WEEK 10: REGRESSION

THOMAS ELLIOTT

Now that we've explored various ways to manage data in R, it now seems appropriate to discuss what you can do with the data once it has been cleaned and formatted. The most common quantitative analysis is regression, so this week will focus on a couple different forms of regression, as well as ways to format the analysis for publication automatically.

1. Ordinary Least Squares Regression

The most straightforward form of regression is OLS, and this is accomplished in R with the `lm()` function (for linear model). To use this function, we first need to discuss formula objects.

1.1. **Formulas.**

In R, there is a special formula object which is used in a variety of contexts, but most commonly when running regressions. The formula object defines the regression model. It's most basic form looks like:

```
y ~ x1 + x2 + x3
```

The left hand side defines the dependent variable and the right hand side defines the independent variables. Interactions can be specified by :

```
y ~ x1 + x2 + x1:x2
```

Or, an alternative way to write the above would be:

```
y ~ x1 * x2
```

The `lm()` function takes as arguments a formula defining the model, and a data frame containing the data on which to run the regression.

```
sal.lm<-lm(salary ~ age + sex + white + married + bachelors,data=hire)
```

Notice that variable names are bare in the formula, meaning they are not in quotation marks nor do they need to be preceded by `hire$` as we would usually need to do to access them. This is because `lm()` knows to look within the data frame if one is passed. The `sal.lm` object now contains the results of the regression. Technically, it is a list with a special class attached so that R knows which `print` and `summary` functions (along with others) to use with it. We can see the internals of the object by looking at the names of the elements of the list:

```
> names(sal.lm)
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "as
 [7] "qr"            "df.residual"   "na.action"     "xlevels"       "call"          "te
```

Date: March 16, 2016.

```
[13] "model"
```

Each of these elements can be accessed just like you would access elements of any list. To see the actual results of the regression, use the `summary()` function:

```
> summary(sal.lm)

Call:
lm(formula = salary ~ age + sex + white + married + bachelors,
    data = hire)

Residuals:
    Min      1Q  Median      3Q     Max
-123.51  -27.49   -5.66   13.19  592.08

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 127.5571     4.5433  28.076  < 2e-16 ***
age           2.0756     0.1247  16.651  < 2e-16 ***
sex         -53.2052     3.0377 -17.515  < 2e-16 ***
white         6.7969     2.2785   2.983  0.00288 **
married       6.9236     2.2228   3.115  0.00186 **
bachelors    49.1997     2.6611  18.489  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57.53 on 3122 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.3108,Adjusted R-squared:  0.3097
F-statistic: 281.6 on 5 and 3122 DF,  p-value: < 2.2e-16
```

The summary function displays the coefficients, standard errors, and significance tests, plus statistics about the model fit over all.

Something to note: the summary function itself calculates the standard errors and significance tests of the coefficients, so these statistics are not accessible in the object returned by `lm()`. However, you can save the object returned by `summary()` and it will contain a coefficients element which holds the value of the coefficients as well as the standard errors, t values, and p values.

```
> sal.sum<-summary(sal.lm)
> names(sal.sum)
 [1] "call"          "terms"         "residuals"     "coefficients"  "aliased"       "si
 [7] "df"            "r.squared"     "adj.r.squared" "fstatistic"    "cov.unscaled"  "na
> sal.sum$coefficients
              Estimate Std. Error    t value      Pr(>|t|)
(Intercept) 127.557057  4.5432871  28.075941 7.567244e-155
age           2.075578  0.1246549  16.650597  1.046265e-59
sex         -53.205198  3.0377420 -17.514719  1.371347e-65
```

```
white         6.796869  2.2785386   2.982995  2.876524e-03
married       6.923561  2.2228023   3.114790  1.857461e-03
bachelors    49.199684  2.6610760  18.488643  1.652980e-72
```

2. Logistic Regression

A second common type of regression is logistic regression, when the dependent variable is binary. While Stata has a `logit` command specifically for logistic regression, in R logistic regression is accomplished with the `glm()` function, for generalized linear models. `glm()` takes as arguments the formula defining the model, the data frame to use, and the family of distribution to use. For logistic regression, we will use the binomial family, which defaults to a logit link function. Thus, to run a logistic regression you would use the following call:

```
except.glm<-glm(except ~ age + sex + white + married + bachelors,family="binomial",data=
```

The family argument can be a character string naming the family, or the result of a call to a family function (see `?family` for available family functions and their arguments), so `glm()` is actually an incredibly powerful function for performing a wide variety of regressions.

As with `lm()`, use `summary()` to see the results of the regression:

```
> summary(except.glm)

Call:
glm(formula = except ~ age + sex + white + married + bachelors,
    family = "binomial", data = hire)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9071  -0.3569  -0.2052  -0.1763   3.0468

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.982926   0.303898 -13.106  < 2e-16 ***
age          0.059496   0.007877   7.553 4.24e-14 ***
sex         -1.779297   0.146392 -12.154  < 2e-16 ***
white        0.534533   0.171205   3.122   0.0018 **
married      0.128163   0.146985   0.872   0.3832
bachelors    2.412536   0.148816  16.211  < 2e-16 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2171.6  on 3109  degrees of freedom
Residual deviance: 1448.4  on 3104  degrees of freedom
  (21 observations deleted due to missingness)
AIC: 1460.4
```

```
Number of Fisher Scoring iterations: 6
```

Again, the coefficients, standard errors, and p-values are reported, as well as details about the model fit. Here, since `glm()` uses MLE to calculate the coefficients, deviance and AIC scores are reported. As an aside, specifying the "gaussian" family in `glm()` will replicate the results of `lm()` but as derived by MLE.

As with OLS, the `summary()` function itself calculates the standard errors and significance tests for the coefficients, so if you want to access those directly you'll want to save the summary object:

```
> except.sum<-summary(except.glm)
> except.sum$coefficients
               Estimate  Std. Error     z value     Pr(>|z|)
(Intercept) -3.98292568 0.303898218 -13.1061173 3.037558e-39
age          0.05949614 0.007876858   7.5532837 4.244190e-14
sex         -1.77929716 0.146392246 -12.1543129 5.441735e-34
white        0.53453349 0.171204544   3.1221922 1.795097e-03
married      0.12816259 0.146984673   0.8719453 3.832382e-01
bachelors    2.41253582 0.148816466  16.2114844 4.183407e-59
```

3. Formatting for Publication

Like Stata, R has a number of third party packages for formatting results in publication-ready formats. The most popular right now is `stargazer` because of its versatility and the aesthetics of the tables it produces. The `stargazer()` function does the bulk of the work. It takes as arguments any number of regression objects (those returned by `lm()`, `glm()`, and a variety of others), and outputs the results in a publication ready table. As an example, here is a plain text output from the two models we ran earlier:

```
> stargazer(sal.lm,except.glm,type="text")

=======================================================
                            Dependent variable:
                    -----------------------------------
                         salary            except
                          OLS             logistic
                          (1)               (2)
-------------------------------------------------------
age                     2.076***          0.059***
                        (0.125)           (0.008)

sex                   -53.205***         -1.779***
                        (3.038)           (0.146)

white                   6.797***          0.535***
                        (2.279)           (0.171)
```

```
married                      6.924***          0.128
                              (2.223)          (0.147)

bachelors                    49.200***        2.413***
                              (2.661)          (0.149)

Constant                    127.557***        -3.983***
                              (4.543)          (0.304)

-------------------------------------------------------
Observations                   3,128            3,110
R2                             0.311
Adjusted R2                    0.310
Log Likelihood                                -724.224
Akaike Inf. Crit.                             1,460.448
Residual Std. Error    57.532 (df = 3122)
F Statistic         281.628*** (df = 5; 3122)
=======================================================
Note:                        *p<0.1; **p<0.05; ***p<0.01
```

Stargazer can also output the above in LaTeX or in HTML. The formatted LaTeX would look like Table 1.

To generate a formatted table you can use in Word, you will want to output HTML, and save it to a file that you can then open in word:

```
stargazer(sal.lm,except.glm,type="html",out="hire_regress.html")
```

Opening hire_regress.html in Word should preserve the formatting of the table you can then copy into your paper.

Stargazer has a very large number of options to modify the formatting of the tables, including being able to provide custom labels for the variables. Check out its help file and its vignette for how to take advantage of these options.

Table 1.

	Dependent variable:	
	salary	except
	OLS	*logistic*
	(1)	(2)
age	2.076^{***}	0.059^{***}
	(0.125)	(0.008)
sex	-53.205^{***}	-1.779^{***}
	(3.038)	(0.146)
white	6.797^{***}	0.535^{***}
	(2.279)	(0.171)
married	6.924^{***}	0.128
	(2.223)	(0.147)
bachelors	49.200^{***}	2.413^{***}
	(2.661)	(0.149)
Constant	127.557^{***}	-3.983^{***}
	(4.543)	(0.304)
Observations	3,128	3,110
R^2	0.311	
Adjusted R^2	0.310	
Log Likelihood		−724.224
Akaike Inf. Crit.		1,460.448
Residual Std. Error	57.532 (df = 3122)	
F Statistic	281.628^{***} (df = 5; 3122)	
Note:		*p<0.1; **p<0.05; ***p<0.01