Using R for Introduction to Econometrics

Christoph Hanck, Martin Arnold, Alexander Gerber and Martin Schmelzer

2018-07-17

Contents

Chapter 1

Introduction

The interest in the freely available statistical programming language and software environment `R` is soaring. By the time we wrote first drafts for this project, more than 11000 addons (many of them providing cutting-edge methods) were made available on the Comprehensive `R` Archive Network (CRAN), an extensive network of FTP servers around the world that store identical and up-to-date versions of `R` code and its documentation. `R` dominates other (commercial) software for statistical computing in most fields of research in applied statistics. The benefits of it being freely available, open source and having a large and constantly growing community of users that contribute to CRAN render `R` more and more appealing for empirical economists and econometricians as well.

A striking advantage of using `R` in econometrics courses is that it enables students to explicitly document their analysis step-by-step such that it is easy to update and to expand. This allows to re-use code for similar applications with different data. Furthermore, `R` programs are fully reproducible which makes it straightforward for others to comprehend and validate results.

Over the recent years, `R` has thus become an integral part of the curricula of econometrics classes we teach at University of Duisburg-Essen. In some sense, learning to code is comparable to learning a foreign language and continuous practice is essential for the learning success. Of course, presenting bare `R` code chunks

on slides has mostly a deterring effect for the students to engage with programming on their own. This is why we offer tutorials where both econometric theory and its applications using R are introduced, for some time now. As for accompanying literature, there are some excellent books that deal with R and its applications to econometrics like Kleiber and Zeileis (2008). However, we have found that these works are somewhat difficult to access, especially for undergraduate students in economics having little understanding of econometric methods and predominantly no experience in programming at all. Consequently, we have started to compile a collection of reproducible reports for use in class. These reports provide guidance on how to implement selected applications from the textbook *Introduction to Econometrics* (Stock and Watson, 2015) which serves as a basis for the lecture and the accompanying tutorials. The process has been facilitated considerably with the release of `knitr` (2018). `knitr` is an R package for dynamic report generation which allows to seamlessly combine pure text, LaTeX, R code and its output in a variety of formats, including PDF and HTML. Being inspired by *Using R for Introductory Econometrics* (Heiss, 2016)[1] and with this powerful toolkit at hand we decided to write up our own empirical companion to Stock and Watson (2015) which resulted in **U**sing **R** **f**or **I**ntroduction **t**o **E**conometrics (*URFITE*).

Similarly to the book by Heiss (2016) this project is neither a comprehensive econometrics textbook nor is it intended to be a general introduction R. *URFITE* is best described as an interactive script in the style of a reproducible research report which aims to provide students of economic sciences with a platform-independent e-learning arrangement by seamlessly intertwining theoretical core knowledge and empirical skills in undergraduate econometrics. Of course the focus is set on empirical applications with R; we leave out tedious derivations and formal proofs wherever we can. *URFITE* is closely aligned on Stock and Watson (2015) which does very well in motivating theory by real-world applications. However, we take it a step further and enable students not only to learn how results of case studies can be replicated with R but we also intend to strengthen their ability in using the newly acquired skills in other empirical applications. To support this, each chapter contains interactive R programming exercises. These exercises are used as supplements to code chunks that display how previously discussed techniques can be implemented within R. They are generated using the DataCamp light widget and are backed by an R-session which is maintained on DataCamp's servers. You may play around with the example exercise presented below.

This interactive application is only available in the HTML version.

As you can see above, the widget consists of two tabs. `script.R` mimics an `.R`-file, a file format that is commonly used for storing R code. Lines starting with a # are commented out, that is they are not recognized as code. Furthermore, `script.R` works like an exercise sheet where you may write down the solution you come up with. If you hit the button *Run*, the code will be executed, submission correctness tests are run and you will be notified whether your approach is correct. If it is not correct, you will receive feedback suggesting improvements or hints. The other tab, `R Console`, is a fully functional R console that can be used for trying out solutions to exercises before submitting them. Of course you may submit (almost any) arbitrary R code and use the console to play around and explore. Simply type a command and hit the enter key on your keyboard.

As an example, consider the following line of code presented in chunk below. It tells R to compute the number of packages available on `CRAN`. The code chunk is followed by the output produced.

```
# compute the number of packages available on CRAN
nrow(available.packages(repos = "http://cran.us.r-project.org"))
```

```
## [1] 12740
```

Each code chunk is equipped with a button on the outer right hand side which copies the code to your clipboard. This makes it convenient to work with larger code segments. In the widget above, you may click on `R Console` and type `nrow(available.packages())` (the command from the code chunk above) and execute it by hitting *Enter* on your keyboard[2].

[1] Heiss (2016) builds on the popular *Introductory Econometrics* by Wooldridge (2016) and demonstrates how to replicate the applications discussed therein using R.

[2] The R session is initialized by clicking anywhere into the widget. This might take a few seconds. Just wait for the indicator next to the button *Run* to turn green

As you might have noticed, there are some out-commented lines in the widget that ask you to assign a numeric value to a variable and then to print the variable's content to the console. You may enter your solution approach to `script.R` and hit the button *Run* in order to get the feedback described further above. In case you do not know how to solve this sample exercise (don't panic, that is probably why you are reading this), a click on *Hint* will prompt you with some advice. If you still can't find a solution, a click on *solution* will provide you with another tab, `Solution.R` which contains sample solution code. It will often be the case that exercises can be solved in many different ways and `Solution.R` presents what we consider as comprehensible and idiomatic.

Conventions Used in this Book

- *Italic* text indicates new terms, names, buttons and alike.
- `Constant width text`, is generally used in paragraphs to refer to `R` code. This includes commands, variables, functions, data types, databases and file names.
- Constant width text on gray background is used to indicate `R` code that can be typed literally by you. It may appear in paragraphs for better distinguishability among executable and non-executable code statements but it will mostly be encountered in shape of large blocks of `R` code. These blocks are referred to as code chunks (see above).

Acknowledgements

We thank Alexander Blasberg and Kim Hermann for proofreading and their constructive criticism.

1.1 A Very Short Introduction to `R` and *RStudio*

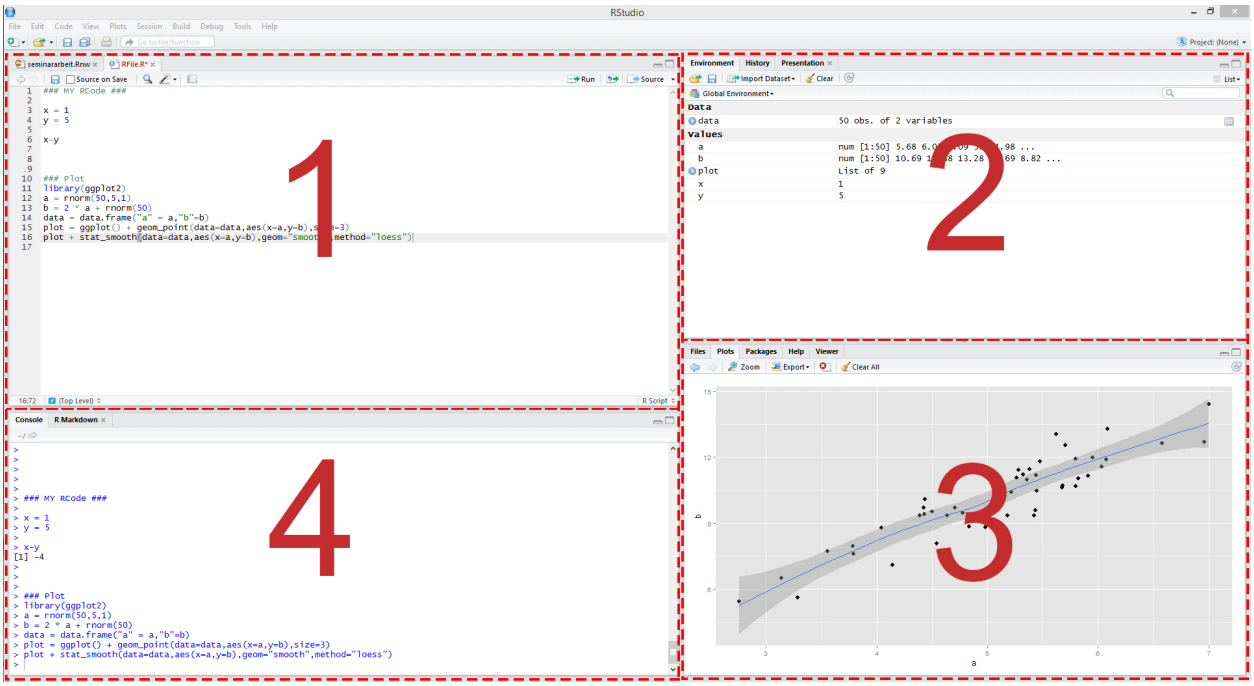


Figure 1.1: RStudio: the four panes

R Basics

This section is meant for those who have never worked with R or *RStudio*. If you at least know how to create objects and call functions, you can skip it. If you would like to refresh your memories or get a feeling for how to work with *RStudio*, keep reading.

First of all start *RStudio* and create a new R Script by selecting *File*, *New File*, *R Script*. In the editor pane type

```
1 + 1
```

and click on the button labeled *Run* in the top right corner of the editor. By doing so, your line of code is send to the console and the result of this operation should be displayed right underneath it. As you can see, R works just like a calculator. You can do all the arithmetic calculations by using the corresponding operator (+, - , *, / or ^). If you are not sure what the last operator does, try it out and check the results.

Vectors

R is of course more sophisticated than that. We can work with variables or more generally objects. Objects are defined by using the assignment operator `<-`. To create a variable named `x` which contains the value `10` type `x <- 10` and click the button *Run* yet again. The new variable should have appeared in the environment pane on the top right. The console however did not show any results, because our line of code did not contain any call that creates output. When you now type `x` in the console and hit return, you ask R to show you the value of `x` and the corresponding value should be printed in the console.

`x` is a scalar, a vector of length 1. You can easily create longer vectors by using the function `c()` (*c* for "concatenate" or "combine"). To create a vector `y` containing the numbers 1 to 5 and print it, do the following.

```
y <- c(1, 2, 3, 4, 5)
y
```

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

You can also create a vector of letters or words. For now just remember that characters have to be surrounded by quotes, else wise they will be parsed as object names.

```
hello <- c("Hello", "World")
```

Here we have created a vector of length 2 containing the words `Hello` and `World`.

Do not forget to save your script! To do so, select *File*, *Save*.

Functions

You have seen the function `c()` that can be used to combine objects. In general, function calls look all the same, a function name is always followed by round parentheses. Sometimes, the parentheses include arguments

Here are two simple examples.

```
z <- seq(from = 1, to = 5, by = 1)

mean(x = z)
```

```
## [1] 3
```

In the first line we use a function called `seq` to create the exact same vector as we did in the previous section but naming it `z`. The function takes on the arguments `from`, `to` and `by` which should be self-explaining.

The function `mean()` computes the arithmetic mean of its argument `x`. Since we pass the vector `z` as the argument `x` to `mean()`, the result is `3`!

If you are not sure what argument a function expects you may consult the function's documentation. Let's say we are not sure how the arguments required for `seq()` work. Then we can type `?seq` in the console and by hitting return the documentation page for that function pops up in the lower right pane of *RStudio*. In there, the section *Arguments* holds the information we seek.

On the bottom of almost every help page you find examples on how to use the corresponding functions. This is very helpful for beginners and we recommend to look out for those.

Chapter 2

Regression with a Binary Dependent Variable

In this chapter, we discuss a special class of regression models that aims to explain a limited dependent variable by multiple regressors.

In particular, we consider models where the dependent variable is binary. We will see that in such models, the regression function can be interpreted as a conditional probability function of the binary dependent variable.

We review the following concepts:

- The linear probability model
- The Probit model
- The Logit model
- Maximum likelihood estimation of nonlinear regression models

Of course, we will also see how to estimate abovementioned models using `R` and discuss an application where we examine the question whether there is racial discrimination in the U.S. mortgage market.

The following packages and their dependencies are needed for reproduction of the code chunks presented throughout this chapter on your computer:

- `AER`
- `stargazer`

Check whether the following code chunk runs without any errors.

```
library(AER)
library(stargazer)
```

2.1 Binary Dependent Variables and the Linear Probability Model

Key Concept 11.1
The Linear Probability Model

The linear regression model

$$Y_i = \beta_0 + \beta_1 + X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + u_i$$

with a binary dependent variable Y_i is called the linear probability model. In the linear probability model we have

$$E(Y|X_1, X_2, \ldots, X_k) = P(Y = 1|X_1, X_2, \ldots, X_3)$$

such that

$$P(Y = 1|X_1, X_2, \ldots, X_k) = \beta_0 + \beta_1 + X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki}.$$

Thus, the coefficient β_j can be interpreted as the change in the probability that $Y_i = 1$, holding constant the other $k - 1$ regressors. Just as in common multiple regression, the β_j can be estimated using OLS and the robust standard error formulas can be used for hypothesis testing and computation of confidence intervals.

In most linear probability models, R^2 has no meaningful interpretation since the regression line can never fit the data perfectly if the dependent variable is binary and the regressors are continuous. This can be seen in the application below.

We emphasize that it is *essential* to use robust standard errors since the u_i in a linear probability model are always heteroskedastic.

Linear probability models are easily estimated in `R` using the function extttlm().

Mortgage Data

Following the book, we start by loading the data set `HMDA` which provides data that relate to mortgage applications filed in Boston in the year of 1990.

```
# load `AER` package and attach the HMDA data
library(AER)
data(HMDA)
```

We continue by inspecting the first few observations and compute summary statistics afterwards.

```
# Inspect data
head(HMDA)
```

```
##   deny pirat hirat     lvrat chist mhist phist unemp selfemp insurance
## 1   no 0.221 0.221 0.8000000     5     2    no   3.9      no        no
## 2   no 0.265 0.265 0.9218750     2     2    no   3.2      no        no
## 3   no 0.372 0.248 0.9203980     1     2    no   3.2      no        no
## 4   no 0.320 0.250 0.8604651     1     2    no   4.3      no        no
## 5   no 0.360 0.350 0.6000000     1     1    no   3.2      no        no
## 6   no 0.240 0.170 0.5105263     1     1    no   3.9      no        no
##   condomin afam single hschool
## 1       no   no     no     yes
```

```
## 2       no   no    yes     yes
## 3       no   no     no     yes
## 4       no   no     no     yes
## 5       no   no     no     yes
## 6       no   no     no     yes
```

```
summary(HMDA)
```

```
##   deny          pirat            hirat            lvrat         chist
##  no :2095   Min.   :0.0000   Min.   :0.0000   Min.   :0.0200   1:1353
##  yes: 285   1st Qu.:0.2800   1st Qu.:0.2140   1st Qu.:0.6527   2: 441
##             Median :0.3300   Median :0.2600   Median :0.7795   3: 126
##             Mean   :0.3308   Mean   :0.2553   Mean   :0.7378   4:  77
##             3rd Qu.:0.3700   3rd Qu.:0.2988   3rd Qu.:0.8685   5: 182
##             Max.   :3.0000   Max.   :3.0000   Max.   :1.9500   6: 201
##  mhist    phist          unemp        selfemp    insurance  condomin
##  1: 747   no :2205   Min.   : 1.800   no :2103   no :2332   no :1694
##  2:1571   yes: 175   1st Qu.: 3.100   yes: 277   yes:  48   yes: 686
##  3:  41              Median : 3.200
##  4:  21              Mean   : 3.774
##                      3rd Qu.: 3.900
##                      Max.   :10.600
##   afam      single     hschool
##  no :2041   no :1444   no :  39
##  yes: 339   yes: 936   yes:2341
##
##
##
##
```

The variable we are interested in modelling is `deny`, an indicator for whether an applicants mortgage application has been accepted (`deny = no`) or denied (`deny = yes`). A regressors that can be presumed to have power in explaining whether a mortgage application has been denied or accepted is `pirat`, the size of the anticipated total monthly loan payments relative to the the applicant's income. It is straightforward to translate this into the simple regression model

$$deny = \beta_0 + \beta_1 P/I\ ratio + u. \tag{2.1}$$

We can estimate this model just as any other linear regression model using `lm()`. Before we do so, the variable `deny` must be converted to a numeric variable using `as.numeric()`. Note that `as.numeric(HMDA\$deny)` will turn `deny = no` into `deny = 1` and `deny = yes` into `deny = 2`, so using `as.numeric(HMDA$deny)-1` we obtain the values `0` and `1`.

```
# convert 'deny' to numeric
HMDA$deny <- as.numeric(HMDA$deny) - 1

# estimate a simple linear probabilty model
denymod1 <- lm(deny ~ pirat, data = HMDA)
denymod1
```

```
##
## Call:
## lm(formula = deny ~ pirat, data = HMDA)
##
## Coefficients:
```

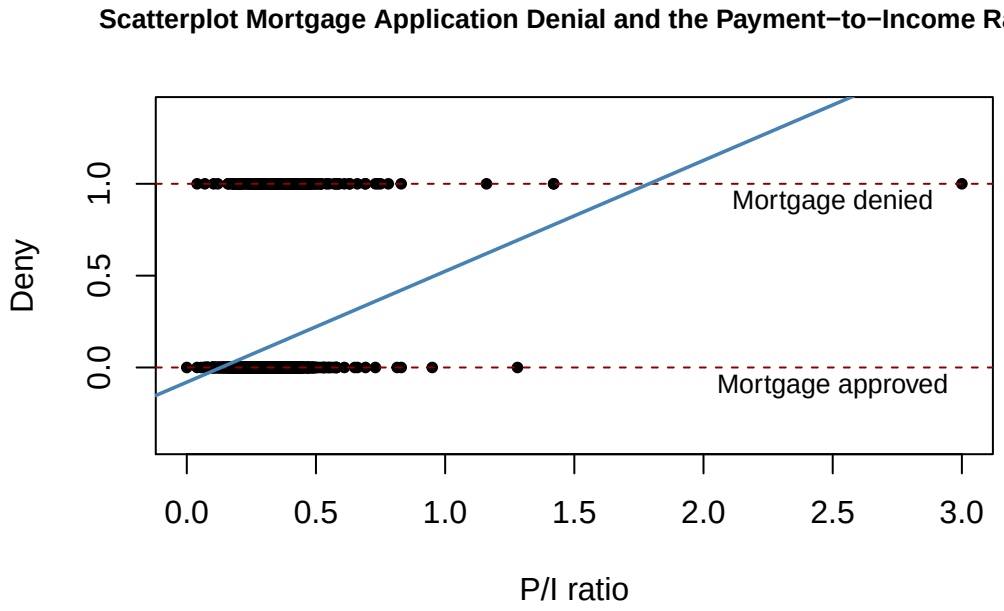
```
## (Intercept)        pirat
##    -0.07991      0.60353
```

We plot the data and the regression line to reproduce Figure 11.1 of the book next.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
     main = "Scatterplot Mortgage Application Denial and the Payment-to-Income Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.8
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line
abline(denymod1,
       lwd = 1.8,
       col = "steelblue")
```



We observe that, according to the estimated model equation, a payment-to-income ratio of 1 is associated with an expected probability of mortgage application denial of roughly 50%. The model indicates that there is a positive relation between the payment-to-income ratio and the probability of a denied mortgage application so individuals with a high ratio of loan payments to income are more likely to be rejected.

We may use `coeftest()` to obtain robust standard errors for both coefficient estimates.

```
# model coefficient summary
coeftest(denymod1, vcov. = vcovHC(denymod1, type = "HC1"))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.079910   0.031967 -2.4998   0.01249 *
## pirat        0.603535   0.098483  6.1283 1.036e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated regression line is

$$\widehat{deny} = \underset{(0.032)}{-0.080} + \underset{(0.098)}{0.604} P/I\ ratio.$$

The true coefficient on P/I *ratio* is statistically different from 0 at the 1% level. Its estimate can be interpreted as follows: a 1% increase in P/I *ratio* leads to an increase in the probability of a loan denial by $0.604 * 0.01 = 0.00604 \approx 0.6\%$.

Following the book we augment the simple model (2.1) by an additional regressor *black* which equals 1 if the applicant is an African American and equals 0 if the applicant is white. Such a specification is the baseline for investigation of the question whether there is racial discrimination in the mortgage market: if being black has a significant (positive) influence on the probability of a loan denial when we control for factors that allow for an objective assessment of an applicants credit worthiness, this is an indicator for discrimination.

```
# rename variable `afam`
colnames(HMDA)[colnames(HMDA) == "afam"] <- "black"

# estimate the model
denymod2 <- lm(deny ~ pirat + black, data = HMDA)
coeftest(denymod2, vcov. = vcovHC(denymod2))
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -0.090514   0.033430 -2.7076  0.006826 **
## pirat        0.559195   0.103671  5.3939 7.575e-08 ***
## blackyes     0.177428   0.025055  7.0815 1.871e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So the estimated regression function is

$$\widehat{deny} = \underset{(0.029)}{-0.091} + \underset{(0.089)}{0.559} P/I\ ratio + \underset{(0.025)}{0.177} black. \tag{2.2}$$

The coefficient on *black* is positive and significantly different from zero at the 0.01% level. The interpretation is that, holding constant the P/I *ratio*, being black increases the probability of a mortgage application denial by about 17.7%. This suggests racial discrimination. However, this result might be distorted by omitted variable bias so this could be a premature conclusion.

2.2 Probit and Logit Regression

The linear probability model has a major flaw: it assumes the conditional probability function to be linear. This does not restrict the probability of observing $Y = 1$ conditional on some regressor to lie between 0 and 1. We can easily see this in our reproduction of Figure 11.1 of the book: for P/I *ratio* ≥ 1.75, model

(2.1) predicts the probability of a mortgage application denial to be bigger than 1. For applications with P/I *ratio* close to 0, the predicted probability of denial is even negative so the model has no meaningful interpretation here.

This circumstance demands for an approach that use a *nonlinear* function to model the conditional probability function of a binary dependent variable. Commonly used methods are Probit and Logit regression.

Probit Regression

In Probit regression, the cumulative standard normal distribution function Φ is used to model the regression function when the dependent variable is binary, that is we assume

$$E(Y|X) = P(Y = 1|X) = \Phi(\beta_0 + \beta_1 X). \tag{2.3}$$

$\beta_0 + \beta_1 X$ in (2.3) plays the role of a quantile z. Remember that

$$\Phi(z) = P(Z \leq z) \ , \ Z \sim \mathcal{N}(0, 1).$$

such that the Probit coefficient β_1 in (2.3) is the change in z associated with a one unit change in X. Note that, although the effect on z of a change in X is linear, the link between z and the dependent variable Y is nonlinear since Φ is a nonlinear function of X.

Since in Probit models the dependent variable is a nonlinear function of the regressors, the coefficient on X has no simple interpretation. According to Key Concept 8.1, the expected change in the probability that $Y = 1$ due to a change in P/I *ratio* can be computed in the following manner:

1. Compute the predicted probability that $Y = 1$ for the original value of X.
2. Compute the predicted probability that $Y = 1$ for $X + \Delta X$.
3. Compute the difference between both predicted probabilities.

Of course we can generalize (2.3) to Probit regression with multiple regressors to mitigate the risk of facing omitted variable bias. The core knowledge on Probit regression is summarized in Key Concept 11.2.

Key Concept 11.2
Probit Model, Predicted Probabilities and Estimated Effects

Assume that Y is a binary variable. The model

$$Y = \beta_0 + \beta_1 + X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + u$$

with

$$P(Y = 1|X_1, X_2, \ldots, X_k) = \Phi(\beta_0 + \beta_1 + X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)$$

is the population Probit model with multiple regressors where $X_1, X_2, \ldots, X_k$ are regressors and Φ is the cumulative standard normal distribution function.

The predicted probability that $Y = 1$ given $X_1, X_2, \ldots, X_k$ can be calculated in two steps:

1. Compute $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$
2. Look up $\Phi(z)$ using a normal distribution table or by calling `pnorm()`.

β_j is the effect on z of a one unit change in regressor X_j, holding constant all other $k - 1$ regressors.

The effect on the predicted probability of a change in a regressor can be computed as shown in Key Concept 8.1.

In `R`, Probit models can be estimated using the function `glm()` from the package `stats`. Using the argument `family` we specify that we want to use a Probit link function.

We now estimate a simple Probit model of the probability of a mortgage denial.

```
# estimate the simple probit model
denyprobit <- glm(deny ~ pirat,
                  family = binomial(link = "probit"),
                  data = HMDA)

coeftest(denyprobit, vcov. = vcovHC(denyprobit, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -2.19415    0.18901 -11.6087 < 2.2e-16 ***
## pirat        2.96787    0.53698   5.5269 3.259e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model equation is

$$\widehat{P(deny|P/I\ ratio)} = \Phi(\underset{(0.19)}{-2.19} + \underset{(0.54)}{2.97} P/I\ ratio). \tag{2.4}$$

Just as in the linear probability model we find that the relation between the probability of denial and the payments-to-income ratio is positive and that the corresponding coefficient is highly significant.

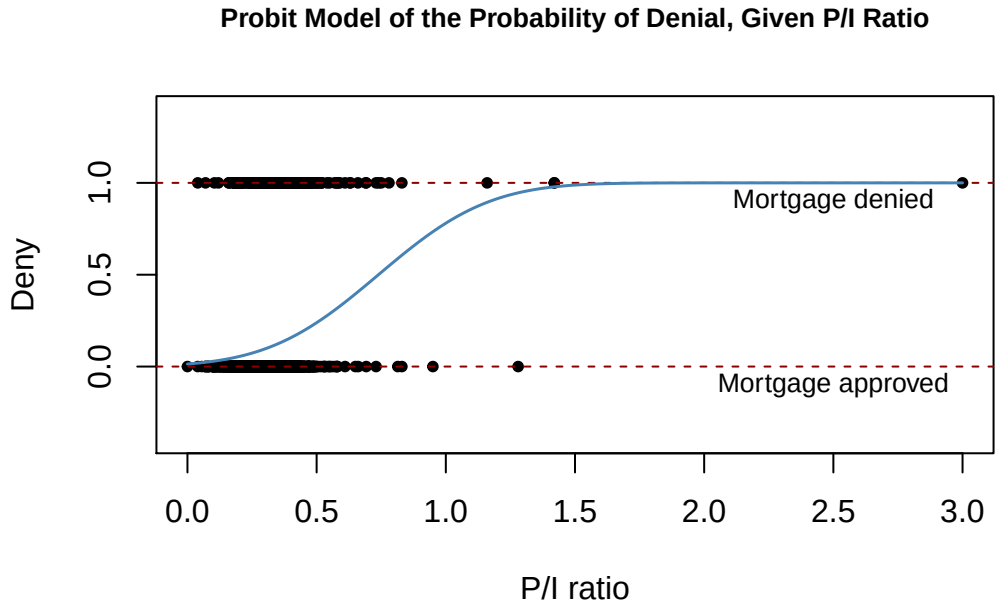
The following code chunk reproduces Figure 11.2 of the book.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
     main = "Probit Model of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.85
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line
x <- seq(0, 3, 0.01)
y <- predict(denyprobit, list(pirat = x), type = "response")

lines(x, y, lwd = 1.5, col = "steelblue")
```



Notice that the estimated regression function has a stretched "S"-shape which is typical for the CDF of a of continuous random variable with symmetric PDF like that of a normally distributed random variable. The function is clearly nonlinear and flattens out for large and small values of P/I *ratio*. More importantly, the functional form ensures that the predicted conditional probabilities of a denial lie between 0 and 1.

We may use `predict()` to compute the predicted change in the denial probability when P/I *ratio* is increased from 0.3 to 0.4.

```
# 1. compute predictions for P/I ratio = 0.3, 0.4
predictions <- predict(denyprobit,
                       newdata = data.frame("pirat" = c(0.3, 0.4)),
                       type = "response"
```

```
                       )

# 2. Compute difference in probabilities
diff(predictions)
```

```
##          2
## 0.06081433
```

We find that an increase in the payment-to-income ratio from 0.3 to 0.4 is predicted to increase the probability of denial by approximately 6.2%.

We continue by using an augmented Probit model to estimate the effect of race on the probability of a mortgage application denial.

```
denyprobit2 <- glm(deny ~ pirat + black,
                   family = binomial(link = "probit"),
                   data = HMDA)

coeftest(denyprobit2, vcov. = vcovHC(denyprobit2, type = "HC1"))
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -2.258787   0.176608 -12.7898 < 2.2e-16 ***
## pirat        2.741779   0.497673   5.5092 3.605e-08 ***
## blackyes     0.708155   0.083091   8.5227 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated model equation is

$$\widehat{P(deny|P/I\ ratio, black)} = \Phi(\underset{(0.18)}{-2.26} + \underset{(0.50)}{2.74}\, P/I\ ratio + \underset{(0.08)}{0.71}\, black). \tag{2.5}$$

While all coefficients are highly significant, both the estimated coefficients on the payments-to-income ratio and the indicator for African American descent are positive. Again, the coefficients are difficult to interpret but they indicate that first, African Americans have a higher probability of denial than white applicants do, holding constant the payments-to-income ratio and second, applicants with a high payments-to-income ratio face a higher risk of being rejected.

How big is the estimated difference in denial probabilities between two hypothetical applicants with the same payments-to-income ratio? As before, we may use `predict()` to compute this difference.

```
# 1. compute predictions for P/I ratio = 0.3,0.4
predictions <- predict(denyprobit2,
                       newdata = data.frame("black" = c("no", "yes"),
                                            "pirat" = c(0.3, 0.3)),
                       type = "response"
                       )

# 2. Compute difference in probabilities
diff(predictions)
```

```
##         2
## 0.1578117
```

In this case, the estimated difference in denial probabilities is about 15.8%.

Logit Regression

Key Concept 11.3 summarizes the main differences between Logit and Probit regression.

Key Concept 11.3
Logit Regression

The population Logit regression function is

$$\begin{aligned} P(Y = 1|X_1, X_2, \dots, X_k) &= F(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k) \\ &= \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}}. \end{aligned}$$

The idea is similar to Probit regression except that a different CDF is used:

$$F(x) = \frac{1}{1 + e^{-x}}$$

is the CDF of a standard logistically distributed random variable.

As for Probit regression, there is no simple interpretation of the model coefficients and it is best to consider predicted probabilities or differences in predicted probabilities. Here again, t-statistics and confidence intervals based on large sample normal approximations can be computed as usual.

It is fairly easy to estimate a Logit regression model using R.

```
denylogit <- glm(deny ~ pirat,
                 family = binomial(link = "logit"),
                 data = HMDA)

coeftest(denylogit, vcov. = vcovHC(denylogit, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -4.02843    0.35898 -11.2218 < 2.2e-16 ***
## pirat        5.88450    1.00015   5.8836 4.014e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The subsequent code chunk reproduces Figure 11.3 of the book.

```
# plot data
plot(x = HMDA$pirat,
     HMDA$deny,
     main = "Probit and Logit Models Model of the Probability of Denial, Given P/I Ratio",
     xlab = "P/I ratio",
     ylab = "Deny",
     pch = 20,
     ylim = c(-0.4, 1.4),
     cex.main = 0.9
```

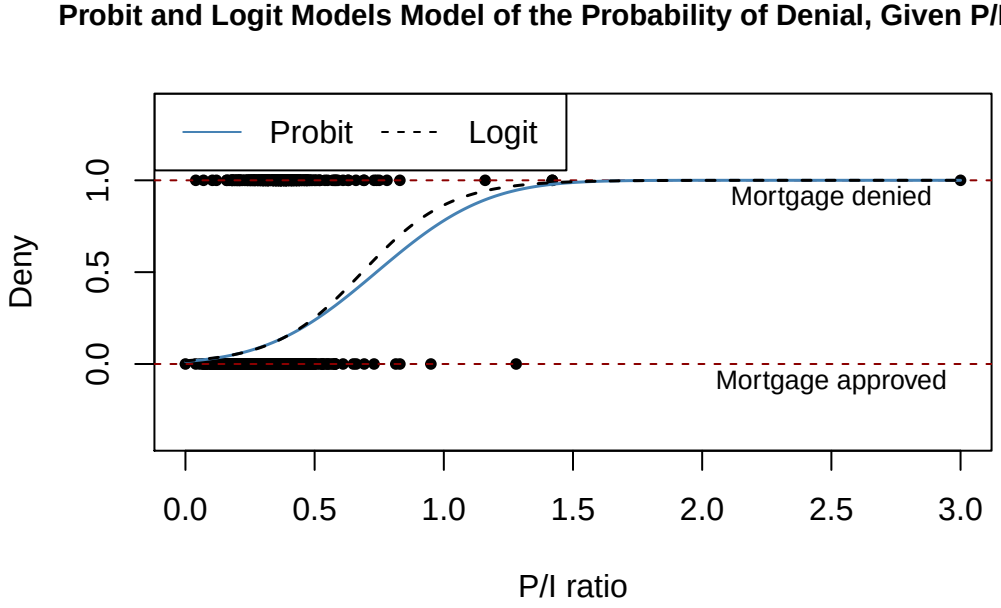
```
)

# add horizontal dashed lines and text
abline(h = 1, lty = 2, col = "darkred")
abline(h = 0, lty = 2, col = "darkred")
text(2.5, 0.9, cex = 0.8, "Mortgage denied")
text(2.5, -0.1, cex= 0.8, "Mortgage approved")

# add estimated regression line of Probit and Logit models
x <- seq(0, 3, 0.01)
y_probit <- predict(denyprobit, list(pirat = x), type = "response")
y_logit <- predict(denylogit, list(pirat = x), type = "response")

lines(x, y_probit, lwd = 1.5, col = "steelblue")
lines(x, y_logit, lwd = 1.5, col = "black", lty = 2)

# add a legend
legend("topleft",
       horiz = TRUE,
       legend = c("Probit", "Logit"),
       col = c("steelblue", "black"),
       lty = c(1, 2))
```



Notice that both models produce very similar estimates of the probability that a mortgage application will be denied depending on the applicants payment-to-income ratio.

Following the book we expand the simple Logit model of mortgage denial with the additional regressor *black*.

```
# estimate Logit regression with multiple regressors
denylogit2 <- glm(deny ~ pirat + black,
                  family = binomial(link = "logit"),
                  data = HMDA)

coeftest(denylogit2, vcov. = vcovHC(denylogit2, type = "HC1"))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error  z value  Pr(>|z|)
## (Intercept) -4.12556    0.34597 -11.9245 < 2.2e-16 ***
## pirat        5.37036    0.96376   5.5723 2.514e-08 ***
## blackyes     1.27278    0.14616   8.7081 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We obtain

$$\widehat{P(deny = 1|P/I ratio, black)} = F(\underset{(0.35)}{-4.13} + \underset{(0.96)}{5.37}\, P/I\ ratio + \underset{(0.15)}{1.27}\, black). \tag{2.6}$$

As for the Probit model (2.5) all model coefficients are highly significant and we obtain positive estimates for the coefficients on P/I *ratio* and *black*. For comparison purposes we compute the predicted probability of denial for two hypothetical applicants that differ in race and have a P/I *ratio* of 0.3.

```
# 1. compute predictions for P/I ratio = 0.3
predictions <- predict(denylogit2,
                       newdata = data.frame("black" = c("no", "yes"),
                                            "pirat" = c(0.3, 0.3)),
                       type = "response"
                      )

predictions
```

```
##          1          2
## 0.07485143 0.22414592
```

```
# 2. Compute difference in probabilities
diff(predictions)
```

```
##         2
## 0.1492945
```

We find that the white applicant faces a denial probability of only 7.5%, while the African American is rejected with a probability of 22.4% with makes a difference of 14.9%.

Comparison of the Models

The Probit model and the Logit model deliver only approximations to the unknown population regression function $E(Y|X)$. It is not unambiguous to decide which model one should use in practice. The linear probability model has the clear drawback of not being able to capture the nonlinear nature of the population regression function and it predicts probabilities to lie outside the interval $[0, 1]$ for extreme regressor values. Probit and Logit models are harder to interpret but can capture the nonlinearities better than the linear approach: both models produce predictions of probabilities that lie inside the interval $[0, 1]$. Predictions of Probit and Logit models are often close to each other. The book suggests to use the method that is easiest to use in the statistical software of choice. As we have seen, it is equally easy to estimate Probit and Logit model using `R`. We can therefore give no general recommendation which method to use.

2.3 Estimation and Inference in the Logit and Probit Models

So far nothing has been said about *how* Logit and Probit models are estimated by statistical software. The reason why this is interesting is that both models are *nonlinear in the parameters* and thus cannot be estimated using OLS. Instead one relies on a method called *maximum likelihood estimation* (MLE). Another approach is estimation by *nonlinear least squares* (NLS).

Nonlinear Least Squares

Consider the multiple regression Probit model

$$E(Y_i|X_{1i},\dots,X_{ki}) = P(Y_i = 1|X_{1i},\dots,X_{ki}) = \Phi(\beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki}). \tag{2.7}$$

Similarly to OLS, NLS estimates the parameters $\beta_0, \beta_1, \dots, \beta_k$ by minimizing the sum of squared mistakes

$$\sum_{i=1}^{n} \left[Y_i - \Phi(b_0 + b_1 X_{1i} + \cdots + b_k X_{ki})\right]^2.$$

NLS estimation is a consistent approach that produces estimates which are normally distributed in large samples. In `R` there are functions like `nls()` from package `stats` which provide algorithms for solving nonlinear least squares problems. However, NLS is inefficient, meaning that there are estimation techniques that have a smaller variance which is why we will not dwell any further on this topic.

Maximum Likelihood Estimation

In MLE we seek to estimate the unknown parameters choosing them such that the likelihood of drawing the sample observed is maximized. This probability is measured by means the likelihood function, the joint probability distribution of the data treated as a function of the unknown parameters. Put differently, the maximum likelihood estimates of the unknown parameters are the values that result in a model which is most likely to produce the data observed. It turns out that MLE is more efficient than NLS.

As maximum likelihood estimates are normally distributed in large samples, statistical inference for coefficients in nonlinear models like Logit and Probit regression can be made using the same tools that are used for linear regression models: we can compute t-statistics and confidence intervals.

Many software packages use an MLE algorithm for estimation of nonlinear models. The function `glm()` uses an algorithm named *iteratively reweighted least squares*.

Measures of Fit

It is important to be aware that the usual R^2 and $\overline{R^2}$ are *invalid* for nonlinear regression models. The reason for this is simple: both measures are derived under the assumption that the relation between the dependent and the explanatory variable(s) is linear. This obviously does not hold for Probit and Logit models thus R^2 does not fall between 0 and 1 and there is no meaningful interpretation. However, statistical software sometimes reports these measures anyway.

There are many measures of fit for nonlinear regression models and there is no consensus which one should be reported. The situation is even more complicated because there is no measure of fit that is generally applicable. For models with a binary response variable like *deny* one could use the following rule:
If $Y_i = 1$ and $P(Y_i|\widehat{X_{i1},\dots,X_{ik}}) > 0.5$ or if $Y_i = 0$ and $P(Y_i|\widehat{X_{i1},\dots,X_{ik}}) < 0.5$, consider the Y_i as correctly predicted. Otherwise Y_i is said to be incorrectly predicted. The measure of fit is the share of correctly

predicted observations. The downside of such an approach is that it does not yield information on the quality of the prediction.[1]

An alternative to the latter are so called pseudo-R^2 measures. In order to measure the quality of the fit, these measures compare the value of the maximized (log-)likelihood of the model with all regressors (the *full model*) to the likelihood of a model with no regressors (*null model*, regression on a constant).

For example, consider a Probit regression. The pseudo-R^2 is given by

$$\text{pseudo-}R^2 = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}$$

where $f_j^{max} \in [0,1]$ denotes the maximized likelihood for model j.

The reasoning behind this is that the maximized likelihood increases as additional regressors are added to the model, similarly to the decrease in SSR when regressors are added in a linear regression model. If the full model has a similar maximized likelihood than the null model, the full model does not really improve upon a model that uses only the information in the dependent variable, so pseudo-$R^2 \approx 0$. If the full model fits the data very good, the maximized likelihood should be close to 1 such that $\ln(f_{full}^{max}) \approx 0$ and pseudo-$R^2 \approx 1$. See Appendix 11.2 of the book for more on MLE and pseudo-R^2 measures.

`summary()` does not report pseudo-R^2 for models estimated by `glm()` but we can use the entries *residual deviance* (`deviance`) and *null deviance* (`null.deviance`) instead. These are computed as

$$\text{deviance} = -2 \times \left[\ln(f_{saturated}^{max}) - \ln(f_{full}^{max})\right]$$

and

$$\text{null deviance} = -2 \times \left[\ln(f_{saturated}^{max}) - \ln(f_{null}^{max})\right]$$

where $f_{saturated}^{max}$ is the maximized likelihood for a model which assumes that each observation has its own parameter (there are $n+1$ parameters to be estimated which leads to a perfect fit). For models wit ah binary dependent variable, it holds that

$$\text{pseudo-}R^2 = 1 - \frac{\text{deviance}}{\text{null deviance}} = 1 - \frac{\ln(f_{full}^{max})}{\ln(f_{null}^{max})}.$$

We now compute the pseudo-R^2 for the augmented Probit model of mortgage denial.

```
# compute pseudo-R2 for the probit model of mortgage denial
pseudoR2 <- 1 - (denyprobit2$deviance) / (denyprobit2$null.deviance)
pseudoR2
```

```
## [1] 0.08594259
```

Another way to obtain the pseudo-R^2 is to estimate the null model using `glm()` and extract the maximized log-likelihoods for both the null and the full model using the function `logLik()`.

```
# compute null model
denyprobit_null <- glm(formula = deny ~ 1,
                       family = binomial(link = "probit"),
                       data = HMDA)

# compute the pseudo-R2 using `logLik`
1 - logLik(denyprobit2)[1]/logLik(denyprobit_null)[1]
```

```
## [1] 0.08594259
```

[1] Note that this is in contrast to the case of a numeric dependent variable where we use the squared errors for assessment of the quality of the prediction.

2.4 Application to the Boston HMDA Data

Models (2.5) and (2.6) indicate that denial rates are higher for African American applicants holding constant payment-to-income ratio. Both results could be wrong due to omitted variable bias. In order to obtain a more trustworthy estimate of the effect of being black on the probability of a mortgage application denial we estimate a linear probability model as well as several Logit and Probit models. We thereby control for financial variables and additional applicant characteristics which are likely to influence the probability of denial and differ between black and white applicants.

Sample averages as shown in Table 11.1 of the book can be easily reproduced using the functions `mean()` (as usual for numeric variables) and `prop.table()` (for factor variables).

```
# Mean P/I ratio
mean(HMDA$pirat)
```

```
## [1] 0.3308136
```

```
# inhouse expense-to-total-income ratio
mean(HMDA$hirat)
```

```
## [1] 0.2553461
```

```
# loan-to-value ratio
mean(HMDA$lvrat)
```

```
## [1] 0.7377759
```

```
# consumer credit score
mean(as.numeric(HMDA$chist))
```

```
## [1] 2.116387
```

```
# mortgage credit score
mean(as.numeric(HMDA$mhist))
```

```
## [1] 1.721008
```

```
# public bad credit record
mean(as.numeric(HMDA$phist)-1)
```

```
## [1] 0.07352941
```

```
# denied mortgage insurance
prop.table(table(HMDA$insurance))
```

```
##
##         no        yes
## 0.97983193 0.02016807
```

```
# self-employed
prop.table(table(HMDA$selfemp))
```

```
##
##        no       yes
## 0.8836134 0.1163866
```

```
# single
prop.table(table(HMDA$single))
```

```
##
##        no       yes
## 0.6067227 0.3932773
```

```
# high school dimploma
prop.table(table(HMDA$hschool))
```

```
##
##         no        yes
## 0.01638655 0.98361345
```

```
# unemployment rate
mean(HMDA$unemp)
```

```
## [1] 3.774496
```

```
# condominium
prop.table(table(HMDA$condomin))
```

```
##
##        no       yes
## 0.7117647 0.2882353
```

```
# black
prop.table(table(HMDA$black))
```

```
##
##       no      yes
## 0.857563 0.142437
```

```
# deny
prop.table(table(HMDA$deny))
```

```
##
##         0         1
## 0.8802521 0.1197479
```

See Chapter 11.4 of the book or use `R`'s help function for more info on variables contained in the `HMDA` data set.

Before estimating the models we transform the loan-to-value ratio (`lvrat`) into a factor variable, where

$$lvrat = \begin{cases} \text{low} & \text{if } lvrat < 0.8, \\ \text{medium} & \text{if } 0.8 \leq lvrat \leq 0.95, \\ \text{high} & \text{if } lvrat > 0.95 \end{cases}$$

and convert both credit scores to numeric variables.

```
# define low, medium and high loan-to-value ratio
HMDA$lvrat <- factor(
  ifelse(HMDA$lvrat < 0.8, "low",
  ifelse(HMDA$lvrat >= 0.8 & HMDA$lvrat <= 0.95, "medium", "high")),
  levels = c("low", "medium", "high")
  )

# convert credit scores to numeric
HMDA$mhist <- as.numeric(HMDA$mhist)
HMDA$chist <- as.numeric(HMDA$chist)
```

In the next step we reproduce the estimation results presented in Table 11.2 of the book.

```
# estimate all 6 models for the denial probability
lpm_HMDA <- lm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
               + insurance + selfemp, data = HMDA)

logit_HMDA <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                  + insurance + selfemp,
                  family = binomial(link = "logit"),
                  data = HMDA)

probit_HMDA_1 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                     + insurance + selfemp,
                     family = binomial(link = "probit"),
                     data = HMDA)

probit_HMDA_2 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist + phist
                     + insurance + selfemp + single + hschool + unemp,
                     family = binomial(link = "probit"),
                     data = HMDA)

probit_HMDA_3 <- glm(deny ~ black + pirat + hirat + lvrat + chist + mhist
                     + phist + insurance + selfemp + single + hschool + unemp + condomin
                     + I(mhist==3) + I(mhist==4) + I(chist==3) + I(chist==4) + I(chist==5)
                     + I(chist==6),
                     family = binomial(link = "probit"),
                     data = HMDA)

probit_HMDA_4 <- glm(deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist
                     + insurance + selfemp + single + hschool + unemp,
                     family = binomial(link = "probit"),
                     data = HMDA)
```

Just as in previous chapters, we use the package `sandwich` for computation of heteroskedasticity-robust standard errors of the coefficient estimators in all models and store these in an object of the type `list` which is then used as the argument `se` in `stargazer()`.

```
library(stargazer)

rob_se <- list(
  sqrt(diag(vcovHC(lpm_HMDA, type = "HC1"))),
  sqrt(diag(vcovHC(logit_HMDA, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_1, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_2, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_3, type = "HC1"))),
  sqrt(diag(vcovHC(probit_HMDA_4, type = "HC1")))
)

stargazer(lpm_HMDA, logit_HMDA, probit_HMDA_1, probit_HMDA_2, probit_HMDA_3, probit_HMDA_4,
          digits = 3,
          type = "latex",
          header = FALSE,
          se = rob_se,
          model.numbers = FALSE,
          column.labels = c("(1)", "(2)", "(3)", "(4)", "(5)", "(6)")
          )
```

Table 2.1: HMDA Data: LPM, Probit and Logit Models

	Dependent variable:					
	deny					
	OLS	*logistic*		*probit*		
	(1)	(2)	(3)	(4)	(5)	(6)
blackyes	0.084*** (0.023)	0.688*** (0.183)	0.389*** (0.099)	0.371*** (0.100)	0.363*** (0.101)	0.246 (0.479)
pirat	0.449*** (0.114)	4.764*** (1.332)	2.442*** (0.673)	2.464*** (0.654)	2.622*** (0.665)	2.572*** (0.728)
hirat	−0.048 (0.110)	−0.109 (1.298)	−0.185 (0.689)	−0.302 (0.689)	−0.502 (0.715)	−0.538 (0.755)
lvratmedium	0.031** (0.013)	0.464*** (0.160)	0.214*** (0.082)	0.216*** (0.082)	0.215** (0.084)	0.216*** (0.083)
lvrathigh	0.189*** (0.050)	1.495*** (0.325)	0.791*** (0.183)	0.795*** (0.184)	0.836*** (0.185)	0.788*** (0.185)
chist	0.031*** (0.005)	0.290*** (0.039)	0.155*** (0.021)	0.158*** (0.021)	0.344*** (0.108)	0.158*** (0.021)
mhist	0.021* (0.011)	0.279** (0.138)	0.148** (0.073)	0.110 (0.076)	0.162 (0.104)	0.111 (0.077)
phistyes	0.197*** (0.035)	1.226*** (0.203)	0.697*** (0.114)	0.702*** (0.115)	0.717*** (0.116)	0.705*** (0.115)
insuranceyes	0.702*** (0.045)	4.548*** (0.576)	2.557*** (0.305)	2.585*** (0.299)	2.589*** (0.306)	2.590*** (0.299)
selfempyes	0.060*** (0.021)	0.666*** (0.214)	0.359*** (0.113)	0.346*** (0.116)	0.342*** (0.116)	0.348*** (0.116)
singleyes				0.229*** (0.080)	0.230*** (0.086)	0.226*** (0.081)
hschoolyes				−0.613*** (0.229)	−0.604** (0.237)	−0.620*** (0.229)
unemp				0.030* (0.018)	0.028 (0.018)	0.030 (0.018)
condominyes					−0.055 (0.096)	
I(mhist == 3)					−0.107 (0.301)	
I(mhist == 4)					−0.383 (0.427)	
I(chist == 3)					−0.226 (0.248)	
I(chist == 4)					−0.251 (0.338)	
I(chist == 5)					−0.789* (0.412)	
I(chist == 6)					−0.905* (0.515)	
blackyes:pirat						−0.579 (1.550)
blackyes:hirat						1.232 (1.709)
Constant	−0.183*** (0.028)	−5.707*** (0.484)	−3.041*** (0.250)	−2.575*** (0.350)	−2.896*** (0.404)	−2.543*** (0.370)
Observations	2,380	2,380	2,380	2,380	2,380	2,380
R^2	0.266					
Adjusted R^2	0.263					
Log Likelihood		−635.637	−636.847	−628.614	−625.064	−628.332
Akaike Inf. Crit.		1,293.273	1,295.694	1,285.227	1,292.129	1,288.664
Residual Std. Error	0.279 (df = 2369)					
F Statistic	85.974*** (df = 10; 2369)					
Note:						*p<0.1; **p<0.05; ***p<0.01

In Table 2.1, models (1), (2) and (3) are base specifications that include several financial control variables. They differ only in the way they model the denial probability. Model (1) is a linear probability model, model (2) is a Logit regression and model (3) uses the Probit approach.

Since model (1) is a linear model, the coefficients have direct interpretation. For example, an increase in the consumer credit score by 1 unit is estimated to increase the probability of a loan denial by about 0.031 percentage points. We also find that having a high loan-to-value ratio is predicted not to be conductive for credit approval: the coefficient for a loan-to-value ratio higher than 0.95 is 0.189 so clients with this property are estimated to face an almost 19% larger risk of denial than those with a low loan-to-value ratio, ceteris paribus. For the linear probability model, the coefficient on the race dummy is estimated to be 0.084 which indicates the denial probability for African Americans is 8.4% larger than for white applicants with the same characteristics except for race. Apart from housing expense-to-income ratio and the mortgage credit score, all coefficients are significant.

Models (2) and (3) provide similar evidence that there is racial discrimination in the U.S. mortgage market. All coefficients except for housing expense-to-income ratio (which is not significantly different from zero) are significant at the 1% level. As discussed above, the nonlinearity makes the interpretation of the coefficient estimates more difficult than for model (1). In order to make a statement about the effect of being black, we need to compute the estimated denial probability for two individuals that differ only in race. For the comparison we consider two individuals that share mean values for all numeric regressors. For all qualitative variables we assign the property that is the most representative for the data at hand. As an example, consider self-employment: we have seen that about 88% of all individuals in the sample are not self-employed such that we set `selfemp = no`. Using this approach, the estimate for the effect on the denial probability of being African American obtained using the Logit model (2) is about 4%. The next code chunk shows how to apply this approach for models (1) to (7) using `R`.

```
# regressor values for average person, black = "yes"
new <- data.frame(
  "pirat" = mean(HMDA$pirat),
  "hirat" = mean(HMDA$hirat),
  "lvrat" = "low",
  "chist" = mean(HMDA$chist),
  "mhist" = mean(HMDA$mhist),
  "phist" = "no",
  "insurance" = "no",
  "selfemp" = "no",
  "black" = c("no", "yes"),
  "single" = "no",
  "hschool" = "yes",
  "unemp" = mean(HMDA$unemp),
  "condomin" = "no"
)

# differnce predicted by the LPM
predictions <- predict(lpm_HMDA, newdata = new)
diff(predictions)
```

```
##          2
## 0.08369674
```

```
# differnce predicted by the logit model
predictions <- predict(logit_HMDA, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.04042135
```

```
# difference predicted by probit model (3)
predictions <- predict(probit_HMDA_1, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.05049716
```

```
# difference predicted by probit model (4)
predictions <- predict(probit_HMDA_2, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.03978918
```

```
# difference predicted by probit model (5)
predictions <- predict(probit_HMDA_3, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.04972468
```

```
# difference predicted by probit model (6)
predictions <- predict(probit_HMDA_4, newdata = new, type = "response")
diff(predictions)
```

```
##          2
## 0.03955893
```

The estimates of the impact on the denial probability of being black are similar for models (2) and (3). In particular, it is interesting that the magnitude of the estimated effects is much smaller than for Probit and Logit models that do not control for financial characteristics (see section 11.2). This indicates that these simple models produce biased estimates due to omitted variables.

Regressions (4) to (6) use regression specifications that include different sets of applicant characteristics and credit rating indicator variables as well as interactions. However, most of the corresponding coefficients are not significant and the estimates of the coefficient on `black` obtained for these models as well as the estimated difference in denial probabilities do not deviate much from those obtained for the similar specifications (2) and (3).

An interesting question related to racial discrimination can be investigated using the Probit model (6) where the interactions `blackyes:pirat` and `blackyes:hirat` are added to the the specification of model (4). If the coefficient on `blackyes:pirat` was different from zero, the effect of the payment-to-income ratio on the denial probability would be different for black and white applicants. Similarly, a non-zero coefficient on `blackyes:hirat` would indicate that loan officers weight the risk of bankrupt associated with a high loan-to-value ratio differently for black and white mortgage applicants. We can test whether these coefficients are jointly significant at the 5% level using an F-Test.

```
linearHypothesis(probit_HMDA_4,
                 test = "F",
                 c("blackyes:pirat=0", "blackyes:hirat=0"),
                 vcov = vcovHC(probit_HMDA_4, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## blackyes:pirat = 0
## blackyes:hirat = 0
##
```

```
## Model 1: restricted model
## Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +
##     insurance + selfemp + single + hschool + unemp
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1   2366
## 2   2364  2 0.2616 0.7698
```

Since p-value ≈ 0.77 for this test, the null cannot be rejected. Nonetheless, we cannot reject the hypothesis that there is no racial discrimination at all since the corresponding F-test has a p-value of about 0.002.

```
linearHypothesis(probit_HMDA_4,
                 test = "F",
                 c("blackyes=0", "blackyes:pirat=0", "blackyes:hirat=0"),
                 vcov = vcovHC(probit_HMDA_4, type = "HC1"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## blackyes = 0
## blackyes:pirat = 0
## blackyes:hirat = 0
##
## Model 1: restricted model
## Model 2: deny ~ black * (pirat + hirat) + lvrat + chist + mhist + phist +
##     insurance + selfemp + single + hschool + unemp
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1   2367
## 2   2364  3 4.8886 0.002168 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary

Models (1) to (6) provide evidence that there is an effect of being African American on the probability of a mortgage application denial: in all specifications, the effect is estimated to be positive (ranging from 4% to 5% for the fictive individuals considered) and is significantly different from zero at the 1% level. While the linear probability model seems to overestimate this effect slightly, it still can be used as an approximation to an intrinsic nonlinear relationship.

See Chapters 11.4 and 11.5 of the book for a discussion of external and internal validity of this study and some concluding remarks on regression models where the dependent variable is binary.

Bibliography

Heiss, F. (2016). *Using R for Introductory Econometrics*. CreateSpace Independent Publishing Platform.

Kleiber, C. and Zeileis, A. (2008). *Applied econometrics with R*. Springer.

Stock, J. and Watson, M. (2015). *Introduction to Econometrics, Third Update, Global Edition*. Pearson Education Limited.

Wooldridge, J. (2016). *Introductory econometrics*. Cengage Learning, sixth editon edition.