Econometria Financeira com o R – OLS e GLM

Análise de Dados Financeiros e Econômicos com o R

Marcelo S. Perlin

EA-UFRGS

12/11/2026

Sumário

- Introduction
- Linear Models
- Statistical Inference in Linear Models
- Generalized Linear Models (GLM)

Introduction

Why Financial Econometrics?

> The modelling tools from financial econometrics allow the researcher to simulate, predict and test a particular hypothesis about the data.

- allow **correctly** testing different hypothesis about the real world
- models scale well! (likely to be usable for large data)
- can be very flexible towards a functional form of the data generating process

Most popular models:

- **Linear models (OLS)**
- **Generalized linear models (GLM)**
- ~~**Panel data models (PDM)**~~
- **Arima models (Integrated Autoregressive Moving Averages)**
- **Garch models (generalized autoregressive conditional heteroskedasticity)**
- ~~**Regime switching models**~~

Please read the references!

- Introductory econometrics for finance (Brooks 2014)
- Econometric analysis (Greene 2003)
- The econometrics of financial markets (Campbell et al. 1997)
- Time series analysis (Hamilton 1994)

Linear Models

Linear Models

> A linear model is, without a doubt, one of the most used econometric models in R and finance.

Why??

- simplicity
- quick estimation, even for HUGE amounts of data
- easy to interpret and process out of sample data

A linear model with *N* explanatory variables can be represented as follows:

$$y_t = \alpha + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \ldots + \beta_N x_{N,t} + \epsilon_t$$

y_t – dependent (or response) variable

$x_{i,t}$ – independent (or explanatory) variables.

β_i – coefficients to be estimated

ϵ_t – Residual from linear equation

Simulating a Linear Model

Consider the following equation:

$$y_t = 0.5 + 2x_t + \epsilon_t$$

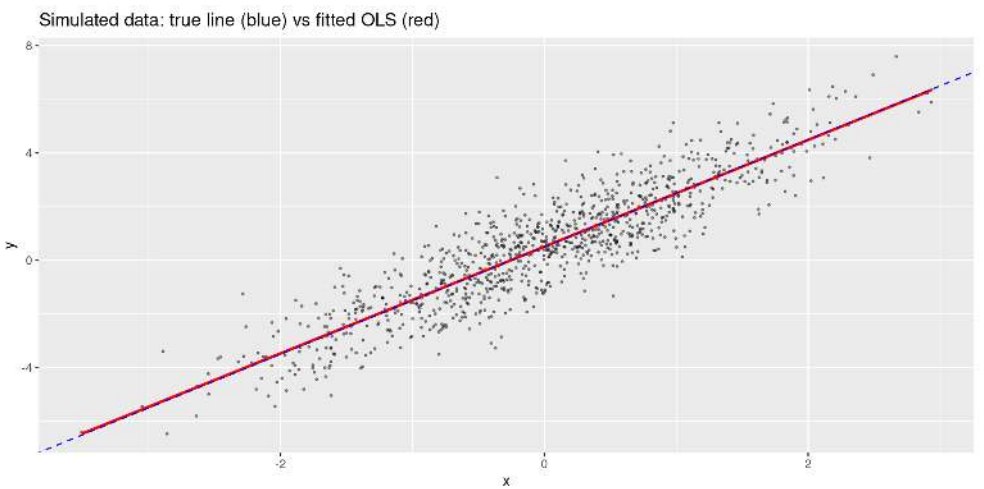
```r
set.seed(50)

# number of obs
nT <- 1000

# set x as Normal (0, 1)
x <- rnorm(nT)

# set coefficients
my_alpha <- 0.5
my_beta <- 2

# build y
```



Estimating a Linear Model

In R, the main function for estimating a linear model is `lm`. Let's use it to estimate a model from the previous simulated data.

```r
# set df
df_reg <- dplyr::tibble(x

# estimate linear model
my_lm <- lm(data = df_reg

print(my_lm)
```

```
Call:
lm(formula = y ~ x, data =
df_reg)

Coefficients:
(Intercept)            x
     0.5083       1.9891
```

```r
library(broom)

# tidy table of coefficie
tidy(my_lm, conf.int = TR
```

```
# A tibble: 2 × 7
  term        estimate
std.error statistic  p.value
conf.low conf.high
  <chr>          <dbl>
```

```
<dbl>     <dbl>    <dbl>
<dbl>     <dbl>
1 (Intercept)    0.508
0.0311      16.4 1.74e-53
0.447     0.569
```

more information about the previously estimated can be found with `summary`:

```
1 print(summary(my_lm))
```

```
Call:
lm(formula = y ~ x, data = df_reg)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0444 -0.6906 -0.0244  0.6807  3.2892

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.50820    0.03107   16.36   <2e-16 ***
```

Reporting a single Linear Model

The typical model table can be constructed in R with `texreg` or `modelsummary`:

```
1 library(texreg)
2
3 htmlreg(my_lm,
4          include.rownames
5          caption = "Simula
```

	Model 1
(Intercept)	0.51***
	(0.03)
x	1.99***
	(0.03)
R^2	0.81
Adj. R^2	0.81
Num. obs.	1000
***p < 0.001; **p < 0.01; *p < 0.05	

Simulated linear model

Reporting many models

```
 1 library(texreg)
 2
 3 # nested models: does the
 4 model_1 <- lm(data = df_r
 5 model_2 <- lm(data = df_r
 6 model_3 <- lm(data = df_r
 7
 8 htmlreg(l = list(model_1,
 9          custom.model.name
10          include.rownames
```

	Linear	Quadratic	Cubic
(Intercept)	0.51^{***}	0.50^{***}	0.50^{***}
	(0.03)	(0.04)	(0.04)
x	1.99^{***}	1.99^{***}	1.97^{***}
	(0.03)	(0.03)	(0.05)
x^2		0.01	0.01
		(0.02)	(0.02)
x^3			0.01
			(0.01)
R^2	0.81	0.81	0.81
Adj. R^2	0.81	0.81	0.81

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

	Linear	Quadratic	Cubic
Num. obs.	1000	1000	1000

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

Statistical models

A real world example

> The table is from published paper titled "The Academic Inbreeding Controversy: Analysis and Evidence from Brazil", published in Journal of Informetrics

Table 3: Estimation Results for Productivity (Eq. 1)

	Dependent variable:						
	NPublic	JIF	SJR	PercA1A2	NBooks	NSupervisions	hindexGS
	Poisson	*Poisson*	*Poisson*	*logistic*	*Poisson*	*Poisson*	*Poisson*
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Male	0.207***	−0.034***	0.019	0.047	0.249***	0.203***	0.118***
PhD Abroad	−0.001	0.015	0.176***	0.068	0.068	0.201***	0.087***
Pos-Doc	0.353***	0.507***	0.361***	0.618***	0.155***	0.284***	0.109***
Heritage (% A1 A2)	−0.141**	4.235***	3.857***	10.356***	−2.081***	−0.419***	0.587***
Heritage (# papers)	0.082***	−0.006***	−0.011***	−0.011**	0.027***	0.056***	0.028***
Inbred	−0.008	0.052***	0.049***	0.011	−0.055	0.050**	0.004
Pure Inbred	0.029**	0.205***	0.151***	0.275***	−0.022	0.137***	0.061***
Mobile Inbred	0.349***	0.457***	0.355***	0.607***	−0.152	0.462***	0.216***
Silver Corded	0.296***	0.374***	0.284***	0.477***	0.098	0.230***	0.122***
Constant	−1.274***	−1.580***	−1.627***	−4.255***	−3.870***	−1.705***	2.117***
Type of Model	GLM-Poisson	GLM-Poisson	GLM-Poisson	GLM-Binomial	GLM-Poisson	GLM-Poisson	GLM-Poisson
Observations	76,521	76,521	76,521	76,521	76,521	76,521	15,495

Another real world example

The table is from a **working paper** titled "The Academia Downturn: the Impact of COVID-19 in the Brazilian Academic Output"

	Publications	Supervisions	Projects	Conferences
Control				
COVID	0.89 (0.07)***	0.11 (0.05)*	−0.02 (0.02)	−0.00 (0.03)
Gender (Female)	−0.30 (0.08)***	0.25 (0.07)***	0.06 (0.01)***	−0.03 (0.02)
PQ Scholarship	3.67 (0.52)***	2.32 (0.32)***	0.44 (0.04)***	0.40 (0.08)***
High Experience	0.55 (0.09)***	−0.17 (0.07)*	−0.14 (0.02)***	0.20 (0.06)***
Hypothesis				
Gender (female) X COVID	−0.08 (0.03)*	0.05 (0.06)	0.02 (0.01)**	0.06 (0.01)***
PQ Scholarship X COVID	0.24 (0.09)*	−1.12 (0.14)***	−0.20 (0.02)***	−0.09 (0.02)***
High Experience X COVID	−0.44 (0.05)***	−0.30 (0.07)***	−0.02 (0.01)	−0.16 (0.04)***
Year Effects	No	No	No	No
Field of Science Effects	Yes	Yes	Yes	Yes
Clustered Standard Errors	Yes	Yes	Yes	Yes
R^2	0.19	0.03	0.02	0.02
Adj. R^2	0.19	0.03	0.02	0.02
Num. obs.	1457349	1457349	1457349	1457349

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

Table 2 Estimation Results for DiD Model

Statistical Inference in Linear Models

> After estimating a model with function `lm`, the next step is to test some hypothesis about the coefficients.

- F-test: verifies the most basic condition for a model to justify its existence – it tests the assumption that all coefficients, excluding the intercept, are equal to zero.
- T-tests: tests the null hypothesis that individual coefficients (betas) are equal to zero.
- Adjusted R^2: R^2 penalized by the number of parameters, so it does not automatically increase when irrelevant variables are added.

The F-test in `summary()`

When function `summary` is applied to a `lm` output, the F-test is provided by default in the last line of the text output. The null hypothesis of the test is that all slopes are equal to zero. Let's try it:

```
nT <- 100
df <- data.frame(y = runif(nT),
                 x1 = runif(nT),
                 x2 = runif(nT))

my_lm <- lm(data = df,
            formula = y ~ x1 + x2)

print(summary(my_lm))
```

```
Call:
lm(formula = y ~ x1 + x2, data = df)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.51446 -0.26408 -0.01434  0.27508  0.46685

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

Alternative hypothesis in linear regressions

> While T-test calculate the p-values associated with the event of the individual coefficient being equal to zero, we can also design other tests using a Wald Test, such as a particular parameter being equal to a value

As a simple example, let's test a linear hypothesis for a simulated model. Here, we will create artificial data and test the formal hypothesis that the estimated coefficients are equal to the actual values provided in the simulation.

```
1 set.seed(10)
2
3 # number of time periods
4 nT <- 1000
5
6 # set parameters
7 intercept <- 0.5
8 beta <- 1.5
9
```

```
# simulate
x <- rnorm(nT)
y <- intercept + beta*x + rnorm(nT)

```

- After the estimation of the model, we use function `LinearHypothesis` from package `car` (Fox and Weisberg 2019) to implement our formal test.
- The tested hypotheses are represented using a matrix operation. With this matrix operation, we test the joint hypothesis that the intercept is equal to 0.5 and the slope is equivalent to 1.5.

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{hypothesis.matrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \underbrace{\begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}}_{rhs}$$

The actual R code that implements the test is given next.

```
library(car)

# set test matrix
```

```r
test_matrix <- matrix(c(intercept,  # alpha test value
                        beta))  # beta test value

# hypothesis matrix
hyp_mat <- matrix(c(1,0,
                    0,1),nrow = 2)

# do test
my_waldtest <- linearHypothesis(my_lm,
                                hypothesis.matrix = hy
```

```
Linear hypothesis test:
(Intercept) = 0.5
x = 1.5

Model 1: restricted model
Model 2: y ~ x

  Res.Df    RSS Df Sum of Sq      F Pr(>F)
```

1 1000 1089 1

As we can see, the test fails to reject the null hypothesis. This means our simulation worked, and the parameters are correctly estimated as expected.

Testing OLS assumptions

> Another family of tests commonly applied to linear models is related to its assumptions. Every linear model estimated by OLS assumes:

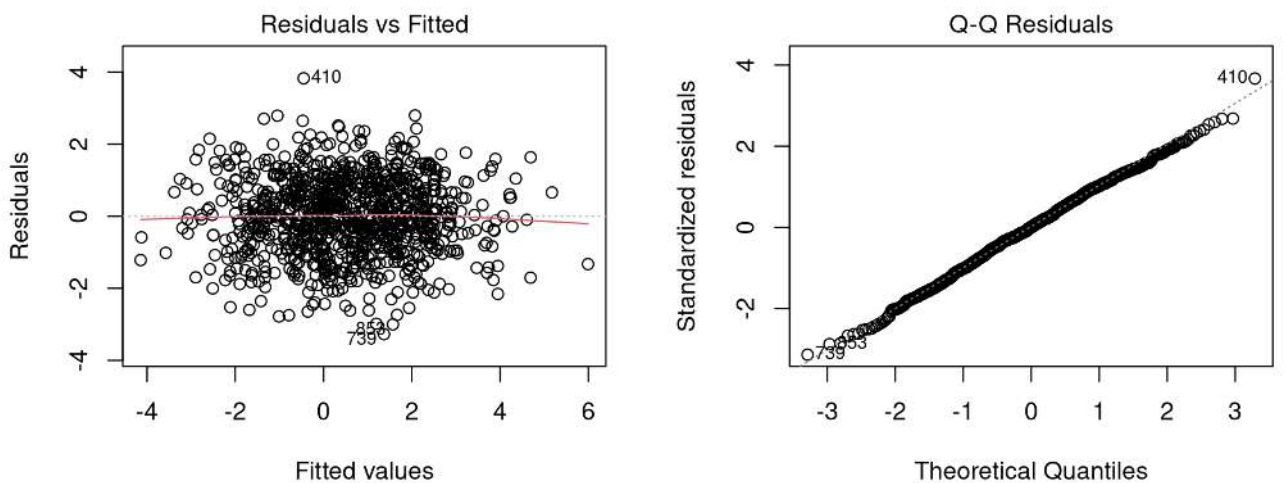
1. linearity in the parameters,
2. independence of the errors,
3. homoscedasticity (constant variance),
4. no perfect multicollinearity,
5. errors approximately Normal (mainly for small-sample inference).

- If these assumptions are not true, the model may be inefficient or biased, meaning some modification or use of robust estimates is required.

Visual diagnostics

> Formal tests are useful, but plots often reveal the problem faster.

```
par(mfrow = c(1, 2))
plot(my_lm, which = 1:2)
```



- **Residuals vs fitted**: a funnel shape suggests heteroscedasticity.
- **Normal Q-Q**: points far from the straight line suggest non-Normal errors.

R packages for testing OLS assumptions

- package `lmtest` (Hothorn et al. 2022) tests for independence with the Breusch-Godfrey and Durbin-Watson tests.
- The Shapiro-Wilk test for normality is available in package `stats`.

```
library(lmtest)

# Breusch-Godfrey test - Serial correlation
# Null hypothesis: no serial correlation in residuals
print(bgtest(my_lm, order = 5))

# Breusch-Pagan test - Non-constant variance
# Null hypothesis: homoscedasticity
print(ncvTest(my_lm))

# Durbin-Watson test - Serial correlation
# Null hypothesis: no serial correlation in residuals
```

```
print(dwtest(my_lm))
```

```
    Breusch-Godfrey test for serial correlation of order
up to 5

data:  my_lm
LM test = 4.2628, df = 5, p-value = 0.5122

Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 1.54328, Df = 1, p = 0.21413
```

As expected, the model with artificial data passed all tests.

`dwtest()` assumes the residuals are in time order. Here the data is i.i.d., so treat this result as illustrative; we will use the test properly with time series.

Robust standard errors

> If heteroscedasticity is present, the OLS point estimates are still unbiased, but the classical standard errors are wrong. A simple fix is to use heteroscedasticity-robust standard errors.

```
 1 set.seed(123)
 2
 3 n <- 500
 4 x <- runif(n, 1, 10)
 5 y <- 1 + 2*x + rnorm(n, sd = x)   # variance grows wit
 6 df_het <- tibble::tibble(x, y)
 7 lm_het <- lm(y ~ x, data = df_het)
 8
 9 library(sandwich)
10 library(lmtest)
11
```

```
12 # classical standard errors
```

```
t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.71554    0.63851  1.1206    0.263
x            2.07969    0.10594 19.6303   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
```

```
1 # heteroscedasticity-robust standard errors (HC1)
2 print(coeftest(lm_het, vcov = vcovHC(lm_het, type = "H
```

```
t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.71554    0.49563  1.4437   0.1494
x            2.07969    0.11804 17.6180   <2e-16 ***
---
```

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

- The coefficients are identical; only the standard errors, t-statistics and p-values change.
- The robust version is usually the safer default when the variance of the errors is not constant.

Global validation

> Package `gvlma` (Pena and Slate 2019) provides a top level function that can execute all sorts of tests in linear models, including the ones described before. The main advantage is that it outputs all tests in a single function call.

```
1 library(gvlma)
2
3 # global validation of model
4 gvmodel <- gvlma(my_lm)
5
6 # print result
7 summary(gvmodel)
```

```
Call:
lm(formula = y ~ x, data = df)

Residuals:
```

```
    Min      1Q  Median      3Q     Max
-3.2703 -0.6898  0.0063  0.7346  3.8266

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

Generalized Linear Models (GLM)

Introduction

> The generalized linear model (GLM) is a flexible alternative to a linear model. It allows the user to change the distribution of the response and the link function, which specifies how the expected response relates to the linear predictor.

- GLM is the right choice when the response is not continuous and unbounded: binary (`binomial`), counts (`poisson`), or positive continuous (`Gamma`).
- Applications of GLM in finance are mostly in credit models, which are used to predict the likelihood of an individual or company to default on its debt.

Specification

We can write a general univariate GLM as:

$$E\left(y_t\right) = g\left(\alpha + \sum_{i=1}^{N} \beta_i x_{i,t}\right)$$

Function *g()* can take many shapes. For example, if we are modelling a binary variable, we can use *g()* as the *logit* function:

$$g(x) = \frac{\exp(x)}{1 + \exp(x)}$$

- function *g()* ensures any value of *x* will result in a number between 0 and 1.

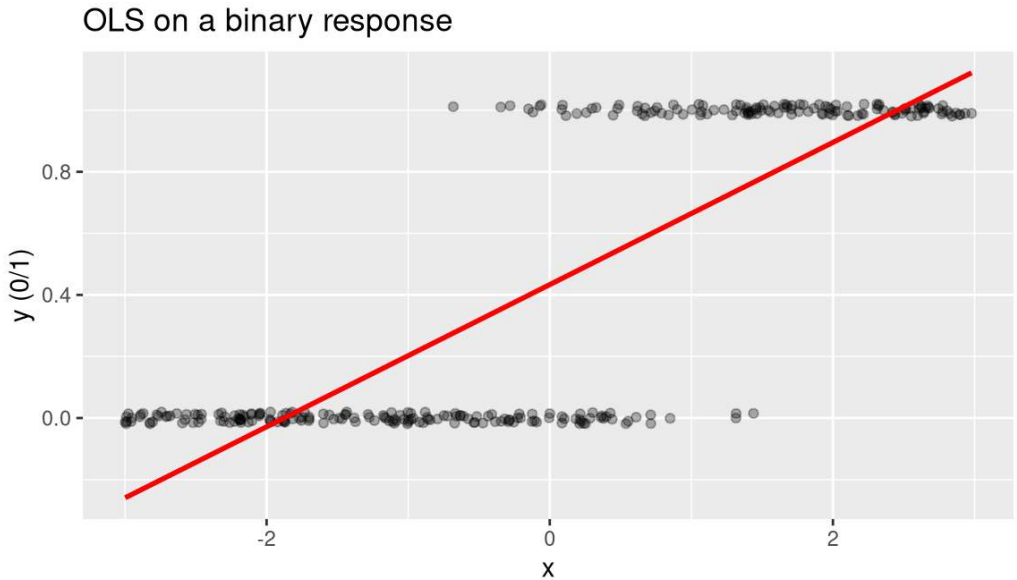
Why not just use OLS?

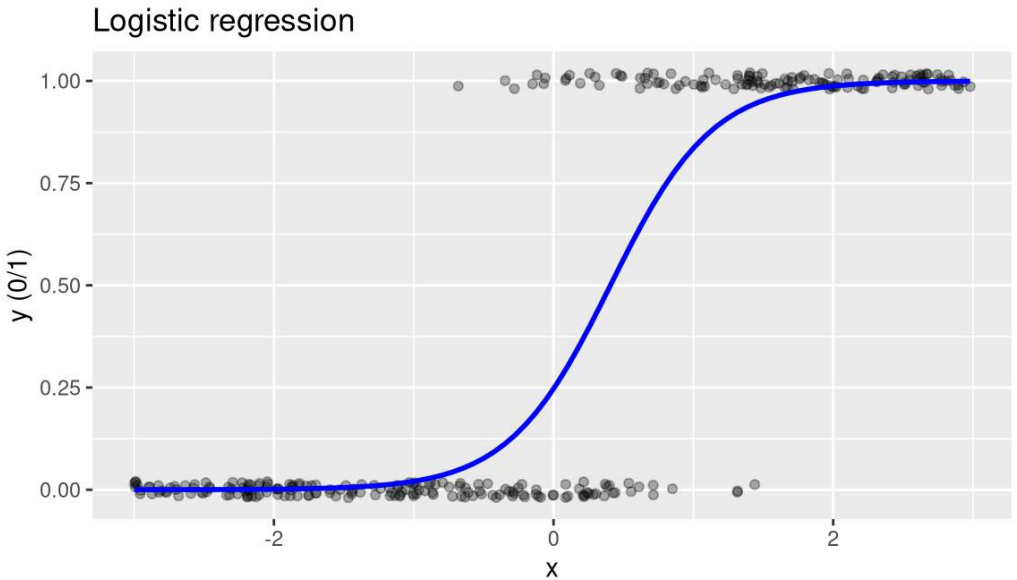
> With a binary response, OLS can produce predicted values below 0 or above 1, and the constant-variance assumption is violated.

```r
set.seed(42)
n <- 300
x <- runif(n, -3, 3)
p <- plogis(-1 + 3*x)
y <- rbinom(n, 1, p)
df_bin <- tibble::tibble(x, y)

p_ols <- ggplot(df_bin, aes(x, y)) +
  geom_jitter(height = 0.02, alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "red"
  labs(title = "OLS on a binary response", y = "y (0/1

p_glm <- ggplot(df_bin, aes(x, y)) +
```





- The OLS line is a straight line and can leave the [0, 1] interval.
- The logistic curve respects the natural bounds of a probability.

Simulating a GLM Model

As an example, let's simulate the following GLM model, where the response vector is a Bernoulli variable that takes value 1 with probability p_t. The probabilities are calculated from the non-linear transformation of x_t.

$$p_t = \frac{\exp(2 + 5x_t)}{1 + \exp(2 + 5x_t)}$$

In R, we use the following code to build the response vector.

```r
set.seed(15)

# set number of obs
n_T <- 500

# set x
x <- rnorm(n_T)

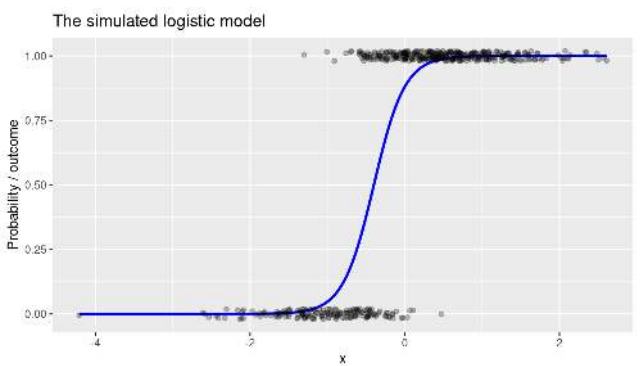
my_alpha <- 2
```

```
10 my_beta <- 5
11
12 # set probabilities
13 z <- my_alpha + my_beta*x
```

The simulated data

```r
sim_df <- tibble::tibble(x, p, y)

ggplot(sim_df, aes(x = x, y = p)) +
  geom_line(color = "blue", linewidth = 1) +
  geom_jitter(aes(y = y), height = 0.02, alpha = 0.25) +
  labs(x = "x", y = "Probability / outcome",
       title = "The simulated logistic model")
```



- The blue curve is the true probability $p_t = \exp(2 + 5x_t)/(1 + \exp(2 + 5x_t))$.
- The points are the realized binary outcomes $y_t \in \{0, 1\}$.

Estimating a GLM Model

- In R, the estimation of GLM models is accomplished with function **stats::glm()**.

Let's use the previously simulated data to estimate a *logit* model:

```
# estimate GLM
df <- tibble::tibble(x, y)
my_family <- binomial(link = "logit")
my_glm <- glm(data = df,
              formula = y ~ x ,
              family = my_family)

# print it with summary
print(summary(my_glm))
```

```
Call:
glm(formula = y ~ x, family = my_family, data = df)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.1488     0.2622   8.197 2.47e-16 ***
x             4.9050     0.5110   9.598  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
```

An example with data

> Let's use a credit card dataset from Kaggle[1] to model credit risk.

```
# read default data
my_f <- afedR3::data_path('CH11_UCI-Credit-Card.csv')
df_default <- readr::read_csv(my_f)

dplyr::glimpse(df_default)
```

```
Rows: 30,000
Columns: 25
$ ID                                  <dbl> 1, 2, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, …
$ LIMIT_BAL                           <dbl> 20000, 120000, 90000,
50000, 50000, 50000, …
$ SEX                                 <dbl> 2, 2, 2, 2, 1, 1, 1,
2, 2, 1, 2, 2, 2, 1, 1…
$ EDUCATION                           <dbl> 2, 2, 2, 2, 2, 1, 1,
```

```
2, 3, 3, 3, 1, 2, 2, 1…
```

For now, let's simplify the analysis and focus on four columns: sex, education, marriage and age. For that, we'll also need to decode the values in some columns.

```r
library(dplyr)

# read credit card data
# source:
# www.kaggle.com/uciml/default-of-credit-card-clients-
# COLUMNS: GENDER: (1 = male; 2 = female).
#          EDUCATION: 0 = unknown; 1 = graduate school
#                     2 = university;
#                     3 = high school;
#                     4 = others;
#                     5, 6 = unknown.
#          MARRIAGE: 1 = married;
#                    2 = single;
```

```
Rows: 30,000
Columns: 5
```

```
$ default        <lgl> TRUE, TRUE, FALSE, FALSE, FALSE,
FALSE, FALSE, FALSE, FA…
$ D_male_gender <lgl> FALSE, FALSE, FALSE, FALSE, TRUE,
TRUE, TRUE, FALSE, FAL…
$ age            <dbl> 24, 26, 34, 37, 57, 37, 29, 23, 28,
35, 34, 51, 41, 30, …
$ educ           <chr> "University", "University",
"University", "University", …
```

> **Data caveat:** 14 observations have `EDUCATION = 0` and none of them defaulted. If left as a separate level, they become the reference category and cause complete separation (huge coefficients and standard errors). We recode `0` to `Unknown`.

Much better! Now we only have the columns of interest, with more appealing names and content. Let's proceed to the estimation of the glm model.

```r
# estimate glm model
glm_credit <- glm(data=df_default,
                  formula = default ~ D_male_gender +  age +
                                      educ + D_marriage,
                  family = binomial(link = "logit"))
```

```
6
7 # show output
8 summary(glm_credit)
```

```
Call:
glm(formula = default ~ D_male_gender + age + educ + D_marriage,
    family = binomial(link = "logit"), data = df_default)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.487666   0.060838 -24.453  < 2e-16 ***
D_male_genderTRUE  0.207892   0.028503   7.294 3.01e-13 ***
age               -0.002417   0.001736  -1.393 0.163731
educHigh School    0.331518   0.042230   7.850 4.15e-15 ***
educOthers        -1.370155   0.390129  -3.512 0.000445 ***
educUniversity     0.257734   0.031936   8.070 7.01e-16 ***
```

1. https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

Odds ratios

> The coefficients of a logistic model are on the **log-odds** scale. Exponentiating them gives **odds ratios**, which are easier to interpret.

```
1 library(broom)
2
3 tidy(glm_credit, exponentiate = TRUE, conf.int = TRUE) |>
4   dplyr::select(term, estimate, conf.low, conf.high, p.value) |>
5   dplyr::mutate(dplyr::across(where(is.numeric), \(x) round(x, 3)))
```

```
# A tibble: 8 × 5
  term                estimate conf.low conf.high p.value
  <chr>                  <dbl>    <dbl>     <dbl>   <dbl>
1 (Intercept)            0.226    0.2       0.254   0
2 D_male_genderTRUE      1.23     1.16      1.30    0
3 age                    0.998    0.994     1.00    0.164
4 educHigh School        1.39     1.28      1.51    0
5 educOthers             0.254    0.107     0.506   0
6 educUniversity         1.29     1.22      1.38    0
7 educUnknown            0.339    0.221     0.497   0
8 D_marriageTRUE         1.14     1.07      1.21    0
```

- An odds ratio above 1 means higher odds of default; below 1 means lower odds.
- For example, the odds ratio for `D_male_gender` is approximately 1.23.

Goodness of fit

> The R^2 from OLS does not apply to a binary response. Useful alternatives are the McFadden pseudo-R^2, the AIC and the AUC.

```
# McFadden pseudo-R2
pseudo_r2 <- 1 - as.numeric(logLik(glm_credit) / logLi
print(pseudo_r2)
```

```
[1] 0.007726326
```

```
# AIC (lower is better)
print(AIC(glm_credit))
```

```
[1] 31476.39
```

- McFadden pseudo-R^2 is usually much smaller than the OLS R^2; values between 0.2 and 0.4 already indicate a very good fit for a binary model.
- Here it is close to zero, consistent with the low AUC: demographics alone explain little about default.
- AIC is useful when comparing models estimated on the same data.

From probabilities to decisions

> The model outputs a probability, not a yes/no answer. To classify a client, we must choose a threshold.

```
# predicted probability of default
df_default$p_default <- predict(glm_credit, type = "re

# classify using a 0.5 threshold
df_default$pred_default <- df_default$p_default > 0.5

# confusion matrix
pred_label <- factor(df_default$pred_default,
                     levels = c(TRUE, FALSE),
                     labels = c("Default", "No default
actual_label <- factor(df_default$default,
                       levels = c(TRUE, FALSE),
                       labels = c("Default", "No defau
```

	actual	
predicted	Default	No default
Default	0	0
No default	6636	23364

- At the 0.5 threshold, this model classifies everyone as “no default”.
- The default rate is about 22.1%, so a model that always predicts “no default” would already be right most of the time.
- Accuracy alone is therefore misleading; we need measures such as sensitivity, specificity and AUC.

ROC and AUC

```
library(pROC)

my_roc <- pROC::roc(response = as.numeric(df_default$d
                    predictor = df_default$p_default,
                    quiet = TRUE)

print(pROC::auc(my_roc))
```

```
Area under the curve: 0.5583
```

```
plot(my_roc, print.auc = TRUE, main = "ROC curve - cre
```

- The ROC curve shows the trade-off between sensitivity and specificity for every possible threshold.
- AUC = 0.5 is a useless model; AUC = 1 is perfect. Here, AUC is about 0.558.
- This low AUC is expected: demographic variables alone are weak predictors. Real credit models use payment history, balances and other behavioural variables.

Exercise 1 - Linear Model

1. Using package `yfR`, download adjusted price data for `^BVSP` and `GRND3.SA` from 2015 to 2023.
2. Estimate the beta of GRND3 from the following model:

$$R_t = \alpha + \beta R_{M,t} + \epsilon_t$$

3. Estimate a new market model with a dummy for negative returns:

$$R_t = \alpha + \beta_1 R_{M,t} + \beta_2 D_t R_{M,t} + \epsilon_t$$

Where $D_t = 1$ if $R_{M,t} < 0$ and $D_t = 0$ otherwise

4. With the first model (without dummies), use package `car` to test the hypothesis that $\alpha = 0$ and $\beta = 1$
5. Test the underlying assumptions of the previous linear model with package `gvlma`

Exercise 2 - Linear Models

1. Using package `yfR`, download adjusted price data for PETR3, PETR4, GGBR3, GGBR4 from 2010 to 2018.
2. Using loops, estimate and save the beta of all stocks in a dataframe with two columns (ticker, beta):

$$R_t = \alpha + \beta R_{M,t} + \epsilon_t$$

3. Report the results of all models with `texreg::screenreg` in a single table

Exercise 3 - GLM and credit risk

1. Re-estimate the credit model with `age`, `D_male_gender`, `educ` and `D_marriage`.
2. Report the odds ratios and interpret the effects of gender and marriage.
3. Compute the ROC curve and the AUC of the model.
4. Compare the AUC with a simpler model that uses only `D_male_gender` and `D_marriage`. Does education improve the ranking of clients?

References

Brooks, Chris. 2014. *Introductory Econometrics for Finance*. Cambridge university press.

Campbell, John Y, Andrew Wen-Chuan Lo, Archie Craig MacKinlay, et al. 1997. *The Econometrics of Financial Markets*. Vol. 2. Princeton University press Princeton, NJ.

Fox, John, and Sanford Weisberg. 2019. *An R Companion to Applied Regression*. Third. Sage. https://www.john-fox.ca/Companion/.

Greene, William H. 2003. *Econometric Analysis*. Pearson Education India.

Hamilton, James Douglas. 1994. *Time Series Analysis*. Vol. 2. Princeton university press Princeton.

Hothorn, Torsten, Achim Zeileis, Richard W. Farebrother, and Clint Cummins. 2022. *Lmtest: Testing Linear Regression Models*. https://CRAN.R-project.org/package=lmtest.

Pena, Edsel A., and Elizabeth H. Slate. 2019. *Gvlma: Global Validation of Linear Models Assumptions*. https://CRAN.R-project.org/package=gvlma.