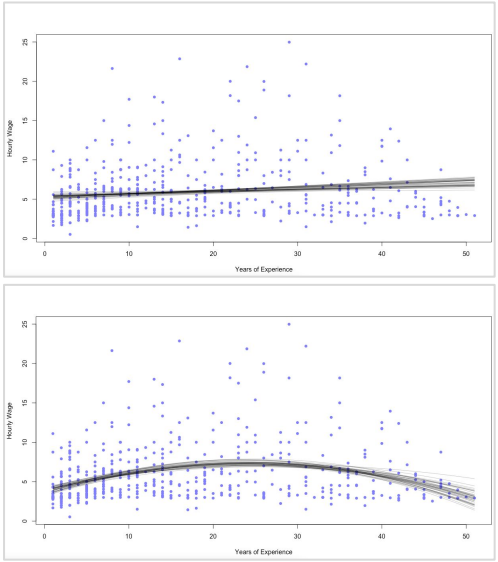
Polynomial regression
(e.g. adding a quadratic term)

Polynomial regression

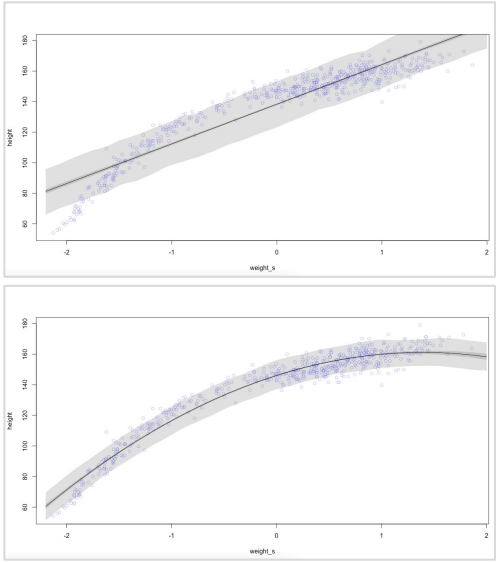
Polynomial regression uses powers of a variable – e.g. squares and cubes – as extra predictors. This is an easy way to build curved associations.

Here are two examples that illustrate the difference between models that predict linear (top panels) and quadratic (bottom panels) associations:

Hourly wage vs Years of experience



Height vs Weight



Modeling in the Bayesian framework

- Step 1. Specify the data model and prior
- Step 2. Estimate the model parameters
- Step 3. Check sampling quality and model fit
- Step 4. Summarize and interpret results

Step 1. Specify and check the priors

Model:

$$h_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = a + b_1 * x_i + b_2 * x_i^2$$

where h_i is height for person i, x_i is weight (standardized) for person i, and x_i^2 is the square of standardized weight, which constructs a parabola instead of a perfectly straight line, to measure the curvature of the relationship

Priors:

First, we'll standardize the predictor to avoid any possible numerical glitches from squaring potentially large numbers

- **Intercept (a)**: defined as the expected height when weight is at its mean value. The range of this prior encompasses a huge range of plausible mean heights for humans:

  a ~ Normal(178,20)

- **Weight slope (b1)**: we know average height increases with average weight, at least up to a point, so we'll assign a log-normal prior, which will force b1 to be strictly positive:

  b1 ~ Log-Normal(0,1)

- **Weight^2 slope (b2)**: if we center it on 0, that indicates no bias for positive or negative; for the standard deviation, let's start with a a guess at 1:
  b2 ~ Normal(0,1)

```
library(rethinking)
data(Howell1)
d <- Howell1

# standardize predictor
d$weight_s <- ( d$weight - mean(d$weight) )/sd(d$weight)

# create quadratic term
d$weight_s2 <- d$weight_s^2
```

```
m4.5 <- quap(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a + b1*weight_s + b2*weight_s2 ,
    a ~ dnorm( 178 , 20 ) ,
    b1 ~ dlnorm( 0 , 1 ) ,
    b2 ~ dnorm( 0 , 1 ) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d )
precis( m4.5 )
```

rstanarm **default** priors

By default, `stan_glm` uses a weakly informative family of prior distributions (type `vignette("priors", package="rstanarm")`)

To understand the default prior, recall that a regression coefficient corresponds to the expected difference in the outcome, y, comparing two people who differ by 1 unit in a predictor, x

If x and y have both been standardized, then the coefficient is the expected difference in standard deviations in y corresponding to a change of 1 standard deviation in x

We typically expect such a difference to be less than 1 in absolute value, hence a normal prior with mean 0 and scale 2.5 will partially pool noisy coefficient estimates toward that range

```
# rstanarm version
prior_summary( stan_glm(height ~ weight_s + weight_s2, data = d) )
```

```
Priors for model 'stan_glm(height ~ weight_s + weight_s2, data = d)'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 138, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 138, scale = 69)

Coefficients
  Specified prior:
    ~ normal(location = [0,0], scale = [2.5,2.5])
  Adjusted prior:
    ~ normal(location = [0,0], scale = [69.01,66.97])

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.036)
------
See help('prior_summary.stanreg') for more details
```

rstanarm **default** priors

Priors (from specified to adjusted):

- **Intercept (a)** *after predictors centered*

  `prior_intercept = normal( mean_y , 2.5*sd_y )`

```
> # Prior: Intercept (after predictor standardized)
> # Mean of y , Standard deviation of y , 2.5 * Standard deviation of y
> paste( "mean_y:", round(mean(d$height),2),
+        "; sd_y:", round(sd(d$height),2),
+        "; 2.5*sd_y:", round(2.5*sd(d$height),2) )
[1] "mean_y: 138.26 ; sd_y: 27.6 ; 2.5*sd_y: 69.01"
```

- **Coefficients (b)**

  `prior = normal( 0 , 2.5*sd_y/sd_x )`

```
> # Prior: Coefficients
> # SD of y , SD of x , 2.5 * SD of y / SD of x
> paste( "sd_y:", round(sd(d$height),2),
+        "; sd_x:", round(sd(d$weight_s),2),
+        "; 2.5*sd_y/sd_x:", round(2.5*sd(d$height)/sd(d$weight_s),2) )
[1] "sd_y: 27.6 ; sd_x: 1 ; 2.5*sd_y/sd_x: 69.01"
>
> paste( "sd_y:", round(sd(d$height),2),
+        "; sd_x:", round(sd(d$weight_s2),2),
+        "; 2.5*sd_y/sd_x:", round(2.5*sd(d$height)/sd(d$weight_s2),2) )
[1] "sd_y: 27.6 ; sd_x: 1.03 ; 2.5*sd_y/sd_x: 66.97"
```

- **Auxiliary (sigma)**

  `prior_aux = exponential( 1/sd_y )`

```
> # Prior: Auxiliary (sigma)
> # SD of y , 1/SD of y
> paste( "sd_y:", round(sd(d$height),2),
+        "; 1/sd_y:", round(1/sd(d$height),3) )
[1] "sd_y: 27.6 ; 1/sd_y: 0.036"
```

```
# rstanarm version
prior_summary( stan_glm(height ~ weight_s + weight_s2, data = d) )
```

```
Priors for model 'stan_glm(height ~ weight_s + weight_s2, data = d)'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 138, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 138, scale = 69)

Coefficients
  Specified prior:
    ~ normal(location = [0,0], scale = [2.5,2.5])
  Adjusted prior:
    ~ normal(location = [0,0], scale = [69.01,66.97])

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.036)
------
See help('prior_summary.stanreg') for more details
```

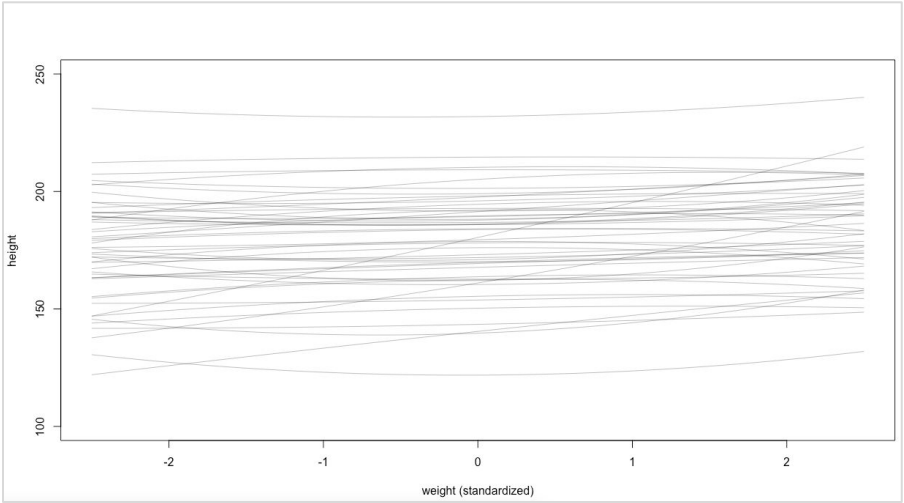
rstanarm **custom** priors

Model with custom priors:

```
# ROS pg. 125 Informed priors
prior_summary( stan_glm(height ~ weight_s + weight_s2, data = d,
                        prior_intercept = normal( location = c(178) , scale = c(20) ) ,
                        prior = normal( location = c(0,0) , scale = c(1,1) ) ,
                        prior_aux = exponential( 1 )) )
```

Plot lines implied by priors:

```
# Plot lines implied by prior: SR pg. 107
# set up the plot dimensions
plot( NULL , xlim=c(-2.5,2.5) , ylim=c(100,250) ,
      xlab="weight (standardized)" , ylab="height")

# draw 50 lines from the prior
N <- 50
a <- rnorm(N, 178, 20)
b1 <- rlnorm(N, 0, 1)
b2 <- rnorm(N, 0, 1)
mu.link <- function(weight) a + b1*weight + b2*weight^2
weight_seq <- seq( from=-2.5 , to=2.5 , length.out=50 )
mu <- sapply( weight_seq , mu.link )
for ( i in 1:N ) lines( weight_seq , mu[i,] , col=col.alpha("black",0.3) )
```



Step 2. Estimate the parameters

`rstanarm`

- The `stan_glm` function is similar in syntax to `glm` but rather than performing MLE, full Bayesian estimation is performed via MCMC. The Bayesian model adds priors (independent by default) on the coefficients of the GLM

`rethinking`

- `quap` uses quadratic approximation and the model definition to define the posterior probability at each combination of parameter values, climb the posterior distribution and find the peak, its maximum a posteriori (MAP), and estimate the quadratic curvature at the MAP to produce an approximation of the posterior distribution
- `ulam`, named after Stanisław Ulam, who was one of the parents of the Monte Carlo method and is the namesake of the Stan project as well, translates the model definition into a Stan model and returns an object with summary information and samples from the posterior distribution

rstanarm:

```
m4.5_rstanarm <- stan_glm(height ~ weight_s + weight_s2, data = d,
                          prior_intercept = normal( location = c(178) , scale = c(20) ) ,
                          prior = normal( location = c(0,0) , scale = c(1,1) ) ,
                          prior_aux = exponential( 1 ))
summary( m4.5_rstanarm , digits=2 )
```

rethinking:

```
m4.5 <- quap(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a + b1*weight_s + b2*weight_s2 ,
    a ~ dnorm( 178 , 20 ) ,
    b1 ~ dlnorm( 0 , 1 ) ,
    b2 ~ dnorm( 0 , 1 ) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d )
precis( m4.5 )
```

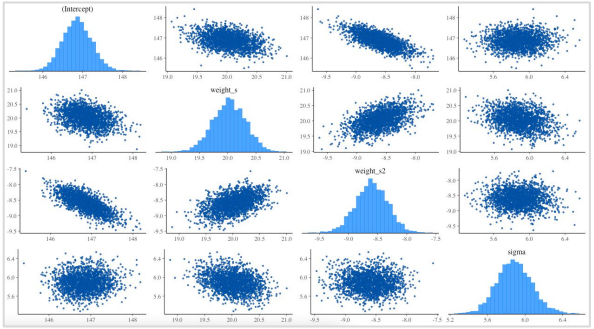
```
m4.5 <- ulam(
  alist(
    height ~ dnorm( mu , sigma ) ,
    mu <- a + b1*weight_s + b2*weight_s2 ,
    a ~ dnorm( 178 , 20 ) ,
    b1 ~ dlnorm( 0 , 1 ) ,
    b2 ~ dnorm( 0 , 1 ) ,
    sigma ~ dunif( 0 , 50 )
  ) , data=d , chains=4 , cores=4 )
precis( m4.5 )
```

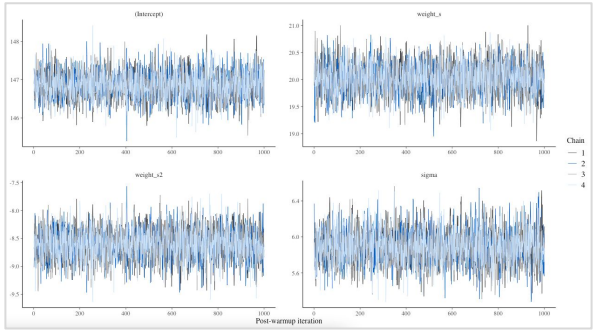
Step 3. Check sampling quality and model fit

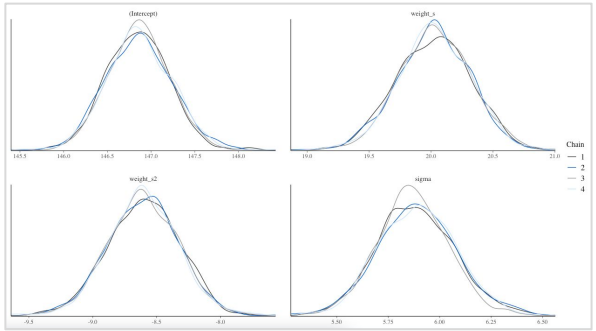
`rstanarm`

By default, all `rstanarm` modeling functions will run four randomly initialized Markov chains, each for 2000 iterations (including a warmup period of 1000 iterations that is discarded). All chains must converge to the target distribution for inferences to be valid

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      height ~ weight_s + weight_s2
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 544
 predictors:   3

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.01 1.00 3864
weight_s      0.00 1.00 3536
weight_s2     0.00 1.00 3694
sigma         0.00 1.00 3684
mean_PPD      0.01 1.00 4184
log-posterior 0.03 1.00 1941
```



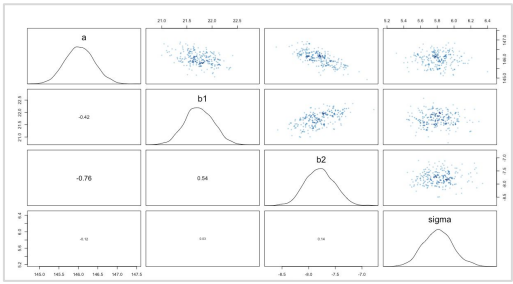




Step 3. Check sampling quality and model fit

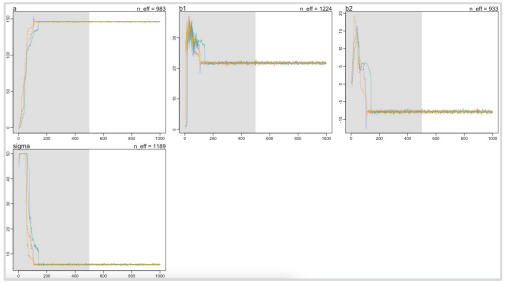
`rethinking`

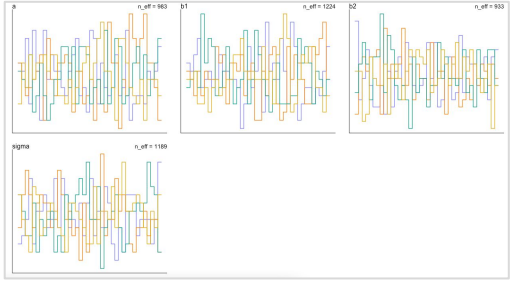
By default, the `ulam` modeling function will run one randomly initialized Markov chain for 1000 iterations (including a warmup period of 500 iterations that is discarded). Use `chains=4`, `iter=2000`, and `cmdstan=TRUE` to increase the number of independent chains, increase the number of iterations, and to use `cmdstanr` instead of `rstan` to run chains. All chains must converge to the target distribution for inferences to be valid




```
Hamiltonian Monte Carlo approximation
2000 samples from 4 chains

Sampling durations (seconds):
        warmup sample total
chain:1   0.29   0.17  0.46
chain:2   0.24   0.18  0.42
chain:3   0.23   0.20  0.43
chain:4   0.27   0.19  0.46

Formula:
height ~ dnorm(mu, sigma)
mu <- a + b1 * weight_s + b2 * weight_s2
a ~ dnorm(178, 20)
b1 ~ dlnorm(0, 1)
b2 ~ dnorm(0, 1)
sigma ~ dunif(0, 50)
```

	mean	sd	5.5%	94.5%	n_eff	Rhat4
a	146.04	0.37	145.45	146.63	983	1
b1	21.74	0.28	21.29	22.19	1224	1
b2	-7.79	0.27	-8.21	-7.35	933	1
sigma	5.81	0.18	5.54	6.09	1189	1





Step 4. Summarize and interpret results

Compared to the linear model shown at the beginning of this section, this quadratic model does a better job of finding a central path through the data

However, remember that a better fit to sample might not be a better model and this model contains no biological information. We aren't learning any causal relationship between height and weight

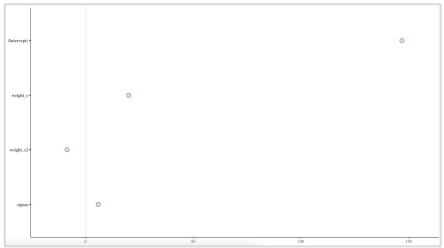
Estimates:

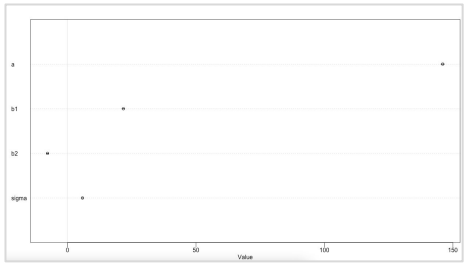
	mean	sd	10%	50%	90%
(Intercept)	146.9	0.4	146.4	146.9	147.3
weight_s	20.0	0.3	19.7	20.0	20.4
weight_s2	-8.6	0.3	-9.0	-8.6	-8.3
sigma	5.9	0.2	5.7	5.9	6.1

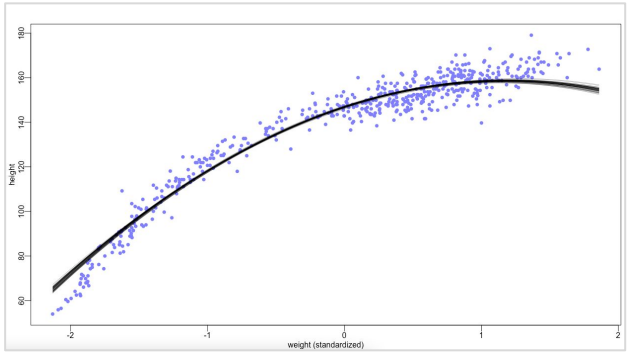
	mean	sd	5.5%	94.5%	n_eff	Rhat4
a	146.04	0.37	145.45	146.63	983	1
b1	21.74	0.28	21.29	22.19	1224	1
b2	-7.79	0.27	-8.21	-7.35	933	1
sigma	5.81	0.18	5.54	6.09	1189	1

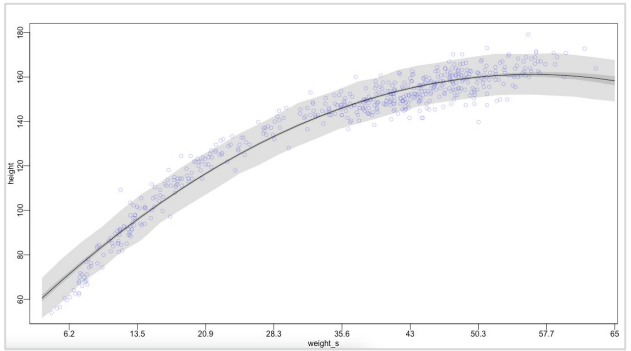
Fit Diagnostics:

	mean	sd	10%	50%	90%
mean_PPD	138.3	0.4	137.8	138.3	138.7









Multiple Predictors

Multiple Predictors: Types of Predictors

Categorical

- Binary
- Several categories

Ordered

- Ordinal
- Quantitative (Continuous)

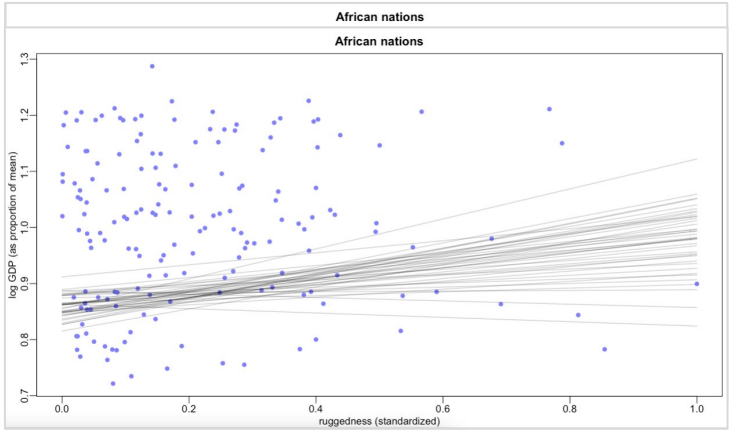
Building an interaction

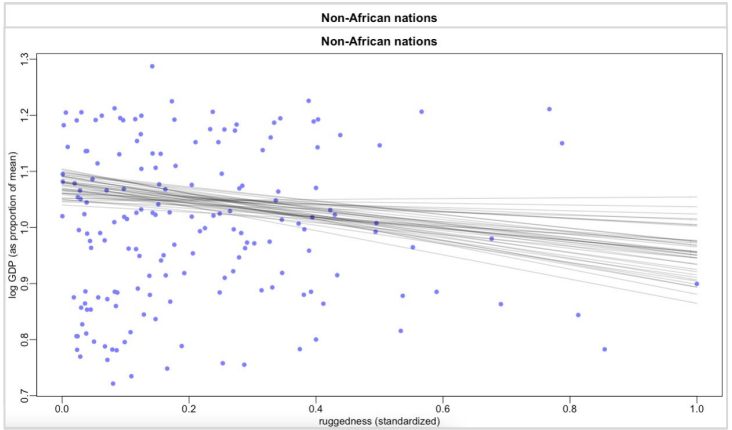
Binary x Continuous

Interactions: Binary and Continuous

Separate linear regressions inside and outside of Africa, for log-GDP against terrain ruggedness. The slope is positive inside Africa, but negative outside.

How can we recover this reversal of the slope, using the combined data?





Interactions: Binary and Continuous

We could split the data into two data frames, one for Africa and one for all other continents, but that's not a good idea (here are four reasons):

- First, there are usually some parameters, such as σ, that the model says do not depend upon continent. By splitting the data, you're hurting the accuracy of the estimates for these parameters
- Second, to acquire probability statements about the variable you used to split the data, you need to include it in the model
- Third, we may want to use information criteria or another method to compare models. To compare a model that treats all continents the same to a model that allows different slopes in different continents, we need models that use all the same data
- Fourth, once we start using multilevel models, we'll see there are advantages to borrowing information across categories like "Africa" and "not Africa", especially when sample sizes vary across categories

Modeling in the Bayesian framework

- Step 1. Specify the data model and prior
- Step 2. Estimate the model parameters
- Step 3. Check sampling quality and model fit
- Step 4. Summarize and interpret results

Step 1. Specify and check the priors

Model:

$$\log(y_i) \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = a + b * (r_i - \text{mean}(r))$$

where y_i is GDP for nation i, r_i is terrain ruggedness for nation i, and mean(r) is the average ruggedness in the whole sample. Its value is 0.215 – most nations aren't that rugged. Using mean(r) makes it easier to assign a prior to the intercept a

Priors:

We may not have much scientific knowledge about the relationship between log GDP and terrain ruggedness, but the scaled outcome and predictor measurements constrain the priors in useful ways

- **Intercept (a)**: defined as the log GDP when ruggedness is at the sample mean. It must be close to 1 because we scaled the outcome so the mean is 1:

  a ~ Normal(1,1)

- **Slope (b)**: if we center it on 0, that indicates no bias for positive or negative; for the standard deviation, let's start with a a guess at 1:

  b ~ Normal(0,1)

```
library(rethinking)
data(rugged)
d <- rugged

# make log version of outcome
d$log_gdp <- log( d$rgdppc_2000 )

# extract countries with GDP data
dd <- d[ complete.cases(d$rgdppc_2000) , ]

# rescale variables
dd$log_gdp_std <- dd$log_gdp / mean(dd$log_gdp)
dd$rugged_std <- dd$rugged / max(dd$rugged)
```

```
m8.1 <- quap(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a + b*( rugged_std - 0.215 ) ,
    a ~ dnorm( 1 , 1 ) ,
    b ~ dnorm( 0 , 1 ) ,
    sigma ~ dexp( 1 )
  ) , data=dd )
```

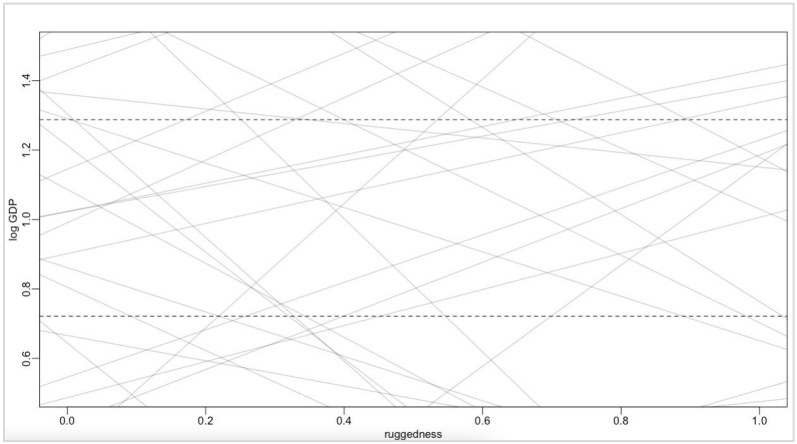
Step 1. Specify and check the priors

Intercept (a):

- The horizontal dashed lines show the maximum and minimum observed log GDP values
- Many of the lines are in impossible territory. Considering the measurement scales, the lines have to pass closer to the point where ruggedness is average (0.215 on the x-axis) and proportional log GDP is 1. Instead, there are lots of lines that expected average GDP outside observed ranges
- So we need a tighter standard deviation on the a prior. Let's try Normal(0, 0.1), which assigns 95% of the plausibility between 0.8 and 1.2

Slope (b):

- The slopes are too variable. It's not plausible that terrain ruggedness explains most of the observed variation in log GDP.
- An implausibly strong association would be, for example, a line that goes from minimum ruggedness and extreme GDP on one end to maximum ruggedness and the opposite extreme of GDP on the other end. The slope of such a line must be about 1.3 – 0.7 = 0.6, the difference between the maximum and minimum observed proportional log GDP. Over half of all slopes have an absolute value greater than 0.6!
- So we need a less extreme b prior. Let's try Normal(0, 0.3), which makes a slope of 0.6 two standard deviations out

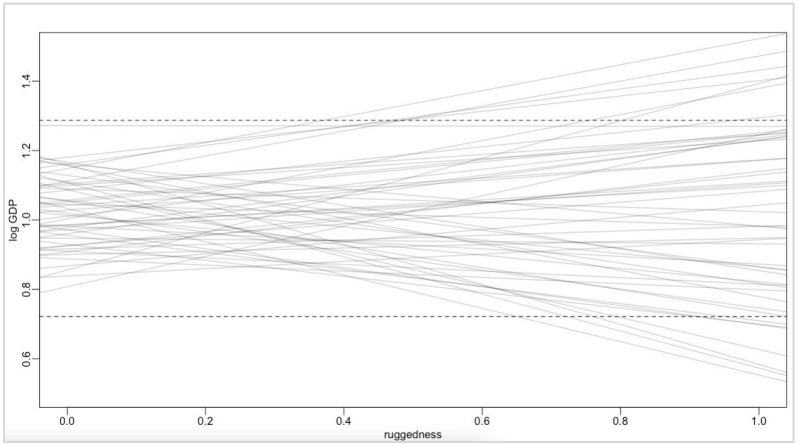



```
> ## SR code 8.4
> round( sum( abs(prior$b) > 0.6 ) / length(prior$b) , 3 )
[1] 0.545
```

Step 1. Specify and check the priors

Model with updated priors for a and b:

```
m8.1 <- quap(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a + b*( rugged_std - 0.215 ) ,
    a ~ dnorm( 1 , 0.1 ) ,
    b ~ dnorm( 0 , 0.3 ) ,
    sigma ~ dexp( 1 )
  ) , data=dd )
```



Intercept (a):

- Now many of the lines pass closer to the point where ruggedness is average (0.215 on the x-axis) and proportional log GDP is 1

Slope (b):

- Now the slopes are less extreme. Some of them are still implausibly strong, but these are much better than before. Only five percent of all slopes have an absolute value greater than 0.6

```
> ## SR code 8.4
> round( sum( abs(prior$b) > 0.6 ) / length(prior$b) , 3 )
[1] 0.05
```

rstanarm **default** priors

By default, `stan_glm` uses a weakly informative family of prior distributions (type `vignette("priors", package="rstanarm")`)

To understand the default prior, recall that a regression coefficient corresponds to the expected difference in the outcome, y, comparing two people who differ by 1 unit in a predictor, x

If x and y have both been standardized, then the coefficient is the expected difference in standard deviations in y corresponding to a change of 1 standard deviation in x

We typically expect such a difference to be less than 1 in absolute value, hence a normal prior with mean 0 and scale 2.5 will partially pool noisy coefficient estimates toward that range

```
# rstanarm version
dd$rugged_std_centered <- dd$rugged_std - 0.215

prior_summary( stan_glm(log_gdp_std ~ rugged_std_centered, data = dd) )
```

```
Priors for model 'stan_glm(log_gdp_std ~ rugged_std_centered, data = dd)'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 1, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 1, scale = 0.34)

Coefficients
  Specified prior:
    ~ normal(location = 0, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 0, scale = 1.8)

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 7.3)
------
See help('prior_summary.stanreg') for more details
```

rstanarm **default** priors

Priors (from specified to adjusted):

- **Intercept (a)** *after predictors centered*

  `prior_intercept = normal( mean_y , 2.5*sd_y )`

```
> # Mean of y , Standard deviation of y , 2.5 * Standard deviation of y
> paste( "mean_y:", mean(dd$log_gdp_std),
+        "; sd_y:", round(sd(dd$log_gdp_std),2),
+        "; 2.5*sd_y:", round(2.5*sd(dd$log_gdp_std),2) )
[1] "mean_y: 1 ; sd_y: 0.14 ; 2.5*sd_y: 0.34"
```

- **Coefficients (b)**

  `prior = normal( 0 , 2.5*sd_y/sd_x )`

```
> # SD of y , SD of x , 2.5 * SD of y / SD of x
> paste( "sd_y:", round(sd(dd$log_gdp_std),2),
+        "; sd_x:", round(sd(dd$rugged_std_centered),2),
+        "; 2.5*sd_y/sd_x:", round(2.5*sd(dd$log_gdp_std)/sd(dd$rugged_std_centered),2) )
[1] "sd_y: 0.14 ; sd_x: 0.19 ; 2.5*sd_y/sd_x: 1.82"
```

- **Auxiliary (sigma)**

  `prior_aux = exponential( 1/sd_y )`

```
> # SD of y , 1/SD of y
> paste( "sd_y:", round(sd(dd$log_gdp_std),2),
+        "; 1/sd_y:", round(1/sd(dd$log_gdp_std),2) )
[1] "sd_y: 0.14 ; 1/sd_y: 7.3"
```

```
# rstanarm version
dd$rugged_std_centered <- dd$rugged_std - 0.215

prior_summary( stan_glm(log_gdp_std ~ rugged_std_centered, data = dd) )
```

```
Priors for model 'stan_glm(log_gdp_std ~ rugged_std_centered, data = dd)'
------
Intercept (after predictors centered)
  Specified prior:
    ~ normal(location = 1, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 1, scale = 0.34)

Coefficients
  Specified prior:
    ~ normal(location = 0, scale = 2.5)
  Adjusted prior:
    ~ normal(location = 0, scale = 1.8)

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 7.3)
------
See help('prior_summary.stanreg') for more details
```

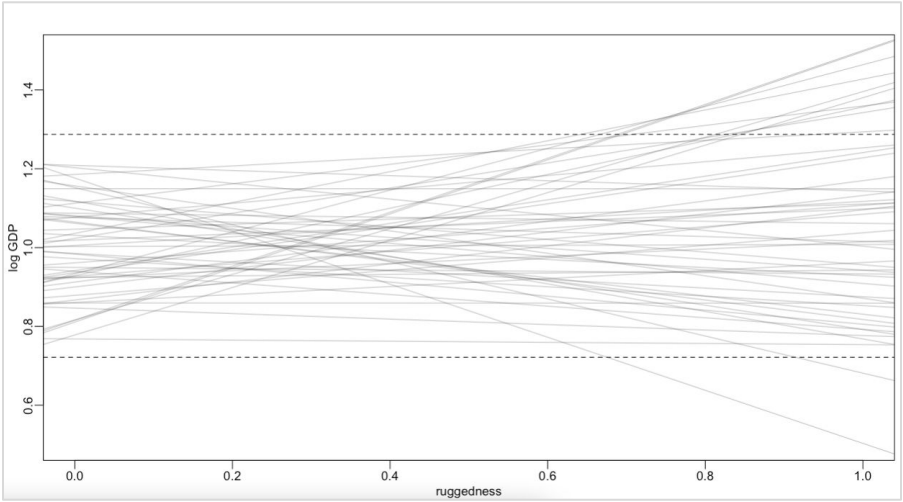
rstanarm **custom** priors

Model with custom priors for a and b:

```
# ROS pg. 125 Informed priors
prior_summary(stan_glm(log_gdp_std ~ rugged_std_centered, data = dd, refresh = 0 ,
                       prior_intercept = normal( location = c(1) , scale = c(0.1) ) ,
                       prior = normal( location = c(0) , scale = c(0.3) ) ,
                       prior_aux = exponential( 1 ) ) )
```

Plot lines implied by priors:

```
# Plot lines implied by prior: SR pg. 107
# set up the plot dimensions
plot( NULL , xlim=c(0,1) , ylim=c(0.5,1.5) ,
      xlab="ruggedness" , ylab="log GDP" )
abline( h=min(dd$log_gdp_std) , lty=2 )
abline( h=max(dd$log_gdp_std) , lty=2 )

# draw 50 lines from the prior
N <- 50
a <- rnorm(N, 1, 0.1)
b <- rnorm(N, 0, 0.3)
mu.link <- function(ruggedness) a + b*( ruggedness - 0.215 )
rugged_seq <- seq( from=-0.1 , to=1.1 , length.out=30 )
mu <- sapply( rugged_seq , mu.link )
for ( i in 1:N ) lines( rugged_seq , mu[i,] , col=col.alpha("black",0.3) )
```



Step 2. Estimate the parameters

`rstanarm`

- The `stan_glm` function is similar in syntax to `glm` but rather than performing MLE, full Bayesian estimation is performed via MCMC. The Bayesian model adds priors (independent by default) on the coefficients of the GLM

`rethinking`

- `quap` uses quadratic approximation and the model definition to define the posterior probability at each combination of parameter values, climb the posterior distribution and find the peak, its maximum a posteriori (MAP), and estimate the quadratic curvature at the MAP to produce an approximation of the posterior distribution
- `ulam`, named after Stanisław Ulam, who was one of the parents of the Monte Carlo method and is the namesake of the Stan project as well, translates the model definition into a Stan model and returns an object with summary information and samples from the posterior distribution

rstanarm:

```
m8.1_rstanarm <- stan_glm(log_gdp_std ~ rugged_std_centered,
                          data = dd,
                          prior_intercept = normal( location = c(1) , scale = c(0.1) ) ,
                          prior = normal( location = c(0) , scale = c(0.3) ) ,
                          prior_aux = exponential( 1 ) )
```

rethinking:

```
m8.1 <- quap(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a + b*( rugged_std - 0.215 ) ,
    a ~ dnorm( 1 , 0.1 ) ,
    b ~ dnorm( 0 , 0.3 ) ,
    sigma ~ dexp( 1 )
  ) , data=dd )
```

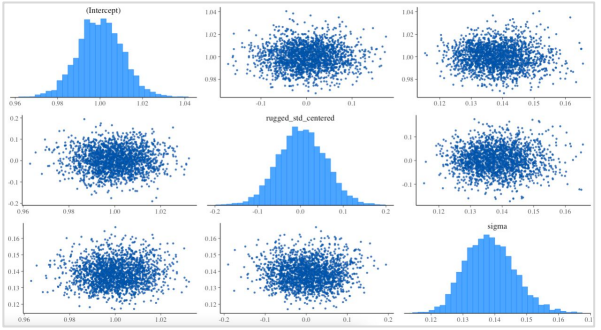
```
m8.1 <- ulam(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a + b*( rugged_std - 0.215 ) ,
    a ~ dnorm( 1 , 0.1 ) ,
    b ~ dnorm( 0 , 0.3 ) ,
    sigma ~ dexp( 1 )
  ) , data=dat_slim  , chains=4 , cores=4 )
```

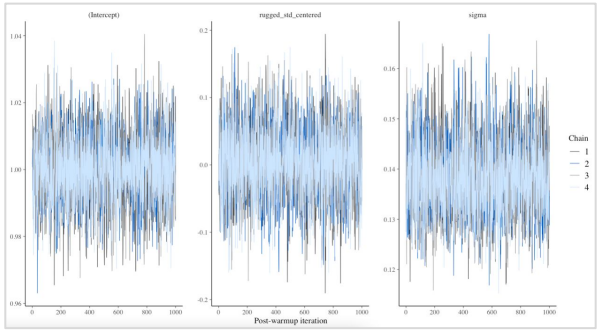
Step 3. Check sampling quality and model fit

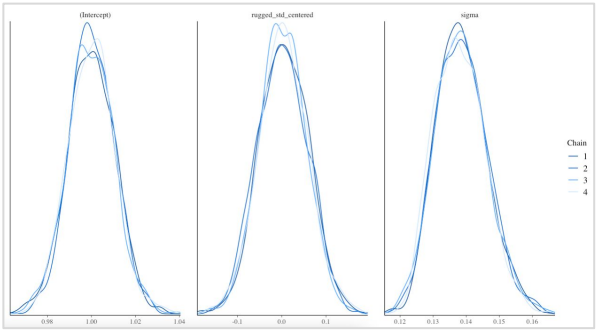
`rstanarm`

By default, all `rstanarm` modeling functions will run four randomly initialized Markov chains, each for 2000 iterations (including a warmup period of 1000 iterations that is discarded). All chains must converge to the target distribution for inferences to be valid

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      log_gdp_std ~ rugged_std_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 170
 predictors:   2

MCMC diagnostics
                      mcse Rhat n_eff
(Intercept)           0.00 1.00 3950
rugged_std_centered   0.00 1.00 3009
sigma                 0.00 1.00 3384
mean_PPD              0.00 1.00 3754
log-posterior         0.03 1.00 1636
```



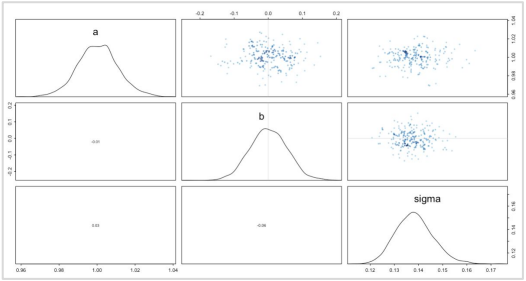




Step 3. Check sampling quality and model fit

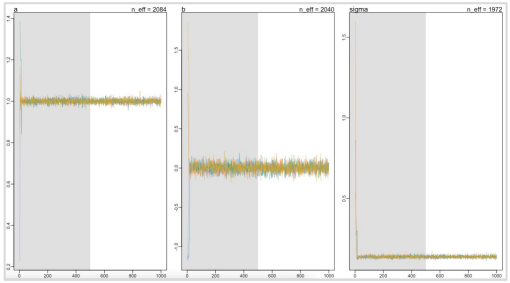
`rethinking`

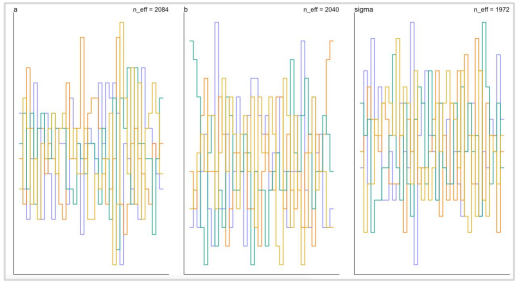
By default, the `ulam` modeling function will run one randomly initialized Markov chain for 1000 iterations (including a warmup period of 500 iterations that is discarded). Use `chains=4`, `iter=2000`, and `cmdstan=TRUE` to increase the number of independent chains, increase the number of iterations, and to use `cmdstanr` instead of `rstan` to run chains. All chains must converge to the target distribution for inferences to be valid




```
Hamiltonian Monte Carlo approximation
2000 samples from 4 chains

Sampling durations (seconds):
        warmup sample total
chain:1   0.04   0.03  0.06
chain:2   0.04   0.03  0.06
chain:3   0.03   0.02  0.06
chain:4   0.04   0.03  0.06

Formula:
log_gdp_std ~ dnorm(mu, sigma)
mu <- a + b * (rugged_std - 0.215)
a ~ dnorm(1, 0.1)
b ~ dnorm(0, 0.3)
sigma ~ dexp(1)
```

	mean	sd	5.5%	94.5%	n_eff	Rhat4
a	1.00	0.01	0.98	1.02	2084	1
b	0.00	0.06	-0.09	0.09	2040	1
sigma	0.14	0.01	0.13	0.15	1972	1

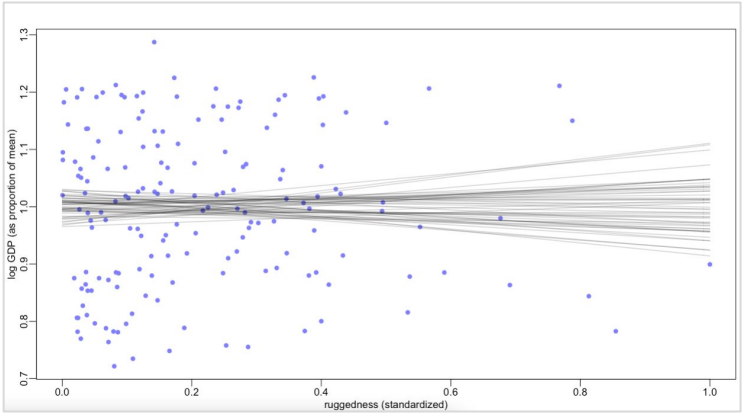




Step 4. Summarize and interpret results

Without splitting apart the continents, there is really no overall association between terrain ruggedness and log GDP.

However, now let's see how to split apart the continents using an interaction to uncover the association we saw at the beginning of this section



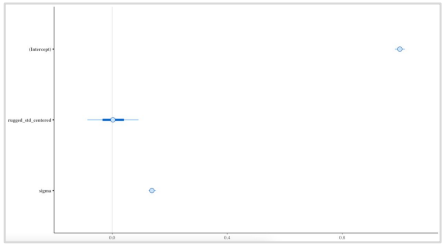
Estimates:

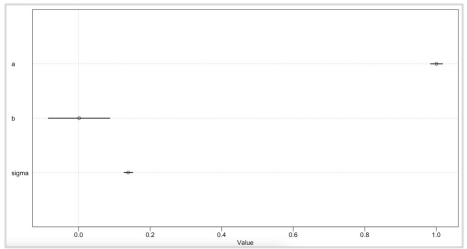
	mean	sd	10%	50%	90%
(Intercept)	1.00	0.01	0.99	1.00	1.01
rugged_std_centered	0.00	0.06	-0.07	0.00	0.07
sigma	0.14	0.01	0.13	0.14	0.15

	mean	sd	5.5%	94.5%	n_eff	Rhat4
a	1.00	0.01	0.98	1.02	2084	1
b	0.00	0.06	-0.09	0.09	2040	1
sigma	0.14	0.01	0.13	0.15	1972	1

Fit Diagnostics:

	mean	sd	10%	50%	90%
mean_PPD	1.00	0.01	0.98	1.00	1.02





Interactions: Adding an index variable isn't enough

To build a model that allows nations inside and outside Africa to have different intercepts, we need to modify the model for μ_i so the mean is conditional on continent:

$$\log(y_i) \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \mathbf{a_{CID[i]}} + b * (r_i - \text{mean}(r))$$

where **CID** is an *index variable*, continent ID. It takes the value 1 for African nations and 2 for all other nations. This means there are two parameters, a_1 and a_2, one for each unique index value. Again, y_i is GDP for nation i, r_i is terrain ruggedness for nation i, and mean(r) is the average ruggedness in the whole sample.

Using this index variable approach, instead of the conventional approach of adding another term with the 0/1 indicator, doesn't force us to say the mean for Africa is inherently less certain than the mean for all other continents. We can just reuse the same prior as before

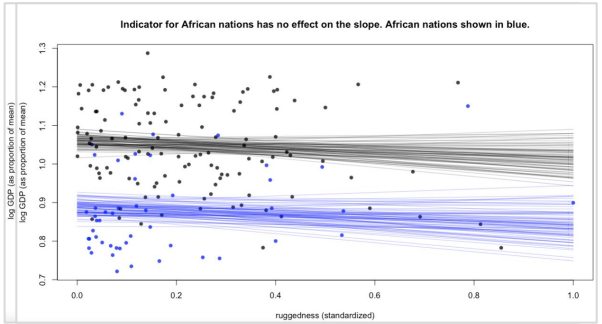
The African nations do have lower overall economic development, so the blue regression line is below, but parallel to, the black line. All including a dummy variable for African nations has done is allow the model to predict a lower mean for African nations. It can't do anything to the slope of the line

rethinking:

```
# Create variable to index Africa (1) and not Africa (2)
dd$cid <- ifelse( dd$cont_africa==1 , 1 , 2 )

# Model with index variable for intercepts
m8.2 <- quap(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a[cid] + b*( rugged_std - 0.215 ) ,
    a[cid] ~ dnorm( 1 , 0.1 ) ,
    b ~ dnorm( 0 , 0.3 ) ,
    sigma ~ dexp( 1 )
  ) , data=dd )
```

rstanarm:

```
m8.2_rstanarm <- stan_glm(log_gdp_std ~ 0 + as.factor(cid) + rugged_std_centered, data = dd ,
                          prior_intercept = normal( location = c(1) , scale = c(0.1) ) ,
                          prior = normal( location = c(0) , scale = c(0.3) ) ,
                          prior_aux = exponential( 1 ) )  # adapt_delta = 0.999
summary(m8.2_rstanarm, digits=2)
```



	mean	sd	5.5%	94.5%
a[1]	0.88	0.02	0.85	0.91
a[2]	1.05	0.01	1.03	1.07
b	-0.05	0.05	-0.12	0.03
sigma	0.11	0.01	0.10	0.12

Interactions: Adding an interaction does work

To build a model that allows nations inside and outside Africa to have different **intercepts and slopes**, we need to modify the model for μ_i so **both** are conditional on continent:

$$\log(y_i) \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \mathbf{a}_{\mathbf{CID[i]}} + \mathbf{b}_{\mathbf{CID[i]}} * (r_i - \text{mean}(r))$$

where **CID** is an *index variable*, continent ID. It takes the value 1 for African nations and 2 for all other nations. This means there are two intercept parameters, a_1 and a_2, and two slope parameters, b_1 and b_2, one for each unique index value.

Again, using this index variable approach doesn't force us to say the slope for Africa is inherently less certain than the slope for all other continents. Instead, we can just reuse the same prior for the slope

With this new model, the slope is essentially reversed inside Africa, 0.13 instead of –0.14.

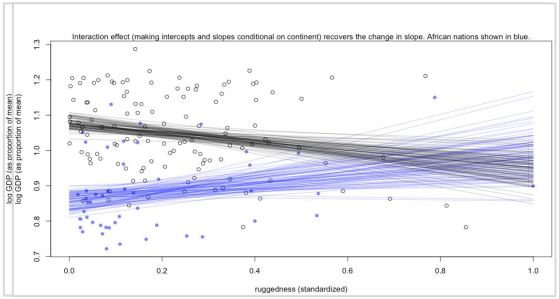
Using PSIS to compare this model with the previous ones (*we'll cover model comparison in a future lecture*), this model receives more than 95% of the weight; very strong support for including the interaction effect, if prediction is our goal

rethinking:

```
m8.3 <- quap(
  alist(
    log_gdp_std ~ dnorm( mu , sigma ) ,
    mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
    a[cid] ~ dnorm( 1 , 0.1 ) ,
    b[cid] ~ dnorm( 0 , 0.3 ) ,
    sigma ~ dexp( 1 )
  ) , data=dd )
```

rstanarm:

```
m8.3_rstanarm <- stan_glm(log_gdp_std ~ f_cid + rugged_std_centered + f_cid:rugged_std_centered, data = dd ,
                          prior_intercept = normal( location = c(1) , scale = c(0.1) ) ,
                          prior = normal( location = c(0) , scale = c(0.3) ) ,
                          prior_aux = exponential( 1 ) )
summary( m8.3_rstanarm, digits=2 )
```



	mean	sd	5.5%	94.5%
a[1]	0.89	0.02	0.86	0.91
a[2]	1.05	0.01	1.03	1.07
b[1]	0.13	0.07	0.01	0.25
b[2]	-0.14	0.05	-0.23	-0.06
sigma	0.11	0.01	0.10	0.12

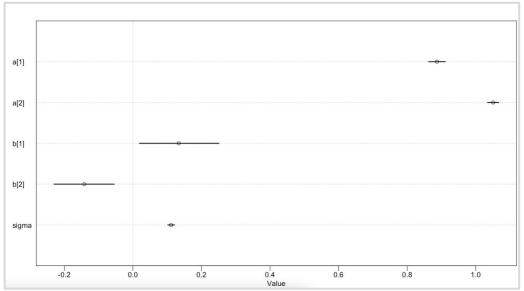
Step 4. Summarize and interpret results

By splitting apart the continents within the model, we can now see the association between terrain ruggedness and log GDP depends on whether the nation is inside or outside of Africa

However, interpreting a table of coefficient estimates isn't ideal and, as we'll see, it becomes even more difficult when there are more than two categories. *Let's complement them with plots*!

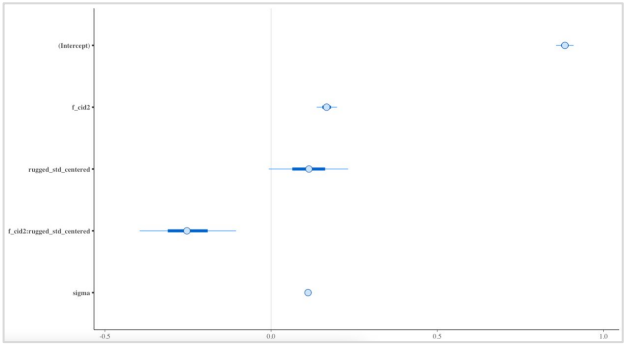
rethinking:

```
       mean   sd  5.5% 94.5% n_eff Rhat4
a[1]   0.89 0.02  0.86  0.91  2389     1
a[2]   1.05 0.01  1.03  1.07  2665     1
b[1]   0.13 0.07  0.02  0.25  1998     1
b[2]  -0.14 0.06 -0.23 -0.05  1835     1
sigma  0.11 0.01  0.10  0.12  2434     1
```



rstanarm:

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      log_gdp_std ~ f_cid + rugged_std_centered + f_cid:rugged_std_centered
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 170
 predictors:   4

Estimates:
                              mean   sd   10%   50%   90%
(Intercept)                   0.88   0.02  0.86  0.88  0.90
f_cid2                        0.17   0.02  0.14  0.17  0.19
rugged_std_centered           0.11   0.07  0.02  0.11  0.21
f_cid2:rugged_std_centered   -0.25   0.09 -0.37 -0.25 -0.14
sigma                         0.11   0.01  0.10  0.11  0.12
```



Step 4. Summarize and interpret results

Using the *`rethinking`* version, the fitted model is:

log(GDP) = 0.89 + 0.13 * ruggedness + error (African nations)
log(GDP) = 1.05 – 0.14 * ruggedness + error (Non-African nations)

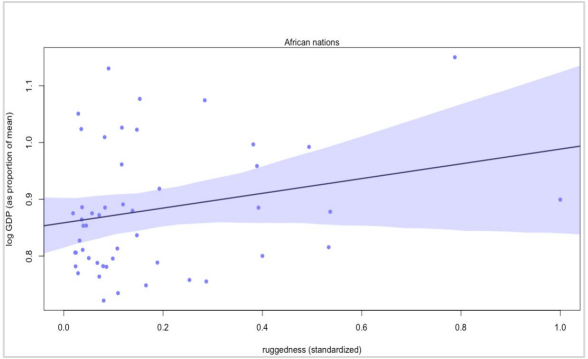
Using the *`rstanarm`* version, the fitted model is:

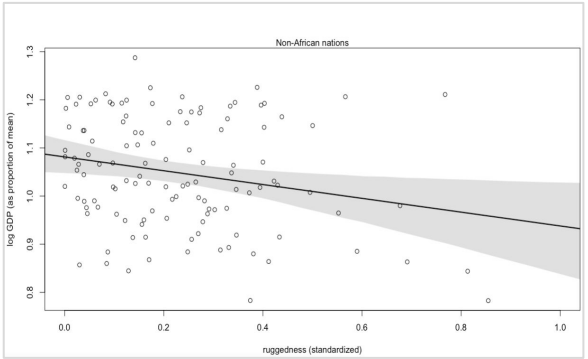
log(GDP) = (0.88 + 0.17*0) + (0.11 – 0.25*0) * ruggedness + error (African nations)
= 0.88 + 0.11 * ruggedness + error (African nations)

log(GDP) = (0.88 + 0.17*1) + (0.11 – 0.25*1) * ruggedness + error (Non-African nations)
= 1.05 – 0.14 * ruggedness + error (Non-African nations)

The `rstanarm` version requires some post-processing to compute the coefficients for the two categories (i.e. African and Non-African)

Now to interpret the coefficients…



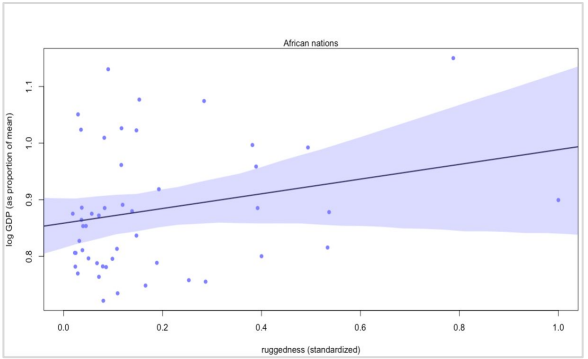


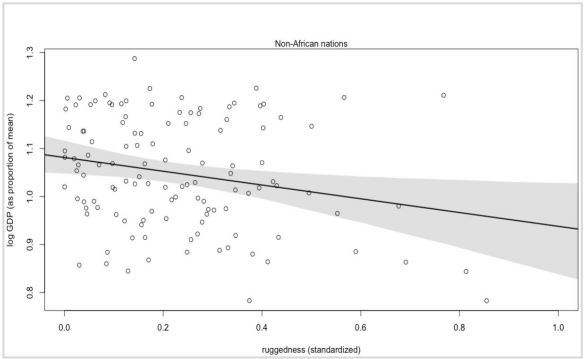
Step 4. Summarize and interpret results

With the ***rethinking*** version, the coefficients are more directly interpretable (compared to the `rstanarm` version):

- a_1 represents the predicted proportional log GDP for African nations that are of average ruggedness
- a_2 represents the predicted proportional log GDP for Non-African nations that are of average ruggedness
- b_1 can be thought of as the comparison of mean proportional log GDPs across nations *inside* Africa that differ by 1 unit of ruggedness
- b_2 can be thought of as the comparison of mean proportional log GDPs across nations *outside* Africa that differ by 1 unit of ruggedness

More than the tables of coefficients, the plots clearly show the association between terrain ruggedness and log GDP depends on whether the nation is inside or outside of Africa



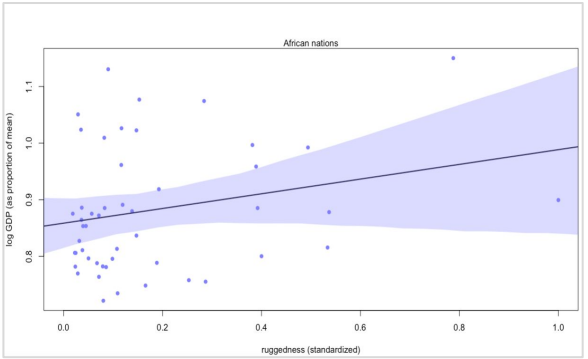


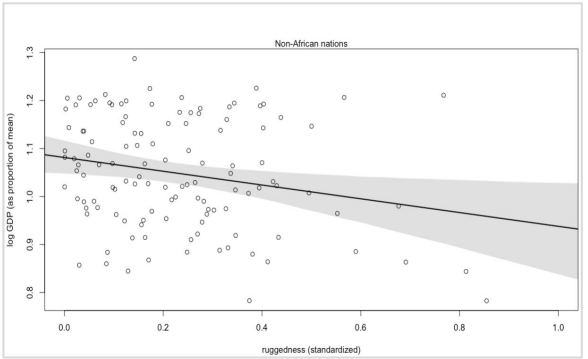
Step 4. Summarize and interpret results

With the ***rstanarm*** version, care must be taken in interpreting the coefficients:

- The *intercept* represents the predicted proportional log GDP for African nations that are of average ruggedness
- The *coefficient of Non-African nations (f_cid2)* can be thought of as the *difference* between the predicted proportional log GDP for African and Non-African nations that are of average ruggedness
- The *coefficient for ruggedness (rugged_std_centered)* can be thought of as the comparison of mean proportional log GDPs across nations inside Africa that differ by 1 unit of ruggedness
- The *coefficient on the interaction term (f_cid2:rugged_std_centered)* represents the *difference* in slope for ruggedness, comparing nations inside and outside of Africa

More than the tables of coefficients, the plots clearly show the association between terrain ruggedness and log GDP depends on whether the nation is inside or outside of Africa





Building an interaction

Several categories x Continuous

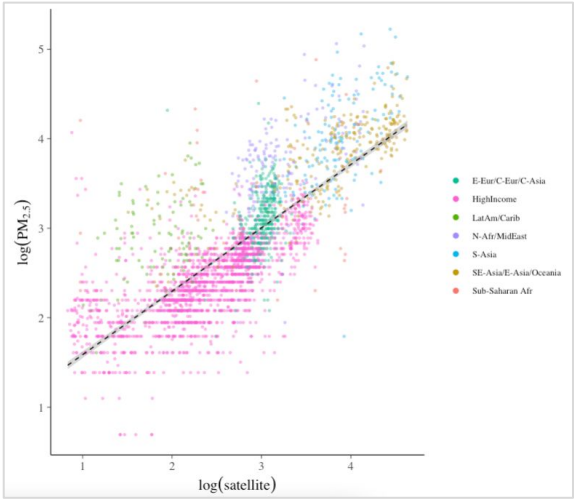
Multiple Predictors: Several categories

Example from [Visualization in Bayesian workflow]

The statistical problem is that we have direct measurements of $PM_{2.5}$ from only a sparse network of 2980 ground monitors with heterogeneous spatial coverage (especially poor coverage across Africa, central Asia, and Russia)

To estimate the public health effect of $PM_{2.5}$, we need estimates of its concentration at the same spatial resolution as the population data. To obtain these estimates, we supplement the direct measurements with high resolution satellite data

The hope is that we can use the ground monitor data to calibrate the approximate satellite measurements and hence obtain estimates of $PM_{2.5}$ concentration at the required spatial resolution



Visual inspection of the data, colored by World Health Organization super-region, suggests that a single linear trend for all super-regions is insufficient. For example, we know that developed and developing countries have different levels of industrialization and hence different air pollution. If these differences are not appropriately captured, fitting only a single regression line could leave us in danger of falling prey to Simpson's paradox (that a trend can reverse when data are grouped).

How can we allow for varying intercepts and varying slopes, using the entire dataset?

Several categories: Varying intercepts

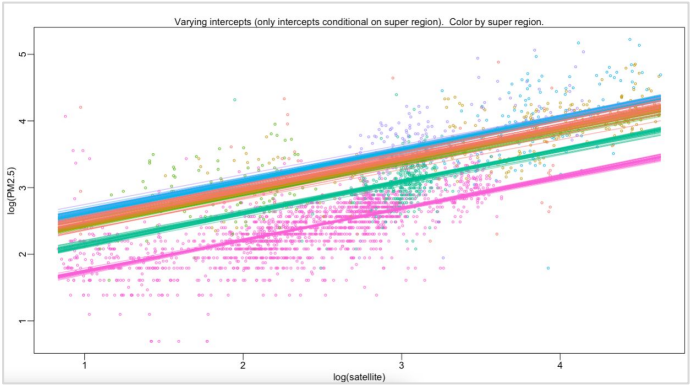
Example from [Visualization in Bayesian workflow]

To build a model that allows the super-regions to have different intercepts, we need to specify a model for μ_i such that the mean is conditional on super-region:

$$\log(y_i) \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \mathbf{a_{SR[i]}} + b * (\log(\text{sat})_i - \text{mean}(\log(\text{sat})))$$

where **SR** is an *index variable*, super-region ID. It takes the values 1 through 7 for the seven super-regions. This means there are seven parameters, $a_1 \ldots a_7$, one for each unique index value. y_i is log($PM_{2.5}$) *ground* $PM_{2.5}$ concentration for country i, $\log(\text{sat})_i$ is *satellite* $PM_{2.5}$ concentration for country i, and mean(log(sat)) is the average satellite $PM_{2.5}$ concentration in the whole sample.




```
fit_pm <- ulam(
  alist(
    log_pm25 ~ dnorm( mu , sigma ) ,
    mu <- a[super_region] + b*( log_sat - mean_log_sat ) ,
    a[super_region] ~ dnorm( 1 , 2 ) ,
    b ~ dnorm( 0 , 1 ) ,
    sigma ~ dexp( 1 )
  ) ,
  data=GM@data , chains=4 , cores=4 )

precis( fit_pm , depth=2 )
```

rstanarm:

```
fit_pm <- stan_glm(log_pm25 ~ f_super_region + c_log_sat + f_super_region:c_log_sat,
                   data=GM@data)

summary( fit_pm , digits=2 )
```

	mean	sd	5.5%	94.5%	n_eff	Rhat4
a[1]	2.53	0.01	2.52	2.54	2132	1
a[2]	3.42	0.03	3.37	3.46	1845	1
a[3]	3.42	0.03	3.37	3.47	1530	1
a[4]	2.93	0.02	2.90	2.96	2384	1
a[5]	3.24	0.03	3.18	3.30	2234	1
a[6]	3.27	0.02	3.23	3.31	1691	1
a[7]	3.29	0.06	3.20	3.38	2017	1
b	0.47	0.01	0.45	0.49	1329	1
sigma	0.35	0.00	0.34	0.35	2832	1

This varying-intercepts model allows the mean log($PM_{2.5}$) to be different for each super-region, but it forces the slope of the regression of ground $PM_{2.5}$ on satellite $PM_{2.5}$ to be the same for each super-region. This isn't quite what we need, so let's add an interaction!

Several categories: Varying intercepts & slopes

Example from [Visualization in Bayesian workflow]

To build a model that allows the super-regions to have different **intercepts and slopes**, we need to modify the model for μ_i so **both** are conditional on super-region:

$$\log(y_i) \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \mathbf{a_{SR[i]}} + \mathbf{b_{SR[i]}} * (\log(\text{sat})_i - \text{mean}(\log(\text{sat})))$$

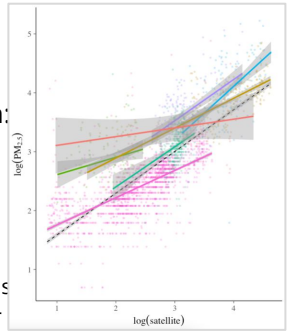
where **SR** is an *index variable*, super-region ID. It takes the values 1 through 7 for the seven super-regions. This means there are seven intercept parameters, $a_1 \ldots a_7$, and seven slope parameters, $b_1 \ldots b_7$, one for each unique index value.

```
fit_pm_rethinking <- ulam(
  alist(
    log_pm25 ~ dnorm( mu , sigma ) ,
    mu <- a[super_region] + b[super_region]*( log_sat - mean_log_sat ) ,
    a[super_region] ~ dnorm( 1 , 2 ) ,
    b[super_region] ~ dnorm( 0 , 1 ) ,
    sigma ~ dexp( 1 )
  ) ,
  data=GM@data , chains=4 , cores=4 )

precis( fit_pm_rethinking , depth=2 )
```

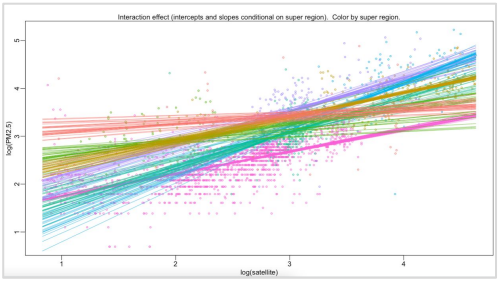
	mean	sd	5.5%	94.5%	n_eff	Rhat4
a[1]	2.53	0.01	2.51	2.54	3448	1
a[2]	3.29	0.05	3.21	3.37	2324	1
a[3]	2.89	0.10	2.73	3.06	1775	1
a[4]	2.86	0.05	2.79	2.93	2137	1
a[5]	3.10	0.08	2.97	3.23	1371	1
a[6]	3.23	0.04	3.17	3.29	1937	1
a[7]	3.35	0.06	3.26	3.44	2531	1
b[1]	0.46	0.01	0.44	0.48	2945	1
b[2]	0.70	0.07	0.59	0.81	2380	1
b[3]	0.91	0.08	0.78	1.05	1812	1
b[4]	0.67	0.12	0.48	0.87	1958	1
b[5]	0.29	0.10	0.14	0.44	1419	1
b[6]	0.51	0.03	0.46	0.55	1854	1
b[7]	0.15	0.06	0.06	0.24	4585	1
sigma	0.34	0.00	0.33	0.35	3941	1

Finally, this is what we need. This varying-intercepts, varying-slopes model allows both the mean log($PM_{2.5}$) and the slope of the regression of ground $PM_{2.5}$ on satellite $PM_{2.5}$ to be different for each super-region.

Figure from paper:



Draws from posterior:



rstanarm:

```
fit_pm_rstanarm <- stan_glm(log_pm25 ~ f_super_region + c_log_sat + f_super_region:c_log_sat,
                            data=GM@data)

summary( fit_pm_rstanarm , digits=2 )
```

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      log_pm25 ~ f_super_region + c_log_sat + f_super_region:c_log_sat
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 2980
 predictors:   14

Estimates:
                                mean   sd   10%   50%   90%
(Intercept)                     2.53   0.01  2.52  2.53  2.54
f_super_region2                 0.76   0.05  0.71  0.76  0.82
f_super_region3                 0.37   0.10  0.23  0.37  0.50
f_super_region4                 0.33   0.05  0.27  0.33  0.39
f_super_region5                 0.57   0.09  0.46  0.57  0.68
f_super_region6                 0.70   0.04  0.66  0.70  0.75
f_super_region7                 0.83   0.06  0.75  0.83  0.90
c_log_sat                       0.46   0.01  0.45  0.46  0.48
f_super_region2:c_log_sat       0.24   0.07  0.15  0.24  0.32
f_super_region3:c_log_sat       0.45   0.08  0.34  0.45  0.56
f_super_region4:c_log_sat       0.22   0.13  0.06  0.22  0.38
f_super_region5:c_log_sat      -0.17   0.10 -0.30 -0.17 -0.04
f_super_region6:c_log_sat       0.04   0.03  0.00  0.04  0.09
f_super_region7:c_log_sat      -0.31   0.06 -0.39 -0.31 -0.24
sigma                           0.34   0.00  0.34  0.34  0.35
```

Building an interaction

Continuous x Continuous

Continuous interactions

Interpretation challenges

Interaction effects are difficult to interpret, especially using only posterior means and standard deviations

It's one thing to make a slope conditional upon a *category*. Here, the model reduces to estimating a difference slope for each category. It's quite a lot harder to understand that a slope varies in a continuous fashion with a continuous variable

To visualize this two-way interaction between two continuous variables, let's plot it using a *triptych* plot, a panel of three figures that comprise a whole picture of the regression results

There's nothing special about having three figures – in other cases you might want more or less. It's useful because it allows one to see how the interaction alters a slope, across changes in a chosen variable

Example: Tulip blooms

Outcome:
the size of blooms from beds of tulips grown in greenhouses, under different soil and light conditions

Predictors:

- Water - three ordered levels of soil moisture, from low (1) to high (3)
- Shade - three ordered levels of light exposure, from high (1) to low (3)

Since both water and light help plants grow, it stands to reason the independent effect of each will be to produce bigger blooms.

But we'll also be interested in their interaction – in the absence of light, it's hard to see how water helps a plant and, in the absence of water, sunlight does a plant little good. We can model this dependency with an interaction!

Step 1. Specify and check the priors

Model:

$$B_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = a + b_W W_i + b_S S_i + b_{WS} W_i S_i$$

where B_i is size of blooms in bed i, W_i is water amount (*centered*) for bed i, S_i is sunlight amount (*centered*) for bed i, and $W_i S_i$ is the product of water and sunlight amounts for bed i.

```
# Continuous interactions
library(rethinking)
data(tulips)
d <- tulips

# standardize blooms to between 0 and 1
d$blooms_std <- d$blooms / max(d$blooms)

# center the predictors
d$water_cent <- d$water - mean(d$water)
d$shade_cent <- d$shade - mean(d$shade)
```

```
m8.5 <- quap(
  alist(
    blooms_std ~ dnorm( mu , sigma ) ,
    mu <- a + bw*water_cent + bs*shade_cent + bws*water_cent*shade_cent ,
    a ~ dnorm( 0.5 , 0.25 ) ,
    bw ~ dnorm( 0 , 0.25 ) ,
    bs ~ dnorm( 0 , 0.25 ) ,
    bws ~ dnorm( 0 , 0.25 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )
```

Priors:

Again, the scaled outcome and predictors constrain the priors in useful ways:

- **Intercept (a)**: defined as bloom size when water and shade are at their mean values; the model expects blooms to be halfway to the observed maximum; a standard deviation of 0.25 puts only 5% of mass outside the valid range (0 < a < 1):

  a ~ Normal(0.5 , 0.25)

- **Water Slope (b_W)**: centering on 0 indicates no bias for positive or negative; a standard deviation of 0.25 means 95% of prior slopes are from -0.5 to 0.5, which allows blooms from min (0) to max (1):

  b ~ Normal(0 , 0.25)

- **Sunlight Slope (b_S)**: centering on 0 indicates no bias for positive or negative; a standard deviation of 0.25 means 95% of prior slopes are from -0.5 to 0.5, which allows blooms from min (0) to max (1):

  b ~ Normal(0 , 0.25)

- **Interaction Slope (b_{WS})**: centering on 0 indicates no bias for positive or negative; a standard deviation of 0.25 allows the interaction to be the largest conceivable (equal, opposite magnitude as the main effect):

  b ~ Normal(0 , 0.25)

Step 1. Specify and check the priors

Model with interaction:

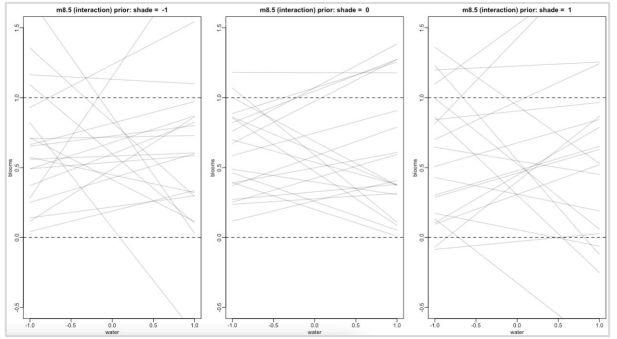
```
m8.5 <- quap(
  alist(
    blooms_std ~ dnorm( mu , sigma ) ,
    mu <- a + bw*water_cent + bs*shade_cent + bws*water_cent*shade_cent ,
    a ~ dnorm( 0.5 , 0.25 ) ,
    bw ~ dnorm( 0 , 0.25 ) ,
    bs ~ dnorm( 0 , 0.25 ) ,
    bws ~ dnorm( 0 , 0.25 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )
```

What can we say about these priors, overall?

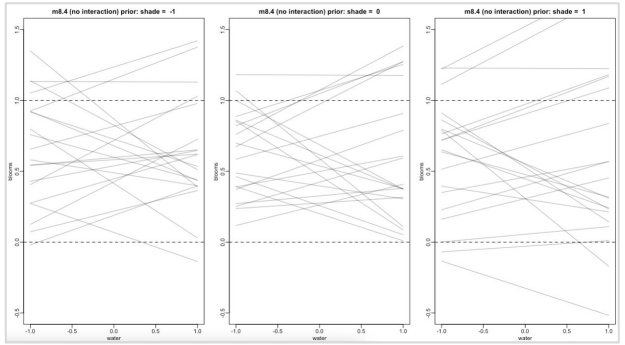
- The lines are very scattered in the prior, so the prior is not very informative
- They are harmless, but only weakly realistic
- Most of the lines stay within the valid outcome space, but silly trends are not rare

The priors contain no bias towards positive or negative effects, and at the same time they very weakly bound the effects to realistic ranges

Model with interaction



For comparison: Model without interaction



Step 2. Estimate the parameters

`rstanarm`

- The `stan_glm` function is similar in syntax to `glm` but rather than performing MLE, full Bayesian estimation is performed via MCMC. The Bayesian model adds priors (independent by default) on the coefficients of the GLM

rstanarm:

```
m8.5_rstanarm <- stan_glm(blooms_std ~ water_cent + shade_cent + water_cent:shade_cent,
                          data=d,
                          prior_intercept = normal( location = c(0.5) , scale = c(0.25) ) ,
                          prior = normal( location = c(0,0,0) , scale = c(0.25,0.25,0.25) ) ,
                          prior_aux = exponential( 1 ) )
```

`rethinking`

- `quap` uses quadratic approximation and the model definition to define the posterior probability at each combination of parameter values, climb the posterior distribution and find the peak, its maximum a posteriori (MAP), and estimate the quadratic curvature at the MAP to produce an approximation of the posterior distribution
- `ulam`, named after Stanisław Ulam, who was one of the parents of the Monte Carlo method and is the namesake of the Stan project as well, translates the model definition into a Stan model and returns an object with summary information and samples from the posterior distribution

rethinking:

```
m8.5 <- quap(
  alist(
    blooms_std ~ dnorm( mu , sigma ) ,
    mu <- a + bw*water_cent + bs*shade_cent + bws*water_cent*shade_cent ,
    a ~ dnorm( 0.5 , 0.25 ) ,
    bw ~ dnorm( 0 , 0.25 ) ,
    bs ~ dnorm( 0 , 0.25 ) ,
    bws ~ dnorm( 0 , 0.25 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )
```

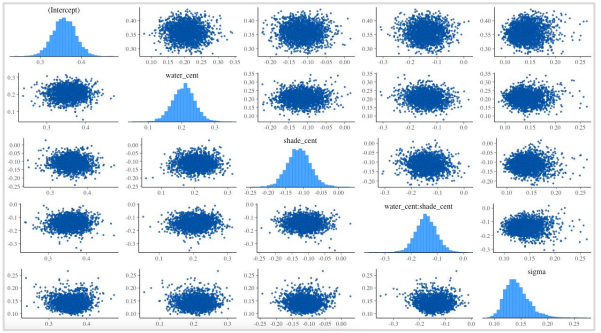
```
m8.5 <- ulam(
  alist(
    blooms_std ~ dnorm( mu , sigma ) ,
    mu <- a + bw*water_cent + bs*shade_cent + bws*water_cent*shade_cent ,
    a ~ dnorm( 0.5 , 0.25 ) ,
    bw ~ dnorm( 0 , 0.25 ) ,
    bs ~ dnorm( 0 , 0.25 ) ,
    bws ~ dnorm( 0 , 0.25 ) ,
    sigma ~ dexp( 1 )
  ) , data=d , chains=4 , cores=4 )
```

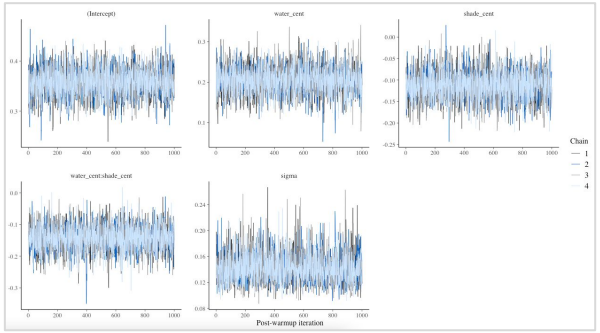
Step 3. Check sampling quality and model fit

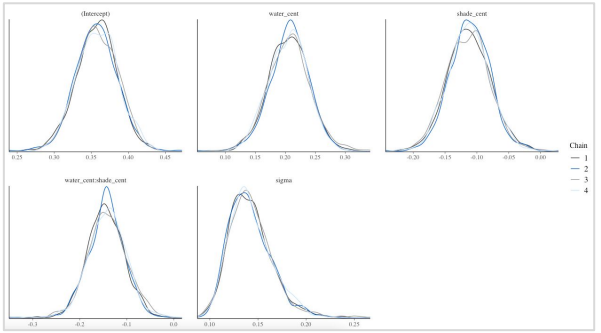
`rstanarm`

By default, all `rstanarm` modeling functions will run four randomly initialized Markov chains, each for 2000 iterations (including a warmup period of 1000 iterations that is discarded). All chains must converge to the target distribution for inferences to be valid

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      blooms_std ~ water_cent + shade_cent + water_cent:shade_cent
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 27
 predictors:   4

MCMC diagnostics
                      mcse Rhat n_eff
(Intercept)           0.00 1.00 3565
water_cent            0.00 1.00 4024
shade_cent            0.00 1.00 3425
water_cent:shade_cent 0.00 1.00 3552
sigma                 0.00 1.00 2819
mean_PPD              0.00 1.00 3960
log-posterior         0.04 1.00 1506
```







Step 4. Summarize and interpret results

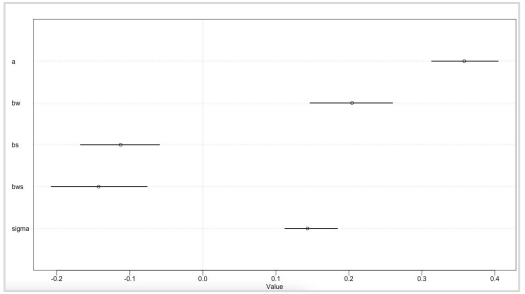
The fitted model is:

blooms = 0.36 + 0.2*water – 0.11*shade – 0.14*water*shade + error

Again, interpreting a table of coefficient estimates isn't ideal!
Let's complement them with plots!

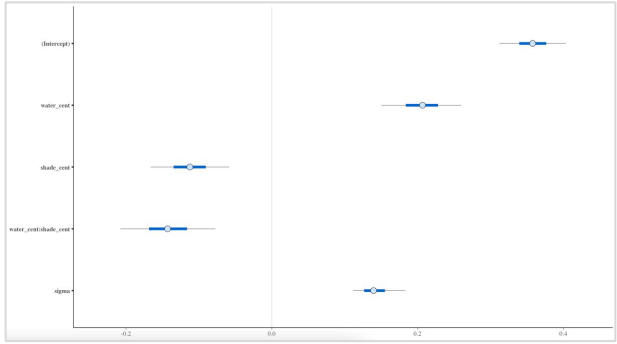
rethinking:

	mean	sd	5.5%	94.5%	n_eff	Rhat4
a	0.36	0.03	0.31	0.40	2115	1
bw	0.20	0.04	0.15	0.26	1748	1
bs	-0.11	0.03	-0.17	-0.06	2268	1
bws	-0.14	0.04	-0.21	-0.08	2428	1
sigma	0.14	0.02	0.11	0.18	1232	1



rstanarm:

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      blooms_std ~ water_cent + shade_cent + water_cent:shade_cent
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 27
 predictors:   4

Estimates:
                      mean   sd   10%   50%   90%
(Intercept)           0.36   0.03  0.32  0.36  0.39
water_cent            0.21   0.03  0.16  0.21  0.25
shade_cent           -0.11   0.03 -0.15 -0.11 -0.07
water_cent:shade_cent -0.14   0.04 -0.19 -0.14 -0.09
sigma                 0.14   0.02  0.12  0.14  0.17
```



Step 4. Summarize and interpret results

Now to interpret the model…

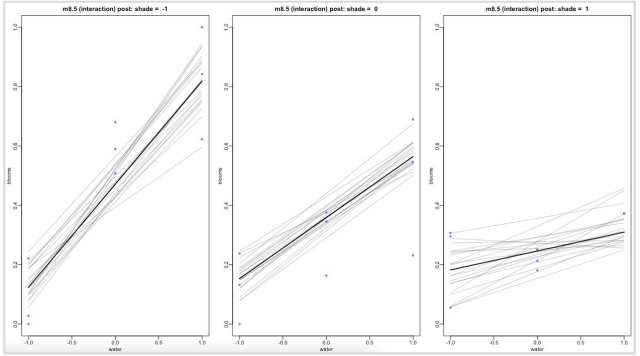
Top (model w/ interaction):

- The model believes water helps – there is a positive slope in each plot – and that shade hurts – the lines sink lower moving from left to right; however,
- The model believes that the effect of water decreases as shade increases. The lines get flat. What's going on?
- The likely explanation is that tulips need both water and light to produce blooms. At low light levels, water can't have much of an effect, because the tulips don't have enough light. At higher light levels, water can matter more, because the tulips have enough light to produce blooms. The same explanation works symmetrically for shade.

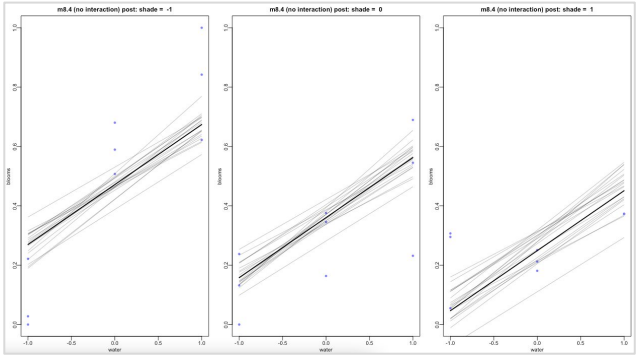
Bottom (model w/o interaction):

- The model believes water helps – there is a positive slope in each plot – and that shade hurts – the lines sink lower moving from left to right. But the slope with water doesn't vary across shade levels. Without the interaction, it cannot vary.

Model with interaction



For comparison: Model without interaction



Partial Effects
(“holding all others constant”)

What does it mean to “hold all other predictors constant”?

The fitted model is:

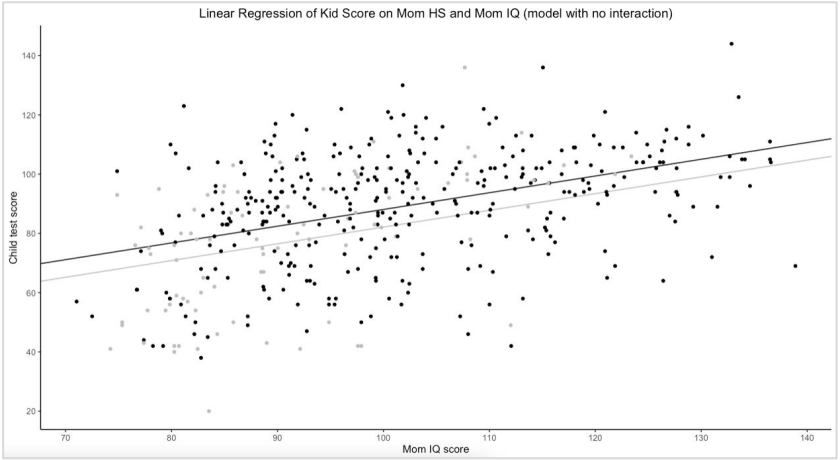
kid_score = 82 + 6 * mom_hs + 0.6 * mom_iq_centered + error

Understanding the fitted model:

Coefficient of maternal HS completion:
Comparing children whose mothers have the same IQ, but who differed in whether they completed HS, …

How do we estimate the relationship between X_1 and Y while mathematically adjusting for X_1’s correlations with X_2, X_3, … ?

Similarly, how do we estimate the relationship between X_2 and Y while mathematically adjusting for X_2’s correlations with X_1, X_3, … ?




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

Estimates:
                Median   MAD_SD   10%   50%   90%
(Intercept)     81.9      2.0     79.6  81.9  84.6
mom_hs           6.2      2.4      3.2   6.2   8.9
mom_iq_centered  0.6      0.1      0.5   0.6   0.6
sigma           18.2      0.6     17.4  18.2  19.1
```

What does it mean to “hold all other predictors constant”?

The original fitted model is:

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

1) Fit kid_score (Y) on mom_iq_centered (X2):

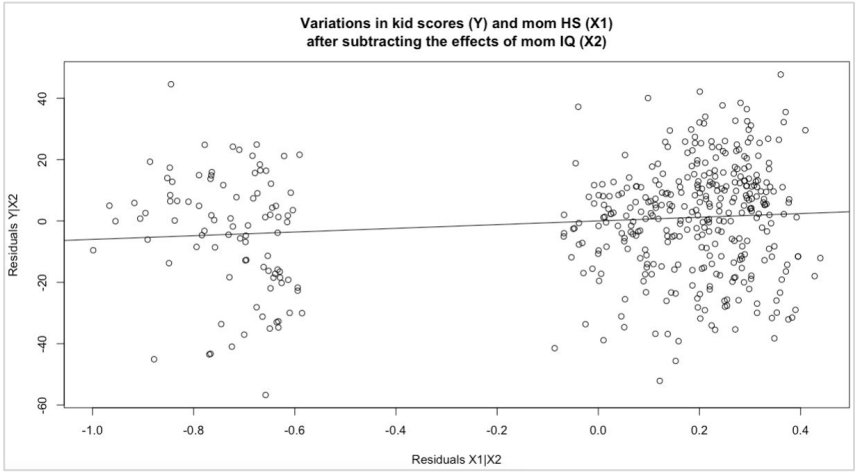
```
stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_iq_centered
 observations: 434
 predictors:   2
------
                Median MAD_SD
(Intercept)     86.8   0.8
mom_iq_centered  0.6   0.1
```

2) Fit mom_hs (X1) on mom_iq_centered (X2):

```
stan_glm
 family:       gaussian [identity]
 formula:      mom_hs ~ mom_iq_centered
 observations: 434
 predictors:   2
------
                Median MAD_SD
(Intercept)     0.8    0.0
mom_iq_centered 0.0    0.0
```

3) Fit residuals from 1 on residuals from 2:

```
stan_glm
 family:       gaussian [identity]
 formula:      y_x2_residuals ~ x1_x2_residuals
 observations: 434
 predictors:   2
------
                  Median MAD_SD
(Intercept)       0.0    0.9
x1_x2_residuals   6.0    2.2
```



The coefficient on x1_x2_residuals in 3 (b_1 = 6.0) is the same as the partial coefficient on X_1 in the original fitted model.

Thus b_1 describes the relation between Y and X_1, adjusting for relations both variables have with X_2.

What does it mean to "hold all other predictors constant"?

The original fitted model is:

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

1) Fit kid_score (Y) on mom_hs (X1):

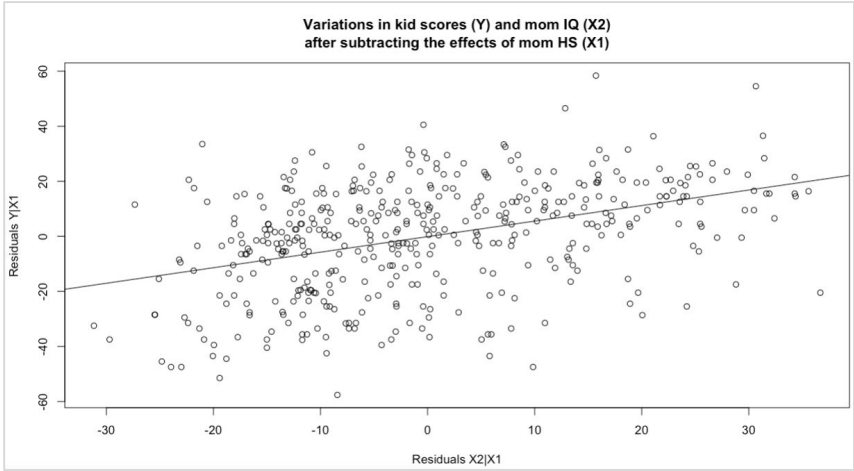
```
stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs
 observations: 434
 predictors:   2
------
            Median MAD_SD
(Intercept) 77.6    2.0
mom_hs      11.7    2.3
```

2) Fit mom_iq_centered (X2) on mom_hs (X1):

```
stan_glm
 family:       gaussian [identity]
 formula:      mom_iq_centered ~ mom_hs
 observations: 434
 predictors:   2
------
            Median MAD_SD
(Intercept) -7.9    1.6
mom_hs      10.2    1.8
```

3) Fit residuals from 1 on residuals from 2:

```
stan_glm
 family:       gaussian [identity]
 formula:      y_x1_residuals ~ x2_x1_residuals
 observations: 434
 predictors:   2
------
                 Median MAD_SD
(Intercept)      -0.1   0.9
x2_x1_residuals  0.6    0.1
```



The coefficient on x2_x1_residuals in 3 (b_1 = 0.6) is the same as the partial coefficient on X_2 in the original fitted model.

Thus b_1 describes the relation between Y and X_2, adjusting for relations both variables have with X_1.

What does it mean to “hold all other predictors constant”?

More generally, a partial regression leverage plot for variables Y and X_k depicts a regression in which:

- The “Y” variable equals the residual from the regression of Y on all X variables except X_k
- The “X” variable equals the residual from the regression of X_k on all X variables except X_k

The line in a leverage plot always has an intercept of zero

Its slope equals b_k, the partial coefficient on X_k from the regression of Y on all X variables including X_k

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