MQE: Economic Inference from Data: Odds and Ends

Claire Duquennois

2020

Odds and Ends

- Non-Standard Standard Errors
  - Robust standard errors
  - Clustered standard errors
  - Newey West Standard Errors
  - Conley Standard errors
- Confidence intervals for prediction
  - For a particular average
  - For a particular unit
- Standardizing

Non-standard standard errors

A standard error estimates the uncertainty around an estimated parameter.

Formally we have

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

Just like calculating point estimates, it is incredibly important to get your standard errors right.

You have to know what you don't know!

- Robust standard errors
- Clustered standard errors

Robust standard errors

Using the diamonds data set from `ggplot2`:

```r
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
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

Robust standard errors

Regress price on carats and depth.

```r
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
```

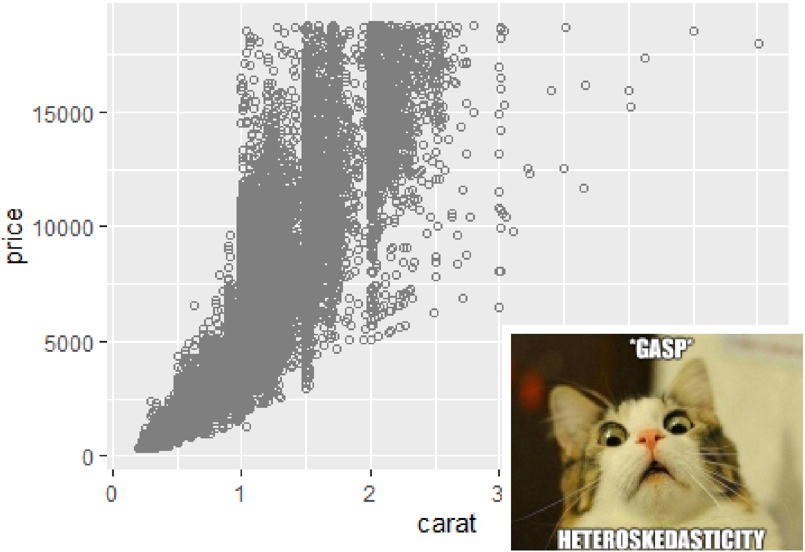
Robust standard errors

Cool.

Plot the data to check OLS assumptions:

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

Robust standard errors



Robust standard errors

You should have the econometric heebie jeebies.

Homoskedastic assumption needed for OLS is not valid!

- The higher the carat, the greater the variance in price.
- $\Rightarrow$ OLS standard errors are likely to be wrong.

Thankfully all is not lost!

Robust standard errors

Lets 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?

Eicker, Huber and White to the rescue!

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 $Var(b\hat{e}ta_1)$, 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".

Robust standard errors

How can we find these in R?

```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
```

Robust standard errors

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

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

Table 2

	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

Note: 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).

Econometricians Haiku

T-stats looks too good

Try cluster standard errors

significance gone.

from Angrist and Pischke 2008

Clustered standard errors

Suppose that every observation belongs to (only) one of G groups.

The assumption we make when we cluster:

- there is no correlation across groups
- we allow for arbitrary within-group correlation.

Clustered standard errors

Example: consider individuals within a village.

It may be reasonable to think that individuals' error terms are:

- correlated within a village
- aren't correlated across villages

Clustered standard errors

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.

Clustered standard errors

Clustering in R:

I use the `NOxEmissions` dataset from the `robustbase` package.

- hourly NO_x readings, including NO_x concentration, auto emissions and windspeed.
- 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:

Clustered standard errors

```
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, omit.stat = "all")
```

Table 3

	Dependent variable:	
	LNOx	
	(1)	(2)
sqrtWS	-0.864^{***}	-0.864^{***}
	(0.020)	(0.048)
Constant	5.559^{***}	5.559^{***}
	(0.029)	(0.065)
Note:	*p$<$0.1; **p$<$0.05; ***p$<$0.01	

Here, the regular standard errors are smaller than the clustered standard errors.

This need not necessarily be the case and depends on the correlation between observations within a cluster.

Newey West Standard Errors

For time series data.

Conley Standard Errors

For spatial data.

Confidence intervals for predictions

You know how to "predict" a value of the dependent variable, y, given certain values of the independent variables.

This prediction is just a guess, with uncertainty.

We can construct a confidence interval to give a range of possible values for this prediction.

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 $x_1, x_2 \ldots x_k$.

Confidence intervals for predictions

Using Wooldridge's birth weight data:

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

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

Confidence intervals for predictions

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

```r
#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
```

Confidence intervals for predictions: for a specific average

Our model gives us the expected value:

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

and our regression gives us an estimate of this:

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

$\hat{y}$ is the expected value of y given the particular values for the explanatory variables.

Confidence intervals for predictions: for a specific average

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,
- 2 older siblings (parity=3).

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

How do we find a standard error for $\hat{y}$ at these particular values of the explanatory variables?

Confidence intervals for predictions: for a specific average

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.

Confidence intervals for predictions: for a specific average

Recall that we have the following regression in mind

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

Then

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

Confidence intervals for predictions: for a specific average

If we modify the regression by subtracting our particular values from the independent variables, we get

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

Then

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

The new intercept is the predicted birthweight for babies with the particular values we are interested in.

If we run this regression in `R`, we can then grab the standard errors for the intercept.

Confidence intervals for predictions: for a specific average

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.

Confidence intervals for predictions: for a specific average

```r
#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
```

Confidence intervals for predictions: for a specific average

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:

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

Confidence Interval for prediction: a specific unit

A confidence interval for a particular average is not the same as a confidence interval for a particular individual.

For individual observations, we must account for the variance in the unobserved error, u_i, which measures our ignorance of the unobserved factors that affect y_i.

Confidence Interval for prediction: a specific unit

We want a confidence interval for $bweight_{i=1}$, the birthweight of baby $i = 1$, 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

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

Confidence Interval for prediction: a specific unit

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:

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

Confidence Interval for prediction: a specific unit

Finding the variance we get

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

Confidence Interval for prediction: a specific unit

There are two sources of variation in $\hat{u}_{i=1}$.

- the sampling error in $\widehat{bweight}_{i=1}$ which arises because we have estimated the population parameters β.
- the variance of the error in the population ($u_{i=1}$).

We can compute:

- $Var(\widehat{bweight}_{i=1})$ exactly the way we did before.
- $\hat{\sigma}^2$ from our regression output.

The 95% confidence interval for $bweight_{i=1}$ is then

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

Confidence Interval for prediction: a specific unit

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.

Confidence Interval for prediction: a specific unit

```r
#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
```

Confidence Interval for prediction: a specific unit

```r
#step 4: get the estimate of the variance
summary(lm(bwght~lfaminc_0+motheduc_0+parity_0,bwght))$sigma^2
```

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

The 95% confidence interval for a particular baby's birthweight with:

- family income of \$14,500 (ln(14.5=2.674)),
- a mother with 12 years of education
- 2 older siblings is:

$$SE = \sqrt{se(\tilde{\beta}_0)^2 + \hat{\sigma}^2} = \sqrt{(1.007^2) + 408.59} = 20.239$$

$$CI = [119.64 - 1.96 * (20.239); 119.64 + 1.96 * (20.239)]$$
$$= [79.972; 159.308]$$

Standardizing

Standardizing variables eliminates the units:

- makes it possible to compare the magnitude of estimates across independent variables.
- makes interpretation easier if you have variables with weird arbitrary units that are unfamiliar to people.

Standardizing

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

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

A linear regression must go through the point of averages:

- if we plugged in $\bar{x}_1$ and $\bar{x}_2$, we would predict $\bar{y}$:

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

Standardizing

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

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

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 regression into standard units:

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

Standardizing

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.

This is the standardized coefficient or "beta" coefficient.

It is also possible to only standardize some variables (you will need an intercept in this case).

Standardizing

In R, we can get these coefficients by using the `scale()` command in `R`.

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.

Standardizing

```
bwght<-bwght2

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

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

Standardizing

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

Table 4

	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
Note:				*p<0.1; **p<0.05; ***p<0.01

Standardizing

Interpreting column 1:

- A mother that is **a year** older predicts a birthweight that is 3.992 **grams** less (not significant).
- A father that is **a year** older predicts a birthweight that is 9.313 **grams** more (significant).

Interpreting column 2:

- A mother whose age is **one standard deviation higher** predicts a birthweight that is 0.033 **standard deviations** lower (not significant).
- A father whose age is **one standard deviation higher** predicts a birthweight that is 0.092 **standard deviations** higher (significant).

Interpreting column 3:

- A mother that is **a year** older predicts a birthweight that is 0.007 **standard deviations** lower (not significant).
- A father that is **a year** older predict a birthweight that is 0.016 **standard deviations** higher (significant).

Interpreting column 4:

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

Binary Dependent variables

What happens if the outcome variable is binary?

- Linear Probability Models
- Logits

Linear Probability Models

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

- 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:
  - y either changes from $0 \rightarrow 1$, from $1 \rightarrow 0$ or doesn't change.
- $\Rightarrow \beta_j$ measures the change in the probability of success when x_j changes by one unit holding all other factors constant.

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

Linear Probability Models

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

- *inlf* ("in the labor force") is a binary variable indicating labor force participation by a married woman in 1975.
- 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*)

Linear Probability Models

Using the `mroz` data that is part of the `wooldridge` package:

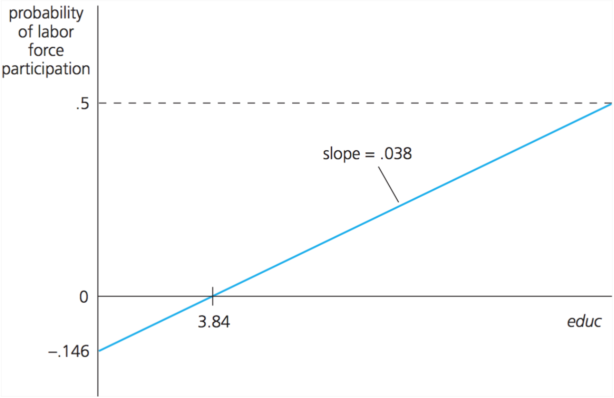
```r
mroz$exper2<-mroz$exper^2
reg1<-lm(inlf~nwifeinc+educ+exper+exper2+age+kidslt6+kidsge6, mroz)
summary(reg1)
```

```
##
## 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
```

Linear Probability Models

For a woman with: $nwifeinc = 50$, $exper = 5$, $age = 30$, $kidslt6 = 1$ and $kidsge6 = 0$, the relationship between years of education and the probability of being in the labor force is given by:

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



Linear Probability Model Drawbacks:

1) The predicted probabilities from our regression aren't bound between 0 and 1.

2) The outcome is implied to be linearly related to the independent variables. This is not possible with probabilities. (Ex: going from 0 to 4 young children reduces the probability by 0.262*4=1.048, 104.8 percentage points, which is impossible).

3) When y is a binary variable $Var(y|x) = p(x)[1 - p(x)]$. 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.

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

Logits differ from the linear probability model in the type of function 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

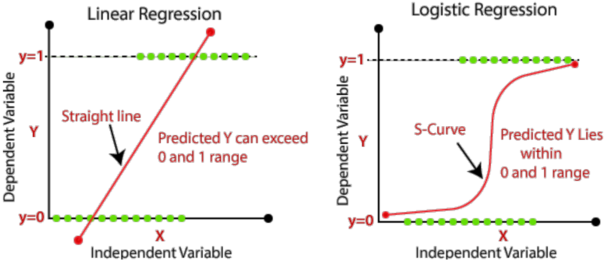
$$Pr(y = 1|x_1, x_2) = \frac{e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}{1 + e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}} = \Lambda(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$$

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

Logits

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),
- it has nice statistical properties (which we will not get into).



Logits

In linear probability models, the marginal effect, $\frac{\partial Pr(y=1|x)}{\partial x_j}$, of x_j on $Pr(y = 1|x)$ is constant.

Logit models are **non-linear**: the exact marginal effect of x_j on $Pr(y = 1|x)$ **changes depending on the other values of x**.

We can select 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.
- we could also specify any values of our independent variables to evaluate our marginal effects precisely at those values (Ex: at the mean values of the x_j's).

Logits

Working with Logits:

- `R` output for a logit will not give you a marginal effect. Instead it reports the **log-odds**.
- To get marginal effects, we need to request the marginal effects and specify which marginal effects we are interested.

Logits

Looking at labor force participation using the `mroz` data.

1) estimate a logistic regression with the `glm()` function.

2) 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)
##
```

Logits

To get the marginal effects:

3) run the estimated `glm()` object through the `margins()` function (from the **margins** package).

- the default returns the **average marginal effect**

```
library(margins)
```

```
## Warning: package 'margins' was built under R version 3.6.3
```

```
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
```

Logits

- you can 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  4.2893 0.0000  0.0207  0.0556
##    exper2 -0.0006 0.0002 -2.8205 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
```

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 in Las Vegas. We are interested in the probability that the favored team actually wins.

We can run the following regression:

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

- *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.

Logit vs. linear probability model

Using the pntsprd data from the wooldridge package:

```
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
```

Logit vs. linear probability model

```r
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
```

Logit vs. linear probability model

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
- by 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.

Logit vs. linear probability model

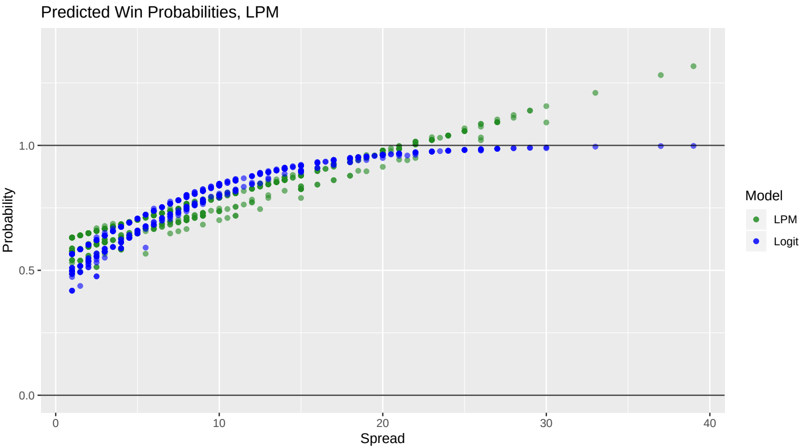
To see the difference, we plot the predicted probabilities

```
#generating the fitted values for both models
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")
```

Logit vs. linear probability model

```
plot
```



Spatial Data in R

Spatial Data in R

Lots of new spatial data becoming available.

With the right tools these can be used to explore all kinds of questions.

Spatial Data in R

Spatial data is just data like anything else. A

The main difference is that spatial data usually come in 2 (or even 3) dimensions (usually latitude and longitude).

Spatial Data in R

To deal with spatial data, we'll need a bunch of new packages:

- `GISTools`, `rgdal`, `rgeos`, `maptools`, `raster` (all spatial utilities),
- `broom` (this will let us turn spatial data into a format that ggplot2 can handle).
- `lubridate` to more easily deal with dates (because why not)
- `RColorBrewer` for new color palettes

Notes:

- The order in which you load packages can matter.
- other programs (often proprietary) are better at dealing with spatial data analysis. `R` does not have a unified spatial toolkit, meaning that different data types and formats and functions don't necessarily get along very well.

Spatial Data in R

```r
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)
```

What the frack?

We're going to use georeferenced data to look at unconventional oil and gas drilling in Pennsylvania.

We'll start with a spatial data file from the US Cansus Bureau with the shape of each county in the state.

This data comes in ArcGIS's the *shapefile* format.

```
counties <- shapefile("data/PA_counties.shp")
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, FUNC
## min values  :        42,        001,      42001,  Adams, Adams County,     06,        H1,   G4020,
## max values  :        42,        133,      42133,   York,  York County,     06,        H6,   G4020,
```

Shapefiles

What's in this object?

- This file contains polygons - outlines of shapes.
- We can check it was read correctly into R by checking its class:

```r
class(counties)
```

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

Perfect. This is `R`'s version of a polygon shapefile.

Shapefiles

What's actually in this dataset?

- We can figure this out by looking at its slots using`slotNames()`
- This is like calling `names()` on a simpler object

```
slotNames(counties)
```

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

What is all this?

- polygons: the spatial component of our spatial dataset. Each polygon is a long two-variable dataframe with longitudes (x) and latitudes (y) (like a giant connect-the-dots).
- data: a dataframe, where each row is actually associated with one of the polygons in our shapefile.

-`bbox` and `plotOrder`: tell the base plotting commands how to display the data.

- `proj4string`: tells R how to display and project the data

Shapefiles

Let's look at the data in this shapefile:

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

```
##   statefpec countyfpec cntyidfpec   nameec      namelsadec lsadec
## 0        42        001      42001    Adams    Adams County     06
## 1        42        027      42027   Centre   Centre County     06
## 2        42        023      42023  Cameron  Cameron County     06
## 3        42        025      42025   Carbon   Carbon County     06
## 4        42        029      42029  Chester  Chester County     06
## 5        42        045      42045 Delaware Delaware County     06
##   classfpec mtfccec funcstatec    alandec awaterec  intptlatec
## 0        H1   G4020          A 1342811635  8560860 +39.8694707
## 1        H1   G4020          A 2877281248  7879260 +40.9091673
## 2        H1   G4020          A 1026698470  5664145 +41.4382882
## 3        H1   G4020          A  988299800 15353755 +40.9178324
## 4        H1   G4020          A 1943960881 22601992 +39.9737772
## 5        H1   G4020          A  476398058 17503777 +39.9160594
##     intptlonec
## 0 -077.2177295
## 1 -077.8478195
## 2 -078.1983134
## 3 -075.7094276
## 4 -075.7503816
## 5 -075.4008573
```

- Mostly identifying information about each county
- Not that interesting: if we want good stuff we will have to merge it in.

Shapefiles: Projections

The world is a sphere (sorry Kyrie Irving) that we are trying to represent on our (flat) computer screens.

- The projection defines what dimensions to stretch to make this representation happen.
- Spatial objects (should) have a projection attached to them

```
counties@proj4string
```

```
## CRS arguments: +proj=longlat +datum=NAD83 +no_defs
```

This tells us that we are using:

- latitude and longitude to identify our data,
- the North American Datum 83 (a common choice for the US).

If you're going to combine multiple geographic datasets, it's important that they all use the same projection.

Let's plot!

1) we extract the shapefile data so we can use it later
2) we convert our polygon into a data frame that `ggplot2` can handle, using `broom`'s `tidy()` function.
3) we merge, or `join()`, the 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)
}
```

Let's plot!

```r
paCounties <- mapToDF(counties)
```

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

```r
head(paCounties)
```

```
##        long      lat order  hole piece group id statefpec countyfpec
## 1 -76.99950 39.79106     1 FALSE     1   0.1  0        42        001
## 2 -76.99943 39.79044     2 FALSE     1   0.1  0        42        001
## 3 -76.99939 39.79013     3 FALSE     1   0.1  0        42        001
## 4 -76.99939 39.79003     4 FALSE     1   0.1  0        42        001
## 5 -76.99931 39.78907     5 FALSE     1   0.1  0        42        001
## 6 -76.99929 39.78852     6 FALSE     1   0.1  0        42        001
##   cntyidfpec nameec   namelsadec lsadec classfpec mtfccec funcstatec
## 1      42001  Adams Adams County     06        H1   G4020          A
## 2      42001  Adams Adams County     06        H1   G4020          A
## 3      42001  Adams Adams County     06        H1   G4020          A
## 4      42001  Adams Adams County     06        H1   G4020          A
## 5      42001  Adams Adams County     06        H1   G4020          A
## 6      42001  Adams Adams County     06        H1   G4020          A
##      alandec awaterec  intptlatec   intptlonec
## 1 1342811635  8560860 +39.8694707 -077.2177295
## 2 1342811635  8560860 +39.8694707 -077.2177295
## 3 1342811635  8560860 +39.8694707 -077.2177295
## 4 1342811635  8560860 +39.8694707 -077.2177295
## 5 1342811635  8560860 +39.8694707 -077.2177295
## 6 1342811635  8560860 +39.8694707 -077.2177295
```

Let's plot!

Once we have this dataset, it's easy to plot using ggplot2.

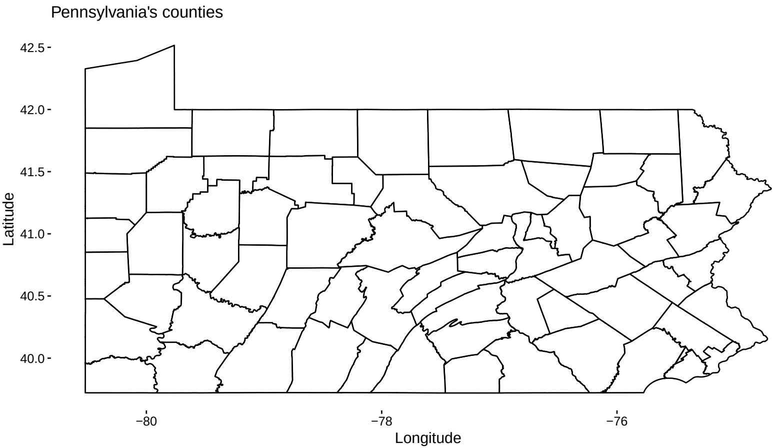
I start by defining a ggplot theme for my maps and then plot the data:

```
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(color = "gray5"),
    axis.text = element_text(color = "black", size = 10),
    axis.title = element_text(color = "black", size = 12),
    legend.key = element_blank()
)

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")
```

Let's plot!

```
paMap
```



A less boring map: Adding wells

Let's bring in another dataset, with the locations of unconventional wells drilled in Pennsylvania between 2002 and 2013 (courtesy of FracTracker Alliance).

```
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
## 1            GAS                      Active 40.46467 -80.47789
## 2            GAS                      Active 40.28316 -80.28402
## 3            GAS                      Active 40.80562 -78.93993
## 4  COMB. OIL&GAS Regulatory Inactive Status 40.15899 -79.70181
## 5  COMB. OIL&GAS                      Active 40.15602 -79.72340
## 6            GAS                      Active 40.43544 -79.58450
##   configuration unconventional
## 1 Vertical Well            Yes
## 2 Vertical Well            Yes
```

A less boring map: Adding wells

Before we add it to our plot we need to make sure that it uses the same projection.

It's currently a dataframe - let's turn it into a SpatialPointsDataFrame (like the polygon, except for points) and 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
```

A less boring map: Adding wells

Since the wells data does not have a projection, we have to assign it ourselves.

The website did not specify a projection so we select WGS84 (a common global projection) and then re-project it to match our county data's.

```
# 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
```

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

```
## Warning in proj4string(wells): CRS object has comment, which is lost in
## output
```

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

```
#convert back to a dataframe  so that ggplot2 can handle it
wellsDF <- as.data.frame(wells)
```

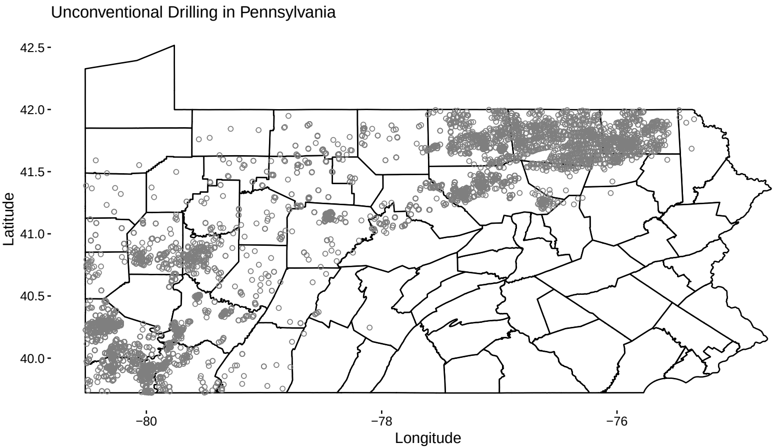
A less boring map: Adding wells

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 in Pennsylvania") +
    xlab("Longitude") +
    ylab("Latitude")
```

A less boring map!

```
paMap
```



Got the blue's?

Let's plot different colors by year of well drilling.

1) convert the spud date (drill date) variable to date format,
2) extract the year (let's actually make pairs of years)
3) convert this into a factor.

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

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

Got the blue's?

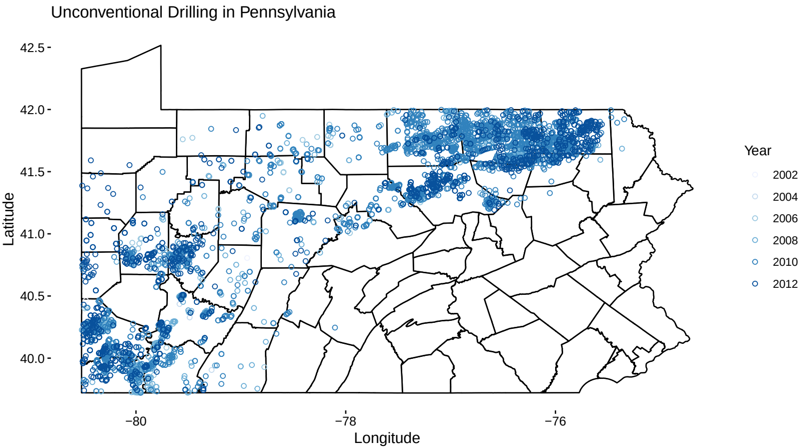
This is also a great excuse to bring in a great R packages: `RColorBrewer`.

You can see all of the available color palettes by typeing `display.brewer.all()` into your consol.

```
paMap <- ggplot() +
    geom_polygon(data = paCounties, aes(x = long, y = lat,
    geom_point(data = wellsDF, aes(x = longitude, y = latit
    scale_color_brewer(palette="Blues") +
    myMapThemeStuff +
    ggtitle("Unconventional Drilling in Pennsylvania") +
    xlab("Longitude") +
    ylab("Latitude") +
    labs(color = "Year")
```

Got the blue's?

```
paMap
```



Playing around

Let's bring in some new data from the EIA: a shapefile of the Marcellus shale play - the rock formation from which you can 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
```

```
playBdry@proj4string
```

```
## CRS arguments: +proj=longlat +datum=WGS84 +no_defs
```

This file is WGS84. We'll have to convert it:

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

Use our mapToDF() function from earlier to convert this into a dataframe:

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

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

Playing around

```
bigPlot <- ggplot(data = bdryDF, aes(x = long, y = lat)) +
        geom_polygon(data = paCounties, aes(x = long, y= la
        geom_path(data = bdryDF, aes(x = long, y = lat), co
        geom_point(data = wellsDF, aes(x = longitude, y = l
        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")
```

Playing around

```
bigPlot
```