Lecture Notes: Causal Inference Fall 2020

Claire Duquennois

7/28/2020

Contents

1 The Hodgepodge

1.1 Regression Interpretation and Variable Transformations

Being able to generate regression results is one thing. Equally important is being able to interpret them correctly. When interpreting regression results, there are typically 3 elements you want to touch upon (the three S'):

1) **Sign**- is the coefficient you are discussing positive or negative? Does the sign of the coefficient match your priors or is it surprising?

2) **Size**- What is the magnitude of the coefficient? Is the effect of x on y economically meaningful or not? While it is not incorrect to say the that a one unit increase in x leads to a β unit change in y, in order to be able to make your interpretation informative to your audience, you will generally want to be more precise:

   - specify the units of measurement
   - interpret the coefficients appropriately: if the regression used a logged variable, a binairy dependent variable, a standardized variable...
   - it may be useful to scale the values into more intuitive units of measurement.
   - If the regression features a polynomial discuss the difference in the magnitude of the marginal effects at different key values.
   - If the regression features interaction terms, interpret the implications for different types of observations
   - it is often informative to compare the coefficient to the mean and standard deviation of the dependent variable to give your audience a point of reference with which to gauge magnitudes

3) **Significance**- Is the estimate statistically significant? Can we reject that the true coefficient is equal to zero? With what confidence level?

1.1.1 Scaling

Sometimes you may find that the units of measurement used for some of the variables in the data you are working with are not very intuitive.

Suppose I am interested in the following regression:

$$sleep = \beta_0 + \beta_1 totwork + \beta_2 educ + u$$

Using the `sleep75` data that is part of the `wooldrige` package, I run the estimation and get the following

```
regsleep1 <- lm(sleep ~ totwrk + educ, sleep75)

summary(regsleep1)
```

```
##
## Call:
## lm(formula = sleep ~ totwrk + educ, data = sleep75)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2441.18  -235.18    18.13   259.76  1329.94
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3756.21512   81.28699  46.209   <2e-16 ***
## totwrk        -0.14945    0.01669  -8.952   <2e-16 ***
## educ         -13.50385    5.68001  -2.377   0.0177 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 419.8 on 703 degrees of freedom
## Multiple R-squared:  0.1104, Adjusted R-squared:  0.1079
## F-statistic: 43.64 on 2 and 703 DF,  p-value: < 2.2e-16
```

What do these coefficients mean? To answer this I need to know the units of measurement for each of the variables. *educ* is measured in years of education, which is fairly intuitive but *sleep* and *totwrk* are measured in minutes per week. Thus, **one additional year of education is associated with 13.5 fewer minutes of sleep per week**. This is probably not the most intuitive interpretation for most audiences. Hours per week would probably be a more natural way to report this.

1.1.1.1 Scaling the dependent variable

One way to do this is to simply make the adjustment in our interpretation. 13.5 min per week is equivalent to $\frac{13.5}{60} = 0.23 \approx 1/4$ hour per week. So we could just rewrite all of our interpretations in terms of hours by dividing the old coefficients by 60:

$$\frac{\hat{\beta}_0}{60}; \frac{\hat{\beta}_1}{60}; \frac{\hat{\beta}_2}{60}.$$

Alternatively, it is often just easier to run the regression after having scaled our *sleep* variable in terms of hours (i.e. dividing the dependent variable by 60 and re-running the regression) as follows:

```
sleep75$sleephrs <- sleep75$sleep/60

regsleep2 <- lm(sleephrs ~ totwrk + educ, sleep75)
```

```
summary(regsleep2)
```

```
##
## Call:
## lm(formula = sleephrs ~ totwrk + educ, data = sleep75)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.686  -3.920   0.302   4.329  22.166
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 62.6035854  1.3547832  46.209   <2e-16 ***
## totwrk      -0.0024909  0.0002782  -8.952   <2e-16 ***
## educ        -0.2250641  0.0946669  -2.377   0.0177 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.996 on 703 degrees of freedom
## Multiple R-squared:  0.1104, Adjusted R-squared:  0.1079
## F-statistic: 43.64 on 2 and 703 DF,  p-value: < 2.2e-16
```

In general, if we re-scale the dependent variable y by a constant c, then the equation we estimate becomes

$$\tilde{y} = \tilde{\beta}_0 + \tilde{\beta}_1 x_1 + ... + \tilde{\beta}_k x_k + u$$
$$cy = c\beta_0 + c\beta_1 x_1 + c\beta_2 x2 + ... + c\beta_k x_k + u.$$

In the example above, $c = \frac{1}{60}$, so the new $\hat{\beta}$'s will be divided by 60 too. Nothing else about the regression changes (R^2, t-stats, p-values).

1.1.1.2 Scaling the independent variable

Things are slightly different is we are scaling an independent variable. Our estimates above say that a 1 minute increase in total work per week predicts a decrease in sleep of 0.0025 hours per week. This is clearly equivalent to saying that a one hour increase in total work predicts a 60*(0.0025)=0.15 hour decrease in sleep per week (i.e. we can multiply our $\hat{\beta}$ estimate by 60).

As above, it often makes sense to scale the variable in R prior to running the regression:

```
sleep75$totwrkhrs <- sleep75$totwrk/60

regsleep3 <- lm(sleephrs ~ totwrkhrs + educ, sleep75)

summary(regsleep3)
```

```
##
## Call:
## lm(formula = sleephrs ~ totwrkhrs + educ, data = sleep75)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.686  -3.920   0.302   4.329  22.166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 62.60359    1.35478  46.209   <2e-16 ***
## totwrkhrs   -0.14945    0.01669  -8.952   <2e-16 ***
## educ        -0.22506    0.09467  -2.377   0.0177 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.996 on 703 degrees of freedom
## Multiple R-squared:  0.1104, Adjusted R-squared:  0.1079
## F-statistic: 43.64 on 2 and 703 DF,  p-value: < 2.2e-16
```

When scaling an independent variable, the other coefficients will be unchanged, but the coefficient on the scaled variable will adjust accordingly.

In general, if we scale x by c, the equation becomes:

$$\begin{aligned} y &= \beta_0 + \tilde{\beta}_1 \tilde{x}_1 + ... + \beta_k x_k + u \\ &= \beta_0 + \frac{\tilde{\beta}_1}{c}(c\tilde{x}_1) + ... + \beta_k x_k + u. \end{aligned}$$

1.1.2 Standardizing

Standardizing variables eliminates the units in order to be able to compare the magnitude of estimates across independent variables. This can make interpretation easier if you have variables with weird arbitrary units that are unfamiliar to people. We can solve this issue by standardizing the variables.

Suppose we have a regression with two variables, x_1 and x_2:

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{u}$$

We know that our regression must go through the point of averages: if we plugged in $\bar{x}_1$ and $\bar{x}_2$, we would predict $\bar{y}$:

$$\bar{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}_1 + \hat{\beta}_2 \bar{x}_2$$

We can subtract the second equation from the first to get:

$$\begin{aligned} \hat{y} - \bar{y} &= (\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{u}) - (\hat{\beta}_0 + \hat{\beta}_1 \bar{x}_1 + \hat{\beta}_2 \bar{x}_2) \\ &= \hat{\beta}_1 (x_1 - \bar{x}_1) + \hat{\beta}_2 (x_2 - \bar{x}_2) + \hat{u} \end{aligned}$$

With a little bit of additional algebra, dividing both sides of this equation by the standard deviation of y, σ_y and multiplying each independent variable by $1 = \frac{\sigma_x}{\sigma_x}$. we can get the entire regression into standard units:

$$(\frac{y - \bar{y}}{\hat{\sigma}_y}) = \frac{\hat{\sigma}_{x_1}}{\hat{\sigma}_y} \hat{\beta}_1 (\frac{x_1 - \bar{x}_1}{\hat{\sigma}_{x_1}}) + \frac{\hat{\sigma}_{x_2}}{\hat{\sigma}_y} \hat{\beta}_2 (\frac{x_2 - \bar{x}_2}{\hat{\sigma}_{x_2}}) + \frac{\hat{u}}{\hat{\sigma}_y}$$

Now we can say that controlling for x_2 a one standard deviation increase in x_1 leads to a $\frac{\hat{\sigma}_{x_1}}{\hat{\sigma}_y} \hat{\beta}_1$ standard deviation increase in the predicted y. We call this new term the standardized coefficient or "beta" coefficient.

We use the bwght2 dataset to look at how parent ages correlate with birth weights. (Note: birth weights here will be measured in grams). I estimate four different regressions of the type

$$birthweight_i = \beta_0 + \beta_1 motherage_i + \beta_2 fatherage_i + \epsilon_i$$

scaling either the dependent and/or independent variables.

```r
reg1 <- lm(bwght ~ mage + fage, bwght2)
reg2 <- lm(scale(bwght) ~ scale(mage) + scale(fage), bwght2)
reg3 <- lm(scale(bwght) ~ mage + fage, bwght2)
reg4 <- lm(bwght ~ scale(mage) + scale(fage), bwght2)

meandep1 <- round(mean(bwght2$bwght), 2)
meandep2 <- round(mean(scale(bwght2$bwght)), 2)
sddep1 <- round(sd(bwght2$bwght), 2)
sddep2 <- round(sd(scale(bwght2$bwght)), 2)

stargazer(reg1, reg2, reg3, reg4, type = "latex", header = FALSE, add.lines = list(c("Mean",
    meandep1, meandep2, meandep2, meandep1), c("SD", sddep1, sddep2, sddep2, sddep1)))
```

Table 1:

	Dependent variable:			
	bwght	scale(bwght)		bwght
	(1)	(2)	(3)	(4)
mage	−3.992		−0.007	
	(3.943)		(0.007)	
fage	9.313***		0.016***	
	(3.291)		(0.006)	
scale(mage)		−0.033		−19.044
		(0.033)		(18.812)
scale(fage)		0.092***		53.205***
		(0.033)		(18.803)
Constant	3,221.030***	−0.001	−0.312**	3,400.304***
	(87.703)	(0.023)	(0.152)	(13.448)
Mean	3401.12	0	0	3401.12
SD	576.54	1	1	576.54
Observations	1,826	1,826	1,826	1,826
R^2	0.005	0.005	0.005	0.005
Adjusted R^2	0.004	0.004	0.004	0.004
Residual Std. Error (df = 1823)	574.677	0.997	0.997	574.677
F Statistic (df = 2; 1823)	4.912***	4.912***	4.912***	4.912***
Note:				*p<0.1; **p<0.05; ***p<0.01

Interpreting column 1:

- Having a mother that is **a year** older at birth predicts a birthweight that is 3.992 **grams** less (not statistically significant).
- Having a father that is **a year** older at birth predicts a birthweight that is 9.313 **grams** more (highly statistically significant).

Interpreting column 2:

- Having a mother whose age is **one standard deviation higher** at birth predicts a birthweight that

is 0.033 **standard deviations** lower (not statistically significant).

- Having a father whose age is **one standard deviation higher** at birth predicts a birthweight that is 0.092 **standard deviations** higher (highly statistically significant).

Interpreting column 3:

- Having a mother that is **a year** older at birth predicts a birthweight that is 0.007 **standard deviations** lower (not statistically significant).
- Having a father that is **a year** older at birth predict a birthweight that is 0.016 **standard deviations** higher (highly statistically significant).

Interpreting column 4:

- Having a mother whose age is **one standard deviation higher** at birth predicts a birthweight that is 19.044 **grams** lower (not statistically significant).
- Having a father whose age is **one standard deviation higher** at birth predicts a birthweight that is 53.205 **grams** higher (highly statistically significant).

Notes:

- You can also standardize y, x_1 and x_2 directly in the data set and then run your regression on the standardized variables though this involves more coding.
- You do not need to standardize all the variables. You could just standardize x_1 and adjust your interpretation accordingly (you will need an intercept in this case).

1.1.3 Logs

Choosing an appropriate functional form is a critical choice in economic modeling. Using log's to transform a variable can greatly improve the fit of your model by making the underlying relationship between the dependent and independent variables linear. Of course, when you do this, you need to then adjust your interpretation accordingly.

1.1.3.1 Linear functions and unit-unit changes

If we assume a linear functional form, the model is:

$$y = \beta_0 + \beta_1 x.$$

Interpretation: First take the derivative of the expression to get: $\frac{dy}{dx} = \beta_1$. Now we can rewrite as:

$$\frac{\Delta y}{\Delta x} = \frac{dy}{dx} = \beta_1.$$

Rearranging we see that

$$\Delta y = \beta_1 \Delta x.$$

Suppose $\Delta x = 1$, so that x changes by 1 **unit**. Plugging into the above expression we see that y will change by β_1 **units**.

1.1.3.2 Logarithmic functions and percent-unit changes

If we assume a logarithmic functional form, the model is:

$$y = \beta_0 + \beta_1 log(x).$$

Interpretation: First take the derivative of our model. Recall: if $w = log(v)$ then $\frac{dw}{dv} = \frac{1}{v}$. Thus for our model: $\frac{dy}{dx} = \frac{\beta_1}{x}$ which, for small changes in x can be approximately rewritten as:

$$\frac{\Delta y}{\Delta x} \approx \frac{dy}{dx} = \frac{\beta_1}{x}.$$

Rearranging we see that

$$\Delta y = \beta_1 \frac{\Delta x}{x}.$$

Suppose we know that x changes by 10 percent, so that the proportional change in x is $0.1 \Rightarrow \frac{\Delta x}{x} = 0.1$. Plugging into the expression above we see that y will change by $\beta_1 * 0.1$ **units**.

1.1.3.3 Exponential functions and Unit-Percent changes

If we assume an exponential functional form, the model is: $y = e^{\beta_0 + \beta_1 x}$ or

$$log(y) = \beta_0 + \beta_1 x.$$

Interpretation: Once again, we take a derivative of our model with respect to x and find $\frac{dlog(y)}{dx} = \beta_1$. For small changes we can rewrite this as

$$\frac{\Delta log(y)}{\Delta x} \approx \frac{dlog(y)}{dx} = \beta_1.$$

Using the fact that $\Delta log(y) = \frac{\Delta y}{y}$:

$$\frac{\frac{\Delta y}{y}}{\Delta x} \approx \frac{dlog(y)}{dx} = \beta_1,$$

we can rearrange and see that

$$\frac{\Delta y}{y} = \beta_1 \Delta x \Rightarrow \underbrace{\frac{\Delta y}{y} \times 100}_{\text{percent change}} = (\beta_1 \Delta x) \times 100$$

Suppose x changes by 1 **unit**. Plugging this into the expression derived above, we see that the proportional change in y is β_1 and the percent change in y is $100 \times \beta_1$ percent.

1.1.3.4 Log-Log functions and Percent-Percent changes (eg elasticities)

If we assume a log-log functional form, the model is

$$log(y) = \beta_0 + \beta_1 log(x).$$

Interpretation: As usual, start by taking the derivative of our model, $\frac{dlog(y)}{dx} = \beta_1 \frac{1}{x}$. Re-writing it in terms of small changes we get:

$$\begin{aligned} \frac{\Delta log(y)}{\Delta x} \approx \frac{dlog(y)}{dx} &= \beta_1(\frac{1}{x}) \\ \Rightarrow \Delta log(y) &= \beta_1(\frac{\Delta x}{x}) \\ \Rightarrow \frac{\Delta y}{y} &= \beta_1(\frac{\Delta x}{x}) \\ \Rightarrow \frac{\Delta y}{y} \times 100 &= \beta_1(\frac{\Delta x}{x}) \times 100 \end{aligned}$$

Suppose we know that x changes by 1 **percent**. Plug the value into this expression and we see that y will change by β_1 **percent**.

1.1.3.5 Summary:

Model	DepVar	IndepVar	How does Δy relate to Δx?	Interpretation
Linear	y	x	$\Delta y = \beta_1 \Delta x$	$\Delta y = \beta_1 \Delta x$
Logarithmic	y	log(x)	$\Delta y = \beta_1 \frac{\Delta x}{x}$	$\Delta y = \beta_1 \frac{\%\Delta x}{100}$
Exponential	log(y)	x	$\frac{\Delta y}{y} = \beta_1 \Delta x$	$\%\Delta y = \beta_1 \Delta x \times 100$
Log-log	log(y)	log(x)	$\frac{\Delta y}{y} = \beta_1 \frac{\Delta x}{x}$	$\%\Delta y = \beta_1 \%\Delta x$

1.1.3.6 Practice:

Example 1: You have data on gas consumption and prices, you estimate the following model

$$log(gas) = 12 - 0.21 price$$

How does gas consumption change when price increases by 1 dollar?

$$\frac{\Delta y}{y} = \beta_1 \Delta x \Rightarrow \frac{\Delta y}{y} = (-0.21) \times 1 = -0.21 \Rightarrow \%\Delta y = -21\%$$

Example 2: You have data on corn and beef prices, you estimate the following model

$$log(P_{beef}) = 0.83 + 0.491 log(P_{corn})$$

How does P_{beef} change is P_{corn} rises by 2%?

$$\frac{\Delta y}{y} = \beta_1 \frac{\Delta x}{x} = 0.49 \times 0.02 = 0.00982 \Rightarrow +0.982\%$$

Example 3: You have data on CEO salaries (in hundred thousand dollars) and annual firm sales (millions of dollars). You estimate:

$$salary = 2.23 + 1.1 log(sales)$$

How does salary change if annual sales increase by 10%?

$$\Delta y = \beta_1 \frac{\Delta x}{x} = 1.1 \times 0.1 = 0.11$$

If sales increase by 10%, CEO salaries are predicted to increase by 0.11 (hundred thousand)= 11,000 dollars.

1.1.4 Quadratics

When interpreting a variable that includes a quadratic (or higher order) polynomial, it is important to recognize that the marginal effect of the variable is not linear, and include this in your interpretation.

Suppose we are interested in the relationship between age and sleep. You estimate the following model using the `sleep75` data from the `wooldridge` package:

$$sleep = \beta_0 + \beta_1 age + \beta_2 age^2 + u$$

This model allows for the marginal effect of age on sleep to change, along a quadratic functional form. To see this, take the derivative of sleep with respect to age:

$$\frac{dsleep}{dage} = \beta_1 + 2 * \beta_2 * age.$$

We see that the predicted effect of age on sleep will depend on an observation's age. When interpreting, the marginal effect in this case, it is often informative to specify the age at which you are interpreting at, and give a sense of the marginal effects at different key points of the age distribution.

Retrieving the coefficients, we get:

```
sleep75$age2 <- sleep75$age * sleep75$age
regquad <- lm(sleep ~ age + age2, sleep75)

summary(regquad)
```

```
##
## Call:
## lm(formula = sleep ~ age + age2, data = sleep75)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2518.1  -250.4     2.6   276.8  1390.0
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3608.0297   230.6457  15.643   <2e-16 ***
## age          -21.4904    11.7367  -1.831   0.0675 .
## age2           0.3012     0.1401   2.150   0.0319 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 441.8 on 703 degrees of freedom
## Multiple R-squared:  0.01464,    Adjusted R-squared:  0.01184
## F-statistic: 5.224 on 2 and 703 DF,  p-value: 0.005598
```

thus

$$\frac{dsleep}{dage} = -21.5 + 0.6 * age.$$

If you are currently 22 years old, our model predicts that a year from now you will be getting 8.3 fewer minutes per week of sleep than you are currently getting.

If you are currently 60, our model predicts that a year from now you will be getting 14.5 more minutes per week of sleep than you are currently getting.

In fact, age is predicted to decrease sleep until a person is 35.8 years old at which point an additional year is associated with an increase in sleep (where $\frac{dsleep}{dage} = 0$).

It is common to interpret non-constant marginal effects at the mean or median value of the explanatory variable, or at any value that would be of particular interest to your audience.

1.1.5 Interactions with continuous variables

When interpreting interactions with continuous variables, it is important to recognize that the marginal effect of the variable are not linear, and, (as with interpretations of quadratics) include this in your interpretation.

Suppose we are interested in the relationship between age, education and sleep. You estimate the following model using the `sleep75` data from the `wooldridge` package:

$$sleep = \beta_0 + \beta_1 age + \beta_2 educ + \beta_3 age \times educ + u$$

This model allows for the marginal effect of age and education on sleep to be a function of one another. To see this, take the derivative of sleep with respect to age:

$$\frac{dsleep}{dage} = \beta_1 + \beta_3 * educ.$$

How age predicts sleeps depends on an observation's education level.

Similarly, take the derivative of sleep with respect to education:

$$\frac{dsleep}{deduc} = \beta_2 + \beta_3 * age.$$

How education predicts sleeps depends on an observation's age.

Retrieving the coefficients, we get:

```
reginter <- lm(sleep ~ age + educ + age * educ, sleep75)

summary(reginter)
```

```
##
## Call:
## lm(formula = sleep ~ age + educ + age * educ, data = sleep75)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2446.38  -242.45     8.67   268.60  1387.11
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2517.3082   293.0368   8.590  < 2e-16 ***
## age           21.2596     6.4662   3.288  0.00106 **
## educ          49.9675    21.9876   2.273  0.02336 *
## age:educ      -1.4656     0.4976  -2.945  0.00333 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 439.6 on 702 degrees of freedom
## Multiple R-squared:  0.02563,    Adjusted R-squared:  0.02146
## F-statistic: 6.155 on 3 and 702 DF,  p-value: 0.0003928
```

thus

$$\frac{dsleep}{dage} = 12.26 - 1.466 * educ$$

and

$$\frac{dsleep}{deduc} = 49.97 - 1.466 * age.$$

If you are interested in predicting the effect of an additional year of education on sleep, you will need to specify the age of the observation in question: we get a prediction of 17.7 minutes per week more sleep if you are 22 years old but 38 minutes per week less sleep if you are 60.

In fact, more education is associated with more sleep until 34 years of age (where $\frac{dsleep}{deduc} = 0$), at which point it becomes associated with less sleep.

It is common to interpret non-constant marginal effects at the mean or median value of the explanatory variable, or at any value that would be of particular interest to your audience. Graphing them is also an option.

1.2 Binary Dependent variables

Most of the models we have seen have had continuous dependent variables: the y variable was some quantitative amount that took on a range of possible values (a test score, a weight,...). Sometimes though y can be a dummy variable, taking on either the value 1 or 0. What changes when we have a binary variable on the left hand side?

1.2.1 Linear Probability Models

A linear probability model is simply a linear regression with a binary variable as the dependent variable y:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u$$

- It no longer makes sense to interpret β_j as the unit change in y given a one-unit increase in x_j holding all other factors fixed. Indeed, y either changes from $0 \rightarrow 1$, from $1 \rightarrow 0$ or doesn't change.
- The β_j's still have a useful interpretation though: each β_j measures the change in the probability of success when x_j changes by one unit holding all other factors constant.

$$Pr(y = 1|x) = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k$$

1.2.1.1 Example:/ Let $inlf$ ("in the labor force") be a binary variable indicating labor force participation by a married woman in 1975. $inlf = 1$ if the woman reports working for a wage outside the home at some point during the year, and zero otherwise. Our independent variables might include:

- husbands earnings ($nwifeinc$, measured in thousands of dollars)
- years of education ($educ$)
- past years of labor market experience ($exper$)
- age (age)
- number of children less than six years old ($kidslt6$)
- number of kids between 6 and 18 years of age ($kidsge6$)

We estimate the following model using the `mroz` data that is part of the `wooldridge` package:

$$inlf = \beta_0 + \beta_1 nwifeinc + \beta_2 educ + \beta_3 exper + \beta_4 exper^2 + \beta_5 age + \beta_6 kidslt6 + \beta_7 kidsge6 + u$$

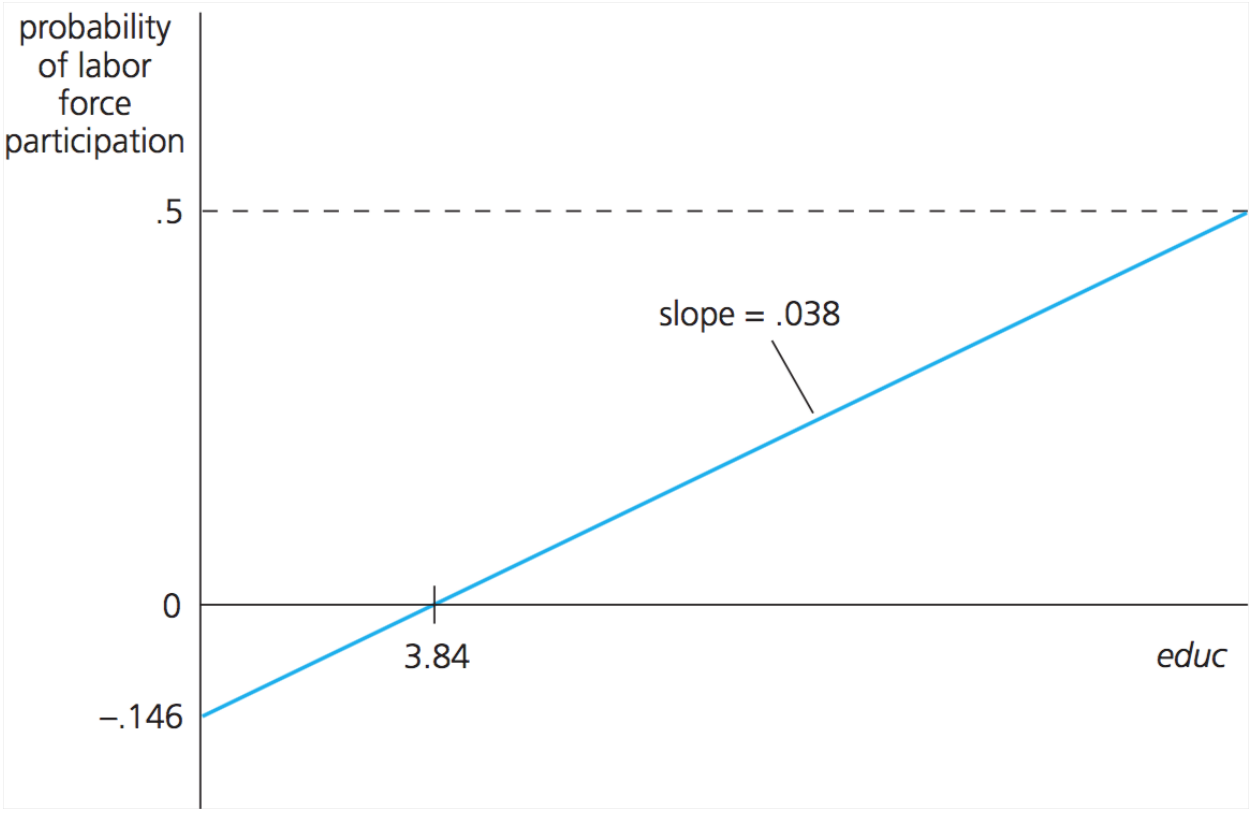
```
mroz$exper2 <- mroz$exper^2
reg1 <- lm(inlf ~ nwifeinc + educ + exper + exper2 + age + kidslt6 + kidsge6, mroz)
summary(reg1)
```

```
##
## Call:
## lm(formula = inlf ~ nwifeinc + educ + exper + exper2 + age +
##     kidslt6 + kidsge6, data = mroz)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93432 -0.37526  0.08833  0.34404  0.99417
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5855192  0.1541780   3.798 0.000158 ***
## nwifeinc    -0.0034052  0.0014485  -2.351 0.018991 *
```

```
## educ         0.0379953  0.0073760   5.151 3.32e-07 ***
## exper        0.0394924  0.0056727   6.962 7.38e-12 ***
## exper2      -0.0005963  0.0001848  -3.227 0.001306 **
## age         -0.0160908  0.0024847  -6.476 1.71e-10 ***
## kidslt6     -0.2618105  0.0335058  -7.814 1.89e-14 ***
## kidsge6      0.0130122  0.0131960   0.986 0.324415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4271 on 745 degrees of freedom
## Multiple R-squared:  0.2642, Adjusted R-squared:  0.2573
## F-statistic: 38.22 on 7 and 745 DF,  p-value: < 2.2e-16
```

As an example, given that this is a linear probability model, we would interpret the coefficient on *educ* as: holding all else fixed, an additional year of education increases predicted probability of labor force participation by 0.038, or 3.8 percentage points.

The graph below depicts the probability of labor force participation and *educ*. The other independent variables, for the purposes of this example, are fixed at the values $nwifeinc = 50$, $exper = 5$, $age = 30$, $kidslt6 = 1$ and $kidsge6 = 0$.



For a woman with these characteristics, the relationship between years of education and the probability of being in the labor force is given by:

$$\begin{aligned} Pr(inlf = 1|educ) &= (0.585 + (-0.003) * 50 + (0.039) * 5 + (-0.001) * 5^2 + (-0.016) * 30 + (-0.262) * 1) + 0.038 * educ \\ &= -0.146 + 0.038 educ \end{aligned}$$

1.2.1.2 Linear Probability Model Drawbacks:

1) Because this is a linear regression, we are trying to fit a line to the data. This means that we can get predicted probabilities that are negative or greater than one. We can see here that the predicted probability of labor force participation for women with education that is below 3.84 years is negative. Similarly, if a woman has 31 years of education it would be greater than one. This isn't too much of a concern here because in the sample few women will fall in those ranges. Nevertheless, the predicted probabilities from our regression aren't bound between 0 and 1.

2) A related problem: a linear probability model implies that the outcome is linearly related to the independent variables. This is not possible with probabilities. In the previous example, an additional child reduces the probability of working by 0.262. Thus our model implies that going from 0 to 4 young children reduces the probability by 0.262*4=1.048, 104.8 percentage points, which is impossible.

3) This model uses the idea that the probability that the dummy is equal to one is actually a function of our x's, which means the variance of the dummy is a function of our x's. Indeed, when y is a binary variable

$$Var(y|x) = p(x)[1 - p(x)]$$

where $p(x)$ is a shorthand for the probability of success $p(x) = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k$. This means that there must be heteroskedasticity in the linear probability model. Thus you should always use **heteroskedasticity robust standard errors** with linear probability models.

1.2.2 Logits

How do we address the drawbacks of linear probability models? With **logistic regressions**, which are estimated via maximum likelihood methods rather than OLS.

Logits differ from the linear probability model in what kind of function is used for the estimated probability.

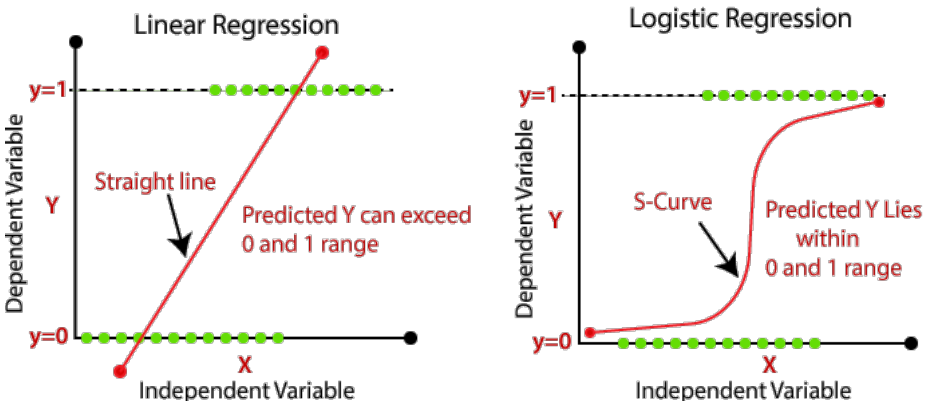
In linear probability models, we have

$$Pr(y = 1|x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

With a logit, we have

$$Pr(y = 1|x_1, x_2) = \frac{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}}{1 + e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}} = \Lambda(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$$

where Λ is commonly used notation to represent the complex logit function.



Why would we ever pick a functional form for the probability that looks as complicated as Λ? Because it is bounded between 0 and 1 (so our predictions make sense), and it has nice statistical properties (which we will not get into).

One important thing to note about the estimation of logit models is that R output will not give you the marginal effect of one more unit of x_j on the probability of y taking the value 1 ($\frac{\partial Pr(y=1|x)}{\partial x_j}$). Instead it reports the **log-odds** by default, which are not the same as marginal effects!

To get marginal effects, we need to do a couple more steps. This is because these models are **non-linear**, such that the exact marginal effect of x_j on y **changes depending on the other values of x**. Therefore we have a bunch of options for where we calculate the marginal effects at. In R, the default is to compute the **average marginal effect** which is the average of the marginal effect for each observation in the sample. In R we can also specify any values of our independent variables to evaluate our marginal effects precisely at those values. Note that this is in contrast to the constant marginal effect of x_j on $Pr(y = 1|x)$ found in the linear probability model, and is another potential benefit of running a logit model.

1.2.2.1 Example:/

I return to the previous example, looking at labor force participation using the `mroz` data.

In R we can estimate a logistic regression with the `glm()` function. Running `summary()` will then get us the log-odds.

```
reglogit1 <- glm(inlf ~ nwifeinc + educ + exper + exper2 + age + kidslt6 + kidsge6,
    mroz, family = "binomial")
summary(reglogit1)
```

```
##
## Call:
## glm(formula = inlf ~ nwifeinc + educ + exper + exper2 + age +
##     kidslt6 + kidsge6, family = "binomial", data = mroz)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1770  -0.9063   0.4473   0.8561   2.4032
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.425452   0.860365   0.495  0.62095
## nwifeinc    -0.021345   0.008421  -2.535  0.01126 *
## educ         0.221170   0.043439   5.091 3.55e-07 ***
## exper        0.205870   0.032057   6.422 1.34e-10 ***
## exper2      -0.003154   0.001016  -3.104  0.00191 **
## age         -0.088024   0.014573  -6.040 1.54e-09 ***
## kidslt6     -1.443354   0.203583  -7.090 1.34e-12 ***
## kidsge6      0.060112   0.074789   0.804  0.42154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  803.53  on 745  degrees of freedom
## AIC: 819.53
##
## Number of Fisher Scoring iterations: 4
```

To get the marginal effects, we need to run the estimated `glm()` object through the `margins()` function from the **margins** package.

```
library(margins)
marg_reglogit1 <- margins(reglogit1)

summary(marg_reglogit1)
```

```
##    factor     AME     SE       z      p   lower   upper
##       age -0.0157 0.0024 -6.6027 0.0000 -0.0204 -0.0111
##      educ  0.0395 0.0073  5.4145 0.0000  0.0252  0.0538
##     exper  0.0368 0.0052  7.1386 0.0000  0.0267  0.0469
##    exper2 -0.0006 0.0002 -3.1759 0.0015 -0.0009 -0.0002
##   kidsge6  0.0107 0.0133  0.8051 0.4207 -0.0154  0.0369
##   kidslt6 -0.2578 0.0319 -8.0696 0.0000 -0.3204 -0.1951
##  nwifeinc -0.0038 0.0015 -2.5714 0.0101 -0.0067 -0.0009
```

Recall that by default this computes the average marginal effect for the sample. To instead compute the marginal effect for a particular type of observation

```
DF <- data.frame(age = 30, nwifeinc = 50, exper = 5, exper2 = 25, kidsge6 = 0, kidslt6 = 1,
    educ = 12, stringsAsFactors = FALSE)

marg_specific <- margins(reglogit1, data = DF)

summary(marg_specific)
```

```
##    factor     AME     SE       z      p   lower   upper
##       age -0.0163 0.0049 -3.3268 0.0009 -0.0259 -0.0067
##      educ  0.0410 0.0088  4.6729 0.0000  0.0238  0.0582
##     exper  0.0382 0.0089  4.2892 0.0000  0.0207  0.0556
##    exper2 -0.0006 0.0002 -2.8204 0.0048 -0.0010 -0.0002
##   kidsge6  0.0111 0.0130  0.8554 0.3924 -0.0144  0.0367
##   kidslt6 -0.2675 0.0567 -4.7195 0.0000 -0.3787 -0.1564
##  nwifeinc -0.0040 0.0011 -3.5885 0.0003 -0.0061 -0.0018
```

1.2.3 Example contrasting a logit to a linear probability model (LPM)

In sports betting, the Las Vegas point spread is the predicted scoring differential between two opponents as quoted by a sports book in Las Vegas. We are interested in the probability that the favored team actually wins.

We can run the following regression:

$$\widehat{favwin}_i = \hat{\beta}_0 + \hat{\beta}_1 spread_i + \hat{\beta}_2 favhome_i + \hat{\beta}_3 fav25_i + \hat{\beta}_4 und25_i + \hat{u}_i$$

Where $favwin$ is a dummy variable indicating whether the favored team won, $spread$ is the Las Vegas point spread, $favhome$ is a dummy variable indicating whether the favored team is playing at home, $fav25$ and $und25$ indicate whether the favored team and the underdog team are in the top 25 teams respectively.

I use the `pntsprd` data from the wooldridge package to first estimate the linear probability model:

```
lpm <- felm(favwin ~ spread + favhome + fav25 + und25, data = pntsprd)

summary(lpm, robust = TRUE)
```

```
##
## Call:
##    felm(formula = favwin ~ spread + favhome + fav25 + und25, data = pntsprd)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9972 -0.1105  0.1470  0.2991  0.4869
##
## Coefficients:
##             Estimate Robust s.e t value Pr(>|t|)
## (Intercept)  0.558815   0.040422  13.825   <2e-16 ***
## spread       0.017763   0.002056   8.637   <2e-16 ***
## favhome      0.054353   0.040923   1.328    0.185
## fav25        0.010982   0.039100   0.281    0.779
## und25       -0.101104   0.089530  -1.129    0.259
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4016 on 548 degrees of freedom
## Multiple R-squared(full model): 0.116   Adjusted R-squared: 0.1095
## Multiple R-squared(proj model): 0.116   Adjusted R-squared: 0.1095
## F-statistic(full model, *iid*):17.97 on 4 and 548 DF, p-value: 7.007e-14
## F-statistic(proj model):  26.2 on 4 and 548 DF, p-value: < 2.2e-16
```

Let's compare the LPM model results to those for the equivalent logit:

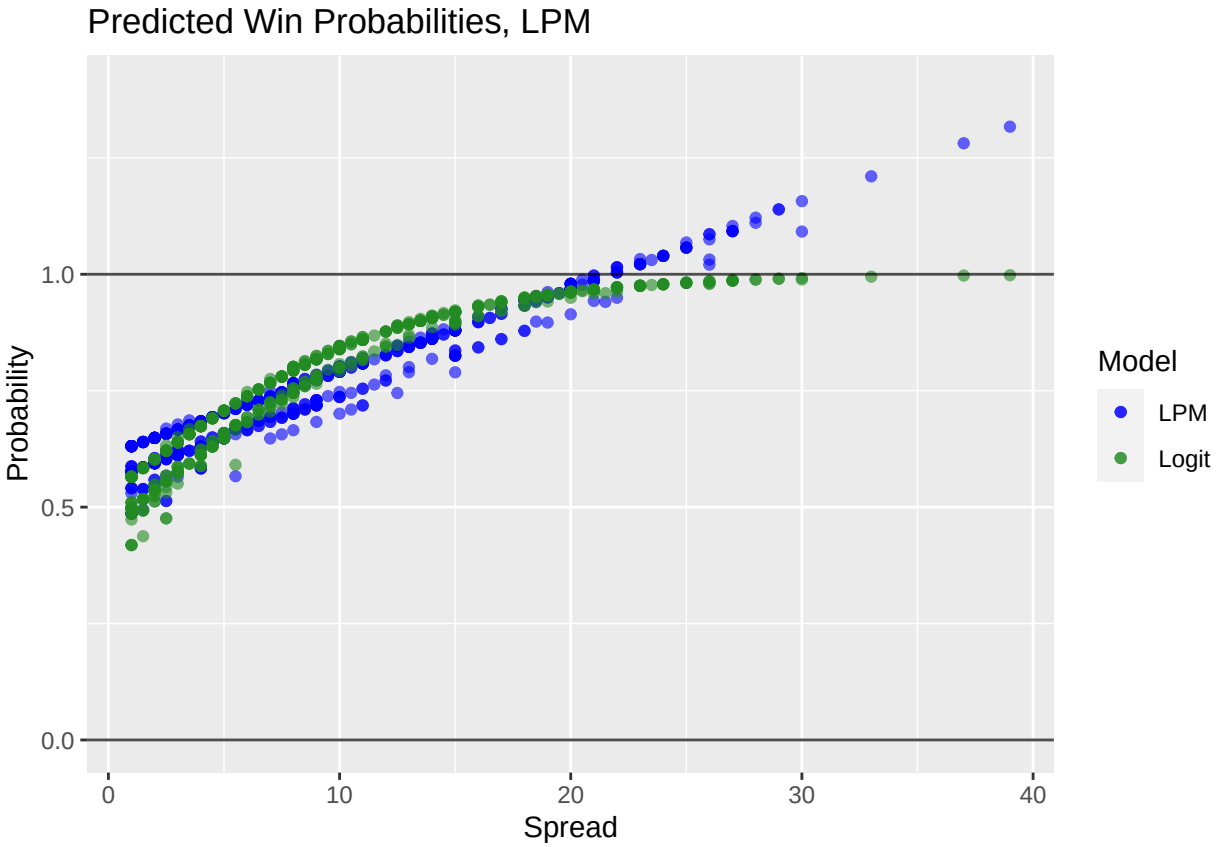
```
logit <- glm(favwin ~ spread + favhome + fav25 + und25, data = pntsprd, family = "binomial")
marg_logit <- margins(logit)
summary(marg_logit)
```

```
##   factor     AME     SE       z      p   lower  upper
##    fav25  0.0076 0.0434  0.1748 0.8612 -0.0774 0.0926
##  favhome  0.0425 0.0362  1.1740 0.2404 -0.0284 0.1134
##   spread  0.0243 0.0033  7.3360 0.0000  0.0178 0.0308
##    und25 -0.0579 0.0650 -0.8912 0.3728 -0.1852 0.0694
```

You can see that all else constant, a 1 point increase in the Vegas point spread is estimated to increase the predicted probability of winning by 1.78 percentage points using the linear probability model and 2.43 percentage points on average using the logit model. It is generally the case that results using these two different methods will often be quite similar.

To really see the difference between them, we can plot the predicted probabilities (obtained with `$fitted.values` on the `lm()` or `glm()` objects).

```
df <- mutate(pntsprd, lpm_prob = lpm$fitted.values, logit_prob = logit$fitted.values)

plot <- df %>%
    ggplot(aes(x = spread)) + geom_point(aes(y = lpm_prob, colour = "LPM"), alpha = 0.6) +
    geom_point(aes(y = logit_prob, colour = "Logit"), alpha = 0.6) + geom_hline(yintercept = 1,
    alpha = 0.7) + geom_hline(yintercept = 0, alpha = 0.7) + scale_colour_manual("Model",
    breaks = c("LPM", "Logit"), values = c("blue", "forestgreen")) + lims(y = c(0,
    1.4)) + labs(title = "Predicted Win Probabilities, LPM", x = "Spread", y = "Probability")
plot
```



We can see straightaway that we estimated probabilities greater than one with the linear probability model (in green) for high point spreads, and that the logit probabilities (blue) never exceed one. Additionally, you can see that the effect of one additional point in the spread has a constant effect for linear probability models, while the slope of the logit probabilities changes depending on the level of the point spread.

1.3 Non-standard standard errors

A standard error is, of course, an estimate of the uncertainty around an estimated parameter. Formally we have

$$se = \sqrt{\widehat{Var(\hat{\beta})}}$$

in other words, the standard error is the square root of the estimated variance of the estimated parameter.

Just like calculating point estimates, it is incredibly important to get your standard errors right. You have to know what you don't know!

1.3.1 Robust standard errors

We are going to use the diamonds data set from `ggplot2` for this exercise so we don't need to load an external data set.

```
knitr::kable(head(diamonds))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31

carat	cut	color	clarity	depth	table	price	x	y	z
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

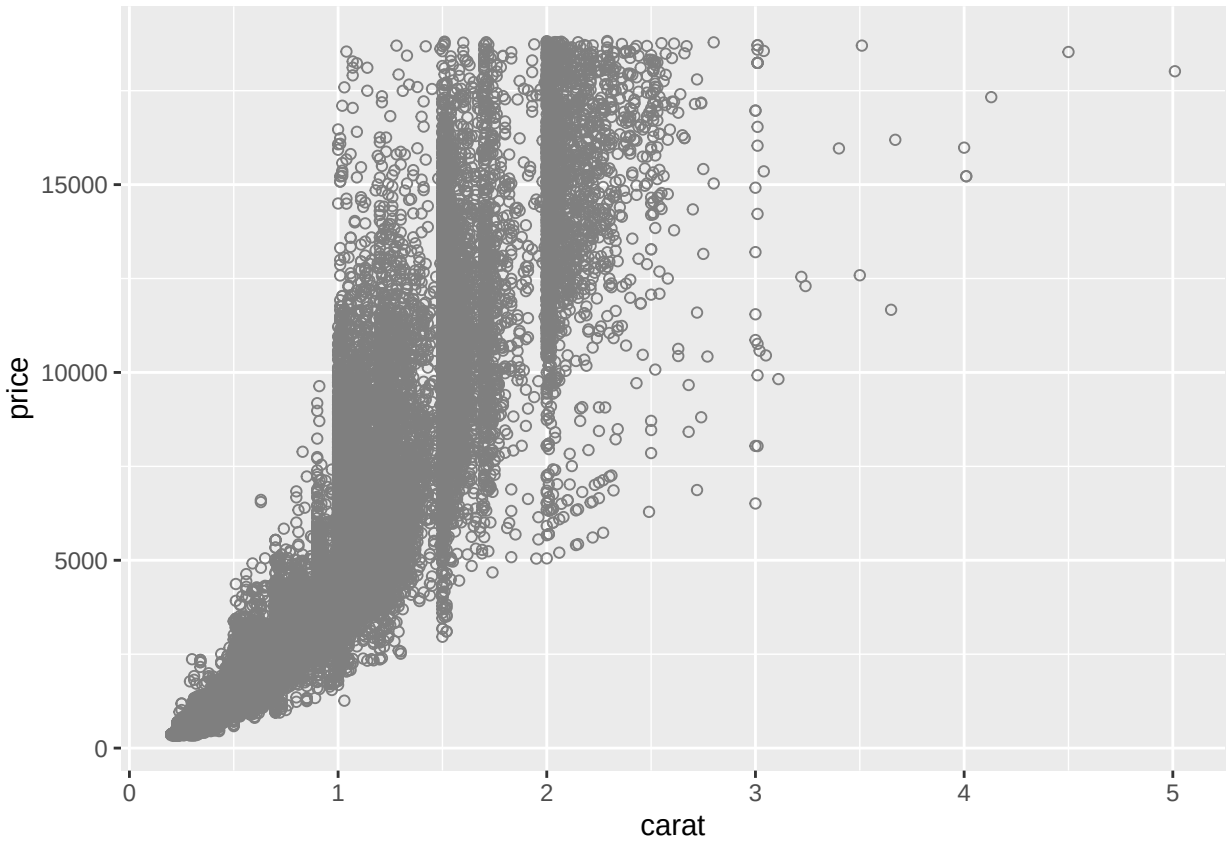
Lets regress price on carats and depth.

```
reg1 <- felm(price ~ carat + depth, diamonds)

summary(reg1)
```

```
##
## Call:
##    felm(formula = price ~ carat + depth, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18238.9   -801.6    -19.6    546.3  12683.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4045.333    286.205   14.13   <2e-16 ***
## carat       7765.141     14.009  554.28   <2e-16 ***
## depth       -102.165      4.635  -22.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1542 on 53937 degrees of freedom
## Multiple R-squared(full model): 0.8507   Adjusted R-squared: 0.8507
## Multiple R-squared(proj model): 0.8507   Adjusted R-squared: 0.8507
## F-statistic(full model):1.536e+05 on 2 and 53937 DF, p-value: < 2.2e-16
## F-statistic(proj model): 1.536e+05 on 2 and 53937 DF, p-value: < 2.2e-16
```

Cool. But of course, we should make sure that our OLS assumptions make sense. One easy way to do this is to plot the data:

```
myPlot <- ggplot(data = diamonds, aes(y = price, x = carat)) + geom_point(color = "gray50",
    shape = 21)

myPlot
```



There are a bunch of things about this plot that should give you the econometric heebie jeebies. From an OLS perspective, you should be very afraid that these data are definitely not homoskedastic. The higher the carat, the greater the variance in price. This means that our OLS standard errors are likely going to get things wrong.

Heteroskedasticity is scary- but thankfully all is not lost. All we have to do is tweak our original assumptions a little bit to relax the homoskedasticity assumption and allow for the variance to depend on the value of x_i.

We know that

$$Var(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2\sigma^2}{(\sum_{i=1}^{n}(x_i - \bar{x})^2)^2}$$

With heteroskedasticity σ^2 is no longer constant and becomes a function of the particular value of x_i an observation has, so

$$Var(u_i|x_i) = \sigma_i^2$$

Where are we going to find all these σ_i^2 for each individual observation?

Econometricians Eicker, Huber and White figured out a way to do this by basically using the square of the estimated residual of each observation, $\hat{u}_i^2$, as a stand-in for σ_i^2. With this trick, a valid estimator for $Var(\hat{\beta}_1)$, with heteroskedasticity of **any** form (including homoskedasticity), is

$$Var(\hat{\beta}_1) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2\hat{u}_i^2}{(\sum_{i=1}^{n}(x_i - \bar{x})^2)^2}$$

We commonly call the resulting standard errors "robust", or "heteroskedasticity-robust".

How can we find these in R?

```
reg1 <- felm(price ~ carat + depth, diamonds)

summary(reg1, robust = TRUE)
```

```
##
## Call:
##    felm(formula = price ~ carat + depth, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18238.9   -801.6    -19.6    546.3  12683.7
##
## Coefficients:
##             Estimate Robust s.e t value Pr(>|t|)
## (Intercept) 4045.333    369.176   10.96   <2e-16 ***
## carat       7765.141     25.105  309.31   <2e-16 ***
## depth       -102.165      5.946  -17.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1542 on 53937 degrees of freedom
## Multiple R-squared(full model): 0.8507   Adjusted R-squared: 0.8507
## Multiple R-squared(proj model): 0.8507   Adjusted R-squared: 0.8507
## F-statistic(full model, *iid*):1.536e+05 on 2 and 53937 DF, p-value: < 2.2e-16
## F-statistic(proj model): 4.878e+04 on 2 and 53937 DF, p-value: < 2.2e-16
```

Or if you want to put them in a stargazer table:

```
stargazer(reg1, type = "latex", se = list(reg1$rse), header = FALSE)
```

Table 4:

	Dependent variable:
	price
carat	7,765.141***
	(25.105)
depth	−102.165***
	(5.946)
Constant	4,045.333***
	(369.176)
Observations	53,940
R^2	0.851
Adjusted R^2	0.851
Residual Std. Error	1,541.649 (df = 53937)
Note:	*p<0.1; **p<0.05; ***p<0.01

It is worth noting that robust standard errors are larger than regular standard errors, and thus more conservative (which is the right thing to be... you want to know what you don't know).

1.3.2 Clustered standard errors

Econometricians Haiku

T-stats looks too good
Try cluster standard errors
significance gone.

from Angrist and Pischke 2008

Suppose that every observation belongs to (only) one of G groups. The assumption we make when we cluster is that there is no correlation across groups- but we will allow for arbitrary within-group correlation. A great example: consider individuals within a village. In many cases it's pretty reasonable to think that individuals' error terms are correlated within a village, but that individuals' errors aren't correlated across villages.

I will spare you the matrix math needed to dive deeper into this. Suffice to say that "cluster-robust" estimates allow for a more complicated set of correlations to exist within observations within a cluster. One thing to be aware of though is that you will need to have a fairly large number of clusters (40+) for the estimate to be credible.

Clustering in R:

I use the `NOxEmissions` dataset from the `robustbase` package. This is a dataset of hourly NO_x readings, including NO_x concentration, auto emissions and windspeed. We are going to use the observation date as our cluster variable. This allows for arbitrary dependence between observations in the same day, and zero correlation across days. Is this reasonable? ... Maybe. But we'll go with it for now:

```
nox <- as.data.frame(NOxEmissions) %>%
    mutate(ones = 1)
noClusters <- felm(data = nox, LNOx ~ sqrtWS)

Clusters <- felm(data = nox, LNOx ~ sqrtWS | 0 | 0 | julday)

stargazer(noClusters, Clusters, type = "latex", header = FALSE)
```

In this case, the regular standard errors are smaller than the clustered standard errors. Be aware that this need not necessarily be the case - depending on the correlation between observations within a cluster, clustered standard errors can be smaller than regular standard errors.

1.3.3 Newey West Standard Errors

For time series data.

1.3.4 Conley Standard Errors

For spatial data.

1.4 Confidence intervals for predictions

You've already had to "predict" a value of the dependent variable, y, given certain values of the independent variables. But this prediction is just a guess, and we can construct a confidence interval to give a range of

Table 5:

	Dependent variable:	
	LNOx	
	(1)	(2)
sqrtWS	−0.864***	−0.864***
	(0.020)	(0.048)
Constant	5.559***	5.559***
	(0.029)	(0.065)
Observations	8,088	8,088
R^2	0.185	0.185
Adjusted R^2	0.185	0.185
Residual Std. Error (df = 8086)	0.846	0.846
Note:	*p<0.1; **p<0.05; ***p<0.01	

possible values for this prediction, to demonstrate this uncertainty.

There are two kinds of predictions we can make:

- A confidence interval for the **average** y given x_1, $x_2 \ldots \ x_k$.
- A confidence interval for a **particular** y given x_1, $x_2 \ldots \ x_k$.

We will use Wooldridge's birth weight data to construct both kinds of confidence intervals to demonstrate the (subtle) difference between them.

$$bweight = \beta_0 + \beta_1 lfaminc + \beta_2 meduc + \beta_3 parity + u$$

where bwght is birth weight in ounces, lfaminc is the log of family income in $1000s, meduc is the education of the mother in years, and parity is the birth order of the child.

Estimating this equation in R, we get the following results:

```
# using the bwght data from the wooldridge package
reg1 <- lm(bwght ~ lfaminc + motheduc + parity, bwght)

summary(reg1)
```

```
##
## Call:
## lm(formula = bwght ~ lfaminc + motheduc + parity, data = bwght)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -94.533 -11.888   0.779  13.136 151.477
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 105.5652     3.3666  31.356  < 2e-16 ***
## lfaminc       2.1313     0.6506   3.276  0.00108 **
## motheduc      0.3172     0.2520   1.259  0.20829
## parity        1.5261     0.6119   2.494  0.01275 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.21 on 1383 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.01633,    Adjusted R-squared:  0.0142
## F-statistic: 7.654 on 3 and 1383 DF,  p-value: 4.482e-05
```

1.4.0.1 Confidence interval for the average Birthweight

Recall that our model gives us the expected value:

$$E[bweight|lfaminc, meduc, parity] = \beta_0 + \beta_1 log(faminc) + \beta_2 meduc + \beta_3 parity$$

and our regression gives us an estimate of this:

$$\hat{E}[bweight|faminc, meduc, parity] = \hat{y} = \hat{\beta}_0 + \hat{\beta}_1 log(faminc) + \hat{\beta}_2 meduc + \hat{\beta}_3 parity$$

In words: when we plug in particular values of the independent variables, we obtain a predication for y, which is an estimate for the expected value of y given the particular values for the explanatory variables.

Say we are interested in a confidence interval for the **average birthweight** for babies with a family income of \$14,500 (ln(14.5)=2.674), mothers with 12 years of education, and with 2 older siblings (parity=3). In other words, we are interested in:

$$\begin{aligned}\hat{E}[bweight|faminc = 14.5, meduc = 12, parity = 3] &= 105.66 + 2.13ln(faminc) + 0.317meduc + 1.53parity \\ \hat{y}_{faminc=14.5, meduc=12, parity=3} &= 105.66 + 2.13(2.674) + 0.317(12) + 1.53(3) \\ &= 119.75 ounces\end{aligned}$$

How do we find a standard error for our estimate of the expected value of y for these particular values of the explanatory variables? Notice that this standard error is complicated because $\widehat{bweight}$ is a function of our $\hat{\beta}$'s which are all random variables. To avoid this computation, we want to transform our data. Before proceeding with the formal steps, recall that we have the following regression in mind

$$bweight = \beta_0 + \beta_1 lfaminc + \beta_2 meduc + \beta_3 parity + u$$

Then

$$\hat{\beta}_0 = \hat{E}(bweight|lfaminc = 0, meduc = 0, parity = 0)$$

If we modify the regression by subtracting our particular values (specified above) for each of the independent variables, then we get the following regression

$$bweight = \beta_0 + \beta_1(lfaminc - 2.674) + \beta_2(meduc - 12) + \beta_3(parity - 3) + u$$

Then

$$\hat{\beta}_0 = \hat{E}(bweight|lfaminc = 2.674, meduc = 12, parity = 3).$$

In other words, the new intercept is the predicted birthweight for babies with family incomes of \$14,500 (ln(14.5)=2.674), mothers with 12 years of education and with 2 older siblings. That's perfect! If we run this regression, `R` always outputs a standard error for the intercept coefficient. We can then grab the standard errors from there.

So step by step we need to:

1) Generate new variables: $\tilde{x}_j = x_j - \alpha_j$

2) Run the regression: $y = \tilde{\beta}_0 + \tilde{\beta}_1 \tilde{x_1} + ... + \tilde{\beta}_k \tilde{x_k} + \tilde{u}$

3) Then $\hat{E}[y|x_1 = \alpha_1, ..., x_k = \alpha_k] = \tilde{\beta}_0$

4) Plug these values into the formula for confidence intervals and interpret.

Below is the code for steps 1 and 2:

```
# Step 1: generate new variables
bwght$lfaminc_0 <- bwght$lfaminc - 2.674
bwght$motheduc_0 <- bwght$motheduc - 12
bwght$parity_0 <- bwght$parity - 3

# step 2: run the new regression

reg2 <- lm(bwght ~ lfaminc_0 + motheduc_0 + parity_0, bwght)

summary(reg2)
```

```
##
## Call:
## lm(formula = bwght ~ lfaminc_0 + motheduc_0 + parity_0, data = bwght)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -94.533 -11.888   0.779  13.136 151.477
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 119.6491     1.0066 118.864  < 2e-16 ***
## lfaminc_0     2.1313     0.6506   3.276  0.00108 **
## motheduc_0    0.3172     0.2520   1.259  0.20829
## parity_0      1.5261     0.6119   2.494  0.01275 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.21 on 1383 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.01633,    Adjusted R-squared:  0.0142
## F-statistic: 7.654 on 3 and 1383 DF,  p-value: 4.482e-05
```

Using this output, the 95% confidence interval for the average birthweight for babies given a family income of \$14,500, a mother with 12 years of education and with 2 older siblings is:

$$[119.64 - 1.96(1.007), 119.64 + 1.96(1.007)] = [117.6653, 121.6158]$$

1.4.0.2 Confidence Interval for a particular individual's birthweight

A confidence interval for the average of observations with specific characteristics is not the same as a confidence interval for a particular individual observation. In forming a confidence interval for a particular unit, we must account for another very important source of variation: the variance in the unobserved error, which measures our ignorance of the unobserved factors that affect y_i.

We would like to construct a confidence interval for the birthweight of baby $i = 1$. Let $bweight_{i=1}$ denote that particular baby's birthweight with

$$bweight_{i=1} = \beta_0 + \beta_1 lfaminc_{i=1} + \beta_2 meduc_{i=1} + \beta_3 parity_{i=1} + u_{i=1}$$

Our best prediction of $bweight_{i=1}$ is $\widehat{bwieght}_{i=1}$ where

$$\widehat{bweight}_{i=1} = \hat{\beta}_0 + \hat{\beta}_1 lfaminc_{i=1} + \hat{\beta}_2 meduc_{i=1} + \hat{\beta}_3 parity_{i=1}$$

Now, there is some error, $\hat{u}_{i=1}$ associated with using $\widehat{bweight}_{i=1}$ to predict $bweight_{i=1}$ where

$$\begin{aligned}\hat{u}_{i=1} &= bweight_{i=1} - \widehat{bweight}_{i=1} \\ &= (\beta_0 + \beta_1 lfaminc_{i=1} + \beta_2 meduc_{i=1} + \beta_3 parity_{i=1} + u_{i=1}) - (\hat{\beta}_0 + \hat{\beta}_1 lfaminc_{i=1} + \hat{\beta}_2 meduc_{i=1} + \hat{\beta}_3 parity_{i=1})\end{aligned}$$

Finding the expected value, we get:

$$\begin{aligned}E[\hat{u}_{i=1}] &= E[bweight_{i=1} - \widehat{bweight}_{i=1}] \\ &= (\beta_0 + \beta_1 lfaminc_{i=1} + \beta_2 meduc_{i=1} + \beta_3 parity_{i=1} + E[u_{i=1}]) - (E[\hat{\beta}_0] + E[\hat{\beta}_1] lfaminc_{i=1} + E[\hat{\beta}_2] meduc_{i=1} + E[\hat{\beta}_3] parity_{i=1}) \\ &= 0\end{aligned}$$

Finding the variance we get

$$\begin{aligned}Var(\hat{u}_{i=1}) &= Var(bweight_{i=1} - \widehat{bweight}_{i=1}) \\ &= Var(\beta_0 + \beta_1 lfaminc_{i=1} + \beta_2 meduc_{i=1} + \beta_3 parity_{i=1} + u_{i=1} - \widehat{bweight}_{i=1}) \\ &= Var(\widehat{bweight}_{i=1}) + Var(u_{i=1}) \\ &= Var(\widehat{bweight}_{i=1}) + \sigma^2 \\ \widehat{Var(\hat{u}_{i=1})} &= Var(\widehat{bweight}_{i=1}) + \hat{\sigma}^2 \\ &= Var(\widehat{bweight}_{i=1}) + \frac{\sum \hat{u}_i^2}{n-k-1} \\ &= Var(\widehat{bweight}_{i=1}) + \frac{SSR}{n-k-1}\end{aligned}$$

So you should see that there are two sources of variation in $\hat{u}_{i=1}$. First we have the sampling error in $\widehat{bweight}_{i=1}$ which arises because we have estimated the population parameters β. Second we have the variance of the error in the population ($u_{i=1}$).

Now we can compute the $Var(\widehat{bweight}_{i=1})$ exactly the way we did before (by subtracting the specific values we are interested in and re-running the regression and looking at the intercept term's standard errors). Second we can compute $\frac{SSR}{n-k-1}$ from our regression output. The 95% confidence interval for $bweight_{i=1}$ is then

$$\hat{y} \pm 1.96 * se(\hat{u}_{i=1})$$

Steps in computing a confidence interval for a particular y when $x_j = \alpha_j$:

1) Generate new variables: $\tilde{x}_j = x_j - \alpha_j$
2) Run the regression: $y = \tilde{\beta}_0 + \tilde{\beta}_1 \tilde{x_1} + ... + \tilde{\beta}_k \tilde{x_k} + \tilde{u}$
3) Then $\hat{E}[y|x_1 = \alpha_1, ..., x_k = \alpha_k] = \tilde{\beta}_0$ and the standard error of the estimate is $se(\tilde{\beta}_0)$
4) Get an estimate for the variance of $\hat{u} = \hat{\sigma}^2$ from the R output
5) compute the standard error: $\sqrt{se(\tilde{\beta}_0)^2 + \hat{\sigma}^2}$
6) Plug these values into the formula for confidence intervals and interpret.

Below is the code:

```
# Step 1: generate new variables
bwght$lfaminc_0 <- bwght$lfaminc - 2.674
bwght$motheduc_0 <- bwght$motheduc - 12
bwght$parity_0 <- bwght$parity - 3

# step 2: run the new regression

reg2 <- lm(bwght ~ lfaminc_0 + motheduc_0 + parity_0, bwght)

summary(reg2)
```

```
##
## Call:
## lm(formula = bwght ~ lfaminc_0 + motheduc_0 + parity_0, data = bwght)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -94.533 -11.888   0.779  13.136 151.477
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 119.6491     1.0066 118.864  < 2e-16 ***
## lfaminc_0     2.1313     0.6506   3.276  0.00108 **
## motheduc_0    0.3172     0.2520   1.259  0.20829
## parity_0      1.5261     0.6119   2.494  0.01275 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.21 on 1383 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.01633,    Adjusted R-squared:  0.0142
## F-statistic: 7.654 on 3 and 1383 DF,  p-value: 4.482e-05
```

```
# step 4: get the estimate of the variance

summary(lm(bwght ~ lfaminc_0 + motheduc_0 + parity_0, bwght))$sigma^2
```

```
## [1] 408.5987
```

The confidence interval for a particular baby's birthweight with a family income of \$14,500 (ln(14.5=2.674)), a mother with 12 years of education and with 2 older siblings we have:

$$\begin{aligned} SE &= \sqrt{se(\tilde{\beta}_0)^2 + \hat{\sigma}^2} = \sqrt{(1.007^2) + 408.59} = 20.239 \\ CI &= [119.64 - 1.96 * (20.239); 119.64 + 1.96 * (20.239)] \\ &= [79.972; 159.308] \end{aligned}$$

1.5 Spatial Data in R

Increasingly, applied econometricians are embedding spatial data into our analyses. A whole host of research questions and designs become possible if you are able to take advantage of, and work with, the huge volume of spatial data that is rapidly becoming available. That said, working with spatial data takes a few new tricks. We'll first go through a variety of spatial data types, then show off the long-promised (and dramatically over-used) nighttime lights dataset. Coooooool.

First off, we often make a big deal out of spatial data - but it's just data like anything else. Anybody who tries to tell you that spatial data is *completely different*(!) from anything you've seen before is trying to sell you a textbook. The main difference between spatial datasets and any other type of dataset is that spatial data usually come in 2 (or even 3) dimensions. For practical purposes, this usually means that a spatial dataset has latitude and longitude variables in some form or another.

To deal with spatial data, we'll need a bunch of new packages: `GISTools`, `rgdal`, `rgeos`, `maptools`, `raster` (all spatial utilities), and `broom` (this will let us turn spatial data into a format that ggplot2 can handle). We'll also get a new package that will let us more easily deal with dates (because why not): `lubridate`, and a final one which gives us access to a bunch of new color palettes: `RColorBrewer`. The number of packages we're installing here should make you uncomfortable. While you certainly can do spatial data analysis in `R`, it's definitely a challenge compared with some other (proprietary) options out there (Matlab and ArcGIS, for example). The benefits are that you'll have a completely integrated work flow, all in one program; and that once you've wrangled your spatial data to turn it into something useful, `R` is great for everything else. The big problem with doing spatial data in `R` is that there's no unified spatial toolkit, meaning that different data types and formats and functions don't necessarily get along very well. I've tried to make this as headache-free as possible, but just be aware that (in my opinion) spatial data work is not necessarily R's comparative advantage.

Note that the order in which you load these packages is important. I've set things up such that they'll work; (consider yourself warned).

```
library(raster)
# library(vctrs) library(tibble) library(GISTools) library(readr)
library(dplyr)
library(lubridate)
# library(xtable)
library(ggplot2)
library(rgeos)
library(rgdal)
library(maptools)
library(broom)
```

1.5.1 What the frack?

Now we can get to the actual spatial stuff. We're going to use some georeferenced data to think about unconventional oil and gas drilling in Pennsylvania. We'll start with a spatial data file from the US Census Bureau with the shape of each county in the state. In keeping with the ultra-creative naming conventions we've seen all semester, this data comes in the *shapefile* format. This is the main format used by ArcGIS, so you'll see a lot of spatial data that comes packaged this way. Bringing it into R is easy:

```
counties <- shapefile("data/PA_counties.shp")
```

```
## Warning in .getSpatDF(x@ptr$df, ...): NAs introduced by coercion to integer
## range
```

```
counties
```

```
## class       : SpatialPolygonsDataFrame
## features    : 67
## extent      : -80.51989, -74.6895, 39.7198, 42.51607  (xmin, xmax, ymin, ymax)
## crs         : +proj=longlat +datum=NAD83 +no_defs
## variables   : 13
## names       : STATEFPEC, COUNTYFPEC, CNTYIDFPEC, NAMEEC,   NAMELSADEC, LSADEC, CLASSFPEC, MTFCCEC, FU
## min values  :        42,        001,      42001,  Adams, Adams County,     06,        H1,   G4020,
## max values  :        42,        133,      42133,   York,  York County,     06,        H6,   G4020,
```

What's in this object? This particular file contains polygons - outlines of shapes. We can make sure that it's read correctly into R by checking its class:

```
class(counties)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

Perfect. This is R's version of a polygon shapefile. What's actually in this dataset? We can figure this out by looking at its slots. Just like calling `names()` on an object can tell us something about its contents, `slotNames()` can do this too, for more complex objects:

```
slotNames(counties)
```

```
## [1] "data"        "polygons"    "plotOrder"   "bbox"        "proj4string"
```

This is cool! Our shapefile contains a bunch of interesting pieces. Let's walk through them quickly. Most importantly, we've got polygons. This is the spatial component of our spatial dataset. Each polygon is essentially a long two-variable dataframe with longitudes (x) and latitudes (y). Think of the polygon piece of a shapefile like a giant connect-the-dots.

Our shapefile also contains data. The dataset attached to a shapefile is just like a regular dataframe, with the additional cool feature that each row is actually associated with one of the polygons in our shapefile. Let's take a look at the data that comes with the Pennsylvania county shapefile (we'll rename our names to all be lower case as well, while we're at it):

```
head(counties@data, 5)
```

```
##   STATEFPEC COUNTYFPEC CNTYIDFPEC  NAMEEC     NAMELSADEC LSADEC CLASSFPEC
## 1        42        001      42001   Adams   Adams County     06        H1
## 2        42        027      42027  Centre  Centre County     06        H1
## 3        42        023      42023 Cameron Cameron County     06        H1
## 4        42        025      42025  Carbon  Carbon County     06        H1
## 5        42        029      42029 Chester Chester County     06        H1
##   MTFCCEC FUNCSTATEC    ALANDEC AWATEREC  INTPTLATEC   INTPTLONEC
## 1   G4020          A 1342811635  8560860 +39.8694707 -077.2177295
## 2   G4020          A         NA  7879260 +40.9091673 -077.8478195
## 3   G4020          A 1026698470  5664145 +41.4382882 -078.1983134
## 4   G4020          A  988299800 15353755 +40.9178324 -075.7094276
## 5   G4020          A 1943960881 22601992 +39.9737772 -075.7503816
```

```
names(counties@data) <- tolower(names(counties@data))
```

This dataset is relatively uninteresting - it just contains a bunch of identifying information about each county (but not much else in terms of demographics or other things we might be interested in). We'll have to bring something else in ourselves (we'll get there).

Also contained in our shapefile: the `bbox` and `plotOrder`. They're largely irrelevant for our purposes - they tell the base plotting commands how to display the data. We've also got a *projection*: `proj4string`. This tells R both how to display and how to "think about" our shapefile. Remember that the world is round. We're trying to represent it with a data object on our (flat) computer screens. The projection defines what dimensions to stretch to make this representation happen. What does it look like? Let's look! Just like we can grab a named object with the $ operator, we can grab a named slot with the @ operator.

```
counties@proj4string
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=NAD83 +no_defs
## WKT2 2019 representation:
```

```
## GEOGCRS["unknown",
##     DATUM["North American Datum 1983",
##         ELLIPSOID["GRS 1980",6378137,298.257222101,
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6269]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

This tells us that we're using latitude and longitude to identify our data, and that we're using the North American Datum 83 (a common choice for looking at the US). Since we're dealing with a relatively small geographic area, this won't affect us too much - but you should be aware of what projection you're using. In particular, if you're going to combine multiple geographic datasets, it's important that they all use the same projection.

1.5.2 Ooh, pretty!

Okay, now that we've had a quick introduction to our data, let's plot our shapefile. Since a polygon is just a big collection of longitude and latitude values, we can just plot it like anything else in `ggplot2`. The first thing we have to do is to convert our polygon into a dataframe that `ggplot2` can handle, using `broom`'s `tidy()` function. In the mean time, we'll merge, or `join()`, the additional data that came with the shapefile, into this new dataframe.

We are going to do this process several times, so we'll write a function to take care of it for us:

```
mapToDF <- function(shapefile) {
    # first assign an identifier to the main dataset
    shapefile@data$id <- rownames(shapefile@data)
    # now 'tidy' our data to convert it into a dataframe that is usable by
    # ggplot2
    mapDF <- tidy(shapefile) %>%
        # and this data onto the information attached to the shapefile
    left_join(., shapefile@data, by = "id") %>%
        as.data.frame()
    return(mapDF)
}
paCounties <- mapToDF(counties)
```

```
## Regions defined for each Polygons
```

```
head(paCounties)
```

```
##        long      lat order  hole piece group id statefpec countyfpec cntyidfpec
## 1 -76.99950 39.79106     1 FALSE     1   1.1  1        42        001      42001
## 2 -76.99943 39.79044     2 FALSE     1   1.1  1        42        001      42001
## 3 -76.99939 39.79013     3 FALSE     1   1.1  1        42        001      42001
## 4 -76.99939 39.79003     4 FALSE     1   1.1  1        42        001      42001
## 5 -76.99931 39.78907     5 FALSE     1   1.1  1        42        001      42001
```

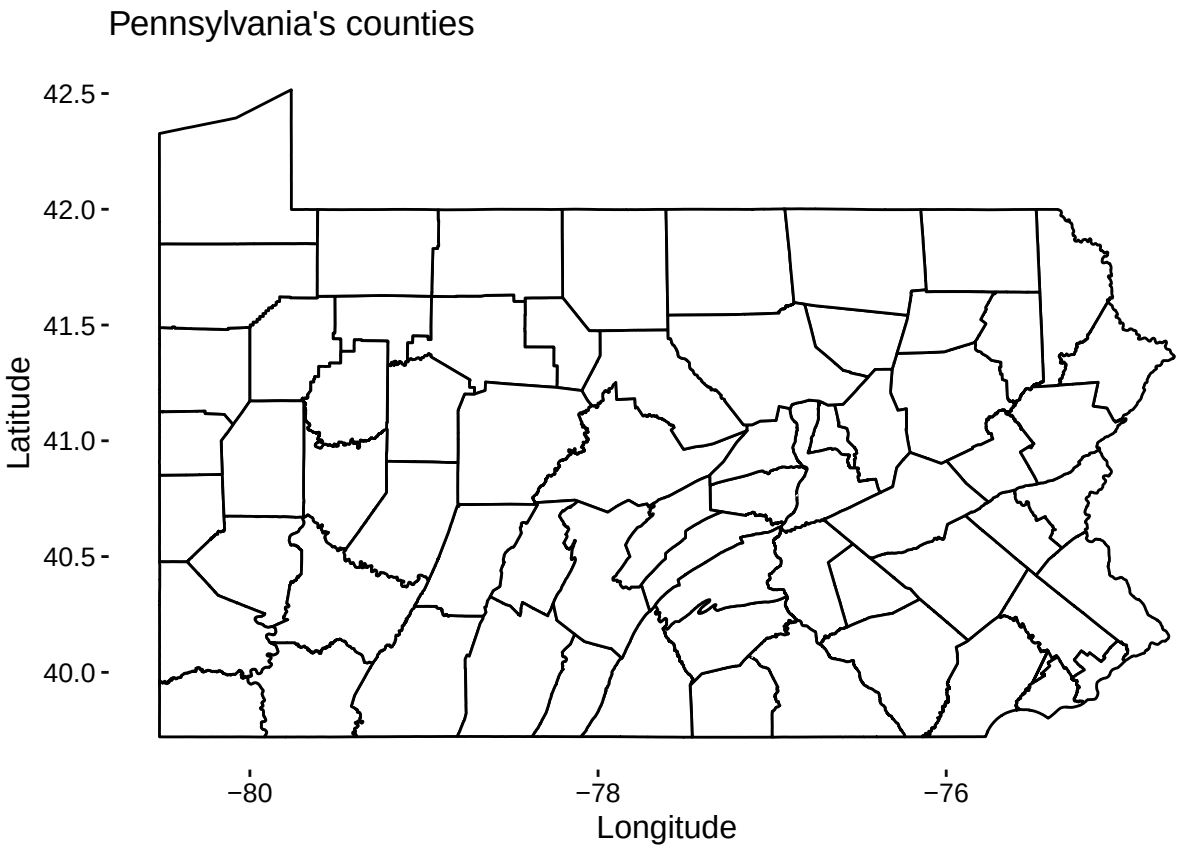
```
## 6 -76.99929 39.78852     6 FALSE     1   1.1  1        42        001      42001
##   nameec   namelsadec lsadec classfpec mtfccec funcstatec    alandec awaterec
## 1  Adams Adams County     06        H1   G4020          A 1342811635  8560860
## 2  Adams Adams County     06        H1   G4020          A 1342811635  8560860
## 3  Adams Adams County     06        H1   G4020          A 1342811635  8560860
## 4  Adams Adams County     06        H1   G4020          A 1342811635  8560860
## 5  Adams Adams County     06        H1   G4020          A 1342811635  8560860
## 6  Adams Adams County     06        H1   G4020          A 1342811635  8560860
##    intptlatec   intptlonec
## 1 +39.8694707 -077.2177295
## 2 +39.8694707 -077.2177295
## 3 +39.8694707 -077.2177295
## 4 +39.8694707 -077.2177295
## 5 +39.8694707 -077.2177295
## 6 +39.8694707 -077.2177295
```

Now that we've got this dataset, it's easy to plot using `ggplot2`. I will start by defining a `ggplot` theme for my maps:

```
myMapThemeStuff <- theme(panel.background = element_rect(fill = NA), panel.border = element_blank(),
    panel.grid.major = element_blank(), panel.grid.minor = element_blank(), axis.ticks = element_line(c
    axis.text = element_text(color = "black", size = 10), axis.title = element_text(color = "black",
        size = 12), legend.key = element_blank())
```

Okay, here we go.

```
paMap <- ggplot(data = paCounties, aes(x = long, y = lat, group = id)) + geom_polygon(color = "black",
    fill = "white") + myMapThemeStuff + ggtitle("Pennsylvania's counties") + xlab("Longitude") +
    ylab("Latitude")

paMap
```



Woohoo! A map! We must've done a lot of work. Just a map by itself isn't very interesting, though. Let's bring in another dataset, containing the locations of every unconventional well drilled in Pennsylvania between 2002 and 2013 (courtesy of FracTracker Alliance). We'll do this with a familiar command:

```
wells <- read.csv("data/PA_wells.csv") %>%
    as.data.frame()

names(wells) <- tolower(names(wells))
head(wells)
```

```
##   spud_date       api   ogo_num                      operator
## 1 5/24/2002 125-22033 OGO-49020           BELDEN & BLAKE CORP
## 2 5/31/2003 125-22074 OGO-60915 RANGE RESOURCES APPALACHIA LLC
## 3 6/14/2003 063-33347 OGO-38958                XTO ENERGY INC
## 4 8/27/2003 129-25004 OGO-60915 RANGE RESOURCES APPALACHIA LLC
## 5  9/6/2003 129-25012 OGO-60915 RANGE RESOURCES APPALACHIA LLC
## 6 2/27/2004 129-25209 OGO-33810          GREAT OAK ENERGY INC
##                  region       county         municipality       farm_name
## 1 EP DOGO SWDO Dstr Off   Washington          Hanover Twp ANDERSON UNIT 1
## 2 EP DOGO SWDO Dstr Off   Washington   Mount Pleasant Twp          RENZ 1
## 3 EP DOGO SWDO Dstr Off      Indiana            Grant Twp LEONARD MUMAU 2
## 4 EP DOGO SWDO Dstr Off Westmoreland South Huntingdon Twp HEPLER CALVIN 1
## 5 EP DOGO SWDO Dstr Off Westmoreland South Huntingdon Twp       STAHL E 1
## 6 EP DOGO SWDO Dstr Off Westmoreland            Salem Twp   EXPORT FUEL 1
##   well_code_desc               well_status latitude longitude configuration
## 1            GAS                    Active 40.46467 -80.47789 Vertical Well
## 2            GAS                    Active 40.28316 -80.28402 Vertical Well
## 3            GAS                    Active 40.80562 -78.93993 Vertical Well
```

```
## 4  COMB. OIL&GAS Regulatory Inactive Status 40.15899 -79.70181 Vertical Well
## 5  COMB. OIL&GAS                     Active 40.15602 -79.72340 Vertical Well
## 6            GAS                     Active 40.43544 -79.58450 Vertical Well
##   unconventional
## 1            Yes
## 2            Yes
## 3            Yes
## 4            Yes
## 5            Yes
## 6            Yes
```

Notice that this dataset includes a latitude and a longitude variable - this means it's spatial. Before we plot it, though, we need to make sure that it'll use the same projection as our our county polygon. It's currently a dataframe - let's turn it into a SpatialPointsDataFrame (like the polygon, except for points). In doing so, we'll attach a projection.

```
# the coordinates() function sets spatial coordinates to define a spatial
# object
coordinates(wells) <- ~longitude + latitude
class(wells)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# the proj4string() function retreives the projection attributes of the wells
# object
proj4string(wells)
```

```
## [1] NA
```

No! Boo. We have to assign it ourselves. The bad news is that I'm not sure exactly what the original projection was (it wasn't labeled on the download website). We'll guess that it's WGS84 (a common global projection), and then re-project it to match our county data's NAD83 (I do this to show you how to modify a projection... teachable moment).

```
# assign a projection (WGS84)... check out
# https://rspatial.org/raster/spatial/6-crs.html for more info on coordinate
# reference systems
proj4string(wells) <- CRS("+proj=longlat +datum=WGS84")
# re-project this to match the county data
wells <- spTransform(wells, CRS(proj4string(counties)))
```

```
## Warning in proj4string(counties): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```
# check
proj4string(wells)
```

```
## Warning in proj4string(wells): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

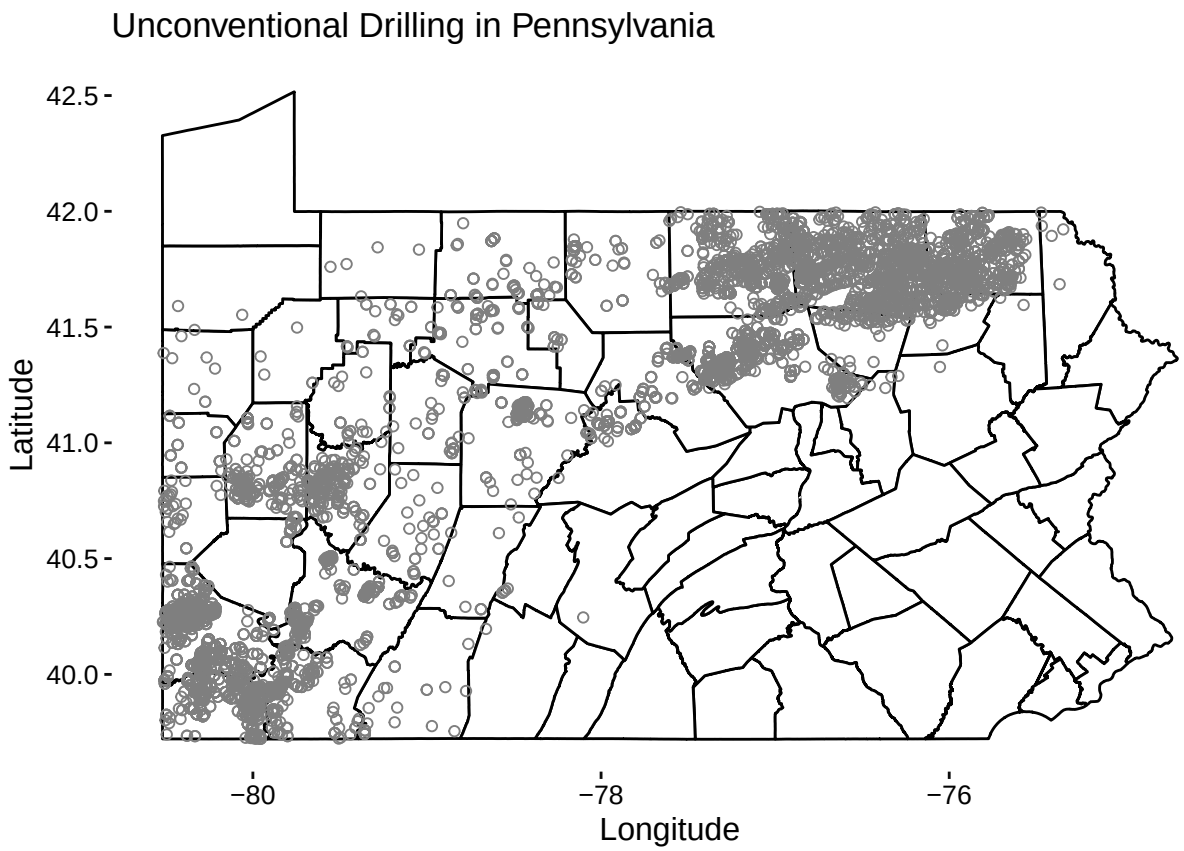
```
## [1] "+proj=longlat +datum=NAD83 +no_defs"
```

Now that the projection is good, we'll convert our dataset back to a dataframe format so that ggplot2 can handle it.

```
wellsDF <- as.data.frame(wells)
```

Now we'll plot both the counties and the wells on our map:

```
paMap <- ggplot() + geom_polygon(data = paCounties, aes(x = long, y = lat, group = id),
    color = "black", fill = "white") + geom_point(data = wellsDF, aes(x = longitude,
    y = latitude), shape = 21, color = "gray50") + myMapThemeStuff + ggtitle("Unconventional Drilling i
    xlab("Longitude") + ylab("Latitude")
paMap
```

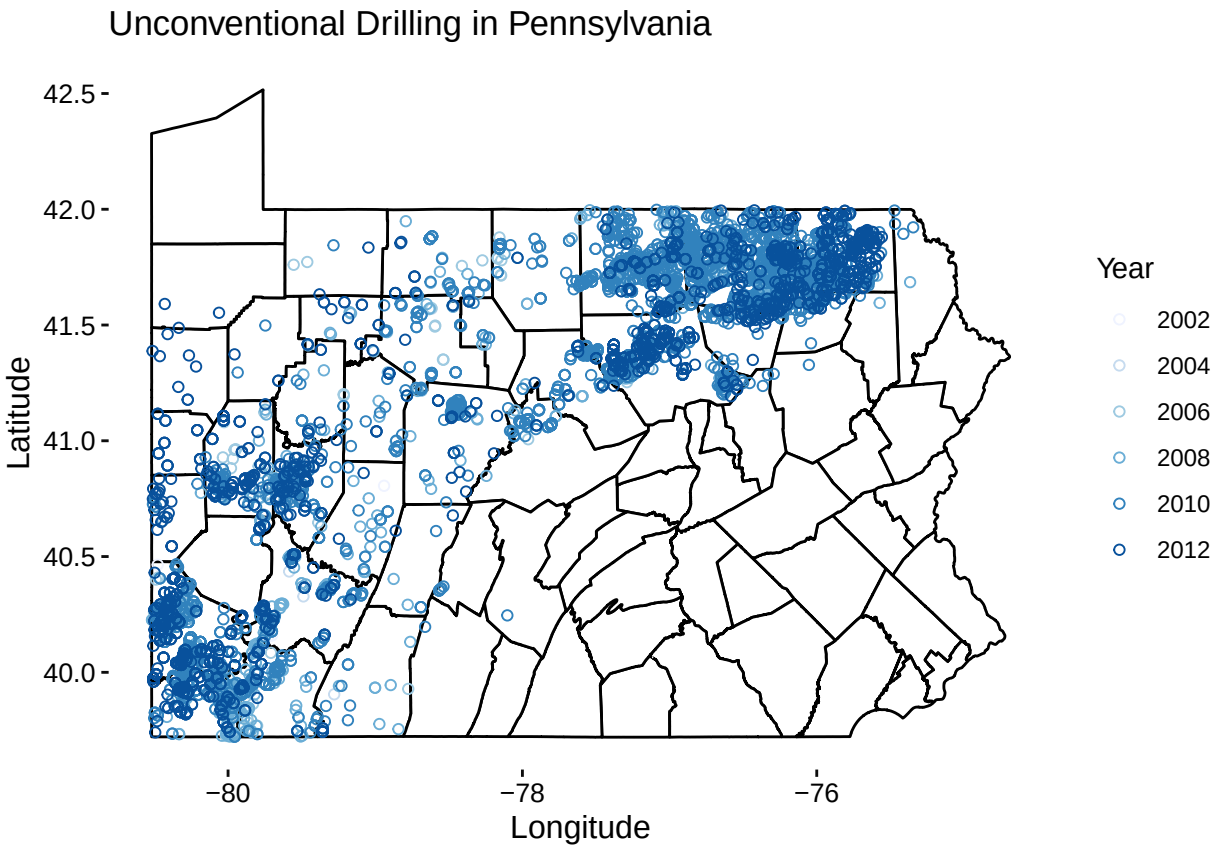


If we want, we can also plot different colors by year of well drilling. To do this, we'll first convert the spud date (drill date) variable to date format, extract the year, and convert this into a factor. In order to make this not impossible to look at, let's actually make pairs of years:

```
wellsDF <- mutate(wellsDF, date = mdy(spud_date), year = year(date)) %>%
    mutate(year = 2 * (floor(year/2)))

wellsDF <- mutate(wellsDF, year = as.factor(year))
```

This is also a great excuse to bring in a great R packages: `RColorBrewer`. If you're interested in seeing all of the available color palettes from this great package, just type `display.brewer.all()` into your consol.

```
paMap <- ggplot() + geom_polygon(data = paCounties, aes(x = long, y = lat, group = id),
    color = "black", fill = "white") + geom_point(data = wellsDF, aes(x = longitude,
    y = latitude, color = year), shape = 21) + scale_color_brewer(palette = "Blues") +
    myMapThemeStuff + ggtitle("Unconventional Drilling in Pennsylvania") + xlab("Longitude") +
    ylab("Latitude") + labs(color = "Year")
paMap
```



1.5.3 Playing around

Whoa. Lots of more recent wells! One other thing that's immediately obvious from this figure is that conventional drilling isn't happening all over the state. Instead, it's constrained to the northwest region. Why is this? Let's bring in some new data from the EIA, which will make this clear. We're going to grab a shapefile of the Marcellus shale play - the rock formation from which you can actually extract hydrocarbons.

```
playBdry <- shapefile("data/ShalePlay_Marcellus_Boundary_EIA_Aug2015_v2.shp")

playBdry
```

```
## class       : SpatialPolygonsDataFrame
## features    : 1
## extent      : -82.52247, -75.20568, 37.18328, 42.76125  (xmin, xmax, ymin, ymax)
## crs         : +proj=longlat +datum=WGS84 +no_defs
## variables   : 7
## names       :       Basin, Lithology, Shale_play, Source, Area_sq_mi, Area_sq_km,       Age_shale
## value       : Appalachian,     Shale,  Marcellus,    EIA,      58326,     151065, Middle Devonian
```

And we'll check the projection:

```
playBdry@proj4string
```

```
## Coordinate Reference System:
## Deprecated Proj.4 representation: +proj=longlat +datum=WGS84 +no_defs
## WKT2 2019 representation:
## GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
```

```
##             LENGTHUNIT["metre",1]],
##         ID["EPSG",6326]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433],
##         ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##         AXIS["longitude",east,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]],
##         AXIS["latitude",north,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433,
##                 ID["EPSG",9122]]]]
```

Whoops - this is WGS84. We'll have to convert it:

```
playBdry <- spTransform(playBdry, CRS(proj4string(counties)))
```

```
## Warning in proj4string(counties): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

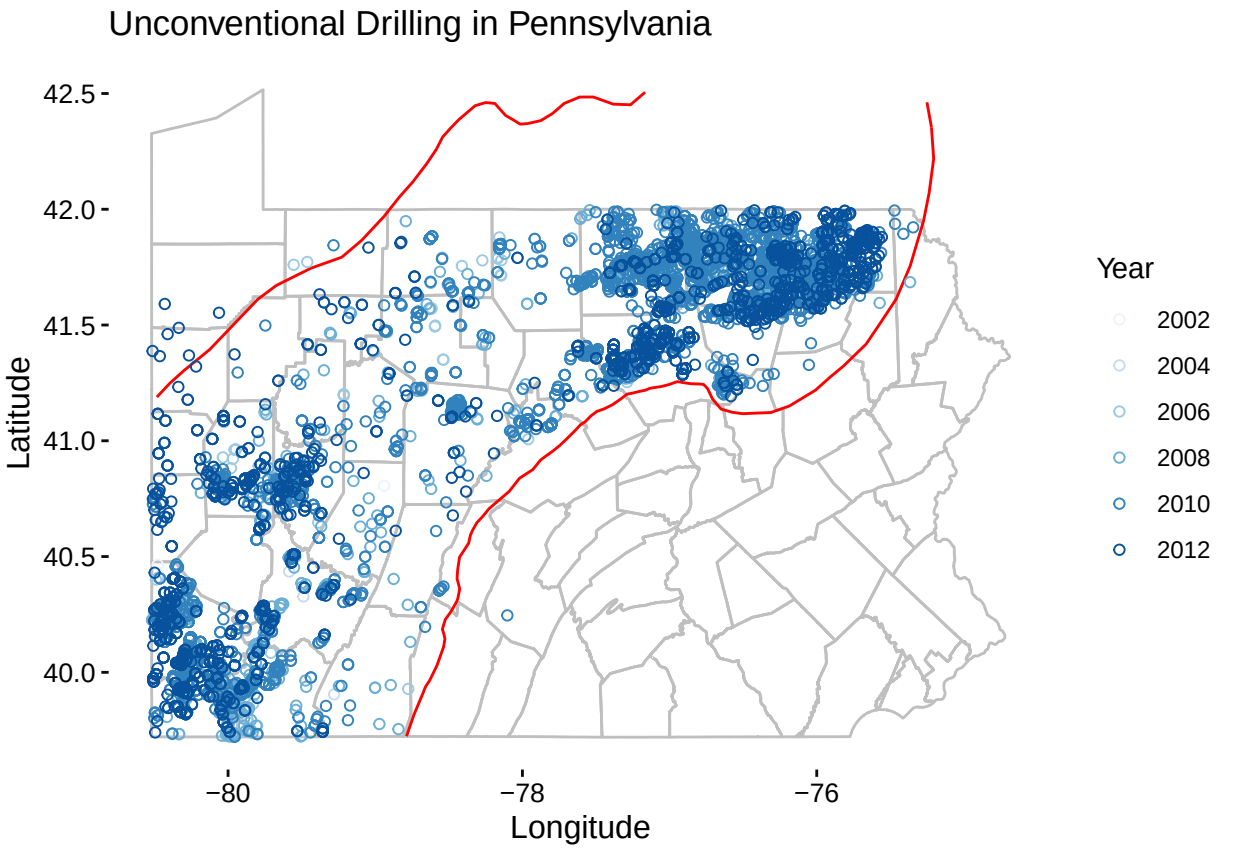
Let's prepare these shapefiles for plotting using our mapToDF() function from earlier to convert this into a dataframe:

```
bdryDF <- mapToDF(playBdry)
```

```
## Regions defined for each Polygons
```

Now we can easily plot these things:

```
bigPlot <- ggplot(data = bdryDF, aes(x = long, y = lat)) + geom_polygon(data = paCounties,
    aes(x = long, y = lat, group = id), color = "gray75", fill = "NA") + geom_path(data = bdryDF,
    aes(x = long, y = lat), color = "red") + geom_point(data = wellsDF, aes(x = longitude,
    y = latitude, color = year), shape = 21) + scale_color_brewer(palette = "Blues") +
    # put in a bounding box to restrict ourselves to the part of the play in PA
xlim(counties@bbox[1, 1], counties@bbox[1, 2]) + ylim(counties@bbox[2, 1], counties@bbox[2,
    2]) + ggtitle("Unconventional Drilling in Pennsylvania") + xlab("Longitude") +
    ylab("Latitude") + myMapThemeStuff + labs(color = "Year")


bigPlot
```



Nearly all of our wells are within the play - which makes sense, of course. Even within the play, though, there seems to be massive heterogeneity in well locations. Suppose we're interested in seeing whether there's any correlation between a county's demographics and its well count. To do this, we'll have to count the number of wells in each county. We'll also need some demographic data.

```
res <- over(wells, counties)

wellsInCty <- res %>%
    group_by(countyfpec) %>%
    summarise(n = n())
names(wellsInCty) <- c("countyfpec", "wells")
```

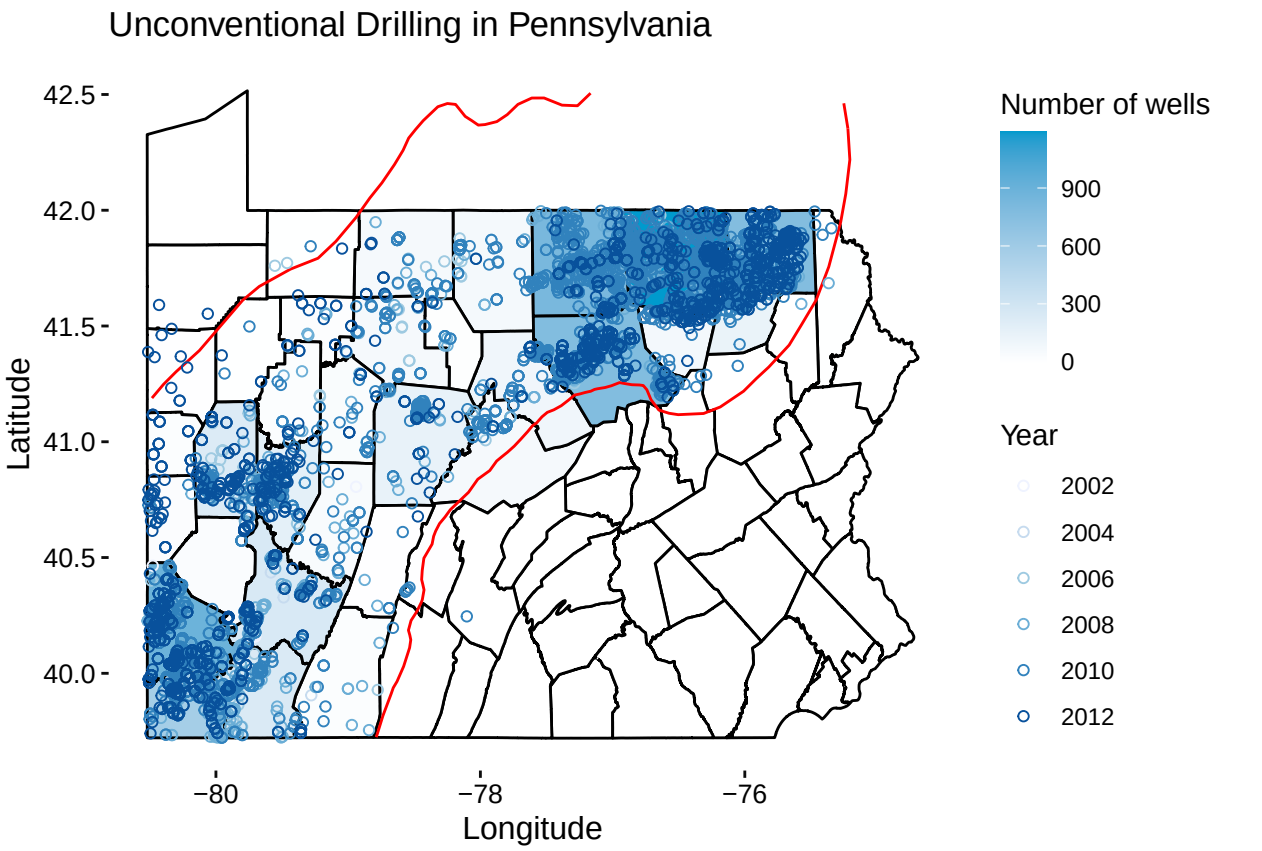
Since this is a spatial object, after all, let's go ahead and plot it. To do this, we'll `left_join()` our new column into our county dataframe:

```
paCounties <- left_join(paCounties, wellsInCty, by = "countyfpec")
paCounties <- mutate(paCounties, wells = ifelse(is.na(wells) == TRUE, 0, wells))
```

And plot:

```
countyPlot <- ggplot(data = bdryDF, aes(x = long, y = lat)) + geom_polygon(data = paCounties,
    aes(x = long, y = lat, group = id, fill = wells), color = "black") + geom_path(data = bdryDF,
    aes(x = long, y = lat), color = "red") + geom_point(data = wellsDF, aes(x = longitude,
    y = latitude, color = year), shape = 21) + scale_color_brewer(palette = "Blues") +
    xlim(counties@bbox[1, 1], counties@bbox[1, 2]) + ylim(counties@bbox[2, 1], counties@bbox[2,
    2]) + ggtitle("Unconventional Drilling in Pennsylvania") + xlab("Longitude") +
    ylab("Latitude") + scale_fill_gradient(low = "white", high = "deepskyblue3") +
    labs(fill = "Number of wells") + myMapThemeStuff + labs(color = "Year")
```

```
countyPlot
```



1.5.4 Regressions in space!

To do actual analysis, we want to merge this (hard-won) column into our county data. But our county data are currently a giant huge dataframe that's kind of unwieldy, because there's one row per lat-long combination. To make things easier for ourselves, let's grab the unique county data from the original shapefile:

```
countyData <- counties@data %>%
    as.data.frame() %>%
    mutate(id = rownames(counties@data)) %>%
    # for easier merging later
rename(fips = countyfpec)
```

Much more manageable. We're actually going to want to bring in some new demographic data for our analysis; all we need out of this county dataset is a way to link our well counts to our other dataset. Let's merge the `county` variable from our `countyData` dataset into our well counts, and only keep the few variables that we need:

```
countyWells <- left_join(paCounties, countyData, by = "id")
countyWells <- countyWells[, c("fips", "id", "wells")]
```

Next, let's bring in our (long-promised) demographic data:

```
countyDemogs <- read.csv("data/PA_county_data.csv") %>%
    # remove the row for the whole state
filter(NAME != "Pennsylvania")
```

```
countyDemogs <- countyDemogs[, c("COUNTY", "NAME", "Total.Population", "Median.Age",
    "Average.Household.Size")]

countyDemogs <- countyDemogs %>%
    rename(fips = "COUNTY", name = "NAME", totp = "Total.Population", medage = "Median.Age",
        avghhsize = "Average.Household.Size")
```

We need to combine this with our quasi-spatial county dataset. We can merge these two datasets on the county name, or better yet, the FIPS code. Let's do it:

```
countyWells$fips <- as.numeric(as.character(countyWells$fips))

analysisData <- left_join(countyDemogs, countyWells, by = "fips") %>%
    mutate(ones = 1)

head(analysisData)
```

```
##   fips         name   totp medage avghhsize id wells ones
## 1    1 Adams County 101407   41.3      2.56  1     0    1
## 2    1 Adams County 101407   41.3      2.56  1     0    1
## 3    1 Adams County 101407   41.3      2.56  1     0    1
## 4    1 Adams County 101407   41.3      2.56  1     0    1
## 5    1 Adams County 101407   41.3      2.56  1     0    1
## 6    1 Adams County 101407   41.3      2.56  1     0    1
```

Wahoo! We can finally run a regression!

```
myReg <- lm(wells ~ totp + medage + avghhsize, analysisData)
summary(myReg)
```

```
##
## Call:
## lm(formula = wells ~ totp + medage + avghhsize, data = analysisData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -201.83  -99.31  -58.64    0.73 1078.22
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.691e+02  1.560e+01   17.25   <2e-16 ***
## totp        -4.828e-05  1.792e-06  -26.95   <2e-16 ***
## medage       9.582e+00  1.906e-01   50.26   <2e-16 ***
## avghhsize   -2.331e+02  4.503e+00  -51.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.8 on 175563 degrees of freedom
## Multiple R-squared:  0.06199,    Adjusted R-squared:  0.06197
## F-statistic:  3867 on 3 and 175563 DF,  p-value: < 2.2e-16
```

... aaaand after all that work nothing is statistically significant. How anticlimactic.

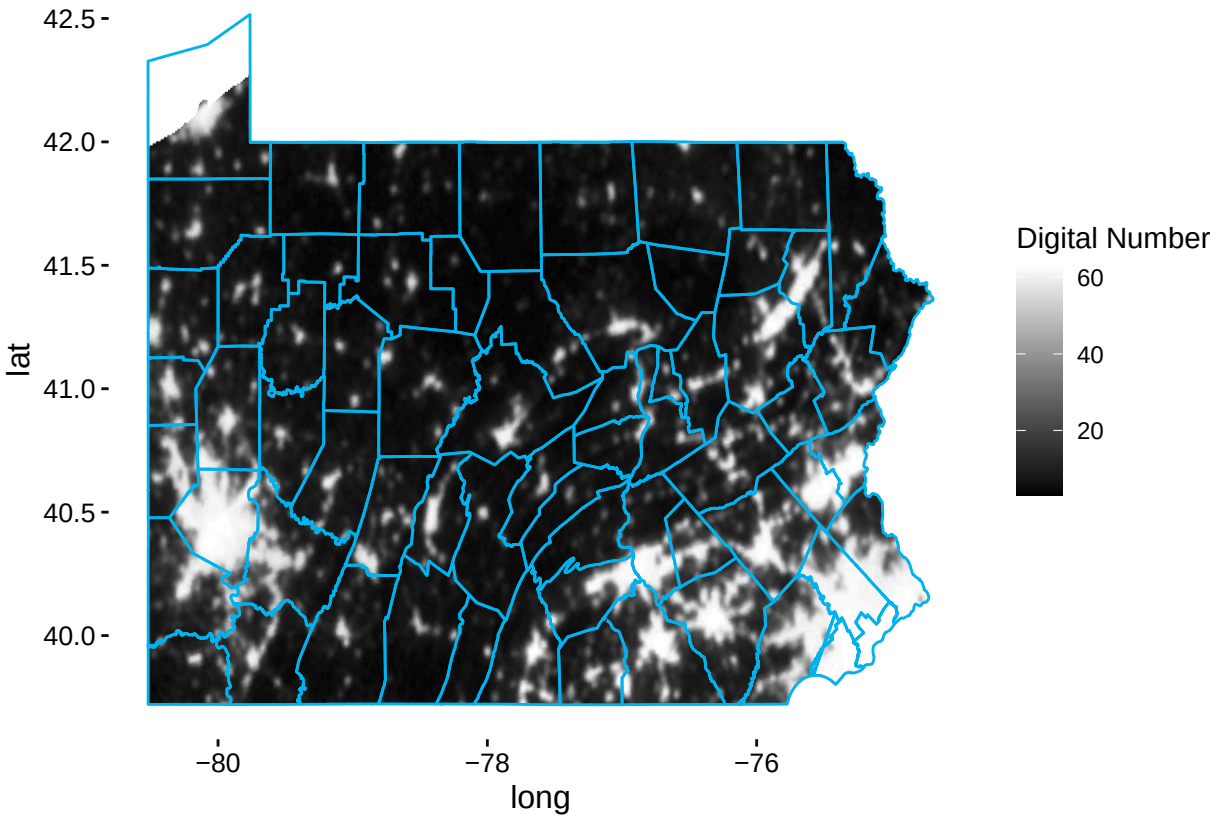
1.5.5 Pointillism

Let's bring in one last dataset - nighttime lights. This dataset grids up the world into approximately 1x1 km squares, and records a luminosity value, which is a measure of the amount of light emanating out of each pixel. This dataset is available annually; we use the 2012 version here. The original file was a TIFF image, which makes this dataset an raster : a giant gridded image. The original file also took up more than 1GB of space, so it has been trimmed down to just Pennsylvania, and exported as an ASCII text file. Let's go ahead and bring it into R, and make sure that R knows that we're treating it as a raster.

```
# make the raster layer
lights <- readAsciiGrid("data/palights.txt") %>%
    raster()
# create a database version
lightsDF <- readAsciiGrid("data/palights.txt") %>%
    as.data.frame()
names(lightsDF) <- c("dn", "long", "lat")
```

And plot:

```
lightsPlot <- ggplot(data = lightsDF, aes(x = long, y = lat)) + geom_raster(aes(fill = dn)) +
    geom_polygon(data = paCounties, aes(x = long, y = lat, group = id), color = "deepskyblue2",
        fill = "NA") + myMapThemeStuff + labs(fill = "Digital Number") + scale_fill_gradient(low = "bla
    high = "white")

lightsPlot
```



The next thing we want to do is to calculate the average lights value for each county (this might take a little while):

```
countyLights <- extract(lights, counties, fun = mean, na.rm = T, df = T) %>%
    as.data.frame()

countyLights <- mutate(countyLights, id = as.character(ID))

countyLights <- countyLights[, c("palights.txt", "id")]

names(countyLights) <- c("dn", "id")
```

And merge into our county data:

```
analysisData <- left_join(analysisData, countyLights, by = "id") %>%
    na.omit()
head(analysisData)
```

```
##   fips         name   totp medage avghhsize id wells ones       dn
## 1    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
## 2    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
## 3    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
## 4    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
## 5    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
## 6    1 Adams County 101407   41.3      2.56  1     0    1 18.21014
```

And again, we can run our regression:

```
myReg2 <- lm(wells ~ dn, analysisData)
summary(myReg2)
```

```
##
## Call:
## lm(formula = wells ~ dn, data = analysisData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -121.00  -97.67  -72.60   -0.71 1071.42
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 139.05639    0.86380  160.98   <2e-16 ***
## dn           -2.58997    0.03515  -73.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 206.2 on 175565 degrees of freedom
## Multiple R-squared:   0.03,  Adjusted R-squared:  0.02999
## F-statistic:  5430 on 1 and 175565 DF,  p-value: < 2.2e-16
```

There doesn't appear to be a correlation between the number of wells drilled in Pennsylvania and nighttime lights. Should we be worried? Not really - these datasets aren't perfect; the wells aren't all active anymore (and we haven't dealt with the time component of these data at all); there's not necessarily a clear reason to think that nighttime lights should be strongly related to fracking anyway (except for gas flaring - but that doesn't happen nearly as much in Pennsylvania as it does in North Dakota). Despite the lack of stars on this regression, I hope that this has been a useful exploration of the ways in which you can use R to manipulate spatial data.[1]

[1] As my econometrics professor once famously said, "This is not Hollywood - do not go looking for stars!"