

# Regression with R: Extract, Tabulate, Plot

Econ 440 - Introduction to Econometrics

Patrick Toche, ptoche@fullerton.edu

28 April 2022

## Extract, Tabulate, Plot

In this notebook we explore how to extract regression coefficients, standard errors, p-values and other pieces of information contained in a linear regression model. We also learn how to augment a regression model, how to tabulate regression results, and how to plot regression lines.

### Dataset on Earnings

The data file **CPS2015** contains data for full-time, full-year workers, ages 25–34, with a high school diploma or B.A./B.S. as their highest degree. A detailed description is given in **CPS2015\_Description**.

```
library(readxl)
df <- read_xlsx("CPS2015.xlsx", trim_ws=TRUE)
head(df)
```

```
## # A tibble: 6 x 5
##   year  ahe bachelor female  age
##   <dbl> <dbl>   <dbl>   <dbl> <dbl>
## 1  2015  11.8         0       0    26
## 2  2015   9.62        0       1    33
## 3  2015  12.0         0       0    31
## 4  2015  18.4         0       0    32
## 5  2015  41.8         0       0    28
## 6  2015  19.2         0       1    31
```

### A Log-quadratic model

This nonlinear regression model is analyzed in Chapter 8 of Stock and Watson's **Introduction to Econometrics**. Run a linear model with `lm()` to produce a model object, then call the `summary()` function to extract basic information.

```
m1 <- lm(log(ahe) ~ age + I(age^2) + bachelor + female, data=df)
summary(m1)
```

```
##
## Call:
## lm(formula = log(ahe) ~ age + I(age^2) + bachelor + female, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.5764 -0.2868  0.0126  0.3041  2.0596
##
```

```
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.418745   0.672088    0.62  0.5333
## age          0.134115   0.045791    2.93  0.0034 **
## I(age^2)     -0.001860   0.000774   -2.40  0.0163 *
## bachelor     0.461629   0.011473   40.24 <2e-16 ***
## female      -0.177364   0.011626  -15.26 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.477 on 7093 degrees of freedom
## Multiple R-squared:  0.209, Adjusted R-squared:  0.209
## F-statistic: 469 on 4 and 7093 DF, p-value: <2e-16
```

## Look inside a model object

See inside the model object: a list that holds useful results obtained from the regression, including the estimates, residuals, and standard errors.

```
str(m1)
```

```
## List of 12
## $ coefficients : Named num [1:5] 0.41874 0.13412 -0.00186 0.46163 -0.17736
##   ..- attr(*, "names")= chr [1:5] "(Intercept)" "age" "I(age^2)" "bachelor" ...
## $ residuals    : Named num [1:7098] -0.182 -0.378 -0.302 0.106 1.018 ...
##   ..- attr(*, "names")= chr [1:7098] "1" "2" "3" "4" ...
## $ effects      : Named num [1:7098] -245.4 -6.08 -1.1 18.32 -7.28 ...
##   ..- attr(*, "names")= chr [1:7098] "(Intercept)" "age" "I(age^2)" "bachelor" ...
## $ rank         : int 5
## $ fitted.values: Named num [1:7098] 2.65 2.64 2.79 2.81 2.72 ...
##   ..- attr(*, "names")= chr [1:7098] "1" "2" "3" "4" ...
## $ assign       : int [1:5] 0 1 2 3 4
## $ qr           :List of 5
##   ..$ qr      : num [1:7098, 1:5] -84.2496 0.0119 0.0119 0.0119 0.0119 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:7098] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:5] "(Intercept)" "age" "I(age^2)" "bachelor" ...
##   .. ..- attr(*, "assign")= int [1:5] 0 1 2 3 4
##   ..$ qraux: num [1:5] 1.01 1.01 1.01 1.01 1.01
##   ..$ pivot: int [1:5] 1 2 3 4 5
##   ..$ tol   : num 1e-07
##   ..$ rank  : int 5
##   ..- attr(*, "class")= chr "qr"
## $ df.residual  : int 7093
## $ xlevels      : Named list()
## $ call         : language lm(formula = log(ahe) ~ age + I(age^2) + bachelor + female, data = df)
## $ terms        :Classes 'terms', 'formula' language log(ahe) ~ age + I(age^2) + bachelor + female
##   .. ..- attr(*, "variables")= language list(log(ahe), age, I(age^2), bachelor, female)
##   .. ..- attr(*, "factors")= int [1:5, 1:4] 0 1 0 0 0 0 0 1 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:5] "log(ahe)" "age" "I(age^2)" "bachelor" ...
##   .. .. .. ..$ : chr [1:4] "age" "I(age^2)" "bachelor" "female"
##   .. ..- attr(*, "term.labels")= chr [1:4] "age" "I(age^2)" "bachelor" "female"
##   .. ..- attr(*, "order")= int [1:4] 1 1 1 1
##   .. ..- attr(*, "intercept")= int 1
```

```
## ..- attr(*, "response")= int 1
## ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
## ..- attr(*, "predvars")= language list(log(ahe), age, I(age^2), bachelor, female)
## ..- attr(*, "dataClasses")= Named chr [1:5] "numeric" "numeric" "numeric" "numeric" ...
## ..- attr(*, "names")= chr [1:5] "log(ahe)" "age" "I(age^2)" "bachelor" ...
## $ model      :'data.frame':  7098 obs. of  5 variables:
## ..$ log(ahe): num [1:7098] 2.47 2.26 2.49 2.91 3.73 ...
## ..$ age      : num [1:7098] 26 33 31 32 28 31 34 33 34 33 ...
## ..$ I(age^2): 'AsIs' num [1:7098] 676 1089 961 1024 784 ...
## ..$ bachelor: num [1:7098] 0 0 0 0 0 0 0 1 0 1 ...
## ..$ female   : num [1:7098] 0 1 0 0 0 1 0 1 0 1 ...
## ..- attr(*, "terms")=Classes 'terms', 'formula' language log(ahe) ~ age + I(age^2) + bachelor + f
## ..- attr(*, "variables")= language list(log(ahe), age, I(age^2), bachelor, female)
## ..- attr(*, "factors")= int [1:5, 1:4] 0 1 0 0 0 0 0 1 0 0 ...
## ..- attr(*, "dimnames")=List of 2
## ..$ : chr [1:5] "log(ahe)" "age" "I(age^2)" "bachelor" ...
## ..$ : chr [1:4] "age" "I(age^2)" "bachelor" "female"
## ..- attr(*, "term.labels")= chr [1:4] "age" "I(age^2)" "bachelor" "female"
## ..- attr(*, "order")= int [1:4] 1 1 1 1
## ..- attr(*, "intercept")= int 1
## ..- attr(*, "response")= int 1
## ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
## ..- attr(*, "predvars")= language list(log(ahe), age, I(age^2), bachelor, female)
## ..- attr(*, "dataClasses")= Named chr [1:5] "numeric" "numeric" "numeric" "numeric" ...
## ..- attr(*, "names")= chr [1:5] "log(ahe)" "age" "I(age^2)" "bachelor" ...
## - attr(*, "class")= chr "lm"
```

You may save the coefficients to a list:

```
coefs <- coef(m1)
names(coefs)
```

```
## [1] "(Intercept)" "age"          "I(age^2)"      "bachelor"      "female"
```

You may extract the coefficients by name:

```
coefs["(Intercept)"]
```

```
## (Intercept)
##      0.41874
```

```
coefs["bachelor"]
```

```
## bachelor
##      0.46163
```

Remember that you can always invoke `str()` on an object to examine its content and thus figure out how to extract its elements. For instance, if you call `str(coefs)` you will find that the regression coefficient on `age` is called “poly(age, 2, raw = TRUE)1”, while the coefficient on `age2` is called “poly(age, 2, raw = TRUE)2”.

Beware: In strings, spaces and cases matter, so the following won’t work!

```
coefs["(intercept)"]
```

```
## <NA>
##      NA
```

The *broom* package offers a more convenient and more versatile interface to extract and transform the regression data.

## Explore the model object with broom

The *broom* package provides convenience functions, including *tidy()*, *glance()* and *augment()*. These functions always return a *tibble* (a modern dataframe).

Extract coefficients: *tidy()* returns a nicely formatted dataframe:

```
tidy(m1)

## # A tibble: 5 x 5
##   term          estimate std.error statistic    p.value
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)  0.419    0.672     0.623 5.33e- 1
## 2 age          0.134    0.0458     2.93 3.41e- 3
## 3 I(age^2)     -0.00186 0.000774    -2.40 1.63e- 2
## 4 bachelor      0.462    0.0115    40.2 4.95e-319
## 5 female       -0.177    0.0116   -15.3 9.83e- 52
```

Same with pipe operator and a slice to select a coefficient of interest. The function *pull()* is then used to extract the desired value, e.g. *pull(std.error)*. The result can be manipulated further, e.g. rounding:

```
m1 %>% tidy() %>% slice(2) %>% pull(std.error) %>% round(.,3)

## [1] 0.046
```

You can add confidence intervals with the *conf.int* argument and extract the desired values with *pull()*:

```
m1 %>% tidy(conf.int=TRUE, conf.level=0.80) %>% pull(conf.low)

## [1] -0.4426506  0.0754267 -0.0028526  0.4469246 -0.1922645
```

The *augment()* function can be used to extract the fitted values and residuals for the original observations. In the augmented dataframe/tibble, each of the new columns begins with a dot, e.g. *.fitted*, to avoid accidentally overwriting existing variable names. This data could be extracted and used to plot a regression line and confidence interval, for instance.

```
ma <- augment(m1, data=df)
names(ma)

## [1] "year"      "ahe"      "bachelor"  "female"   "age"
## [6] ".fitted"   ".resid"   ".hat"      ".sigma"    ".cooksd"
## [11] ".std.resid"
```

The *glance()* function can be used to extract summary statistics for the entire regression, including the R-squared and the F-statistic.

```
mg <- glance(m1, data=df)
names(mg) # glancing the regression statistics

## [1] "r.squared"      "adj.r.squared" "sigma"          "statistic"
## [5] "p.value"        "df"            "logLik"         "AIC"
## [9] "BIC"           "deviance"      "df.residual"    "nobs"
```

For instance, the R-squared can be extracted with:

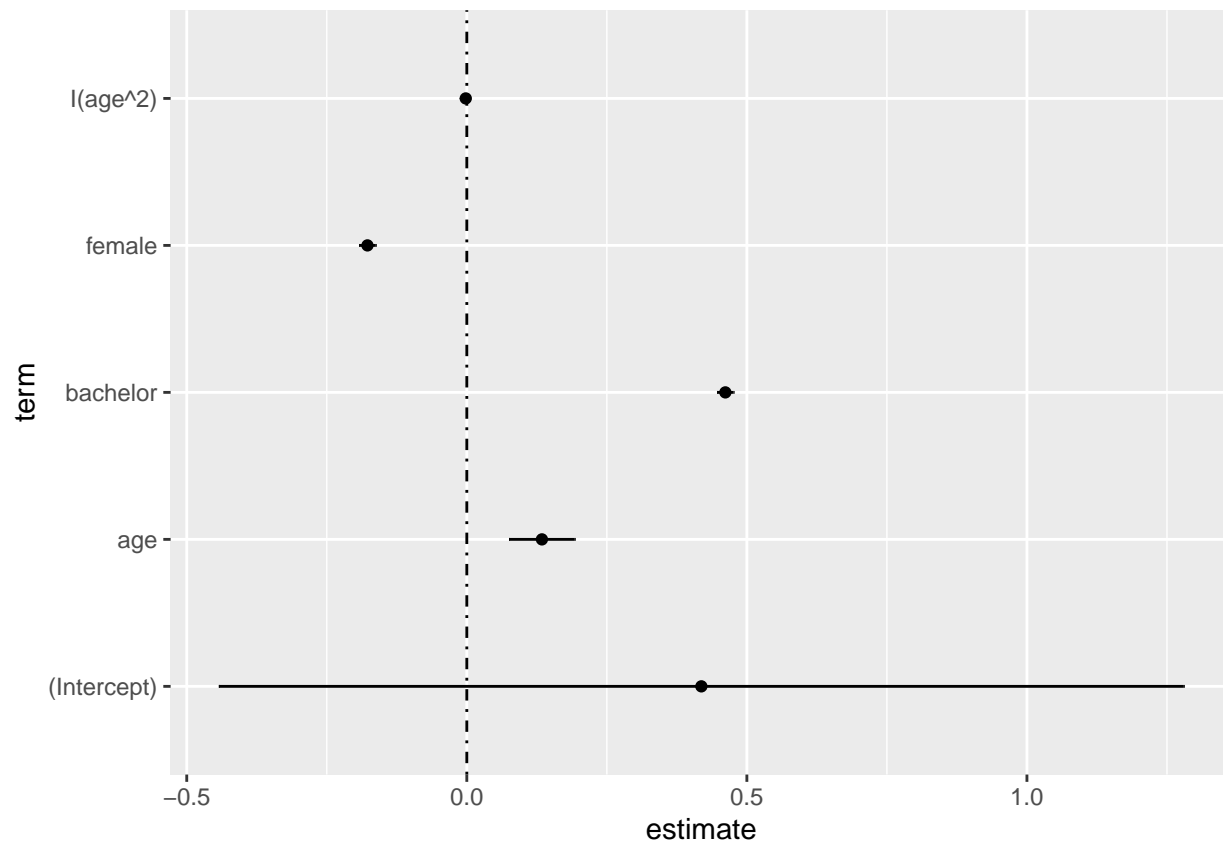
```
R2 <- mg$r.squared # or mg[["r.squared"]]
print(R2)

## [1] 0.20901
```

## Visualize the regression coefficients

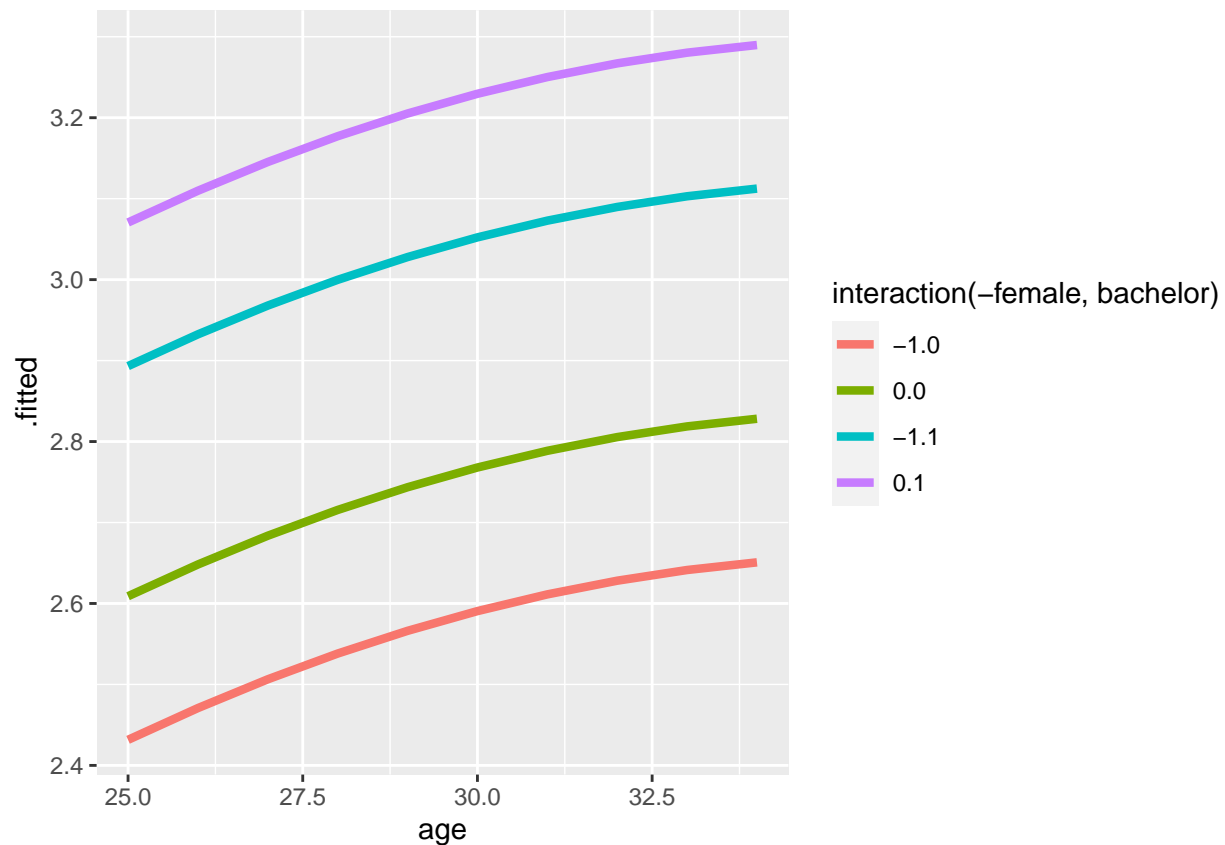
The following plot allows you to quickly visualize the regression coefficients:

```
m1 %>% tidy(conf.int=TRUE, conf.level=0.80) %>%  
  ggplot(., aes(estimate, term, xmin=conf.low, xmax=conf.high, height=0)) +  
  geom_point() +  
  geom_vline(xintercept=0, lty=4) +  
  geom_errorbarh()
```



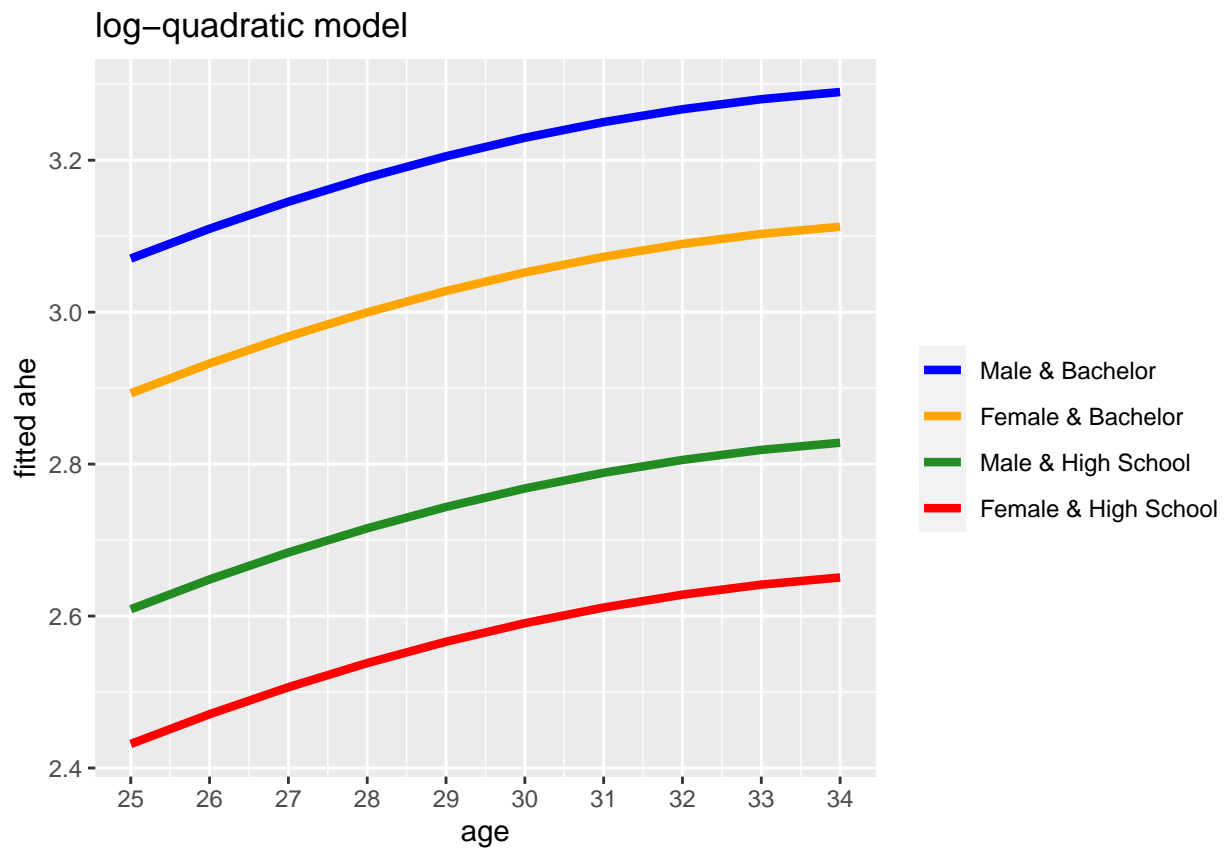
## Plot regression lines

```
augment(m1, df, interval="confidence") %>%  
  ggplot(data=.) +  
  aes(x=age,  
       y=.fitted,  
       group=interaction(-female, bachelor),  
       color=interaction(-female, bachelor)) +  
  geom_line(size=1.5) -> p0  
p0
```



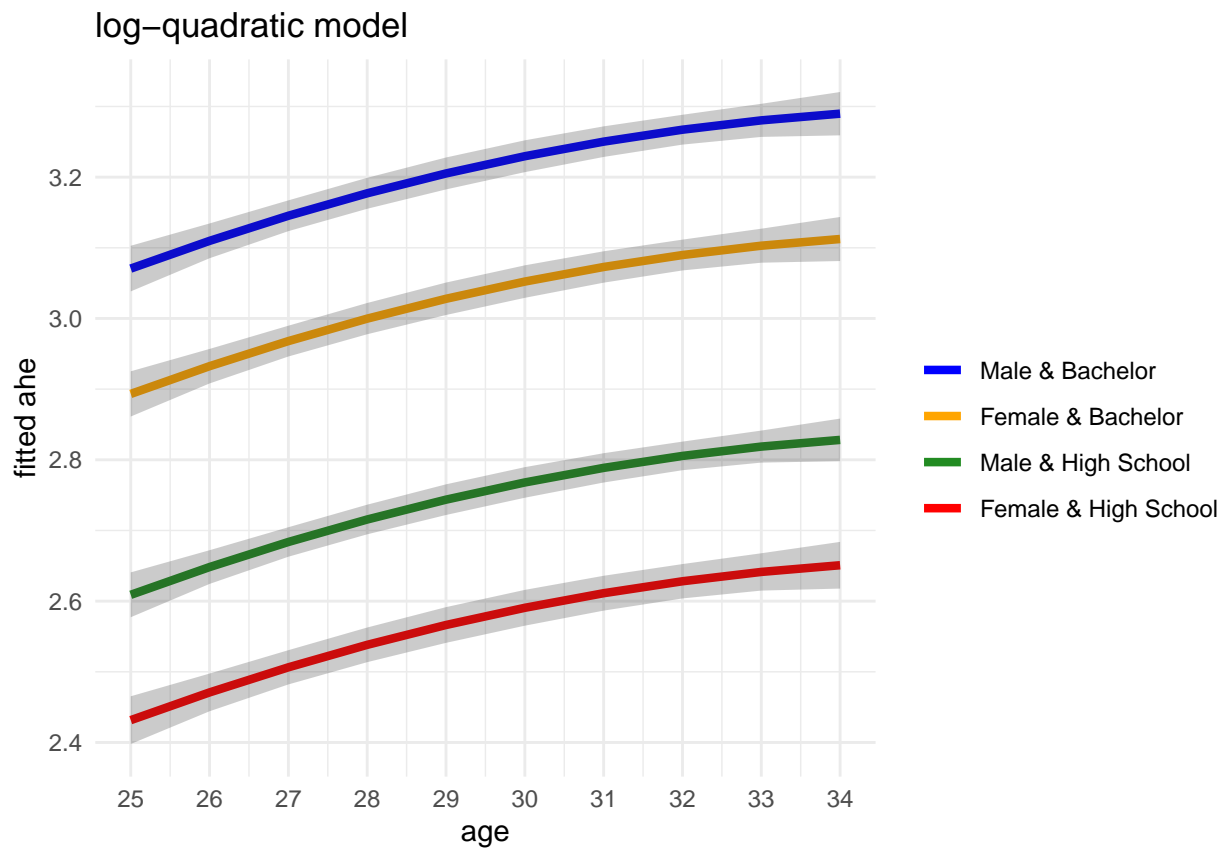
Fix the legend, select colors, tweak labels

```
labs <- c("Female & High School", "Male & High School", "Female & Bachelor", "Male & Bachelor") # crea
p0 + scale_x_continuous(breaks=seq(25,35,1)) +
  scale_color_manual(name="", labels=labs, values=c("red", "forestgreen", "orange", "blue")) +
  guides(color=guide_legend(reverse=TRUE)) +
  ylab("fitted ahe") +
  ggtitle("log-quadratic model") -> p1
p1
```



## Add confidence intervals

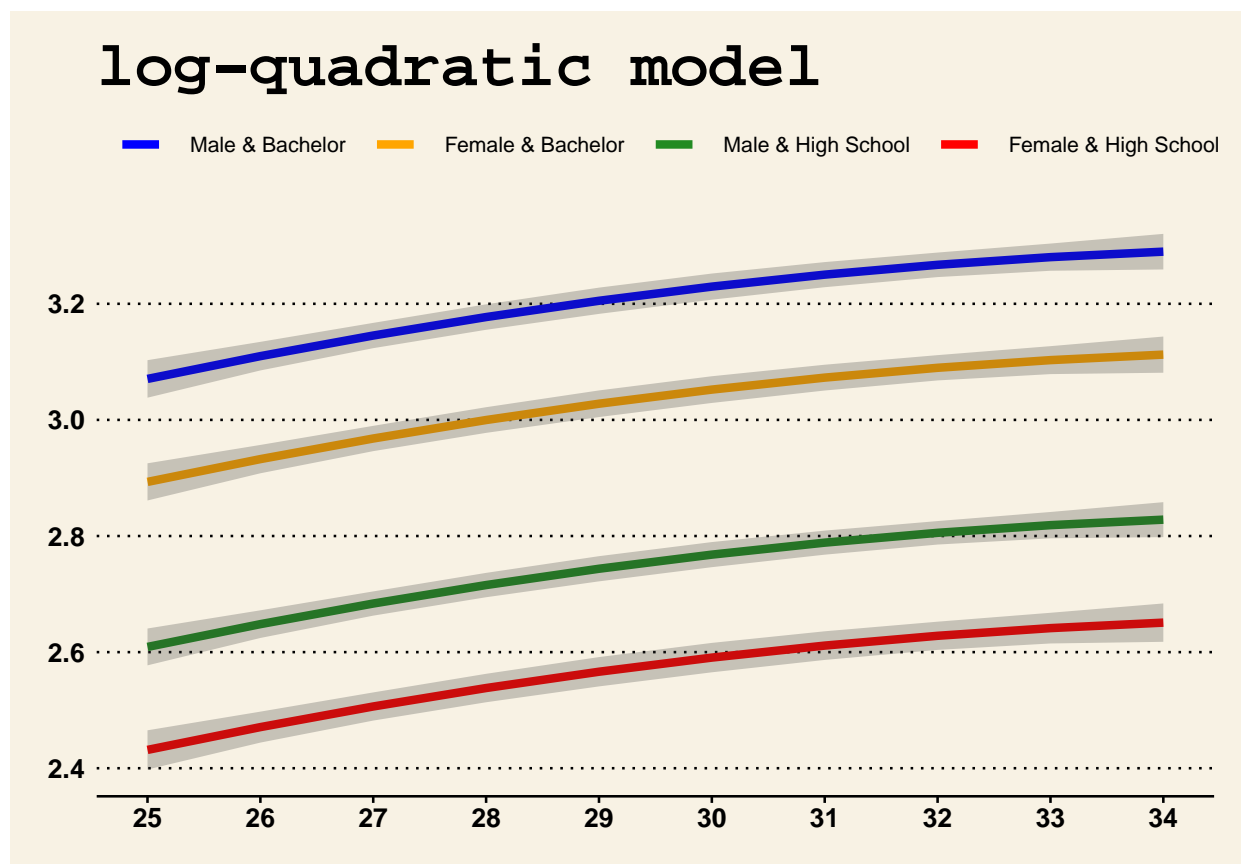
```
p1 + geom_ribbon(aes(ymin=.lower, ymax=.upper), alpha=.25, color=NA) -> p2  
p2 + theme_minimal()
```



Add theme to the plot

```
library(ggthemes) # to theme the plot
p2 + theme_wsj(base_size=10)
```





## Confidence Intervals for Parameters

The `dplyr` package (part of `tidyverse`) contains a convenient `full_join()` function that may be used to merge dataframes containing estimates calculated with `broom` functions like `tidy()` and `glance()`. First, create a dataframe which contains both the data estimates (using `tidy()`) and the model's summary statistics (using `glance()`). Then use `pull()` to get the desired statistics.

```
dplyr::full_join(
  df %>% group_modify(.f = ~ tidy(m1, conf.int=TRUE, conf.level=0.99)),
  df %>% group_modify(.f = ~ glance(m1))
) -> dm
```

```
## Joining, by = c("statistic", "p.value")
```

Then pull the desired statistics:

```
dm %>% pull(estimate)
```

```
## [1] 0.4187449 0.1341152 -0.0018603 0.4616293 -0.1773644 NA
```

```
dm %>% pull(conf.low)
```

```
## [1] -1.3129048 0.0161346 -0.0038551 0.4320687 -0.2073179 NA
```

```
dm %>% pull(conf.high)
```

```
## [1] 2.15039470 0.25209570 0.00013449 0.49118990 -0.14741083 NA
```

## Multiple models

Let's compute several regression models.

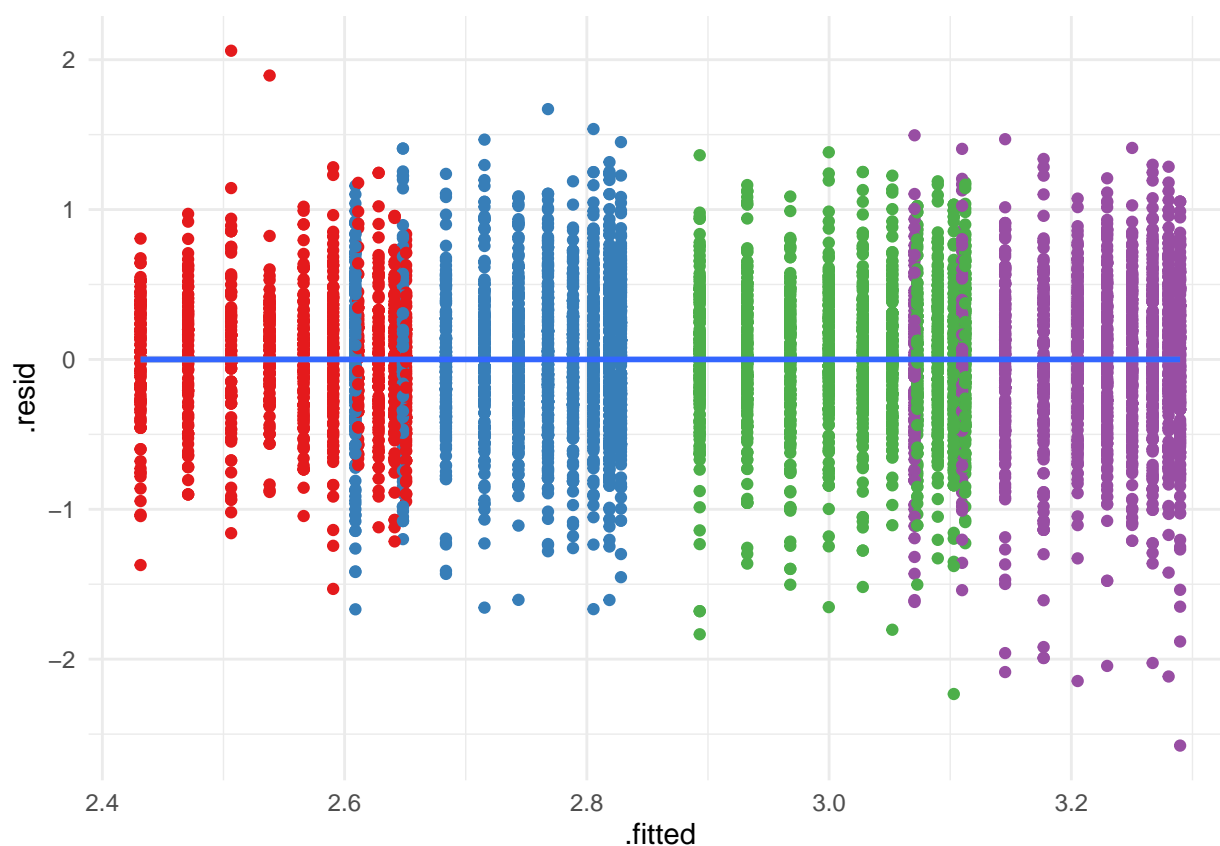
```
m2 <- lm(log(ahe) ~ age + I(age^2) + I(age^3) + bachelor + female, data=df)
m3 <- lm(log(ahe) ~ log(age) + bachelor + female, data=df)
```

## Fitted values and residuals

Clusters of points can potentially cause the errors to be heteroskedastic. To visualize this, we color the points according to the four combinations of Male/Female and Bachelor/High-School. We suppress the legend for clarity and set the color palette with the `scale_color_brewer()` function of the `ggplot2` package.

```
augment(m1, df, interval="confidence") %>%
  ggplot(data=., aes(x=.fitted, y=.resid)) +
  geom_point(aes(color=interaction(-female, bachelor))) +
  geom_smooth(method="lm") +
  scale_color_brewer(palette="Set1") +
  theme_minimal() +
  theme(legend.position="none")
```

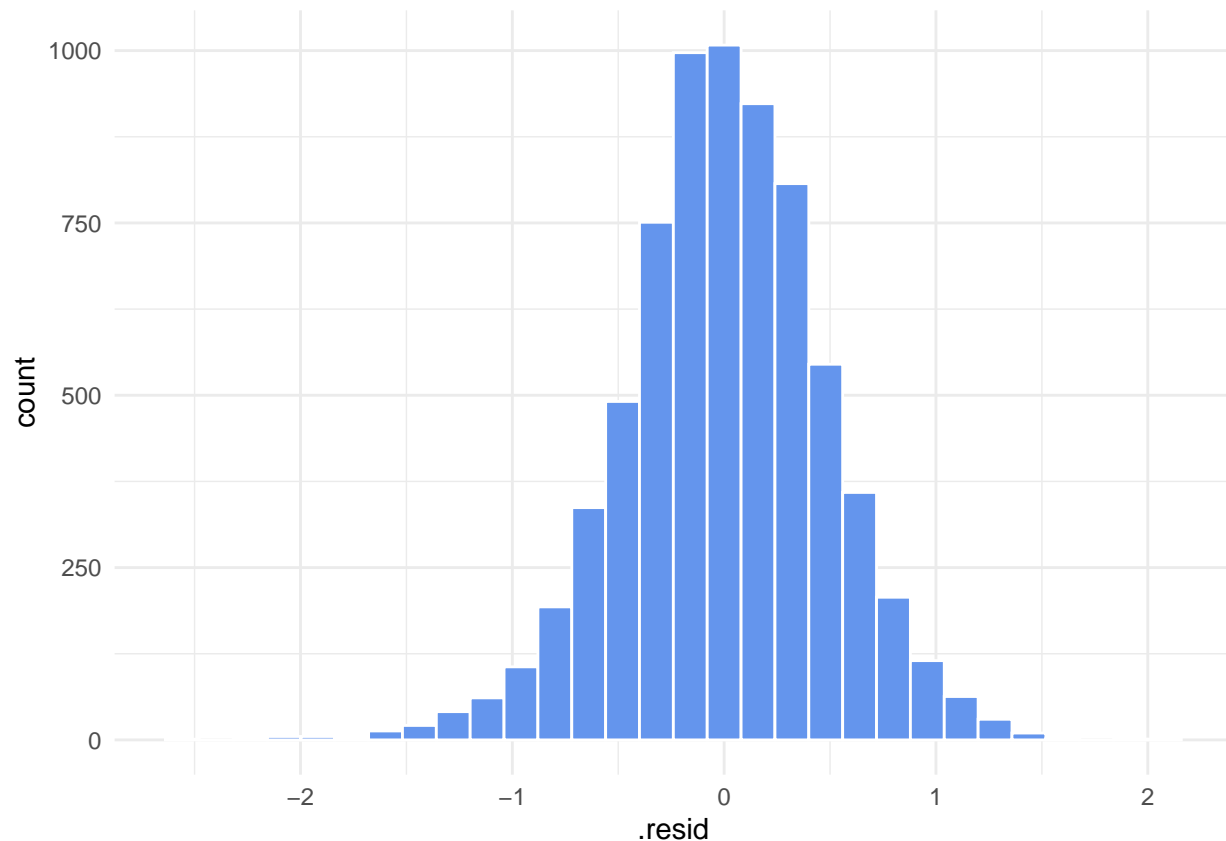
```
## `geom_smooth()` using formula 'y ~ x'
```



Look at the distribution of the residuals for evidence of heteroskedasticity:

```
augment(m1, df, interval="confidence") %>%
  ggplot(data=., aes(x=.resid)) +
  geom_histogram(color="white", fill="cornflowerblue") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



No evidence of heteroskedasticity! However, the clustering suggests that we must use robust standard errors.

## Compute robust standard errors

The `sandwich` package computes robust Heteroscedasticity-Consistent Covariance estimators with the function `vcovHC()`. The `type` argument can be used to specify estimators from *HC0* (White's estimator) to *HC5* (various refinements). The `lmtest` package provides a convenient function, `coeftest()`, to calculate the t test based on the variance-covariance matrix provided in the `vcov` argument. A robust test may be computed as follows:

```
library(sandwich)
library(lmtest) # coeftest, waldtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##   as.Date, as.Date.numeric

coeftest(m1, vcov=vcovHC(m1, type="HC1"))

##
## t test of coefficients:
##
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.418745   0.669575   0.63   0.5317
```

```
## age          0.134115    0.045610    2.94    0.0033 **
## I(age^2)     -0.001860    0.000771   -2.41    0.0159 *
## bachelor     0.461629    0.011456   40.30   <2e-16 ***
## female      -0.177364    0.011499  -15.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Compute robust Wald test

A heteroskedasticity robust F test may be calculated with the *waldtest* function from package *lmtest* (using White standard errors):

```
m1.unrestricted <- lm(log(ahe) ~ age + I(age^2) + I(age^3) + bachelor + female, data=df)
waldtest(m1, m1.unrestricted, vcov=vcovHC(m1.unrestricted, type="HC0"))
```

```
## Wald test
##
## Model 1: log(ahe) ~ age + I(age^2) + bachelor + female
## Model 2: log(ahe) ~ age + I(age^2) + I(age^3) + bachelor + female
##   Res.Df Df    F Pr(>F)
## 1    7093
## 2    7092  1 0.03  0.85
```

## Tabulate Regression results

Tabulating regression results may be automated with the help of several packages. A popular solution is *stargazer*. Here instead I use a newer package, *texreg*.

Output the table to screen with *screenreg()*:

```
suppressMessages(library(texreg))
screenreg(list(m1, m2, m3))

##
## =====
##           Model 1      Model 2      Model 3
## -----
## (Intercept)    0.42      -1.09      0.32
##               (0.67)    (8.22)    (0.20)
## age            0.13 **    0.29
##               (0.05)    (0.84)
## age^2          -0.00 *    -0.01
##               (0.00)    (0.03)
## bachelor        0.46 ***    0.46 ***    0.46 ***
##               (0.01)    (0.01)    (0.01)
## female         -0.18 ***    -0.18 ***    -0.18 ***
##               (0.01)    (0.01)    (0.01)
## age^3              0.00
##               (0.00)
## log(age)              0.72 ***
##                   (0.06)
## -----
## R^2              0.21      0.21      0.21
## Adj. R^2         0.21      0.21      0.21
```

```
## Num. obs.      7098      7098      7098
## =====
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

Export the table to the LaTeX format with *texreg()*:

```
# htmlreg(list(m1, m2, m3), doctype = FALSE, star.symbol = "\\*")
texreg(list(`(1)`=m1, `(2)`=m2, `(3)`=m3), booktabs=TRUE, dcolumn=TRUE)

##
## \usepackage{booktabs}
## \usepackage{dcolumn}
##
## \begin{table}
## \begin{center}
## \begin{tabular}{l D{.}{.}{4.5} D{.}{.}{4.5} D{.}{.}{4.5}}
## \toprule
## & \multicolumn{1}{c}{(1)} & \multicolumn{1}{c}{(2)} & \multicolumn{1}{c}{(3)} & \\
## \midrule
## (Intercept) & 0.42 & -1.09 & 0.32 & \\
## & (0.67) & (8.22) & (0.20) & \\
## age & 0.13^{**} & 0.29 & & \\
## & (0.05) & (0.84) & & \\
## age$^2$ & -0.00^{*} & -0.01 & & \\
## & (0.00) & (0.03) & & \\
## bachelor & 0.46^{***} & 0.46^{***} & 0.46^{***} & \\
## & (0.01) & (0.01) & (0.01) & \\
## female & -0.18^{***} & -0.18^{***} & -0.18^{***} & \\
## & (0.01) & (0.01) & (0.01) & \\
## age$^3$ & & 0.00 & & \\
## & & (0.00) & & \\
## log(age) & & & 0.72^{***} & \\
## & & & (0.06) & \\
## \midrule
## R$^2$ & 0.21 & 0.21 & 0.21 & \\
## Adj. R$^2$ & 0.21 & 0.21 & 0.21 & \\
## Num. obs. & 7098 & 7098 & 7098 & \\
## \bottomrule
## \multicolumn{4}{l}{\scriptsize $^{***}$p<0.001$; $^{**}$p<0.01$; $^{*}$p<0.05$}
## \end{tabular}
## \caption{Statistical models}
## \label{table:coefficients}
## \end{center}
## \end{table}
```

Export table to HTML format with *htmlreg()*

[Not shown as it messes up the PDF output]

Export table to PDF format

(the *texreg* argument *use.packages=FALSE* is set to suppress any package loading instructions in the preamble)

	(1)	(2)	(3)
(Intercept)	0.42 (0.67)	-1.09 (8.22)	0.32 (0.20)
age	0.13** (0.05)	0.29 (0.84)	
age <sup>2</sup>	-0.00* (0.00)	-0.01 (0.03)	
age <sup>3</sup>		0.00 (0.00)	
log(age)			0.72*** (0.06)
bachelor	0.46*** (0.01)	0.46*** (0.01)	0.46*** (0.01)
female	-0.18*** (0.01)	-0.18*** (0.01)	-0.18*** (0.01)
R <sup>2</sup>	0.21	0.21	0.21
Adj. R <sup>2</sup>	0.21	0.21	0.21
Num. obs.	7098	7098	7098

\*\*\* $p < 0.001$ ; \*\* $p < 0.01$ ; \* $p < 0.05$