

# Econometrics in R

## Lecture 4

Louis SIRUGUE

M1 APE - Fall 2022



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv")
```

```
#  
#  
#  
#  
#  
#
```

```
##      Wk Day      Date  Time      Home  xG  Score  xG.1      Away Attendance ...  
## 1     1 Fri 2021-08-06 21:00 Monaco 2.0  1-1  0.3      Nantes      7500 ...  
## 2     1 Sat 2021-08-07 17:00   Lyon 1.4  1-1  0.8      Brest    29018 ...  
## 3     1 Sat 2021-08-07 21:00 Troyes 0.8  1-2  1.2 Paris S-G 15248 ...  
## 4     1 Sun 2021-08-08 13:00  Rennes 0.6  1-1  2.0      Lens    22567 ...  
## 5     1 Sun 2021-08-08 15:00 Bordeaux 0.7  0-2  3.3 Clermont Foot 18748 ...  
## 6     1 Sun 2021-08-08 15:00 Strasbourg 0.4  0-2  0.9      Angers    23250 ...  
## 7     1 Sun 2021-08-08 15:00      Nice 0.8  0-0  0.2      Reims    18030 ...  
## 8     1 Sun 2021-08-08 15:00 Saint-Étienne 2.1  1-1  1.3      Lorient   20461 ...  
## 9     1 Sun 2021-08-08 17:00      Metz 0.7  3-3  1.4      Lille    15551 ...  
... ..
```



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv") %>%  
  select(Home, xG, Score, xG.1, Away) # Keep/drop certain columns  
#  
#  
#  
#  
#
```

##	Home	xG	Score	xG.1	Away
## 1	Monaco	2.0	1-1	0.3	Nantes
## 2	Lyon	1.4	1-1	0.8	Brest
## 3	Troyes	0.8	1-2	1.2	Paris S-G
## 4	Rennes	0.6	1-1	2.0	Lens
## 5	Bordeaux	0.7	0-2	3.3	Clermont Foot
## 6	Strasbourg	0.4	0-2	0.9	Angers
## 7	Nice	0.8	0-0	0.2	Reims
## 8	Saint-Étienne	2.1	1-1	1.3	Lorient
## 9	Metz	0.7	3-3	1.4	Lille
...	...	...	...	...	...



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv") %>%  
  select(Home, xG, Score, xG.1, Away) %>%  
  mutate(home_winner = xG > xG.1)  
  
# Keep/drop certain columns  
# Create a new variable  
#  
#  
#  
#
```

##	Home	xG	Score	xG.1	Away	home_winner
## 1	Monaco	2.0	1-1	0.3	Nantes	TRUE
## 2	Lyon	1.4	1-1	0.8	Brest	TRUE
## 3	Troyes	0.8	1-2	1.2	Paris S-G	FALSE
## 4	Rennes	0.6	1-1	2.0	Lens	FALSE
## 5	Bordeaux	0.7	0-2	3.3	Clermont Foot	FALSE
## 6	Strasbourg	0.4	0-2	0.9	Angers	FALSE
## 7	Nice	0.8	0-0	0.2	Reims	TRUE
## 8	Saint-Étienne	2.1	1-1	1.3	Lorient	TRUE
## 9	Metz	0.7	3-3	1.4	Lille	FALSE
...	...	...	...	...	...	...



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv") %>%  
  select(Home, xG, Score, xG.1, Away) %>%           # Keep/drop certain columns  
  mutate(home_winner = xG > xG.1) %>%              # Create a new variable  
  filter(Home == "Rennes")                          # Keep/drop certain rows  
                                                    #  
                                                    #  
                                                    #
```

```
##      Home  xG Score xG.1      Away home_winner  
## 1 Rennes 0.6  1-1  2.0      Lens      FALSE  
## 2 Rennes 0.9  1-0  0.5      Nantes      TRUE  
## 3 Rennes 1.0  0-2  0.5      Reims      TRUE  
## 4 Rennes 2.4  6-0  0.3 Clermont Foot      TRUE  
## 5 Rennes 0.8  2-0  1.4      Paris S-G      FALSE  
## 6 Rennes 1.5  1-0  0.6      Strasbourg      TRUE  
## 7 Rennes 3.8  4-1  1.1      Lyon      TRUE  
## 8 Rennes 3.1  2-0  0.7      Montpellier      TRUE  
## 9 Rennes 0.8  1-2  0.6      Lille      TRUE  
## ... ..  
## ... ..
```



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv") %>%  
  select(Home, xG, Score, xG.1, Away) %>%           # Keep/drop certain columns  
  mutate(home_winner = xG > xG.1) %>%              # Create a new variable  
  filter(Home == "Rennes") %>%                    # Keep/drop certain rows  
  arrange(-xG)                                     # Sort rows  
#  
#
```

```
##      Home  xG  Score  xG.1      Away  home_winner  
## 1  Rennes 3.8   4-1   1.1      Lyon    TRUE  
## 2  Rennes 3.3   6-0   0.4    Bordeaux  TRUE  
## 3  Rennes 3.3   6-1   0.9      Metz    TRUE  
## 4  Rennes 3.1   2-0   0.7  Montpellier TRUE  
## 5  Rennes 2.7   2-0   0.3      Brest   TRUE  
## 6  Rennes 2.6   4-1   0.4      Troyes  TRUE  
## 7  Rennes 2.4   6-0   0.3  Clermont Foot TRUE  
## 8  Rennes 1.9   2-3   2.9      Monaco  FALSE  
## 9  Rennes 1.7   2-0   0.3      Angers  TRUE  
##    ...  ...  ...  ...  ...  ...
```



# What we've seen so far

## Manipulate data with `dplyr`

```
read.csv("ligue1.csv") %>%  
  select(Home, xG, Score, xG.1, Away) %>%           # Keep/drop certain columns  
  mutate(home_winner = xG > xG.1) %>%              # Create a new variable  
  filter(Home == "Rennes") %>%                    # Keep/drop certain rows  
  arrange(-xG) %>%                                 # Sort rows  
  summarise(expected_wins = mean(home_winner),      # Aggregate into statistics  
            expected_goals = sum(xG))              #
```

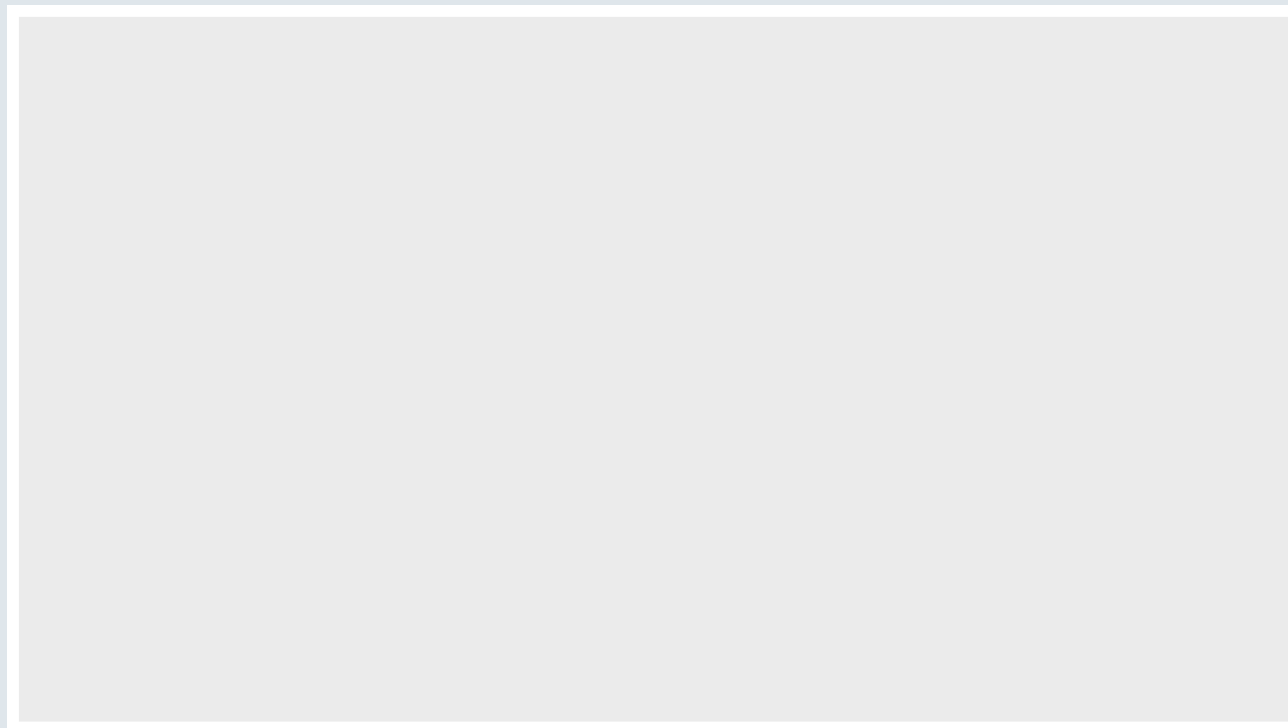
```
##   expected_wins expected_goals  
## 1      0.8421053          36.6
```



# What we've seen so far

## Plot data with `ggplot()`

```
ggplot(read.csv("wid.csv"))           # Data  
                                     #
```



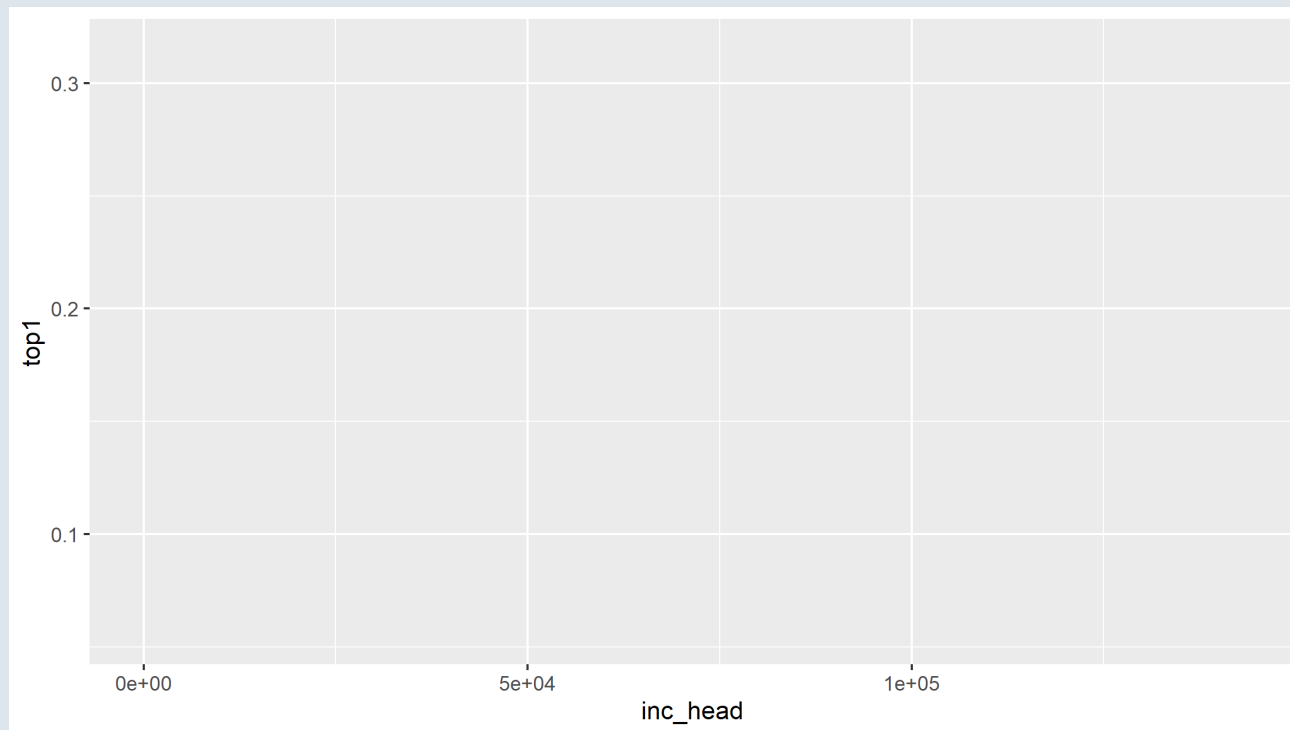




# What we've seen so far

## Plot data with `ggplot()`

```
ggplot(read.csv("wid.csv"), aes(x = inc_head, y = top1))           # Data & aesthetics  
                                                                    #
```



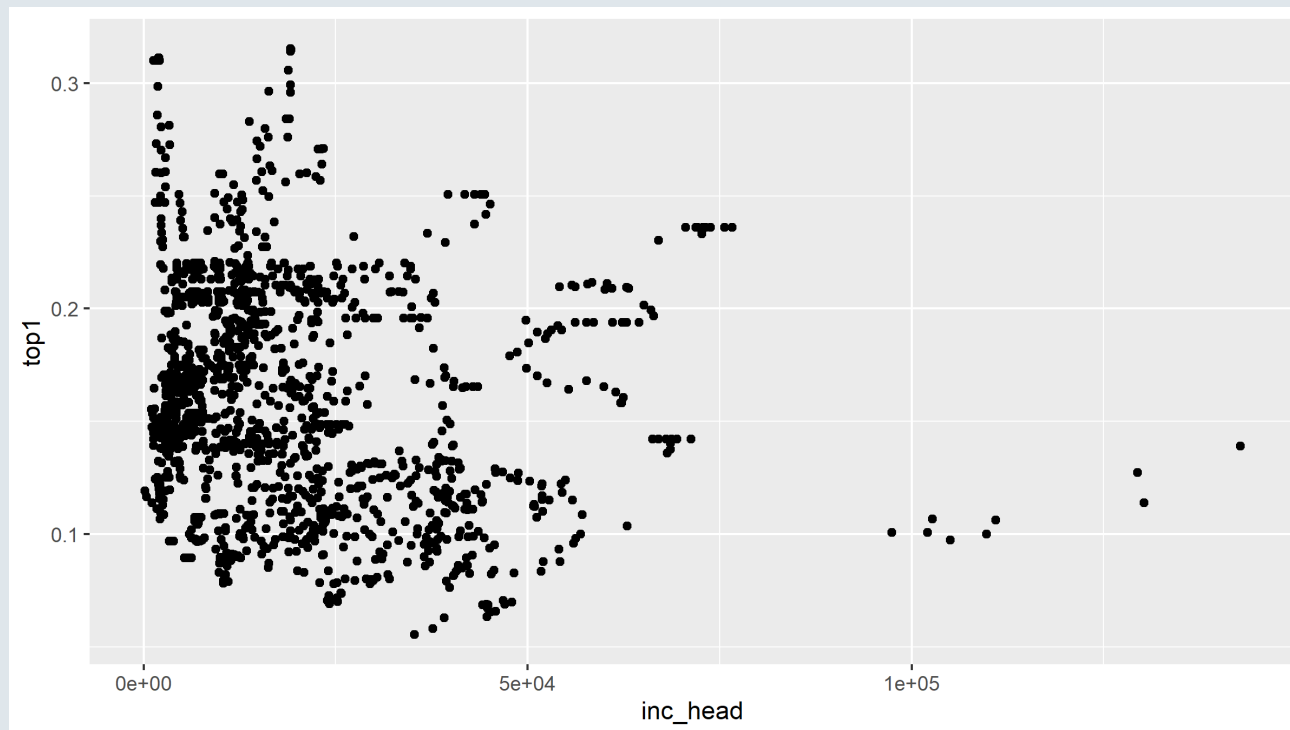


# What we've seen so far

## Plot data with `ggplot()`

```
ggplot(read.csv("wid.csv"), aes(x = inc_head, y = top1)) +  
  geom_point()
```

```
# Data & aesthetics  
# Geometry
```





# What we've seen so far

## Write reports with R markdown

```
---  
title: "Starbucks"  
author: "Louis Sirugue"  
output: html_document  
---
```

# Starbucks

Louis Sirugue



# What we've seen so far

## Write reports with R markdown

```
---  
title: "Starbucks"  
author: "Louis Sirugue"  
output: html_document  
---  
  
```${r, echo = F, message = F, warning = F}  
library(ggplot2) # Load package  
starbucks <- read.csv("starbucks.csv", sep = ";") # Load data  
```${br/>  
Nutritional values of `r nrow(starbucks)` *starbucks* beverages
```

## Starbucks

Louis Sirugue

Nutritional values of 242 *starbucks* beverages



# What we've seen so far

## Write reports with R markdown

```
---
title: "Starbucks"
author: "Louis Sirugue"
output: html_document
---

```{r, echo = F, message = F, warning = F}
library(ggplot2) # Load package
starbucks <- read.csv("starbucks.csv", sep = ";") # Load data
```

Nutritional values of `r nrow(starbucks)` starbucks beverages

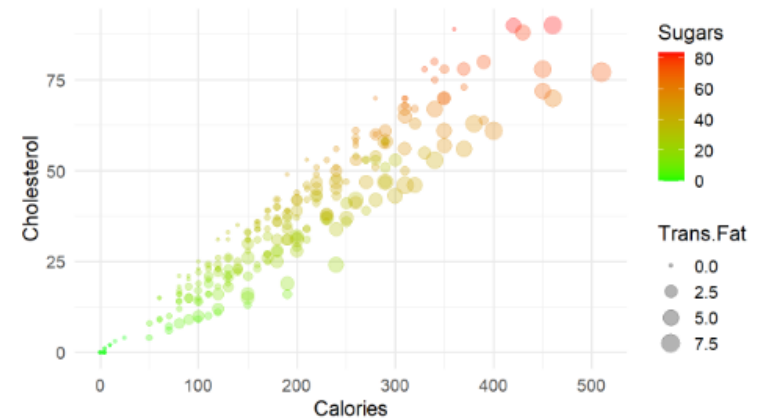
```{r, fig.height = 4}
ggplot(starbucks, aes(x = Calories, y = Cholesterol,
                      size = Trans.Fat, color = Sugars)) +
  geom_point(alpha = .3) + theme_minimal(base_size = 14) +
  scale_color_gradient(low = "green", high = "red")
```
```

## Starbucks

Louis Sirugue

Nutritional values of 242 *starbucks* beverages

```
ggplot(starbucks, aes(x = Calories, y = Cholesterol,
                      size = Trans.Fat, color = Sugars)) +
  geom_point(alpha = .3) + theme_minimal(base_size = 14) +
  scale_color_gradient(low = "green", high = "red")
```





# Today: Econometrics in R!

## 1. Regressions

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results

- 4.1. Regression tables
- 4.2. Plot coefficients

## 5. Wrap up!



# Today: Econometrics in R!

## 1. Regressions

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

# 1. Regressions

## 1.1. On continuous variables

- For this part we're going to work with the '**Great Gatsby Curve**'
  - It refers to the positive relationship between **inequality** and **intergenerational income persistence**
  - The term was coined by Alan Krueger based on the research of Miles Corak

```
ggcurve <- read.csv("ggcurve.csv")
str(ggcurve)
```

```
## 'data.frame':    22 obs. of  3 variables:
## $ country: chr  "Denmark" "Norway" "Finland" "Canada" ...
## $ ige    : num  0.15 0.17 0.18 0.19 0.26 0.27 0.29 0.32 0.34 0.4 ...
## $ gini   : num  0.378 0.325 0.378 0.463 0.439 ...
```

- For **22 countries** we have the following variables
  - **ige**: The intergenerational income elasticity, the higher the closer child income to parent income
  - **gini**: The Gini coefficient of income inequality, the higher the more concentrated the income

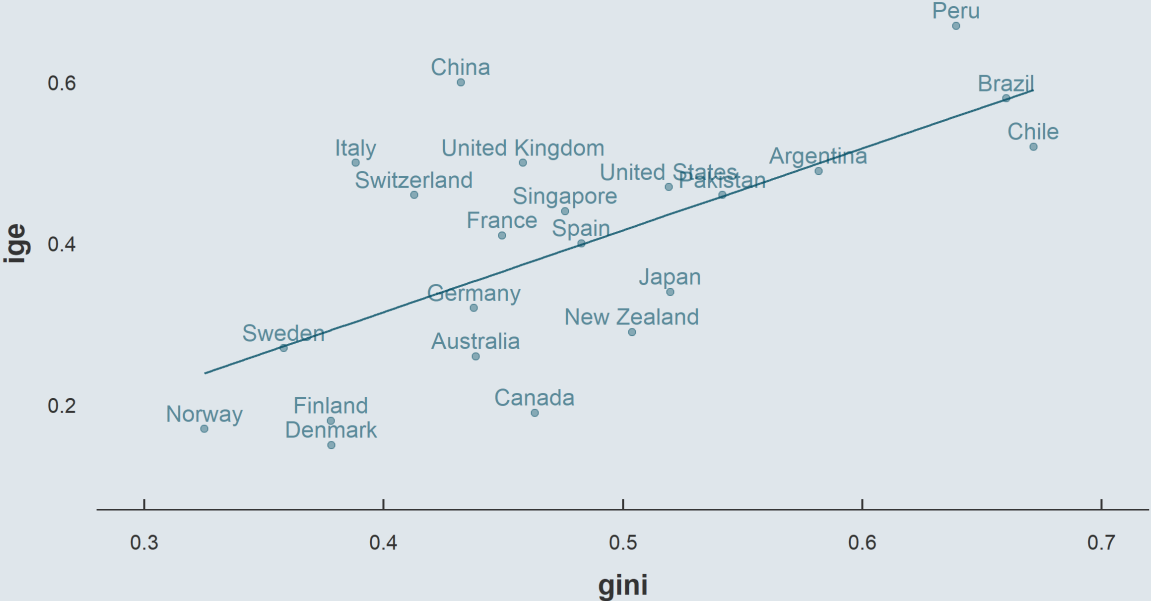




# 1. Regressions

## 1.1. On continuous variables

- You must already be quite familiar with univariate **regressions**  $y = \alpha + \beta x + \varepsilon$

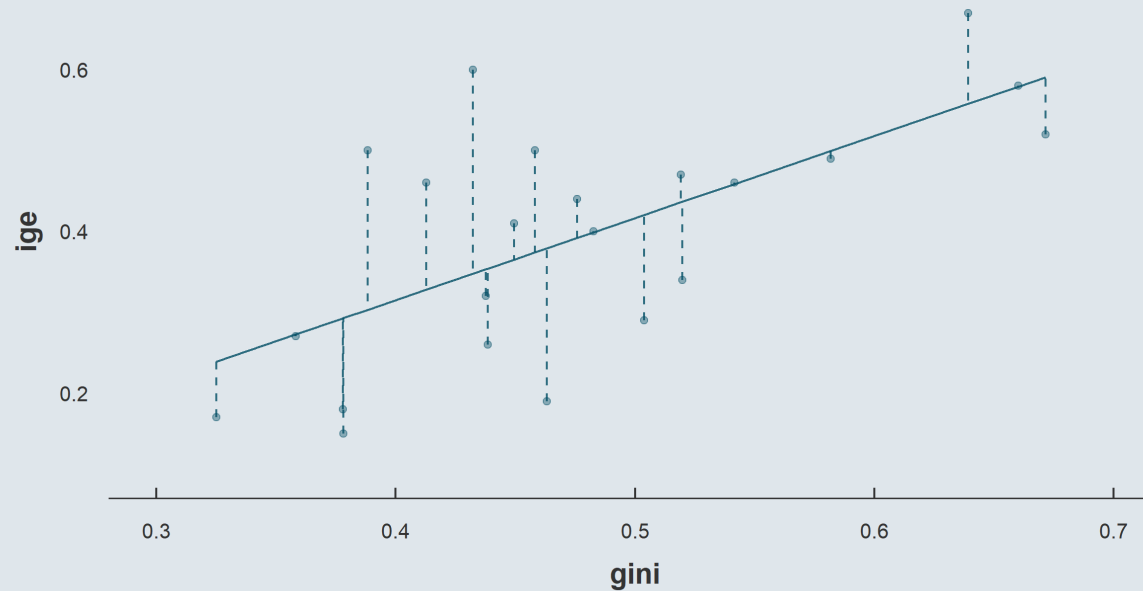




# 1. Regressions

## 1.1. On continuous variables

- You must already be quite familiar with univariate **regressions**  $y = \alpha + \beta x + \varepsilon$ 
  - We're looking for the line  $\hat{y}_i = \hat{\alpha} + \hat{\beta}x_i$  that **mimimizes distance** to the data points  $\sum \varepsilon_i^2$

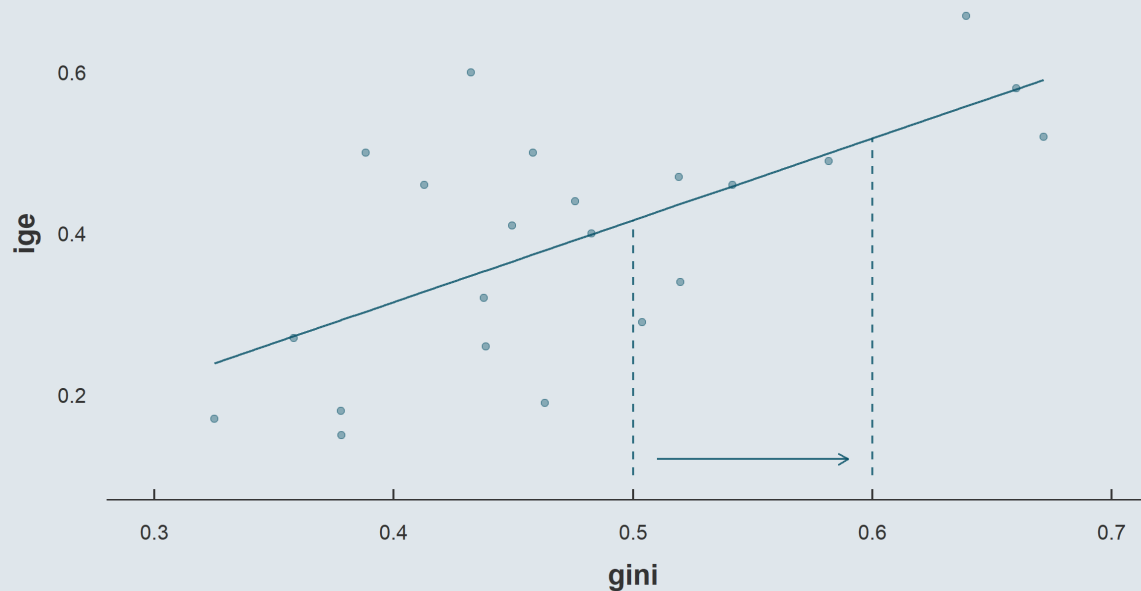




# 1. Regressions

## 1.1. On continuous variables

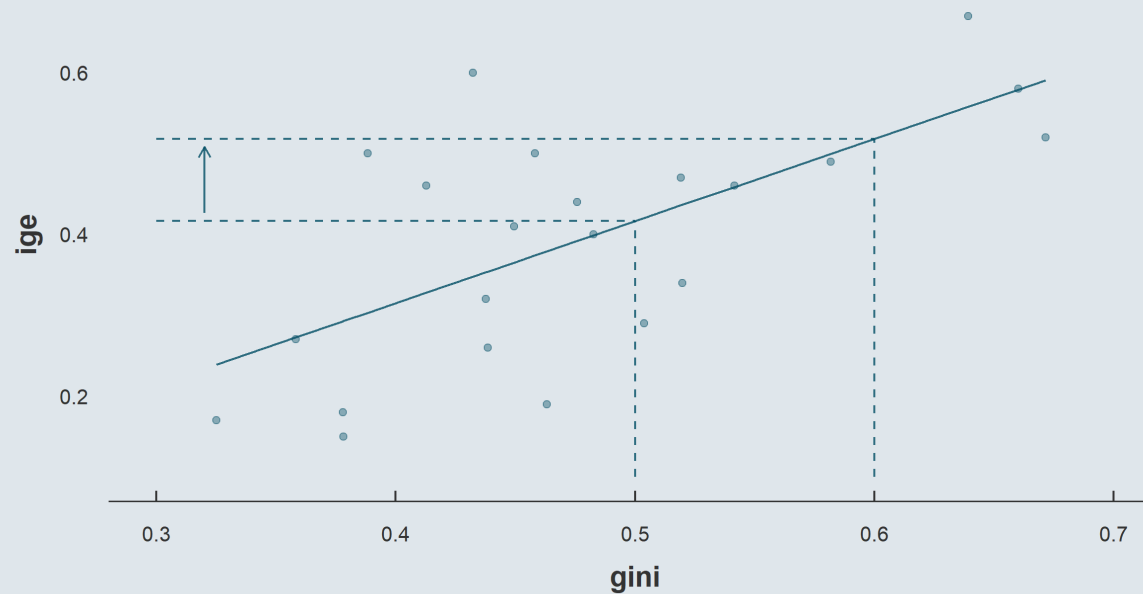
- You must already be quite familiar with univariate **regressions**  $y = \alpha + \beta x + \varepsilon$ 
  - We're looking for the line  $\hat{y}_i = \hat{\alpha} + \hat{\beta}x_i$  that **mimimizes distance** to the data points  $\sum \varepsilon_i^2$
  - Such that for a **one unit increase** in  $x$



# 1. Regressions

## 1.1. On continuous variables

- You must already be quite familiar with univariate **regressions**  $y = \alpha + \beta x + \varepsilon$ 
  - We're looking for the line  $\hat{y}_i = \hat{\alpha} + \hat{\beta}x_i$  that **mimimizes distance** to the data points  $\sum \varepsilon_i^2$
  - Such that for a **one unit increase** in  $x$
  - Its slope  $\hat{\beta}$  indicates the associated **expected change** in  $y$





# 1. Regressions

## 1.1. On continuous variables

- In R we can **estimate a regression** model using the **lm()** command (*for **L**inear **M**odel*)
  - 1.
  - 2.

```
lm( , )
```



# 1. Regressions

## 1.1. On continuous variables

- In R we can **estimate a regression** model using the **lm()** command (*for Linear Model*)
  1. The first argument is the **formula**, written as **y ~ x**
  - 2.

```
lm(ige ~ gini, )
```

# 1. Regressions

## 1.1. On continuous variables

- In R we can **estimate a regression** model using the **lm()** command (*for Linear Model*)
  1. The first argument is the **formula**, written as **y ~ x**
  2. The second argument is the **data** containing the variables

```
lm(ige ~ gini, ggcurve)
```

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve)  
##  
## Coefficients:  
## (Intercept)          gini  
##   -0.09129         1.01546
```

- This is great but the **output** is a bit minimalistic
  - To get a more **exhaustive description** of our regression we can apply the **summary()** function to **lm()**

```
ggmodel <- lm(ige ~ gini, ggcurve) %>% summary()
```



# 1. Regressions

## 1.1. On continuous variables

```
ggmodel
```

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -0.188991 -0.088238 -0.000855  0.047284  0.252310   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept) -0.09129    0.12870  -0.709  0.48631      
## gini         1.01546    0.26425   3.843  0.00102 **   
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.1159 on 20 degrees of freedom  
## Multiple R-squared:  0.4247,    Adjusted R-squared:  0.396   
## F-statistic: 14.77 on 1 and 20 DF,  p-value: 0.001016
```









# 1. Regressions

## 1.1. On continuous variables

- **Coefficients** along with their **standard error**, **t-value**, and **p-value**

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -0.188991 -0.088238 -0.000855  0.047284  0.252310   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept) -0.09129     0.12870  -0.709  0.48631      
## gini         1.01546     0.26425   3.843  0.00102      
##  
##  
##  
##  
##  
##
```

← Command

← Residuals distribution

← Coefs, s.e., t-/p-values



# 1. Regressions

## 1.1. On continuous variables

- **Significance** thresholds with symbols

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve)  
##  
## Residuals:  
##      Min      1Q   Median      3Q      Max  
## -0.188991 -0.088238 -0.000855  0.047284  0.252310  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -0.09129    0.12870  -0.709  0.48631  
## gini         1.01546    0.26425   3.843  0.00102 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
##  
##  
##
```

← Command

← Residuals distribution

← Coefs, s.e., t-/p-values

← Significance



# 1. Regressions

## 1.1. On continuous variables

- The **residual standard error**  $\sqrt{\sum (y_i - \hat{y}_i)^2 / df}$  and **degrees of freedom**

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve) ← Command  
##  
## Residuals:  
##      Min      1Q   Median      3Q      Max ← Residuals distribution  
## -0.188991 -0.088238 -0.000855  0.047284  0.252310  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|) ← Coefs, s.e., t-/p-values  
## (Intercept) -0.09129    0.12870  -0.709  0.48631  
## gini         1.01546    0.26425   3.843  0.00102 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 ← Significance  
##  
## Residual standard error: 0.1159 on 20 degrees of freedom ← Residual s.e. & df.  
##  
##
```



# 1. Regressions

## 1.1. On continuous variables

- The  $R^2$  and **adjusted  $R^2$**

```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve)                                     ← Command  
##  
## Residuals:  
##      Min      1Q   Median      3Q      Max                                     ← Residuals distribution  
## -0.188991 -0.088238 -0.000855  0.047284  0.252310  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)                                     ← Coefs, s.e., t-/p-values  
## (Intercept) -0.09129    0.12870  -0.709  0.48631  
## gini         1.01546    0.26425   3.843  0.00102 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1                                     ← Significance  
##  
## Residual standard error: 0.1159 on 20 degrees of freedom                                     ← Residual s.e. & df.  
## Multiple R-squared:  0.4247,    Adjusted R-squared:  0.396                                     ← R2 & adjusted R2  
##
```

# 1. Regressions

## 1.1. On continuous variables

- The result of an **F-test** ( $H_0 : \beta_k = 0 \ \forall k$ )

```
##
## Call:
## lm(formula = ige ~ gini, data = ggcurve)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.188991 -0.088238 -0.000855  0.047284  0.252310
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.09129    0.12870  -0.709  0.48631
## gini         1.01546    0.26425   3.843  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1159 on 20 degrees of freedom
## Multiple R-squared:  0.4247,    Adjusted R-squared:  0.396
## F-statistic: 14.77 on 1 and 20 DF,  p-value: 0.001016
```

← Command

← Residuals distribution

← Coefs, s.e., t-/p-values

← Significance

← Residual s.e. & df.

← R<sup>2</sup> & adjusted R<sup>2</sup>

← F-test results

# 1. Regressions

## 1.1. On continuous variables

- All these elements are then easily accessible using the `$` operator

```
str(ggmodel, give.attr = F)
```

```
## List of 11
## $ call      : language lm(formula = ige ~ gini, data = ggcurve)
## $ terms     :Classes 'terms', 'formula' language ige ~ gini
## $ residuals : Named num [1:22] -0.1427 -0.0687 -0.1125 -0.189 -0.094 ...
## $ coefficients : num [1:2, 1:4] -0.0913 1.0155 0.1287 0.2642 -0.7093 ...
## $ aliases    : Named logi [1:2] FALSE FALSE
## $ sigma      : num 0.116
## $ df         : int [1:3] 2 20 2
## $ r.squared   : num 0.425
## $ adj.r.squared: num 0.396
## $ fstatistic  : Named num [1:3] 14.8 1 20
## $ cov.unscaled : num [1:2, 1:2] 1.23 -2.48 -2.48 5.19
```

→ *Let's try it out!*





# 1. Regressions

## 1.1. On continuous variables

- Take the **coefficients** for instance

```
ggmodel$coefficients
```

```
##           Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) -0.09129311  0.1287045 -0.7093234 0.486311455
## gini        1.01546204  0.2642477  3.8428420 0.001015706
```

- We can **subset** this matrix like we would do with a regular `data.frame`

```
ggmodel$coefficients[2, 1]
```

```
## [1] 1.015462
```

```
ggmodel$coefficients[, "Pr(>|t|)"]
```

```
## (Intercept)      gini
## 0.486311455 0.001015706
```

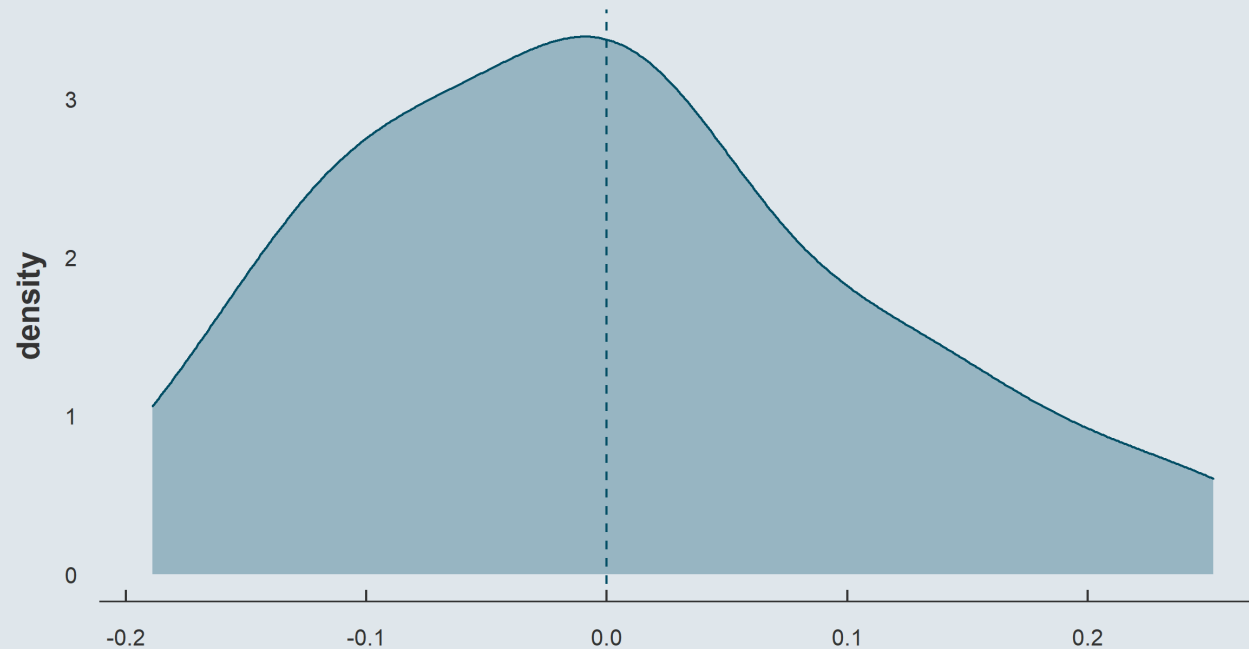


# 1. Regressions

## 1.1. On continuous variables

- We can also easily **plot** the **distribution** of our **residuals**

```
ggplot(data.frame(x = ggmodel$residuals), aes(x = x)) +  
  geom_density() + geom_vline(xintercept = 0, linetype = "dashed")
```



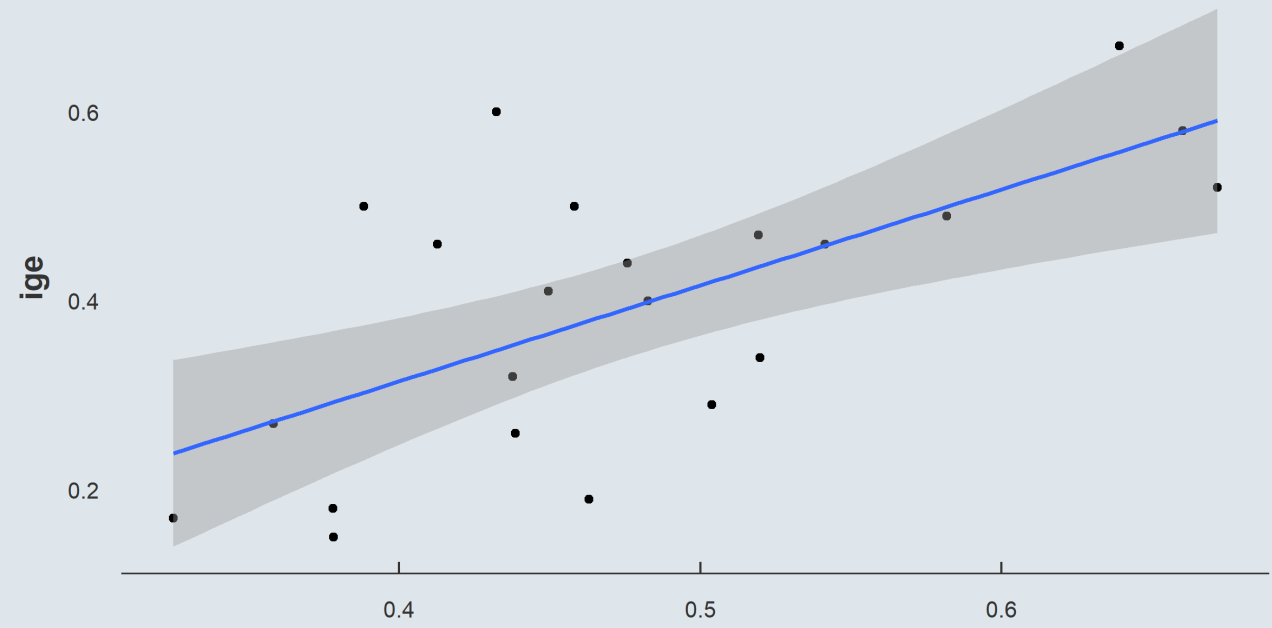


# 1. Regressions

## 1.1. On continuous variables

- Note that ggplot() has a dedicated **geometry for fitted values**: `geom_smooth()`

```
ggplot(ggcurve, aes(x = gini, y = ige)) +  
  geom_point() + geom_smooth(method = "lm")
```



# Practice

10:00

*Check that `lm()` works fine by computing the  $R^2$  manually*

1) Start by creating a variable for  $\hat{\beta}$ , then for  $\hat{\alpha}$ ,  $\hat{y}_i$ , and  $\hat{\varepsilon}_i$ .

$$\hat{\beta} = \frac{\text{Cov}(x, y)}{\text{Var}(x)} \quad ; \quad \hat{\alpha} = \bar{y} - \hat{\beta} \times \bar{x}$$

You're gonna need the `cov()` and `var()` functions

2) Summarise the data into only  $\hat{\alpha}$ ,  $\hat{\beta}$ , and the  $R^2$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2}$$

*You've got 10 minutes!*

# Solution

```
ggcurve %>%  
  mutate(beta = cov(gini, ige) / var(gini))  
#  
#  
#  
#  
#  
#
```

```
##      country ige      gini      beta  
## 1      Denmark 0.15 0.3781796 1.015462  
## 2      Norway 0.17 0.3250102 1.015462  
## 3      Finland 0.18 0.3779868 1.015462  
## 4      Canada 0.19 0.4631237 1.015462  
## 5      Australia 0.26 0.4385511 1.015462  
## 6      Sweden 0.27 0.3582480 1.015462  
## 7      New Zealand 0.29 0.5039373 1.015462  
## 8      Germany 0.32 0.4377010 1.015462  
## 9      Japan 0.34 0.5198383 1.015462  
## 10     Spain 0.40 0.4826243 1.015462  
## 11     France 0.41 0.4495654 1.015462  
## ..     ... ..     ...     ..
```

# Solution

```
ggcurve %>%  
  mutate(beta = cov(gini, ige) / var(gini),  
         alpha = mean(ige) - beta * mean(gini))  
#  
#  
#  
#  
#
```

```
##      country  ige      gini      beta      alpha  
## 1      Denmark 0.15 0.3781796 1.015462 -0.09129311  
## 2      Norway  0.17 0.3250102 1.015462 -0.09129311  
## 3      Finland 0.18 0.3779868 1.015462 -0.09129311  
## 4      Canada  0.19 0.4631237 1.015462 -0.09129311  
## 5      Australia 0.26 0.4385511 1.015462 -0.09129311  
## 6      Sweden  0.27 0.3582480 1.015462 -0.09129311  
## 7      New Zealand 0.29 0.5039373 1.015462 -0.09129311  
## 8      Germany 0.32 0.4377010 1.015462 -0.09129311  
## 9      Japan   0.34 0.5198383 1.015462 -0.09129311  
## 10     Spain   0.40 0.4826243 1.015462 -0.09129311  
## 11     France  0.41 0.4495654 1.015462 -0.09129311  
## ..      ...    ...      ...      ...      ...
```

# Solution

```
ggcurve %>%
  mutate(beta = cov(gini, ige) / var(gini),
         alpha = mean(ige) - beta * mean(gini),
         fit = alpha + beta * gini)
#
#
#
#

##      country  ige      gini      beta      alpha      fit
## 1    Denmark 0.15 0.3781796 1.015462 -0.09129311 0.2927339
## 2     Norway 0.17 0.3250102 1.015462 -0.09129311 0.2387424
## 3    Finland 0.18 0.3779868 1.015462 -0.09129311 0.2925381
## 4     Canada 0.19 0.4631237 1.015462 -0.09129311 0.3789915
## 5   Australia 0.26 0.4385511 1.015462 -0.09129311 0.3540389
## 6     Sweden 0.27 0.3582480 1.015462 -0.09129311 0.2724942
## 7 New Zealand 0.29 0.5039373 1.015462 -0.09129311 0.4204361
## 8     Germany 0.32 0.4377010 1.015462 -0.09129311 0.3531757
## 9       Japan 0.34 0.5198383 1.015462 -0.09129311 0.4365829
## 10    Spain 0.40 0.4826243 1.015462 -0.09129311 0.3987935
## 11    France 0.41 0.4495654 1.015462 -0.09129311 0.3652234
## ..      ...      ...      ...      ...      ...
```

# Solution

```
ggcurve %>%  
  mutate(beta = cov(gini, ige) / var(gini),  
         alpha = mean(ige) - beta * mean(gini),  
         fit = alpha + beta * gini,  
         residuals = ige - fit)  
#  
#  
#
```

```
##      country  ige      gini      beta      alpha      fit      residuals  
## 1      Denmark 0.15 0.3781796 1.015462 -0.09129311 0.2927339 -0.1427339244  
## 2      Norway 0.17 0.3250102 1.015462 -0.09129311 0.2387424 -0.0687424324  
## 3      Finland 0.18 0.3779868 1.015462 -0.09129311 0.2925381 -0.1125381050  
## 4      Canada 0.19 0.4631237 1.015462 -0.09129311 0.3789915 -0.1889914561  
## 5      Australia 0.26 0.4385511 1.015462 -0.09129311 0.3540389 -0.0940388831  
## 6      Sweden 0.27 0.3582480 1.015462 -0.09129311 0.2724942 -0.0024941569  
## 7      New Zealand 0.29 0.5039373 1.015462 -0.09129311 0.4204361 -0.1304360777  
## 8      Germany 0.32 0.4377010 1.015462 -0.09129311 0.3531757 -0.0331756674  
## 9      Japan 0.34 0.5198383 1.015462 -0.09129311 0.4365829 -0.0965829037  
## 10     Spain 0.40 0.4826243 1.015462 -0.09129311 0.3987935 0.0012065012  
## 11     France 0.41 0.4495654 1.015462 -0.09129311 0.3652234 0.0447765627  
## ..      ...      ...      ...      ...      ...      ...
```



# Solution

```
ggcurve %>%  
  mutate(beta = cov(gini, ige) / var(gini),  
         alpha = mean(ige) - beta * mean(gini),  
         fit = alpha + beta * gini,  
         residuals = ige - fit) %>%  
  summarise(alpha = alpha[1],  
            beta = beta[1],  
            r2 = 1 - sum(residuals^2)/sum((ige - mean(ige))^2))
```

```
##           alpha      beta      r2  
## 1 -0.09129311  1.015462  0.424749
```

```
ggmodel$coefficients[, "Estimate"]
```

```
## (Intercept)      gini  
## -0.09129311  1.01546204
```

```
ggmodel$r.squared
```

```
## [1] 0.424749
```



# 1. Regressions

## 1.2. On binary variables

- Now consider that we want to know the **relationship** between **not** being a **European country** and the **ige**

```
ggcurve$country
```

```
## [1] "Denmark"      "Norway"      "Finland"    "Canada"
## [5] "Australia"    "Sweden"     "New Zealand" "Germany"
## [9] "Japan"        "Spain"      "France"     "Singapore"
## [13] "Pakistan"     "Switzerland" "United States" "Argentina"
## [17] "Italy"        "United Kingdom" "Chile"      "Brazil"
## [21] "China"       "Peru"
```

```
europe <- c("Denmark", "Norway", "Finland", "Sweden", "Germany", "Spain",
           "France", "Switzerland", "Italy", "United Kingdom")
```

```
ggcurve <- ggcurve %>%
  mutate(continent = ifelse(country %in% europe, "Europe", "Other"))
```

# 1. Regressions

## 1.2. On binary variables

- Can we just **regress** ige **on** continent even though it's a **character** variable?

```
lm(ige ~ continent, ggcurve)
```

```
##  
## Call:  
## lm(formula = ige ~ continent, data = ggcurve)  
##  
## Coefficients:  
##      (Intercept)  continentOther  
##           0.3360           0.1065
```

- Seems like we can!

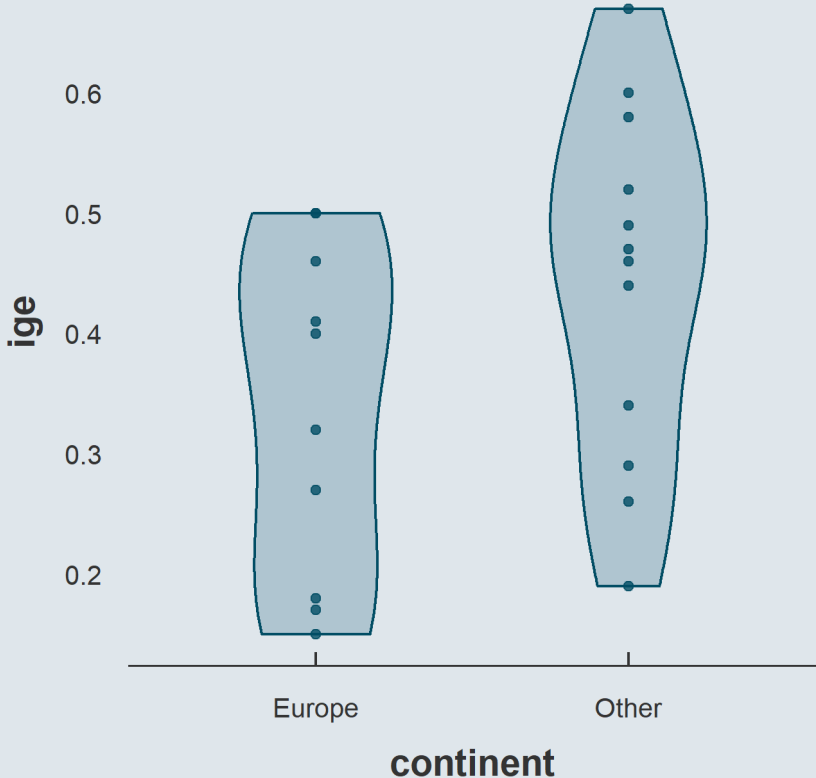
**→ But what's actually going on?**



# 1. Regressions

## 1.2. On binary variables

- R implicitly **converts** character variables into **binary** variables

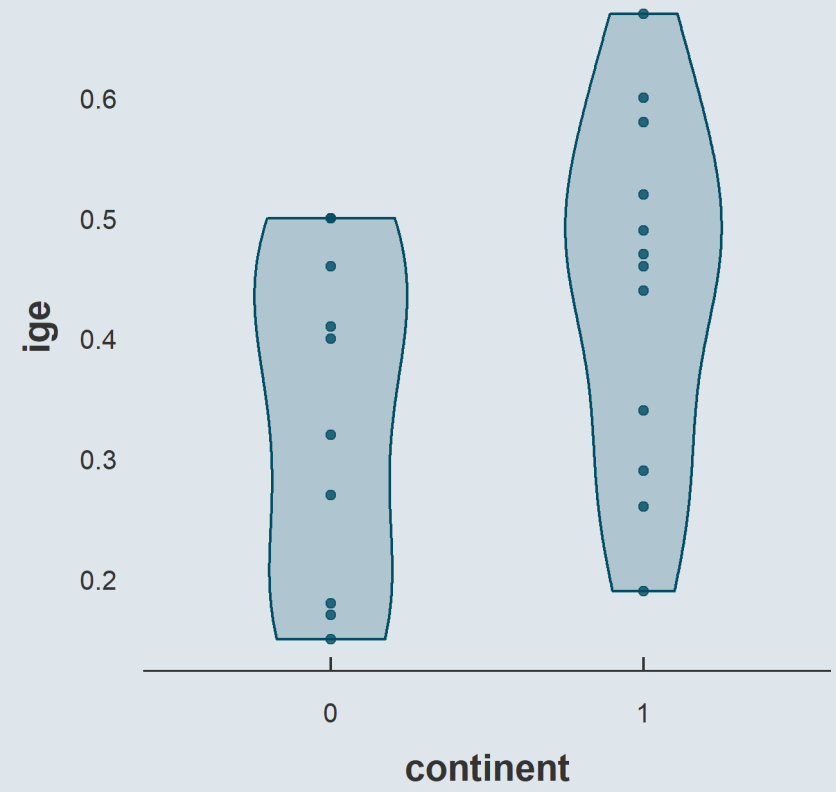




# 1. Regressions

## 1.2. On binary variables

- Such that just **like** in the **continuous** case...

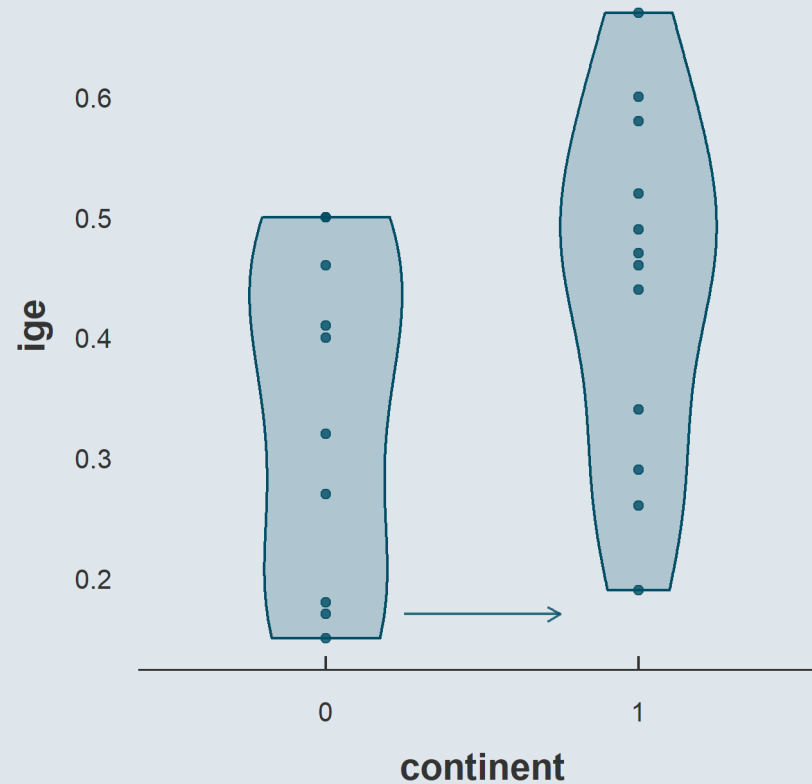




# 1. Regressions

## 1.2. On binary variables

- ... we're looking at a **1-unit increase** in  $x$  (i.e., switching from 0 (Europe) to 1 (Other))

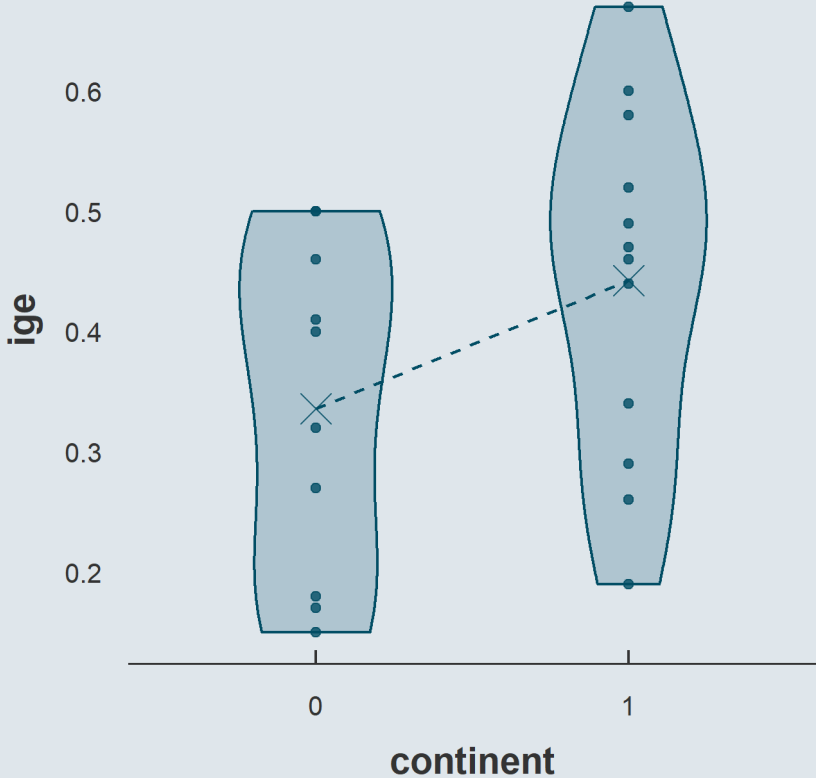




# 1. Regressions

## 1.2. On binary variables

- As you know the **fit** is necessarily going through the **mean of each category**

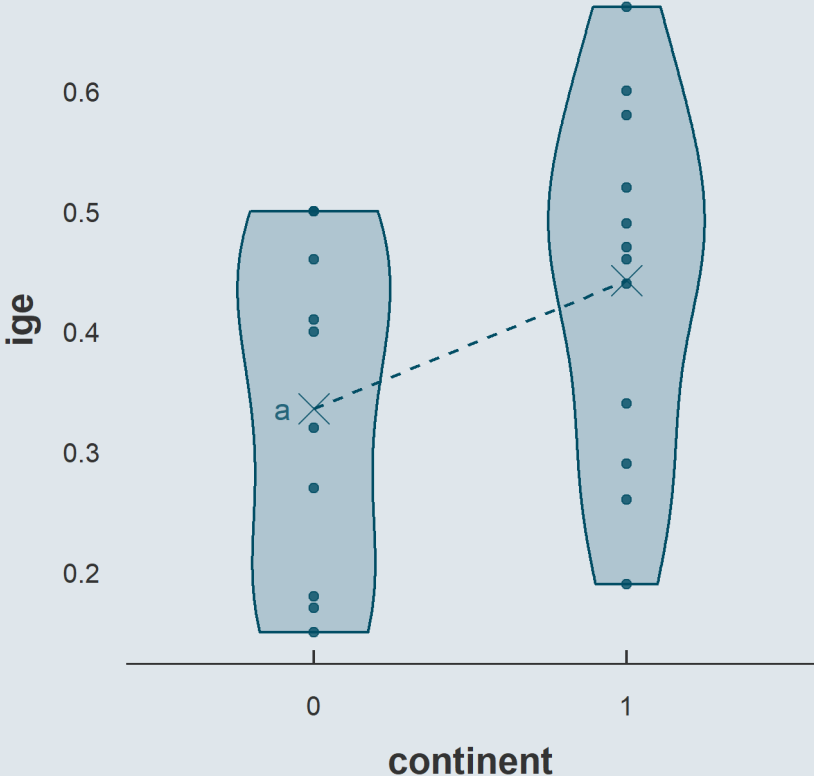




# 1. Regressions

## 1.2. On binary variables

- Such that  $\hat{\alpha}$  is the **mean of the reference group**



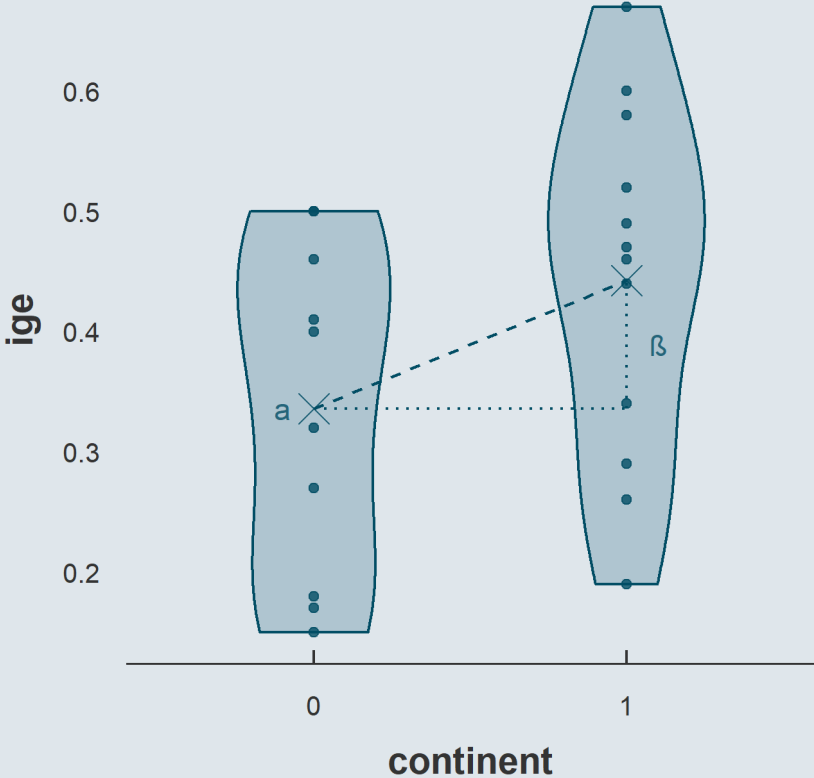




# 1. Regressions

## 1.2. On binary variables

- And  $\hat{\beta}$  is the **difference in means**





# 1. Regressions

## 1.2. On binary variables

- We can verify that easily:

```
ggcurve %>%  
  group_by(continent) %>%  
  summarise(ybar = mean(ige)) %>%  
  mutate(relative = ybar - ybar[1])
```

```
## # A tibble: 2 x 3  
##   continent ybar relative  
##   <chr>     <dbl>   <dbl>  
## 1 Europe    0.336     0  
## 2 Other     0.442    0.106
```

```
lm(ige ~ continent, ggcurve)
```

```
##  
## Call:  
## lm(formula = ige ~ continent, data = ggcurve)  
##  
## Coefficients:  
##   (Intercept) continentOther  
##           0.3360           0.1065
```

→ And what about discrete variables with more than 2 categories?



# 1. Regressions

## 1.3. On categorical variables

- Let's work on the [2020 Annual Social and Economic \(ASEC\) Supplement to the US CPS](#)
  - Here is an extract on 64,336 working individuals with positive earnings
  - For which I kept only 4 variables:

```
asec <- read.csv("asec.csv")
str(asec)
```

```
## 'data.frame':    64336 obs. of  4 variables:
## $ Sex      : chr  "Female" "Male" "Female" "Male" ...
## $ Earnings: int   52500 34000 40000 8424 58000 42000 55000 28000 200 25000 ...
## $ Race     : chr  "White" "White" "White" "White" ...
## $ Hours    : int   40 40 44 21 60 40 40 40 20 40 ...
```

- Let's say we want to regress earnings on Race

```
unique(asec$Race)
```

```
## [1] "White" "Other" "Black"
```



# 1. Regressions

## 1.3. On categorical variables

- Just like the **2-category** variable was equivalent to **1 dummy** variable
  - An **n-category** variable is equivalent to **n-1 dummy** variables

| Continent | Other |  | Race  | Other | White |
|-----------|-------|--|-------|-------|-------|
| Europe    | 0     |  | Black | 0     | 0     |
| Europe    | 0     |  | Black | 0     | 0     |
| Europe    | 0     |  | Other | 1     | 0     |
| Other     | 1     |  | Other | 1     | 0     |
| Other     | 1     |  | White | 0     | 1     |
| Other     | 1     |  | White | 0     | 1     |

```
lm(Earnings ~ Race, asec)
```

```
##  
## Call:  
## lm(formula = Earnings ~ Race, data = asec)  
##  
## Coefficients:  
## (Intercept)      RaceOther      RaceWhite  
##      50577      17477      12303
```

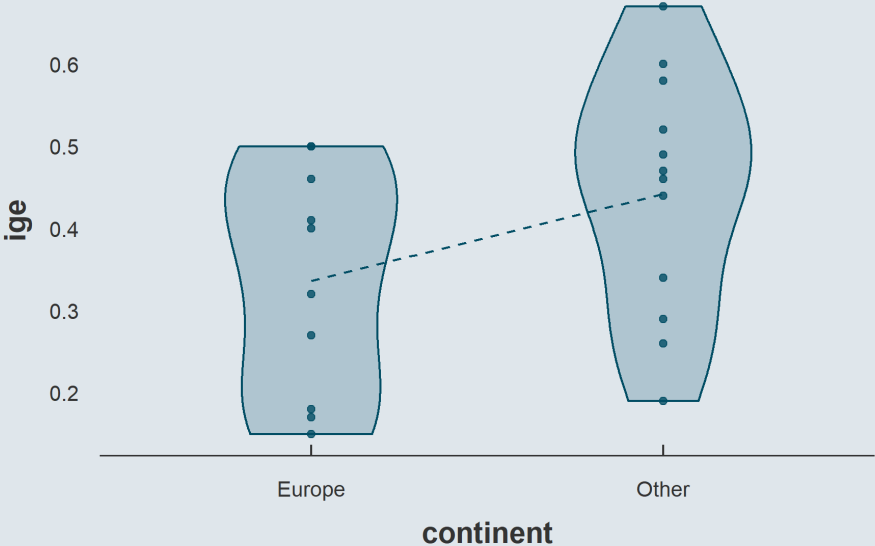
- Instead of 1  $x$  axis we're going to have n-1  $x$  axes



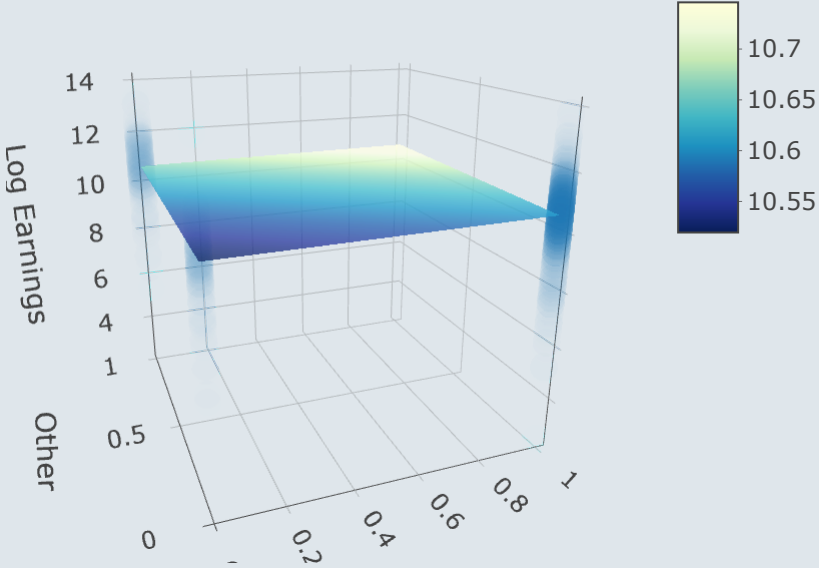
# 1. Regressions

## 1.3. On categorical variables

2-category variable



3-category variable



# 1. Regressions

## 1.3. On categorical variables

- Once again the **constant** is the **average  $y$**  for the reference category
  - And the **slopes** are the relative **differences in means**

```
lm(Earnings ~ Race, asec)
```

```
##
## Call:
## lm(formula = Earnings ~ Race, data = asec)
##
## Coefficients:
## (Intercept)      RaceOther      RaceWhite
##          50577          17477          12303
```

```
asec %>%
  group_by(Race) %>%
  summarise(ybar = mean(Earnings)) %>%
  mutate(relative = ybar - ybar[1])
```

```
## # A tibble: 3 x 3
##   Race      ybar relative
##   <chr> <dbl> <dbl>
## 1 Black  50577.      0
## 2 Other  68055.  17477.
## 3 White  62880.  12303.
```

- Note that R always sorts **character** variables by **alphabetical order**
  - But it would be more natural to interpret relative to the **majority group**
  - Then how to change the **reference category** in the regression?



# 1. Regressions

## 1.3. On categorical variables

- The function to set the reference category is **relevel()**
  - The first argument is the **vector** to indicate the reference of
  - The second argument is the value of the **reference group**

```
lm(Earnings ~ relevel(Race, "White"), asec)
```

```
## Error in relevel.default(Race, "White"): 'relevel' only for (unordered) factors
```

- Oops! I need to introduce you a **new class** of R objects first: **factors**

```
as.factor(asec$Race) %>%  
  levels()
```

```
## [1] "Black" "Other" "White"
```

```
as.factor(asec$Race) %>%  
  relevel("White") %>%  
  levels()
```

```
## [1] "White" "Black" "Other"
```

# 1. Regressions

## 1.3. On categorical variables

- The **factor class** is made for variables whose values **indicate** different **groups**
  - Values are just **arbitrary group classifiers**

```
individuals <- as.factor(c(1, 2, 3, 4, 5))
individuals[1]
```

```
## [1] 1
## Levels: 1 2 3 4 5
```

- With **factors**, R understands that the different values **do not mean anything**
  - And applying **standard operations** to factors **does not make sense**

```
individuals * 2
```

```
## Warning in Ops.factor(individuals, 2): '*' not meaningful for factors
```

```
## [1] NA NA NA NA NA
```



# Practice

05:00

- 1) Open the data from the World Inequality Database we used in lecture 2
- 2) Regress the income share of the top 1% on the year variable
- 3) Redo the same regression after having converted the year variable as a **factor**

*You've got 5 minutes!*

# Solution

## 1) Open the data from the World Inequality Database we used in lecture 2

```
wid <- read.csv("wid.csv")
```

## 2) Regress the income share of the top 1% on the year variable

```
lm(top1 ~ year, wid)
```

```
##  
## Call:  
## lm(formula = top1 ~ year, data = wid)  
##  
## Coefficients:  
## (Intercept)          year  
##  2.242e-01    -3.131e-05
```

# Solution

## 3) Redo the same regression after having converted the year variable as a **factor**

```
lm(top1 ~ year, wid %>% mutate(year = as.factor(year)))
```

```
##  
## Call:  
## lm(formula = top1 ~ year, data = wid %>% mutate(year = as.factor(year)))  
##  
## Coefficients:  
## (Intercept)      year2011      year2012      year2013      year2014      year2015  
##  0.1621494    -0.0011366    -0.0026891    -0.0013401    -0.0004469    -0.0008683  
##   year2016      year2017      year2018      year2019  
## -0.0001450    -0.0004857    -0.0015236    -0.0018488
```

# 1. Regressions

## 1.3. On categorical variables

- Another option to include categorical variables is to **one hot encode** the data
  - It simply means **converting** the discrete variables into dummies such that **everything is numeric**

```
asec <- asec %>% mutate(White = as.numeric(Race == "White"),
                      Black = as.numeric(Race == "Black"),
                      Other = as.numeric(Race == "Other"))
```

- Then we can use the `+` symbol to include n-1 categories in the regression

```
lm(Earnings ~ Black + Other, asec)
```

```
##
## Call:
## lm(formula = Earnings ~ Black + Other, data = asec)
##
## Coefficients:
## (Intercept)      Black      Other
##      62880      -12303       5174
```



# 1. Regressions

## 1.3. On categorical variables

- But do we really need to **omit** one **category**?

- As we are in the multivariate case let's move from  $\hat{\beta} = \frac{\text{Cov}(x,y)}{\text{Var}(x)}$

- To  $\hat{\beta} = (X'X)^{-1}X'y$

```
y <- as.matrix(asec$Earnings)

X <- asec %>%
  mutate(constant = 1) %>%
  select(constant, White, Black, Other) %>%
  as.matrix()
```

```
dim(X)
```

```
## [1] 64336 4
```

```
dim(t(X))
```

```
## [1] 4 64336
```

```
dim(t(X) %*% X)
```

```
## [1] 4 4
```

# 1. Regressions

## 1.3. On categorical variables

- Because of perfect **multicollinearity** it will not be possible to invert  $X'X$

```
solve(t(X) %*% X)
```

```
## Error in solve.default(t(X) %*% X): system is computationally singular
```

- We have to **remove one category**

```
X <- asec %>% mutate(constant = 1) %>%  
  select(constant, Black, Other) %>%  
  as.matrix()
```

```
solve(t(X) %*% X) %*% (t(X) %*% y)
```

```
##           [,1]  
## constant 62880.488  
## Black    -12302.993  
## Other     5174.141
```

- Or to **remove the constant**

```
X <- asec %>% #mutate(constant = 1) %>%  
  select(White, Black, Other) %>%  
  as.matrix()
```

```
solve(t(X) %*% X) %*% (t(X) %*% y)
```

```
##           [,1]  
## White 62880.49  
## Black 50577.49  
## Other 68054.63
```



# 1. Regressions

## 1.3. On categorical variables

- Note that you can remove the constant in `lm()` by adding `- 1` in the formula

```
lm(Earnings ~ White + Black + Other - 1, asec)
```

```
## Coefficients:  
## White   Black   Other  
## 62880   50577   68055
```

- And that it would drop a category anyway

```
lm(Earnings ~ White + Black + Other, asec)
```

```
## Coefficients:  
## (Intercept)      White      Black      Other  
##      68055      -5174      -17477      NA
```

***But it's not because multicollinearity does not break `lm()` that you should not pay attention to it!***



# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results

- 4.1. Regression tables
- 4.2. Plot coefficients

## 5. Wrap up!





# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 2. Case study

### 2.1. Variable transformation

- Imagine you want to estimate the **relationship** between **weekly hours** of work and **annual earnings**

$$\text{Earnings}_i = \alpha + \beta \text{Hours}_i + \varepsilon_i$$

- We can **estimate it in R** using the Annual Social and Economic Supplement to the US CPS

```
lm(Earnings ~ Hours, asec)
```

```
## Coefficients:
## (Intercept)      Hours
##      -20039      2078
```

- Are we done?

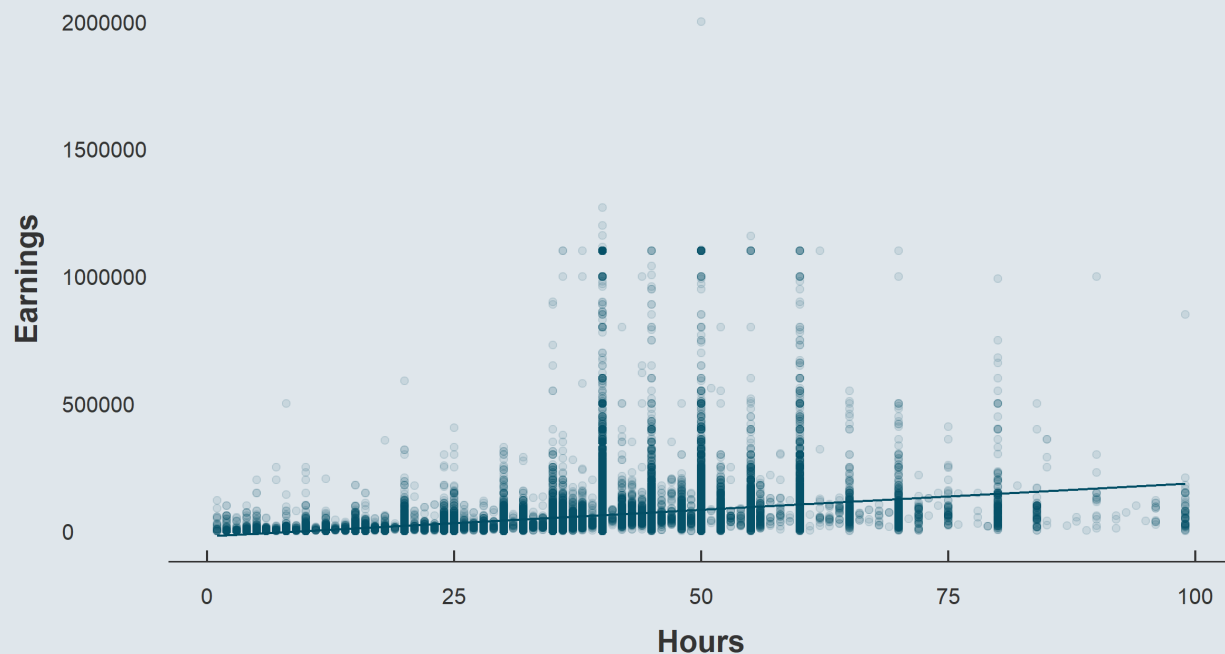
**→ Let's take a look at what we just did!**



## 2. Case study

### 2.1. Variable transformation

```
ggplot(asec, aes(x = Hours, y = Earnings)) +  
  geom_point() + geom_smooth(method = "lm")
```



→ Not very satisfactory

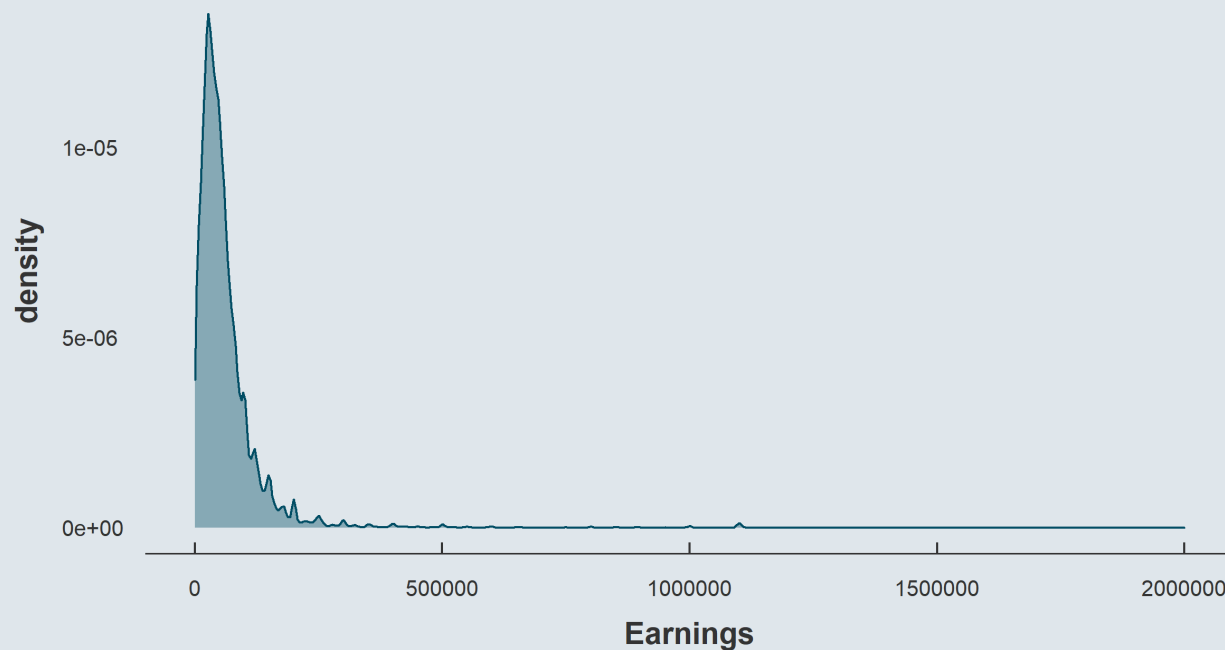
- The joint distribution does **not seem adequate** on the the **y dimension**
- Let's take a **look** at the earnings **distribution**



## 2. Case study

### 2.1. Variable transformation

```
ggplot(asec, aes(x = Earnings)) +  
  geom_density()
```



It's clearly **log-normal**

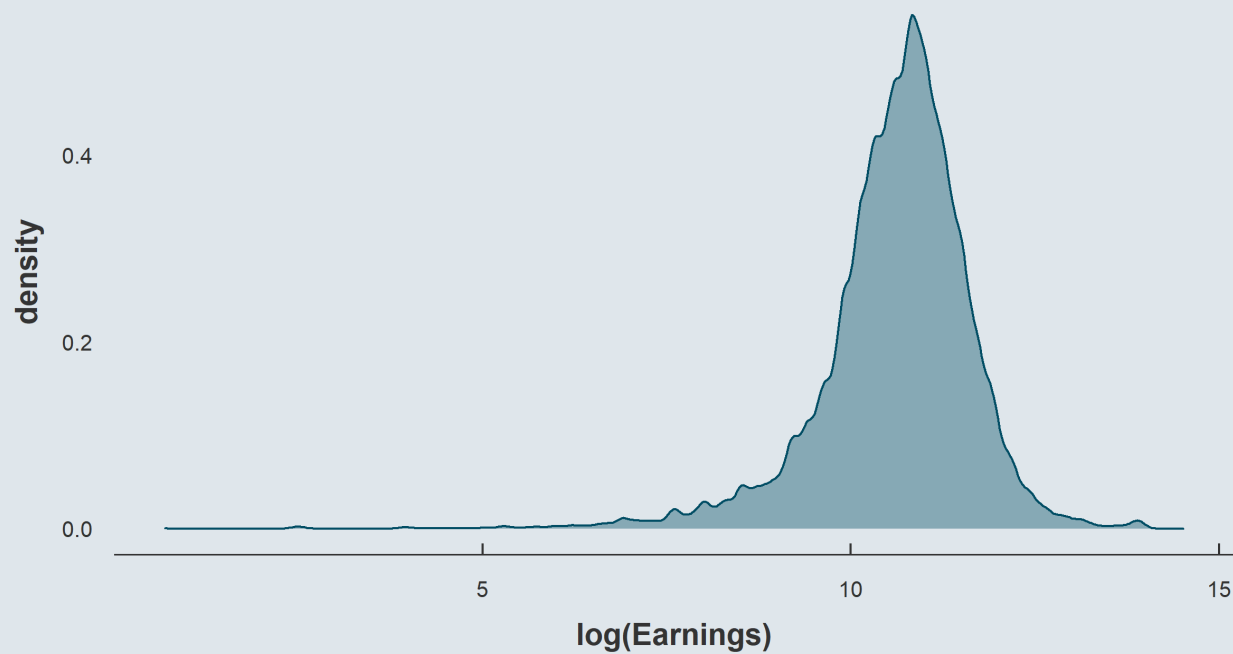
→ Let's **plot the log** of it



## 2. Case study

### 2.1. Variable transformation

```
ggplot(asec, aes(x = log(Earnings))) +  
  geom_density()
```



**Better!**

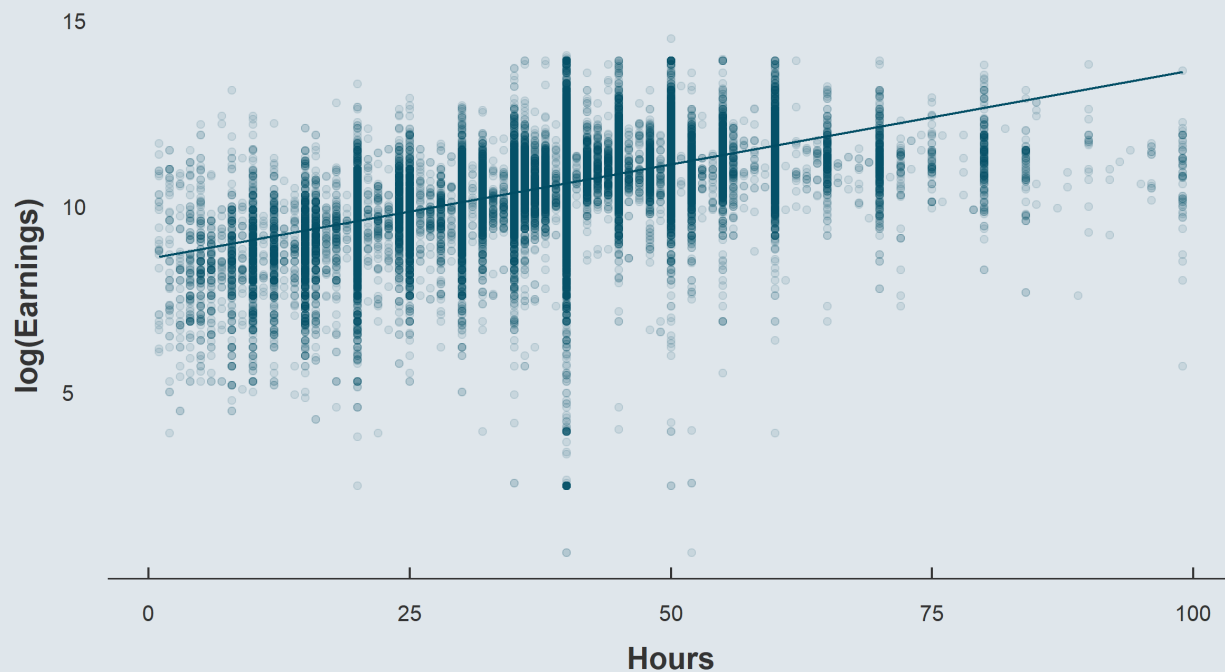
→ Let's **update** the **scatterplot**



## 2. Case study

### 2.2. Functional form

```
ggplot(asec, aes(x = Hours, y = log(Earnings))) +  
  geom_point(alpha = .1) + geom_smooth(method = "lm")
```



Definitely better!

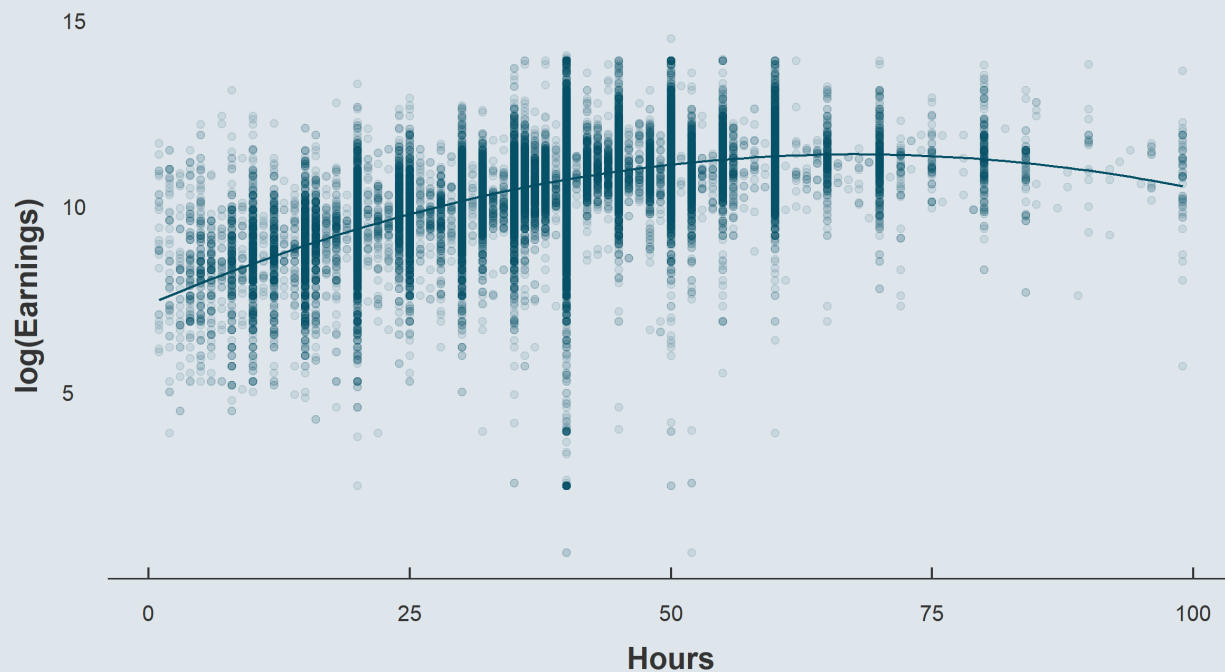
→ But something **still** feels off



## 2. Case study

### 2.2. Functional form

```
ggplot(asec, aes(x = Hours, y = log(Earnings))) +  
  geom_point(alpha = .1) + geom_smooth(method = "lm", formula = y ~ poly(x, 2))
```



→ *There we go*

## 2. Case study

### 2.2. Functional form

- We'd better rewrite our model as:

$$\log(\text{Earnings}_i) = \alpha + \beta_1 \text{Hours}_i + \beta_2 \text{Hours}_i^2 + \varepsilon_i$$

- Create the new variables

```
asec <- asec %>%
  mutate(lEarnings = log(Earnings),
         sqHours = Hours^2)
```

- And run the regression

```
lm(lEarnings ~ Hours + sqHours, asec)
```

```
## Coefficients:
## (Intercept)      Hours      sqHours
##  7.3637848    0.1192609   -0.0008804
```

**Is that it?**

*Isn't there something we could/should add in our regression?*





## 2. Case study

### 2.3. Control variables

- This positive **relationship** could be **driven** by a **third variable**
  - **Males** tend both to work **part time less often** and to **earn more**
  - The **higher** the **hours**, the higher the **probability to be a male**, the higher the **expected earnings**
- Let's **control** for that in the regression

$$\log(\text{Earnings}_i) = \alpha + \beta_1 \text{Hours}_i + \beta_2 \text{Hours}_i^2 + \beta_3 \text{Male}_i + \varepsilon_i$$

- Such that Males and Females would have
  - The **same slope**
  - **Different intercepts**

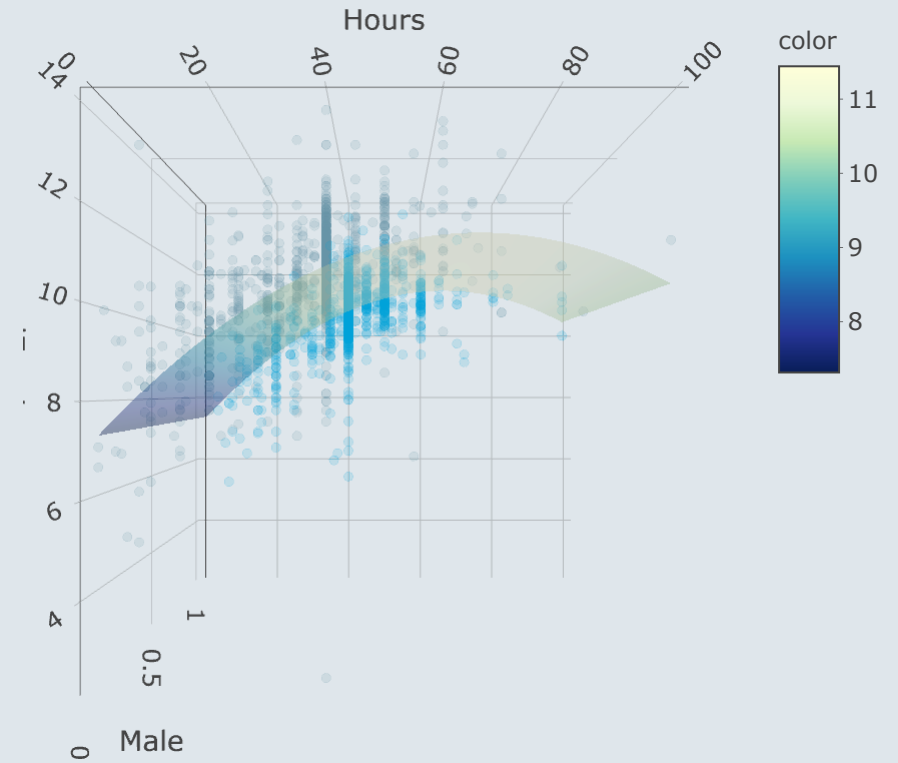
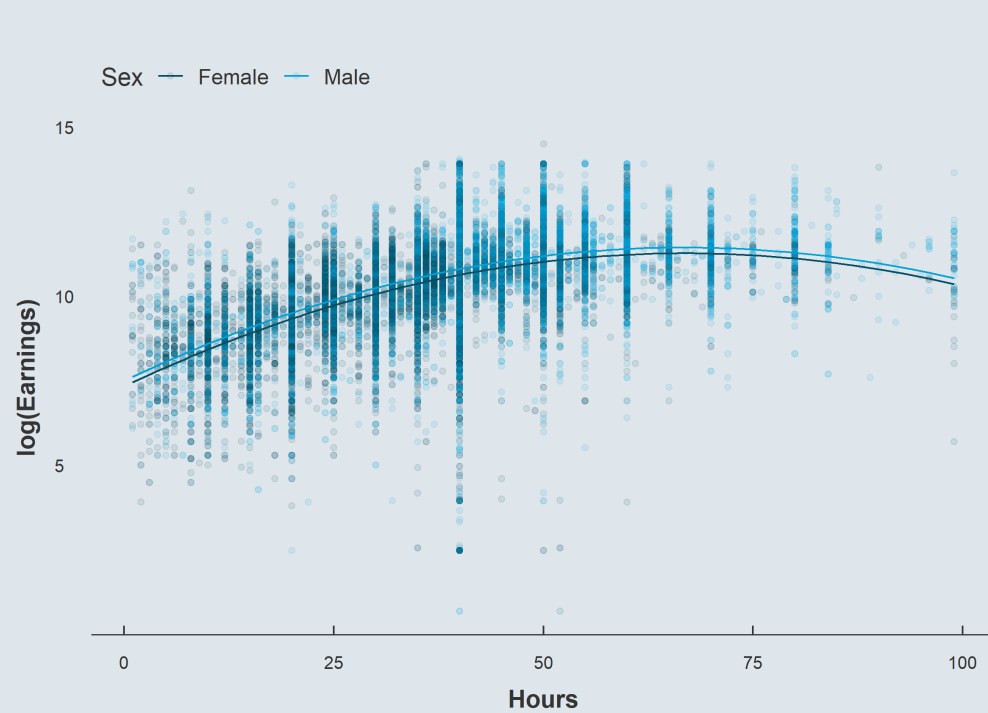
|           | Male = 0                          | Male = 1                          |
|-----------|-----------------------------------|-----------------------------------|
| Intercept | $\alpha$                          | $\alpha + \beta_3$                |
| Slope     | $\beta_1 + 2\beta_2 \text{Hours}$ | $\beta_1 + 2\beta_2 \text{Hours}$ |

*Let's take a look*



## 2. Case study

### 2.3. Control variables



## 2. Case study

### 2.3. Control variables

- Once again we can include a control variable using the + symbol

```
lm(lEarnings ~ Hours + sqHours + Sex, asec)
```

```
## Coefficients:  
## (Intercept)      Hours      sqHours      SexMale  
##  7.3356057    0.1177514   -0.0008806    0.1700972
```

```
lm(lEarnings ~ Hours + sqHours, asec)
```

```
## Coefficients:  
## (Intercept)      Hours      sqHours  
##  7.3637848    0.1192609   -0.0008804
```

*Actually the baseline coefficient was not that inflated*



## 2. Case study

### 2.4. Interactions

- But what if we were to allow for **different slopes**?
  - The **relationship** between hours and earnings might be **heterogeneous** across sex/gender
  - This is what **interactions** allow to account for
- We simply have to include the **product** of the two variables in the model

$$\log(\text{Earnings}_i) = \alpha + \beta_1 \text{Hours}_i + \beta_2 \text{Hours}_i^2 + \beta_3 \text{Male}_i + \beta_4 \text{Hours}_i \times \text{Male}_i + \varepsilon_i$$

- Such that Males and Females would have
  - Not only **different intercepts**
  - But also **different slopes**

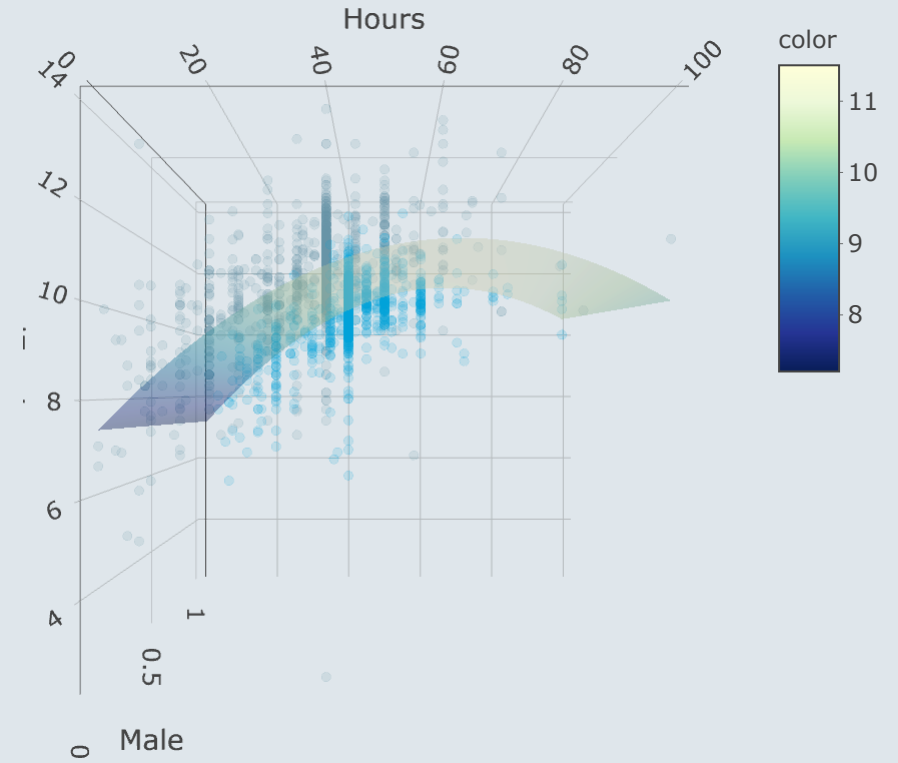
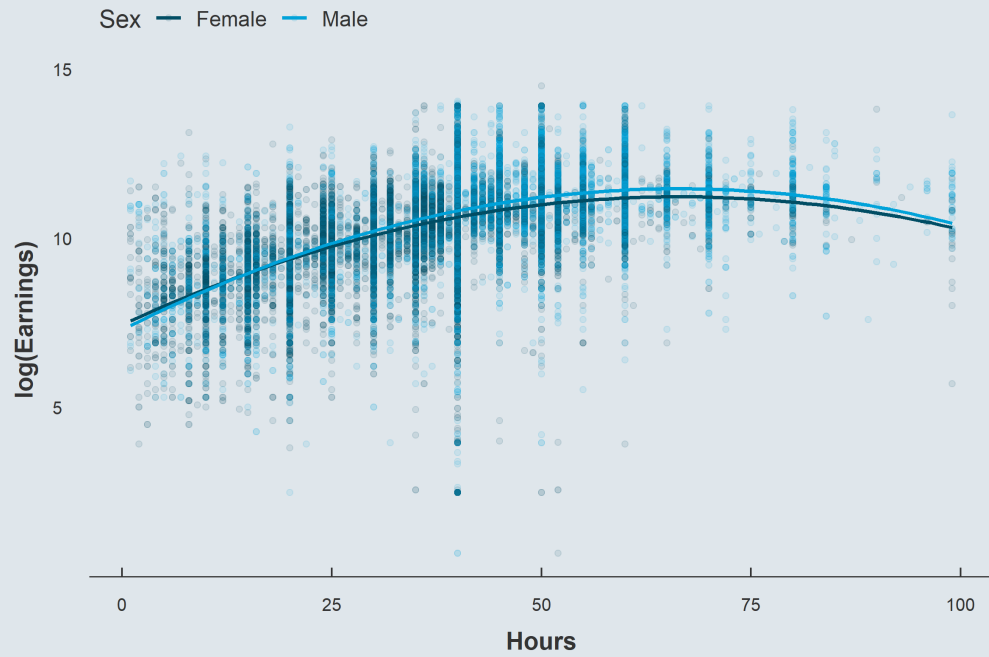
|           | Male = 0                          | Male = 1                                    |
|-----------|-----------------------------------|---|
| Intercept | $\alpha$                          | $\alpha + \beta_3$                          |
| Slope     | $\beta_1 + 2\beta_2 \text{Hours}$ | $\beta_1 + 2\beta_2 \text{Hours} + \beta_4$ |

*Let's take a look*



# 2. Case study

## 2.4. Interactions





## 2. Case study

### 2.4. Interactions

- Now we can use the `*` symbol for products

```
results <- summary(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec))$coefficients
results
```

```
##              Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)  7.3900468002  2.319198e-02  318.646690  0.0000000e+00
## Hours        0.1174998826  1.034522e-03  113.578950  0.0000000e+00
## sqHours     -0.0009103523  1.321706e-05  -68.877048  0.0000000e+00
## SexMale     -0.0282905673  2.753222e-02   -1.027544  3.041682e-01
## Hours:SexMale 0.0050321134  6.767013e-04    7.436240  1.048736e-13
```

What do you conclude ?

```
results[5, 1] / results[2, 1]
```

```
## [1] 0.04282654
```

→ A 4% difference

```
results[5, 4]
```

```
## [1] 1.048736e-13
```

→ Which is highly significant



# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study ✓

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results

- 4.1. Regression tables
- 4.2. Plot coefficients

## 5. Wrap up!



# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study ✓

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference

- 3.1. Hypothesis testing
- 3.2. Confidence intervals



# 3. Inference

## 3.1. Hypothesis testing

- According to our previous regression,  $\hat{\beta}_1$  is significantly **different from 0**
  - But let's pretend that you know that this coefficient is equal to **.12 in Canada**
  - How could we test whether or not our coefficient is **different from .12?**

```
results
```

| ## |               | Estimate      | Std. Error   | t value    | Pr(> t )     |
|----|---------------|---------------|--------------|------------|--------------|
| ## | (Intercept)   | 7.3900468002  | 2.319198e-02 | 318.646690 | 0.000000e+00 |
| ## | Hours         | 0.1174998826  | 1.034522e-03 | 113.578950 | 0.000000e+00 |
| ## | sqHours       | -0.0009103523 | 1.321706e-05 | -68.877048 | 0.000000e+00 |
| ## | SexMale       | -0.0282905673 | 2.753222e-02 | -1.027544  | 3.041682e-01 |
| ## | Hours:SexMale | 0.0050321134  | 6.767013e-04 | 7.436240   | 1.048736e-13 |

- We can compute the t-stat from our results

$$t = \frac{\hat{\beta} - .12}{\text{s.e.}(\hat{\beta})}$$

```
t <- (results[2, 1] - .12) / results[2, 2]
t
```

```
## [1] -2.416689
```



# 3. Inference

## 3.1. Hypothesis testing

- And then we need the value of the **area outside the interval**  $[-t; t]$ 
  - From a **Student-t** distribution with the correct number of **degrees of freedom** (#obs - #parameters)



# 3. Inference

## 3.1. Hypothesis testing

- You can get the **area below** a certain **t-value** with the **pt()** function
  - 1.
  - 2.

```
pt( , )
```



# 3. Inference

## 3.1. Hypothesis testing

- You can get the **area below** a certain **t-value** with the **pt()** function
  1. The first argument is the **t-value**
  - 2.

```
pt(t, )
```

# 3. Inference

## 3.1. Hypothesis testing

- You can get the **area below** a certain **t-value** with the **pt()** function
  - The first argument is the **t-value**
  - The second argument is the number of **degrees of freedom**

```
pt(t, nrow(asec) - nrow(results))
```

```
## [1] 0.007832573
```

- We just have to multiply this value by 2 to obtain the p-value

```
2 * pt(t, nrow(asec) - nrow(results))
```

```
## [1] 0.01566515
```

- Had our t-stat been positive we would have needed to multiply  $1 - t$  by 2

```
2 * (1 - pt(abs(t), nrow(asec)-nrow(results)))
```

```
## [1] 0.01566515
```



# 3. Inference

## 3.1. Hypothesis testing

- A very handy function for hypothesis testing is **linearHypothesis()** from the **car** package
  - 
  -

```
linearHypothesis( , )
```



# 3. Inference

## 3.1. Hypothesis testing

- A very handy function for hypothesis testing is **linearHypothesis()** from the **car** package
  - The first argument is the **model**
  -

```
linearHypothesis(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec), )
```

# 3. Inference

## 3.1. Hypothesis testing

- A very handy function for hypothesis testing is **linearHypothesis()** from the **car** package
  - The first argument is the **model**
  - The second argument is the **hypothesis/es**

```
linearHypothesis(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec), c("Hours = .12"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## Hours = 0.12
##
## Model 1: restricted model
## Model 2: lEarnings ~ Hours + sqHours + Sex + Hours * Sex
##
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1  64332 46102
## 2  64331 46098   1    4.1851 5.8404 0.01567 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```





# 3. Inference

## 3.1. Hypothesis testing

- It can be used for **F tests** like the one from the summary

```
linearHypothesis(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),  
                 c("Hours = 0", "sqHours = 0", "SexMale = 0", "Hours:SexMale = 0"))
```



# 3. Inference

## 3.1. Hypothesis testing

```
linearHypothesis(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),  
                 c("Hours = 0", "sqHours = 0", "SexMale = 0", "Hours:SexMale = 0"))
```

```
## Linear hypothesis test  
##  
## Hypothesis:  
## Hours = 0  
## sqHours = 0  
## SexMale = 0  
## Hours:SexMale = 0  
##  
## Model 1: restricted model  
## Model 2: lEarnings ~ Hours + sqHours + Sex + Hours * Sex  
##  
##   Res.Df  RSS Df Sum of Sq    F    Pr(>F)  
## 1   64335 68395  
## 2   64331 46098  4    22297 7779.1 < 2.2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



## 3. Inference

```
##  
## Call:  
## lm(formula = lEarnings ~ Hours + sqHours + Sex + Hours * Sex,  
##      data = asec)  
##  
## Residuals:  
##      Min      1Q   Median      3Q      Max  
## -10.3453  -0.4299   0.0133   0.4810   4.8506  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept)  7.390e+00  2.319e-02 318.647 < 2e-16 ***  
## Hours        1.175e-01  1.035e-03 113.579 < 2e-16 ***  
## sqHours      -9.103e-04  1.322e-05 -68.877 < 2e-16 ***  
## SexMale      -2.829e-02  2.753e-02  -1.028  0.304  
## Hours:SexMale 5.032e-03  6.767e-04   7.436 1.05e-13 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.8465 on 64331 degrees of freedom  
## Multiple R-squared:  0.326,    Adjusted R-squared:  0.326  
## F-statistic: 7779 on 4 and 64331 DF,  p-value: < 2.2e-16
```

# 3. Inference

## 3.2. Confidence intervals

- Now we know how to get the area below a given value of a Student  $t$  distribution
  - But sometimes the opposite is useful as well
  - In particular to compute **confidence intervals**
- Indeed the confidence interval for a  $\hat{\beta}$  coefficient is given by:

$$\hat{\beta} \pm t_{1-(\alpha/2),df} \times \text{s.e.}(\hat{\beta})$$

- Where  $\alpha$  denotes the desired significance level
- **qt()** gives the  $t$ -statistic below which lies the desired share of the area of a given Student  $t$  distribution
  - 
  -

qt( , )

# 3. Inference

## 3.2. Confidence intervals

- Now we know how to get the area below a given value of a Student  $t$  distribution
  - But sometimes the opposite is useful as well
  - In particular to compute **confidence intervals**
- Indeed the confidence interval for a  $\hat{\beta}$  coefficient is given by:

$$\hat{\beta} \pm t_{1-(\alpha/2),df} \times \text{s.e.}(\hat{\beta})$$

- Where  $\alpha$  denotes the desired significance level
- **qt()** gives the  $t$ -statistic below which lies the desired share of the area of a given Student  $t$  distribution
  - The first argument is  $1 - (\alpha/2)$
  -

```
qt(.975, )
```

# 3. Inference

## 3.2. Confidence intervals

- Now we know how to get the area below a given value of a Student  $t$  distribution
  - But sometimes the opposite is useful as well
  - In particular to compute **confidence intervals**
- Indeed the confidence interval for a  $\hat{\beta}$  coefficient is given by:

$$\hat{\beta} \pm t_{1-(\alpha/2),df} \times \text{s.e.}(\hat{\beta})$$

- Where  $\alpha$  denotes the desired significance level
- **qt()** gives the  $t$ -statistic below which lies the desired share of the area of a given Student  $t$  distribution
  - The first argument is  $1 - (\alpha/2)$
  - The second argument is the number of **degrees of freedom**

```
qt(.975, Inf)
```

```
## [1] 1.959964
```



# 3. Inference

## 3.2. Confidence intervals

- So if we want a 97% **confidence interval** for our coefficients associated with hours, we feed **qt()** with:
  - 
  -

```
t003 <- qt( , )
```

# 3. Inference

## 3.2. Confidence intervals

- So if we want a 97% **confidence interval** for our coefficients associated with hours, we feed **qt()** with:
  - The **share** of the Student  $t$  **distribution** below the desired  $t$ -stat:  $1 - (\alpha/2) = 1 - (0.03/2) = 0.985$
  -

```
t003 <- qt(.985, )
```



# 3. Inference

## 3.2. Confidence intervals

- So if we want a 97% **confidence interval** for our coefficients associated with hours, we feed **qt()** with:
  - The **share** of the Student *t* **distribution** below the desired *t*-stat:  $1 - (\alpha/2) = 1 - (0.03/2) = 0.985$
  - The **degrees of freedom**:  $\#obs. - \#params$

```
t003 <- qt(.985, nrow(asec) - nrow(results))
```

# 3. Inference

## 3.2. Confidence intervals

- So if we want a 97% **confidence interval** for our coefficients associated with hours, we feed **qt()** with:
  - The **share** of the Student *t* **distribution** below the desired *t*-stat:  $1 - (\alpha/2) = 1 - (0.03/2) = 0.985$
  - The **degrees of freedom**:  $\#obs. - \#params$

```
t003 <- qt(.985, nrow(asec) - nrow(results))
t003
```

```
## [1] 2.170139
```

- And we apply the formula:

```
results[2, 1] - (results[2, 2] * t003)
```

```
## [1] 0.1152548
```

```
results[2, 1] + (results[2, 2] * t003)
```

```
## [1] 0.1197449
```



# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study ✓

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference ✓

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results

- 4.1. Regression tables
- 4.2. Plot coefficients

## 5. Wrap up!



# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study ✓

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference ✓

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results

- 4.1. Regression tables
- 4.2. Plot coefficients



# 4. Report and export results

## 4.1. Regression tables

- The output of the **summary()** function is very **practical** but **not very convenient** to report the main results
  - Academic regression tables** rather look like that

**Table 5**  
The effect of distance to the line on production levels.

|   | Dependent Variable Ln Per Capita GDP |                     |                     |                     |                     |                     |
|---|--------------------------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
|   | (1)                                  | (2)                 | (3)                 | (4)                 | (5)                 | (6)                 |
| Ln Distance to Historical Lines             | -0.0617<br>(0.0286)                  | -0.0434<br>(0.0281) | -0.0491<br>(0.0277) | -0.0581<br>(0.0265) | -0.0699<br>(0.0270) | -0.0681<br>(0.0272) |
| Ln Distance to Segment City                 |                                      | 0.065<br>(0.232)    | -0.061<br>(0.266)   | -0.063<br>(0.282)   | -0.178<br>(0.325)   | -0.208<br>(0.298)   |
| Ln Distance to Segment City <sup>2</sup>    |                                      | -0.0276<br>(0.0277) | -0.0089<br>(0.0308) | -0.0071<br>(0.0324) | 0.0072<br>(0.0372)  | 0.0101<br>(0.0344)  |
| Ln Distance to Navigable River              |                                      |                     | 0.318<br>(0.153)    | 0.321<br>(0.141)    | 0.366<br>(0.140)    | 0.385<br>(0.138)    |
| Ln Distance to Navigable River <sup>2</sup> |                                      |                     | -0.0517<br>(0.0200) | -0.0481<br>(0.0186) | -0.0550<br>(0.0188) | -0.0554<br>(0.0183) |
| Ln Area                                     |                                      |                     |                     | -1.572<br>(0.681)   | -1.442<br>(0.635)   | -1.441<br>(0.642)   |
| Ln Area <sup>2</sup>                        |                                      |                     |                     | 0.0983<br>(0.0486)  | 0.0911<br>(0.0459)  | 0.0894<br>(0.0464)  |
| Ln Distance to Coastline                    |                                      |                     |                     |                     | -0.243<br>(0.224)   | -0.207<br>(0.219)   |
| Ln Distance to Coastline <sup>2</sup>       |                                      |                     |                     |                     | 0.0168<br>(0.0265)  | 0.0141<br>(0.0259)  |
| Ln Distance to Country Border               |                                      |                     |                     |                     |                     | -16.49<br>(6.097)   |
| Ln Distance to Country Border <sup>2</sup>  |                                      |                     |                     |                     |                     | 1.241<br>(0.459)    |
| Observations                                | 2744                                 | 2744                | 2744                | 2744                | 2744                | 2744                |
| R-squared                                   | 0.818                                | 0.826               | 0.833               | 0.845               | 0.849               | 0.852               |

Notes: All regressions control for year and province fixed effects. Standard errors are clustered at the county level. These estimates use an unbalanced county-year level panel. GDP data are from Provincial Statistical Yearbooks. All geographic variables are computed by the authors.



## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables.
  - 
  - 
  - 
  - 
  - 
  -

```
outreg <- huxreg()  
#  
#  
#  
#  
#  
#  
#  
#
```

## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - 
  - 
  - 
  - 
  -

```
outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),  
                lm(lEarnings ~ Hours + sqHours, asec),  
                lm(lEarnings ~ Hours + sqHours + Sex, asec),  
                lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec))  
  
#  
#  
#  
#  
#
```

## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - Which **uncertainty statistic** to display (std.error, p.value, conf.low, conf.high)
  - 
  - 
  - 
  -

```
outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),
                 lm(lEarnings ~ Hours + sqHours, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
                 error_format = "{std.error}")
```

```
#
#
#
#
```



## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - Which **uncertainty statistic** to display (std.error, p.value, conf.low, conf.high)
  - Where to **place** the uncertainty statistic (below, same, right)
  - 
  - 
  -

```

outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),
                 lm(lEarnings ~ Hours + sqHours, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
                 error_format = "{std.error}",
                 error_pos = "below")

```

```

#
#
#

```

## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - Which **uncertainty statistic** to display (std.error, p.value, conf.low, conf.high)
  - Where to **place** the uncertainty statistic (below, same, right)
  - Which **general statistics** to display (adj.r.squared, df, ...)
  - 
  -

```

outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),
                 lm(lEarnings ~ Hours + sqHours, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex, asec),
                 lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
                 error_format = "{std.error}",
                 error_pos = "below",
                 statistics = c(N = "nobs", R2 = "r.squared"))

#
#

```

## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - Which **uncertainty statistic** to display (std.error, p.value, conf.low, conf.high)
  - Where to **place** the uncertainty statistic (below, same, right)
  - Which **general statistics** to display (adj.r.squared, df, ...)
  - The desired **significance symbology**
  -

```

outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),
                lm(lEarnings ~ Hours + sqHours, asec),
                lm(lEarnings ~ Hours + sqHours + Sex, asec),
                lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
                error_format = "{std.error}",
                error_pos = "below",
                statistics = c(N = "nobs", R2 = "r.squared"),
                stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1))

```

#

## 4. Report and export results

### 4.1. Regression tables

- The **huxreg()** function from **huxtable** allows to create such tables. Main arguments include:
  - As many **models** as you want, named or not
  - Which **uncertainty statistic** to display (std.error, p.value, conf.low, conf.high)
  - Where to **place** the uncertainty statistic (below, same, right)
  - Which **general statistics** to display (adj.r.squared, df, ...)
  - The desired **significance symbology**
  - What to write in the **table footnote**

```
outreg <- huxreg(Baseline = lm(lEarnings ~ Hours, asec),
                lm(lEarnings ~ Hours + sqHours, asec),
                lm(lEarnings ~ Hours + sqHours + Sex, asec),
                lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
                error_format = "{std.error}",
                error_pos = "below",
                statistics = c(N = "nobs", R2 = "r.squared"),
                stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
                note = "Dependent variable: log annual earnings. {stars}")
```



# 4. Report and export results

## 4.1. Regression tables

```
## -----
##                               Baseline          (2)          (3)          (4)
## -----
## (Intercept)                   8.604 ***          7.364 ***          7.336 ***          7.390 ***
##                               (0.014)          (0.022)          (0.022)          (0.023)
## Hours                          0.051 ***          0.119 ***          0.118 ***          0.117 ***
##                               (0.000)          (0.001)          (0.001)          (0.001)
## sqHours                       -0.001 ***          -0.001 ***          -0.001 ***          -0.001 ***
##                               (0.000)          (0.000)          (0.000)          (0.000)
## SexMale                        0.170 ***          -0.028
##                               (0.007)          (0.028)
## Hours:SexMale                   0.005 ***
##                               (0.001)
## -----
## N                               64336          64336          64336          64336
## R2                              0.268          0.319          0.325          0.326
## -----
## Dependent variable: log annual earnings. *** p < 0.01; ** p < 0.05; * p < 0.1
```

- Then export it with `quick_[latex/html/pdf/docx](outreg, file = "path/filename.format")`



# 4. Report and export results

## 4.1. Regression tables

```
1 \documentclass{article}
2 \usepackage{array}
3 \usepackage{caption}
4 \usepackage{graphicx}
5 \usepackage{siunitx}
6 \usepackage[normalem]{ulem}
7 \usepackage{colortbl}
8 \usepackage{multirow}
9 \usepackage{hhline}
10 \usepackage{calc}
11 \usepackage{tabularx}
12 \usepackage{threeparttable}
13 \usepackage{wrapfig}
14 \usepackage{adjustbox}
15 \usepackage{hyperref}
16 % These are LaTeX packages. You can install them using your LaTeX management
17 % software,
18 % or by running 'huxtable::install_latex_dependencies()' from within R.
19 % Other packages may be required if you use non-standard tabulars (e.g. tabulaxy).
20 \pagenumbering{gobble}
21 \begin{document}
22
23
24 \providecommand{\huxb}[2]{\arrayrulecolor [RGB]{#1}\global\arrayrulewidth=#2pt}
25 \providecommand{\huxvb}[2]{\color [RGB]{#1}\vrule width #2pt}
26 \providecommand{\huxtpad}[1]{\rule{0pt}{#1}}
27 \providecommand{\huxbpad}[1]{\rule[-#1]{0pt}{#1}}
28
29 \begin{table}[ht]
30 \begin{center}
31 \begin{threeparttable}
32 \setlength{\tabcolsep}{0pt}
33 \begin{tabular}{|l|l|l|l|}
34
35
36 \hhline{>{\huxb{0, 0, 0}{0.8}}->{\huxb{0, 0, 0}{0.8}}->{\huxb{0, 0, 0}{0.8}}->
37 {\huxb{0, 0, 0}{0.8}}->{\huxb{0, 0, 0}{0.8}}-}
38 \arrayrulecolor{black}
```

|               | Baseline             | (2)                   | (3)                   | (4)                   |
|---------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)   | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours         | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| sqHours       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| SexMale       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| Hours:SexMale |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N             | 64336                | 64336                 | 64336                 | 64336                 |
| R2            | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1



# 4. Report and export results

## 4.1. Regression tables

```
1 <!DOCTYPE html>
2 <html lang="French-France">
3 <head><meta charset="utf8"><title>hux.html</title></head>
4 <body>
5
6 <p>&nbsp;</p><table class="huxtable" style="border-collapse: collapse; border: 0px; margin-bottom: 2em
7 <col><col><col><col><col><tr>
8 <th style="vertical-align: top; text-align: center; white-space: normal; border-style: solid solid sol
9 <tr>
10 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
11 <tr>
12 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
13 <tr>
14 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
15 <tr>
16 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
17 <tr>
18 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
19 <tr>
20 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
21 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
22 <tr>
23 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
24 <tr>
25 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
26 <tr>
27 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
28 <tr>
29 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
30 <tr>
31 <th style="vertical-align: top; text-align: left; white-space: normal; padding: 6pt 6pt 6pt 6pt; font-t
32 <tr style="vertical-align: top; text-align: left; white-space: normal; border-style: solid solid solid
33 <tr>
34 <th colspan="5" style="vertical-align: top; text-align: left; white-space: normal; border-style: solid
35 </table>
36
37
38 </body></html>
```

|               | Baseline             | (2)                   | (3)                   | (4)                   |
|---------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)   | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours         | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| sqHours       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| SexMale       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| Hours:SexMale |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N             | 64336                | 64336                 | 64336                 | 64336                 |
| R2            | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1



|               | Baseline             | (2)                   | (3)                   | (4)                   |
|---------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)   | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours         | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| sqHours       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| SexMale       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| Hours:SexMale |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N             | 64336                | 64336                 | 64336                 | 64336                 |
| R2            | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1





# 4. Report and export results

## 4.1. Regression tables

|               | Baseline             | (2)                   | (3)                   | (4)                   |
|---------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)   | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours         | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| sqHours       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| SexMale       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| Hours:SexMale |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N             | 64336                | 64336                 | 64336                 | 64336                 |
| R2            | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1

|                      | Baseline             | (2)                   | (3)                   | (4)                   |
|----------------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)          | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours                | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| <u>sqHours</u>       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| <u>SexMale</u>       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| <u>Hours:SexMale</u> |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N                    | 64336                | 64336                 | 64336                 | 64336                 |
| R2                   | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1



# Practice

10:00

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce this table:

```
##                               Dependent variable: Log annual earnings
##                               (1)          (2)          (3)          (4)
##                               _____
## Hours worked                 0.051 ***   0.119 ***   0.118 ***   0.117 ***
##                               (0.000)   (0.000)   (0.000)   (0.000)
## (Hours worked)2                -0.001 ***   -0.001 ***   -0.001 ***
##                               (0.000)   (0.000)   (0.000)
## Male                          0.170 ***   -0.028
##                               (0.000)   (0.304)
## Hours worked x Male                0.005 ***
##                               (0.000)
## Constant                      8.604 ***   7.364 ***   7.336 ***   7.390 ***
##                               (0.000)   (0.000)   (0.000)   (0.000)
##                               _____
## N                             64336     64336     64336     64336
## R2                             0.268     0.319     0.325     0.326
## _____
## Significance: *** p < 0.01; ** p < 0.05; * p < 0.1
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),  
      lm(lEarnings ~ Hours + sqHours, asec),  
      lm(lEarnings ~ Hours + sqHours + Sex, asec),  
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec))  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below")
#
#
#
#
#
#
#
#
#
#
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),  
      lm(lEarnings ~ Hours + sqHours, asec),  
      lm(lEarnings ~ Hours + sqHours + Sex, asec),  
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),  
      error_format = "{p.value}",  
      error_pos = "below",  
      statistics = c(N = "nobs", R2 = "r.squared"))  
  
#  
#  
#  
#  
#  
#  
#  
#  
#  
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below",
      statistics = c(N = "nobs", R2 = "r.squared"),
      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
      note = "Significance: {stars}")
#
#
#
#
#
#
#
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below",
      statistics = c(N = "nobs", R2 = "r.squared"),
      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
      note = "Significance: {stars}",
      align = "c")
#
#
#
#
#
#
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below",
      statistics = c(N = "nobs", R2 = "r.squared"),
      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
      note = "Significance: {stars}",
      align = "c",
      coefs = c("Hours worked" = "Hours",
                "(Hours worked)2" = "sqHours",
                "Male" = "SexMale",
                "Hours worked x Male" = "Hours:SexMale",
                "Constant" = "(Intercept)"))
#
#
```



# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below",
      statistics = c(N = "nobs", R2 = "r.squared"),
      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
      note = "Significance: {stars}",
      align = "c",
      coefs = c("Hours worked" = "Hours",
                "(Hours worked)2" = "sqHours",
                "Male" = "SexMale",
                "Hours worked x Male" = "Hours:SexMale",
                "Constant" = "(Intercept)")) %>%
insert_row(c("", rep("Dependent variable: Log annual earnings", 4)), after = 0)
#
```

# Solution

Use the functions `huxreg()`, `insert_row()`, and `merge_cells()` to reproduce the table

```
huxreg(lm(lEarnings ~ Hours, asec),
      lm(lEarnings ~ Hours + sqHours, asec),
      lm(lEarnings ~ Hours + sqHours + Sex, asec),
      lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),
      error_format = "{p.value}",
      error_pos = "below",
      statistics = c(N = "nobs", R2 = "r.squared"),
      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
      note = "Significance: {stars}",
      align = "c",
      coefs = c("Hours worked" = "Hours",
                "(Hours worked)2" = "sqHours",
                "Male" = "SexMale",
                "Hours worked x Male" = "Hours:SexMale",
                "Constant" = "(Intercept)")) %>%
insert_row(c("", rep("Dependent variable: Log annual earnings", 4)), after = 0) %>%
merge_cells(1, 2:5)
```

# 4. Report and export results

## 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - 
  - 
  - 
  - 
  -

```
plot_summs()
#
#
#
#
#
```

## 4. Report and export results

### 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - First feed it with your **models**
  - 
  - 
  - 
  -

```
plot_summs(lm(lEarnings ~ Hours + Race + Sex, asec),
           lm(lEarnings ~ Hours + Race + Sex + sqHours, asec))
```

```
#
#
#
#
```

## 4. Report and export results

### 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - First feed it with your **models**
  - You can choose to **omit** some **coefficients**
  - 
  - 
  -

```
plot_summs(lm(lEarnings ~ Hours + Race + Sex, asec),
           lm(lEarnings ~ Hours + Race + Sex + sqHours, asec),
           omit.coefs = "(Intercept)")
```

```
#
#
#
```

## 4. Report and export results

### 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - First feed it with your **models**
  - You can choose to **omit** some **coefficients**
  - Change the **level** of the **confidence** intervals
  - 
  -

```
plot_summs(lm(lEarnings ~ Hours + Race + Sex, asec),
           lm(lEarnings ~ Hours + Race + Sex + sqHours, asec),
           omit.coefs = "(Intercept)",
           ci_level = 0.99)
```

```
#
```

```
#
```

## 4. Report and export results

### 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - First feed it with your **models**
  - You can choose to **omit** some **coefficients**
  - Change the **level** of the **confidence** intervals
  - Custom the **color palette**
  -

```
plot_summs(lm(lEarnings ~ Hours + Race + Sex, asec),
           lm(lEarnings ~ Hours + Race + Sex + sqHours, asec),
           omit.coefs = "(Intercept)",
           ci_level = 0.99,
           colors = c("#014D64", "#00A2D9"))
```

```
#
```

## 4. Report and export results

### 4.2. Plot coefficients

- It can also be useful to provide a **graphical representation** of the coefficients
  - By now you should be able to work it around with **dplyr** and **ggplot**
  - But there exists a **shortcut**
- The **plot\_summs()** function from the **jtools** package takes regression models as inputs and **plots the results**
  - First feed it with your **models**
  - You can choose to **omit** some **coefficients**
  - Change the **level** of the **confidence** intervals
  - Custom the **color palette**
  - And add ggplot functions!

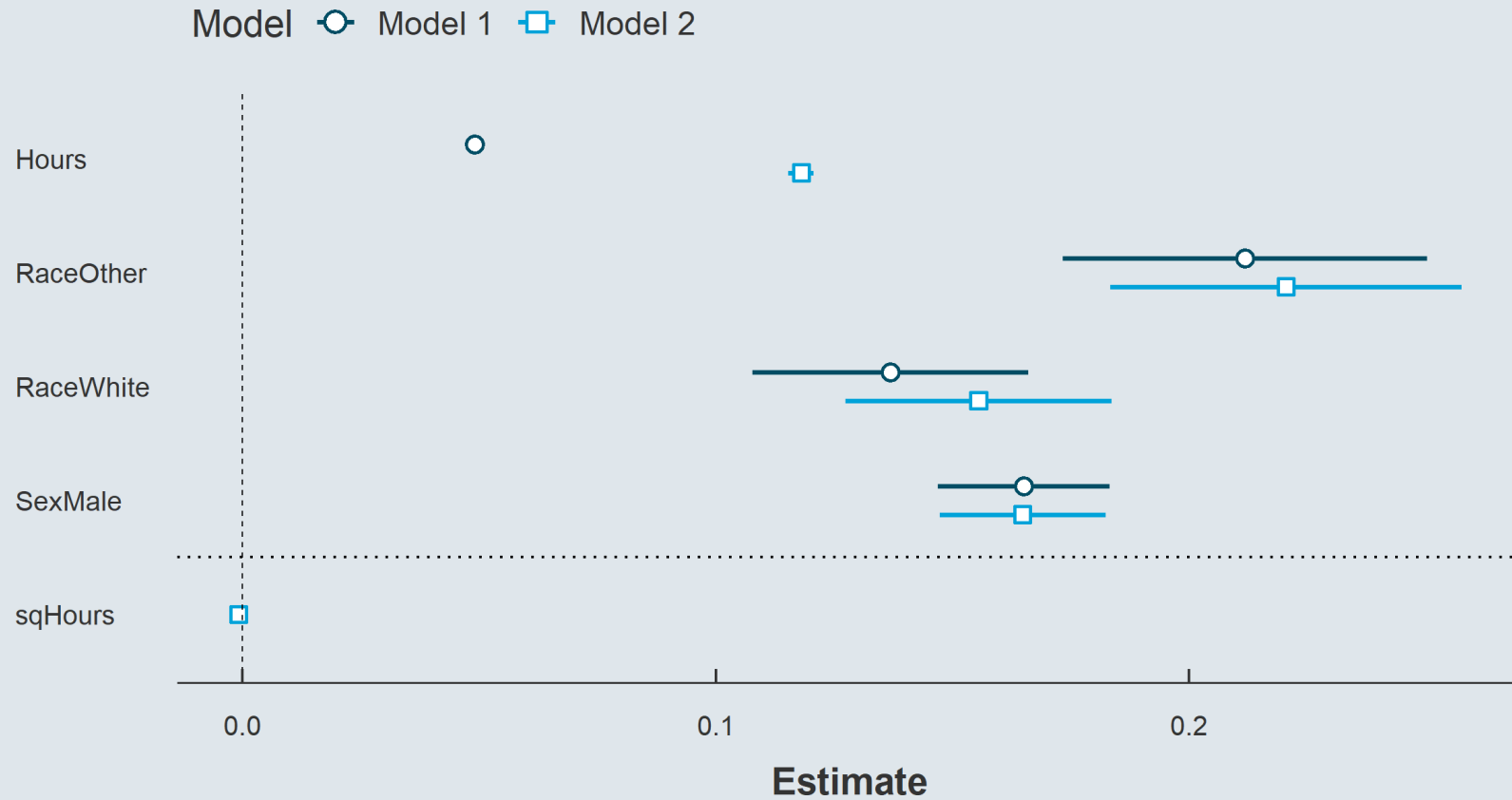
```
plot_summs(lm(lEarnings ~ Hours + Race + Sex, asec),
           lm(lEarnings ~ Hours + Race + Sex + sqHours, asec),
           omit.coefs = "(Intercept)",
           ci_level = 0.99,
           colors = c("#014D64", "#00A2D9")) +
  geom_hline(yintercept = 1.5, linetype = "dotted")
```





# 4. Report and export results

## 4.2. Plot coefficients





# Overview

## 1. Regressions ✓

- 1.1. On continuous variables
- 1.2. On binary variables
- 1.3. On categorical variables

## 2. Case study ✓

- 2.1. Variable transformation
- 2.2. Functional form
- 2.3. Control variables
- 2.4. Interactions

## 3. Inference ✓

- 3.1. Hypothesis testing
- 3.2. Confidence intervals

## 4. Report and export results ✓

- 4.1. Regression tables
- 4.2. Plot coefficients

## 5. Wrap up!



# 5. Wrap up!

## Regressions in R

```
summary(lm(formula = ige ~ gini, data = ggcurve))
```

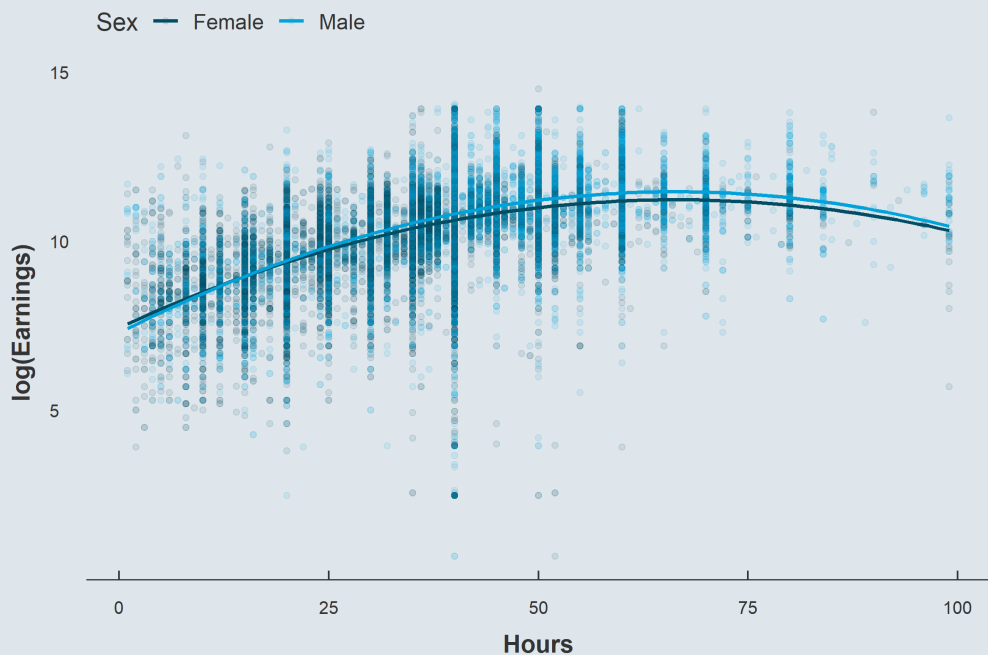
```
##  
## Call:  
## lm(formula = ige ~ gini, data = ggcurve) ← Command  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -0.188991 -0.088238 -0.000855  0.047284  0.252310 ← Residuals distribution  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -0.09129    0.12870  -0.709  0.48631 ← Coefs, s.e., t-/p-values  
## gini         1.01546    0.26425   3.843  0.00102 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.1159 on 20 degrees of freedom ← Residual s.e. & df.  
## Multiple R-squared:  0.4247,    Adjusted R-squared:  0.396 ← R2 & adjusted R2  
## F-statistic: 14.77 on 1 and 20 DF,  p-value: 0.001016 ← F-test results
```



# 5. Wrap up!

## Variable transformations, functional forms, controls, interactions

$$\log(\text{Earnings}_i) = \alpha + \beta_1 \text{Hours}_i + \beta_2 \text{Hours}_i^2 + \beta_3 \text{Male}_i + \beta_4 \text{Hours}_i \times \text{Male}_i + \varepsilon_i$$



```
summary(  
  lm(lEarnings ~ Hours +  
    sqHours +  
    Sex +  
    Hours * Sex,  
    asec)  
)$coefficients[, 1:2]
```

| ##               | Estimate      | Std. Error   |
|------------------|---------------|--------------|
| ## (Intercept)   | 7.3900468002  | 2.319198e-02 |
| ## Hours         | 0.1174998826  | 1.034522e-03 |
| ## sqHours       | -0.0009103523 | 1.321706e-05 |
| ## SexMale       | -0.0282905673 | 2.753222e-02 |
| ## Hours:SexMale | 0.0050321134  | 6.767013e-04 |



## 5. Wrap up!

### Hypothesis testing

```
linearHypothesis(lm(lEarnings ~ Hours + sqHours + Sex + Hours * Sex, asec),  
                 c("Hours = 0", "sqHours = 0", "SexMale = 0", "Hours:SexMale = 0"))
```

```
## Linear hypothesis test  
##  
## Hypothesis:  
## Hours = 0  
## sqHours = 0  
## SexMale = 0  
## Hours:SexMale = 0  
##  
## Model 1: restricted model  
## Model 2: lEarnings ~ Hours + sqHours + Sex + Hours * Sex  
##  
##   Res.Df  RSS Df Sum of Sq    F    Pr(>F)  
## 1   64335 68395  
## 2   64331 46098  4    22297 7779.1 < 2.2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



# 5. Wrap up!

## Report/export results

```

huxreg(
  Baseline = lm(lEarnings ~ Hours, asec),
  lm(lEarnings ~ Hours + sqHours, asec),
  lm(lEarnings ~ Hours + sqHours + Sex, asec),
  lm(lEarnings ~ Hours + sqHours + Sex +
      Hours * Sex, asec),
  error_format = "{std.error}",
  error_pos = "below",
  statistics = c(N="nobs", R2="r.squared"),
  stars = c(`***` = 0.01,
            `**` = 0.05,
            `*` = 0.1),
  note = paste("Dependent variable: log",
              "annual earnings. {stars}")
)

```

|               | Baseline             | (2)                   | (3)                   | (4)                   |
|---------------|----------------------|-----------------------|-----------------------|-----------------------|
| (Intercept)   | 8.604 ***<br>(0.014) | 7.364 ***<br>(0.022)  | 7.336 ***<br>(0.022)  | 7.390 ***<br>(0.023)  |
| Hours         | 0.051 ***<br>(0.000) | 0.119 ***<br>(0.001)  | 0.118 ***<br>(0.001)  | 0.117 ***<br>(0.001)  |
| sqHours       |                      | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) | -0.001 ***<br>(0.000) |
| SexMale       |                      |                       | 0.170 ***<br>(0.007)  | -0.028<br>(0.028)     |
| Hours:SexMale |                      |                       |                       | 0.005 ***<br>(0.001)  |
| N             | 64336                | 64336                 | 64336                 | 64336                 |
| R2            | 0.268                | 0.319                 | 0.325                 | 0.326                 |

Dependent variable: log annual earnings. \*\*\* p < 0.01; \*\* p < 0.05; \* p < 0.1