First of all, is it heteros**k**edasticity or heteros**c**edasticity?H eteros**k**edasticity is the proper spelling, because when transliterating Greek words, scientists use the Latin letter k in place of the Greek letter κ (kappa). κ sometimes is transliterated as the Latin letter c, but only when these words entered the English language through French, such  
as scepter.

Now that this is out of the way, we can get to the meat of this blogpost (foreshadowing pun).  
A random variable is said to be heteroskedastic, if its variance is not constant. For example,  
the variability of expenditures may increase with income. Richer families may spend a similar  
amount on groceries as poorer people, but some rich families will sometimes buy expensive  
items such as lobster. The variability of expenditures for rich families is thus quite large.  
However, the expenditures on food of poorer families, who cannot afford lobster, will not vary much.  
Heteroskedasticity can also appear when data is clustered; for example, variability of  
expenditures on food may vary from city to city, but is quite constant within a city.

To illustrate this, let’s first load all the packages needed for this blog post:

library(robustbase)

library(tidyverse)

library(sandwich)

library(lmtest)

library(modelr)

library(broom)

First, let’s load and prepare the data:

data("education")

education <- education %>%

rename(residents = X1,

per\_capita\_income = X2,

young\_residents = X3,

per\_capita\_exp = Y,

state = State) %>%

mutate(region = case\_when(

Region == 1 ~ "northeast",

Region == 2 ~ "northcenter",

Region == 3 ~ "south",

Region == 4 ~ "west"

)) %>%

select(-Region)

I will be using the education data sat from the {robustbase} package. I renamed some columns  
and changed the values of the Region column. Now, let’s do a scatterplot of per capita expenditures  
on per capita income:

ggplot(education, aes(per\_capita\_income, per\_capita\_exp)) +

geom\_point() +

theme\_dark()

It would seem that, as income increases, variability of expenditures increases too. Let’s look  
at the same plot by region:

ggplot(education, aes(per\_capita\_income, per\_capita\_exp)) +

geom\_point() +

facet\_wrap(~region) +

theme\_dark()

I don’t think this shows much; it would seem that observations might be clustered, but there are  
not enough observations to draw any conclusion from this plot (in any case, drawing conclusions  
from plots only is dangerous).

Let’s first run a good ol’ linear regression:

lmfit <- lm(per\_capita\_exp ~ region + residents + young\_residents + per\_capita\_income, data = education)

summary(lmfit)

##

## Call:

## lm(formula = per\_capita\_exp ~ region + residents + young\_residents +

## per\_capita\_income, data = education)

##

## Residuals:

## Min 1Q Median 3Q Max

## -77.963 -25.499 -2.214 17.618 89.106

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -467.40283 142.57669 -3.278 0.002073 \*\*

## regionnortheast 15.72741 18.16260 0.866 0.391338

## regionsouth 7.08742 17.29950 0.410 0.684068

## regionwest 34.32416 17.49460 1.962 0.056258 .

## residents -0.03456 0.05319 -0.650 0.519325

## young\_residents 1.30146 0.35717 3.644 0.000719 \*\*\*

## per\_capita\_income 0.07204 0.01305 5.520 1.82e-06 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 39.88 on 43 degrees of freedom

## Multiple R-squared: 0.6292, Adjusted R-squared: 0.5774

## F-statistic: 12.16 on 6 and 43 DF, p-value: 6.025e-08

Let’s test for heteroskedasticity using the Breusch-Pagan test that you can find in the {lmtest}  
package:

bptest(lmfit)

##

## studentized Breusch-Pagan test

##

## data: lmfit

## BP = 17.921, df = 6, p-value = 0.006432

This test shows that we can reject the null that the variance of the residuals is constant,  
thus heteroskedacity is present. To get the correct standard errors, we can use the vcovHC()  
function from the {sandwich} package (hence the choice for the header picture of this post):

lmfit %>%

vcovHC() %>%

diag() %>%

sqrt()

## (Intercept) regionnortheast regionsouth regionwest

## 311.31088691 25.30778221 23.56106307 24.12258706

## residents young\_residents per\_capita\_income

## 0.09184368 0.68829667 0.02999882

By default vcovHC() estimates a heteroskedasticity consistent (HC) variance covariance  
matrix for the parameters. There are several ways to estimate such a HC matrix, and by default  
vcovHC() estimates the “HC3” one.

We see that the standard errors are much larger than before! The intercept and regionwest variables  
are not statistically significant anymore.

You can achieve the same in one single step:

coeftest(lmfit, vcov = vcovHC(lmfit))

##

## t test of coefficients:

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -467.402827 311.310887 -1.5014 0.14056

## regionnortheast 15.727405 25.307782 0.6214 0.53759

## regionsouth 7.087424 23.561063 0.3008 0.76501

## regionwest 34.324157 24.122587 1.4229 0.16198

## residents -0.034558 0.091844 -0.3763 0.70857

## young\_residents 1.301458 0.688297 1.8908 0.06540 .

## per\_capita\_income 0.072036 0.029999 2.4013 0.02073 \*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

It’s is also easy to change the estimation method for the variance-covariance matrix:

coeftest(lmfit, vcov = vcovHC(lmfit, type = "HC0"))

##

## t test of coefficients:

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -467.402827 172.577569 -2.7084 0.009666 \*\*

## regionnortheast 15.727405 20.488148 0.7676 0.446899

## regionsouth 7.087424 17.755889 0.3992 0.691752

## regionwest 34.324157 19.308578 1.7777 0.082532 .

## residents -0.034558 0.054145 -0.6382 0.526703

## young\_residents 1.301458 0.387743 3.3565 0.001659 \*\*

## per\_capita\_income 0.072036 0.016638 4.3296 8.773e-05 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

As I wrote above, by default, the type argument is equal to “HC3”.

Another way of dealing with heteroskedasticity is to use the lmrob() function from the  
{robustbase} package. This package is quite interesting, and offers quite a lot of functions  
for robust linear, and nonlinear, regression models. Running a robust linear regression  
is just the same as with lm():

lmrobfit <- lmrob(per\_capita\_exp ~ region + residents + young\_residents + per\_capita\_income,

data = education)

summary(lmrobfit)

##

## Call:

## lmrob(formula = per\_capita\_exp ~ region + residents + young\_residents + per\_capita\_income,

## data = education)

## \--> method = "MM"

## Residuals:

## Min 1Q Median 3Q Max

## -57.074 -14.803 -0.853 24.154 174.279

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -156.37169 132.73828 -1.178 0.24526

## regionnortheast 20.64576 26.45378 0.780 0.43940

## regionsouth 10.79694 29.42747 0.367 0.71549

## regionwest 45.22589 33.07950 1.367 0.17867

## residents 0.03406 0.04412 0.772 0.44435

## young\_residents 0.57896 0.25512 2.269 0.02832 \*

## per\_capita\_income 0.04328 0.01442 3.000 0.00447 \*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Robust residual standard error: 26.4

## Multiple R-squared: 0.6235, Adjusted R-squared: 0.571

## Convergence in 24 IRWLS iterations

##

## Robustness weights:

## observation 50 is an outlier with |weight| = 0 ( < 0.002);

## 7 weights are ~= 1. The remaining 42 ones are summarized as

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## 0.05827 0.85200 0.93870 0.85250 0.98700 0.99790

## Algorithmic parameters:

## tuning.chi bb tuning.psi refine.tol

## 1.548e+00 5.000e-01 4.685e+00 1.000e-07

## rel.tol scale.tol solve.tol eps.outlier

## 1.000e-07 1.000e-10 1.000e-07 2.000e-03

## eps.x warn.limit.reject warn.limit.meanrw

## 1.071e-08 5.000e-01 5.000e-01

## nResample max.it best.r.s k.fast.s k.max

## 500 50 2 1 200

## maxit.scale trace.lev mts compute.rd fast.s.large.n

## 200 0 1000 0 2000

## psi subsampling cov

## "bisquare" "nonsingular" ".vcov.avar1"

## compute.outlier.stats

## "SM"

## seed : int(0)

This however, gives you different estimates than when fitting a linear regression model.  
The estimates should be the same, only the standard errors should be different. This is because  
the estimation method is different, and is also robust to outliers (at least that’s my understanding,  
I haven’t read the theoretical papers behind the package yet).

Finally, it is also possible to bootstrap the standard errors. For this I will use the  
bootstrap() function from the {modelr} package:

resamples <- 100

boot\_education <- education %>%

modelr::bootstrap(resamples)

Let’s take a look at the boot\_education object:

boot\_education

## # A tibble: 100 x 2

## strap .id

##

## 1 001

## 2 002

## 3 003

## 4 004

## 5 005

## 6 006

## 7 007

## 8 008

## 9 009

## 10 010

## # ... with 90 more rows

The column strap contains resamples of the original data. I will run my linear regression  
from before on each of the resamples:

(

boot\_lin\_reg <- boot\_education %>%

mutate(regressions =

map(strap,

~lm(per\_capita\_exp ~ region + residents +

young\_residents + per\_capita\_income,

data = .)))

)

## # A tibble: 100 x 3

## strap .id regressions

##

## 1 001

## 2 002

## 3 003

## 4 004

## 5 005

## 6 006

## 7 007

## 8 008

## 9 009

## 10 010

## # ... with 90 more rows

I have added a new column called regressions which contains the linear regressions on each  
bootstrapped sample. Now, I will create a list of tidied regression results:

(

tidied <- boot\_lin\_reg %>%

mutate(tidy\_lm =

map(regressions, broom::tidy))

)

## # A tibble: 100 x 4

## strap .id regressions tidy\_lm

##

## 1 001

## 2 002

## 3 003

## 4 004

## 5 005

## 6 006

## 7 007

## 8 008

## 9 009

## 10 010

## # ... with 90 more rows

broom::tidy() creates a data frame of the regression results. Let’s look at one of these:

tidied$tidy\_lm[[1]]

## term estimate std.error statistic p.value

## 1 (Intercept) -515.19835839 129.61828003 -3.9747353 2.648477e-04

## 2 regionnortheast -11.75706535 18.72014312 -0.6280436 5.332970e-01

## 3 regionsouth -8.89139412 15.51588707 -0.5730510 5.695950e-01

## 4 regionwest 25.96217940 17.48335459 1.4849656 1.448476e-01

## 5 residents -0.08921914 0.05557157 -1.6054819 1.157079e-01

## 6 young\_residents 1.41203710 0.35297506 4.0003877 2.447887e-04

## 7 per\_capita\_income 0.08609340 0.01191468 7.2258276 6.069534e-09

This format is easier to handle than the standard lm() output:

tidied$regressions[[1]]

##

## Call:

## lm(formula = per\_capita\_exp ~ region + residents + young\_residents +

## per\_capita\_income, data = .)

##

## Coefficients:

## (Intercept) regionnortheast regionsouth

## -515.19836 -11.75707 -8.89139

## regionwest residents young\_residents

## 25.96218 -0.08922 1.41204

## per\_capita\_income

## 0.08609

Now that I have all these regression results, I can compute any statistic I need. But first,  
let’s transform the data even further:

list\_mods <- tidied %>%

pull(tidy\_lm)

list\_mods is a list of the tidy\_lm data frames. I now add an index and  
bind the rows together (by using map2\_df() instead of map2()):

mods\_df <- map2\_df(list\_mods,

seq(1, resamples),

~mutate(.x, resample = .y))

Let’s take a look at the final object:

head(mods\_df, 25)

## term estimate std.error statistic p.value

## 1 (Intercept) -515.19835839 129.61828003 -3.9747353 2.648477e-04

## 2 regionnortheast -11.75706535 18.72014312 -0.6280436 5.332970e-01

## 3 regionsouth -8.89139412 15.51588707 -0.5730510 5.695950e-01

## 4 regionwest 25.96217940 17.48335459 1.4849656 1.448476e-01

## 5 residents -0.08921914 0.05557157 -1.6054819 1.157079e-01

## 6 young\_residents 1.41203710 0.35297506 4.0003877 2.447887e-04

## 7 per\_capita\_income 0.08609340 0.01191468 7.2258276 6.069534e-09

## 8 (Intercept) -526.78466908 142.64004523 -3.6931050 6.207704e-04

## 9 regionnortheast 3.35909252 21.97165783 0.1528830 8.792057e-01

## 10 regionsouth 2.62845462 17.20973263 0.1527307 8.793251e-01

## 11 regionwest 26.40527188 20.50214280 1.2879274 2.046593e-01

## 12 residents -0.04834920 0.05722962 -0.8448282 4.028830e-01

## 13 young\_residents 1.49618012 0.37091593 4.0337445 2.208994e-04

## 14 per\_capita\_income 0.07489435 0.01179888 6.3475800 1.140844e-07

## 15 (Intercept) -466.17160566 130.18954610 -3.5807146 8.659014e-04

## 16 regionnortheast -6.75977787 17.24994600 -0.3918724 6.970880e-01

## 17 regionsouth 6.31061828 16.27953099 0.3876413 7.001938e-01

## 18 regionwest 27.89261897 15.43852463 1.8066894 7.781178e-02

## 19 residents -0.08760677 0.05007780 -1.7494134 8.735434e-02

## 20 young\_residents 1.23763741 0.32965208 3.7543746 5.168492e-04

## 21 per\_capita\_income 0.08469609 0.01236601 6.8491057 2.128982e-08

## 22 (Intercept) -316.44265722 166.96769979 -1.8952328 6.479692e-02

## 23 regionnortheast 11.54907449 14.95746934 0.7721276 4.442620e-01

## 24 regionsouth 9.25759229 16.89974638 0.5477947 5.866654e-01

## 25 regionwest 31.83905855 16.56287888 1.9223143 6.120738e-02

## resample

## 1 1

## 2 1

## 3 1

## 4 1

## 5 1

## 6 1

## 7 1

## 8 2

## 9 2

## 10 2

## 11 2

## 12 2

## 13 2

## 14 2

## 15 3

## 16 3

## 17 3

## 18 3

## 19 3

## 20 3

## 21 3

## 22 4

## 23 4

## 24 4

## 25 4

Now this is a very useful format, because I now can group by the term column and compute any  
statistics I need, in the present case the standard deviation:

(

r.std.error <- mods\_df %>%

group\_by(term) %>%

summarise(r.std.error = sd(estimate))

)

## # A tibble: 7 x 2

## term r.std.error

##

## 1 (Intercept) 226.

## 2 per\_capita\_income 0.0211

## 3 regionnortheast 19.7

## 4 regionsouth 19.1

## 5 regionwest 18.7

## 6 residents 0.0629

## 7 young\_residents 0.509

We can append this column to the linear regression model result:

lmfit %>%

broom::tidy() %>%

full\_join(r.std.error) %>%

select(term, estimate, std.error, r.std.error)

## Joining, by = "term"

## term estimate std.error r.std.error

## 1 (Intercept) -467.40282655 142.57668653 226.08735720

## 2 regionnortheast 15.72740519 18.16259519 19.65120904

## 3 regionsouth 7.08742365 17.29949951 19.05153934

## 4 regionwest 34.32415663 17.49459866 18.72767470

## 5 residents -0.03455770 0.05318811 0.06285984

## 6 young\_residents 1.30145825 0.35716636 0.50928879

## 7 per\_capita\_income 0.07203552 0.01305066 0.02109277

As you see, using the whole bootstrapping procedure is longer than simply using either one of  
the first two methods. However, this procedure is very flexible and can thus be adapted to a very  
large range of situations. Either way, in the case of heteroskedasticity, you can see that  
results vary a lot depending on the procedure you use, so I would advise to use them all as  
robustness tests and discuss the differences.