Motivation

Basic linear regression in R is super easy. Just use the lm() function, summarize the model with summary() and you're done. But as researchers we need more than that. What if our model is more complicated, like if our errors are homoskedastic? And once we have run our regressions with our adjusted standard errors, how do we export the results for presentation in word or latex documents? This post will go over exactly these things, with the help of the stargazer package created by my fellow Harvard grad student Marek Hlavac. More posts about stargazer can be found here on this blog.

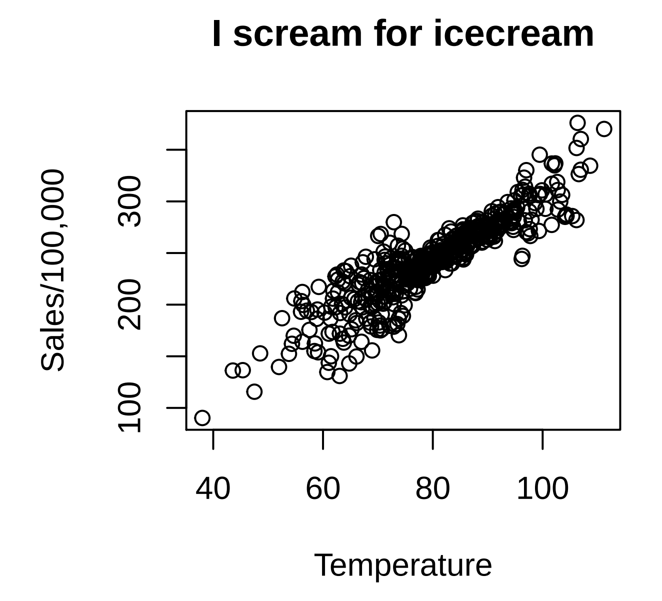
Alright so today we are going to analyze how temperature affects ice cream sales. We just make ourselves a little dataset like so:

```
set.seed(5)
temp<-rnorm(500, mean=80, sd=12)
sales<-2+temp*3+ifelse(temp<75, rnorm(500, mean=0, sd=25),
         ifelse(temp>95, rnorm(500, mean=0, sd=25), rnorm(500, mean=0, sd=8)))
female<-rnorm(500, mean=.5, sd=.01)
icecream<-as.data.frame(cbind(temp, sales, female))
head(icecream)

##      temp sales female
## 1  69.91 209.6 0.5155
## 2  96.61 323.0 0.4934
## 3  64.93 227.5 0.5010
## 4  80.84 256.8 0.4914
## 5 100.54 306.7 0.4867
## 6  72.77 210.4 0.4840
```

So there we have the temperature of 500 cities, sales per 100,000 people, and gender ratio of each of these cities. Let's plot out the temperature and sales.

```
plot(icecream$temp, icecream$sales,
     xlab="Temperature",
     ylab="Sales/100,000",
     main="I scream for icecream")
```



Alright so we definitely see a relationship between the temperature in a city and the sales of icecream. However, we notice that our errors are not exactly homoskedastic. We see that for both lower temperatures and very high temperatures, the variance is larger than the middle temperatures. So we will have to deal with that.

Let's start by running a linear regression using the lm() function and summarizing the results:

```
fit1 <- lm(sales ~ temp + female, data = icecream)
summary(fit1)

##
## Call:
## lm(formula = sales ~ temp + female, data = icecream)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -61.17  -7.90   0.26   8.70  58.03
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.4209    39.6823    0.16     0.87
## temp          3.0074     0.0653   46.09   <2e-16 ***
## female       -7.6077    78.1848   -0.10     0.92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.6 on 497 degrees of freedom
## Multiple R-squared:  0.811,Adjusted R-squared:  0.81
## F-statistic: 1.06e+03 on 2 and 497 DF,  p-value: <2e-16

names(summary(fit1))

##  [1] "call"          "terms"         "residuals"     "coefficients"  "aliased"       "sigma"         "df"
## [10] "fstatistic"    "cov.unscaled"
```

We see the strong association between temperature and icecream sales, but no relationship between the gender ratio and icecream. Don't forget that you can extract many things from the summary of the lm object (read up on extracting information using names() in this post. Now let's deal with the heteroskedasticity. We know that under the constant variance assumption,

$$V[\hat{\beta}] = (X'X)^{-1}X'\Sigma X(X'X)^{-1}$$

where

$$\Sigma = \sigma^2 I$$

and we can find an unbiased estimate of σ^2 by summing the squared the residuals. However, in this case, we see that the variance is not constant, so the naive covariance of the OLS estimator is biased. To fix this we employ the White Huber, or sandwich, method of robust standard errors. The sandwich method assumes that

$$\Sigma = diag(\sigma_1^2, ..., \sigma_n^2)$$

and again the $\hat{\sigma}_i^2$ are estimated via the residuals $\hat{u}_i^2$. In R, we can use the sandwich package to estimate robust standard errors this way:

```
library(sandwich)
cov.fit1 <- vcovHC(fit1, type = "HC")
rob.std.err <- sqrt(diag(cov.fit1))
```

The vcovHC() function returns the variance-covariance matrix under the assumption of "HC" (Heteroskedasticity-consistent) estimation. We then take the diagonal of this matrix and square root it to calculate the robust standard errors. You could do this in one line of course, without creating the cov.fit1 object.

Now, we can put the estimates, the naive standard errors, and the robust standard errors together in a nice little table. We save the naive standard errors into a vector and then column bind the three columns together like so:

```
naive.std.err <- summary(fit1)$coefficients[, 2]
estimate.table <- cbind(Estimate = coef(fit1), `Naive SE` = naive.std.err, `Robust SE` = rob.std.err)
estimate.table

##             Estimate Naive SE Robust SE
## (Intercept)    6.421 39.68231  44.14922
## temp           3.007  0.06526   0.08316
## female        -7.608 78.18482  88.96965
```

That's a good start. We see that the robust standard errors are larger than the naive. But of course we want p-values and, even better, upper and lower 95% confidence intervals, so we can add those to the table:

```
better.table <- cbind("Estimate" = coef(fit1),
                      "Naive SE" = naive.std.err,
                      "Pr(>|z|)" = 2 * pt(abs(coef(fit1)/naive.std.err),
                                          df=nrow(icecream)-2,
                                          lower.tail = FALSE),
                      "LL" = coef(fit1) - 1.96 * naive.std.err,
                      "UL" = coef(fit1) + 1.96 * naive.std.err,
                      "Robust SE" = rob.std.err,
                      "Pr(>|z|)" = 2 * pt(abs(coef(fit1)/rob.std.err),
                                          df=nrow(icecream)-2,
                                          lower.tail = FALSE),
                      "LL" = coef(fit1) - 1.96 * rob.std.err,
                      "UL" = coef(fit1) + 1.96 * rob.std.err)
rownames(better.table)<-c("Constant", "Temperature", "Gender Ratio")
better.table

##              Estimate Naive SE  Pr(>|z|)       LL      UL Robust SE  Pr(>|z|)       LL     UL
## Constant        6.421 39.68231 8.715e-01  -71.356  84.198  44.14922 8.844e-01  -80.112  92.95
## Temperature     3.007  0.06526 9.387e-182   2.879   3.135   0.08316 2.125e-141   2.844   3.17
## Gender Ratio   -7.608 78.18482 9.225e-01 -160.850 145.635  88.96965 9.319e-01 -181.988 166.77
```

A bit wide, but there we are. We are finished with our linear regression analysis. But now we would like to export this into a pdf or word document so we can send it to our advisor who will be very excited about this cutting edge icecream research. There are two options that I'll go over, xtable() and stargazer().

xtable

The first option is xtable. You can put in just about any object (dataframe, table, matrix, lm, glm, etc) and it will create latex code for you that you can copy and paste into latex. This is how the resulting table will look in latex:

```
xtable(fit1)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	6.4209	39.6823	0.16	0.8715
temp	3.0074	0.0653	46.09	0.0000
female	-7.6077	78.1848	-0.10	0.9225

But now what we really want is to compare the naive to the robust in one table. If we want to use xtable, we will have to put those vectors together ourselves, the way we have done above. You can't input multiple models into the xtable function.

```
xtable(better.table, digits = 3, caption = "Comparison of Naive and Robust SEs")
```

	Estimate	Naive SE	Pr(>|z|)	LL	UL	Robust SE	Pr(>|z|)	LL	UL
Constant	6.421	39.682	0.872	-71.356	84.198	44.149	0.884	-80.112	92.953
Temperature	3.007	0.065	0.000	2.879	3.135	0.083	0.000	2.844	3.170
Gender Ratio	-7.608	78.185	0.923	-160.850	145.635	88.970	0.932	-181.988	166.773

Table 1: Comparison of Naive and Robust SEs

This table is not all that attractive though - it's too wide for the amount of information it's imparting and it's not how we are used to seeing regression results. We can move on to another way of exporting tables that is better.

stargazer

The package I encourage you to try is called stargazer. It can make the task of creating tables and reporting results easier and more attractive. The basic function is just stargazer(fit1), which prints out latex output into the R console. You can copy the latex code and paste it into a latex document. Here is the output for the basic stargazer table produced just from the model itself with all of the function defaults:

```
stargazer(fit1)
```

Table 2:

	Dependent variable:
	sales
temp	3.007^{***}
	(0.065)
female	-7.608
	(78.190)
Constant	6.421
	(39.680)
Observations	500
R^2	0.811
Adjusted R^2	0.810
Residual Std. Error	17.590
F Statistic	1,064.000
Note:	*p<0.1; **p<0.05; ***p<0.01

If you want a text file instead to put into a word document, there is an option for that,

```
stargazer(fit1, type = "text", out = "reg_results.txt")
```

and it will export a text file of the same table.

So this is nice, but the great part about it is that if we have multiple models like we do with our icecream example, we can juxtapose them without having to create a table ourselves using cbind. We just put in as many model objects as we want and it will create that many columns. Then we can adjust the standard errors of the models. Finally, we can change a lot of the physical appearance with stargazer's many customizable options.

Here is the code:

```
stargazer(fit1, fit1,
          se = list(naive.std.err,rob.std.err),
          title="Regression Results",
          dep.var.labels=c("Icecream sales"),
          covariate.labels=c("Temperature","Gender ratio"),
          no.space=TRUE,
          omit.stat=c("LL","ser","f", "rsq"),
          column.labels=c("Naive SE", "Robust SE"),
          dep.var.caption="",
          model.numbers=FALSE)
```

This is a lot of code in one bit but all of the options are very self-explanatory (there's also descriptions of every option in the help file). So in this table we are going to:

- Juxtapose two models - the linear model with naive SEs and the linear model with robust SEs - so we put in **fit1, fit1** which takes the same coefficients for both columns of the table.
- Then specify the two different standard errors which we list together: **se=list(naive.std.err, rob.std.err)**. This is not necessary if you just go with the model produced standard errors.
- Then we add a title for the whole thing
- Add a dependent variable label and some covariate labels
- Remove all the spaces
- Take out some of the default stats that show up
- Label the columns
- Take out the dependent variable caption
- and take out the model numbers.

Table 3: Regression Results

	Icecream sales	
	Naive SE	Robust SE
Temperature	3.007^{***}	3.007^{***}
	(0.065)	(0.083)
Gender ratio	-7.608	-7.608
	(78.190)	(88.970)
Constant	6.421	6.421
	(39.680)	(44.150)
Observations	500	500
Adjusted R^2	0.810	0.810
Note:	*p<0.1; **p<0.05; ***p<0.01	

Looks pretty great! So there you go, in this way you can conduct linear regression with robust standard errors and then report your results in a physically attractive and easy way.