R Notebook

Code

Hide

```
library(tidyverse)
```

```
── Attaching packages ────────────────────────────────────────────────────────────────
──────────────────────────────────────────── tidyverse 1.3.1 ──
✓ ggplot2 3.3.5     ✓ purrr   0.3.4
✓ tibble  3.1.6     ✓ dplyr   1.0.7
✓ tidyr   1.1.4     ✓ stringr 1.4.0
✓ readr   2.1.1     ✓ forcats 0.5.1
── Conflicts ─────────────────────────────────────────────────────────────────────────
──────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Hide

```
#Exercise 1
Mind <- datind2009.csv

#1.1

cor(Mind$wage, Mind$age, use = "complete.obs") #I have excluded NA, the result is
-0.1788512
```

```
[1] -0.1788512
```

Hide

```
#1.2
Mind2 <- Mind[!is.na(Mind$age) & !is.na(Mind$wage),] #clear obs with NA wage and N
A age
X <- Mind2 %>%
  select(age) %>%
  mutate(intercept = 1) %>%
  as.matrix()

Beta <- solve(t(X) %*% X) %*% t(X) %*% Mind2$wage
as.numeric(Beta) #the coefficient of age is -180.1765, and the intercept is 22075.
1066
```

```
[1]  -180.1765 22075.1066
```

Hide

```
#1.3
#I estimate the variance of error and than use it to compute standard error
error <- Mind2$wage - (Beta[1,1]) * Mind2$age - Beta[2,1] #Beta is a matrix, the f
irst row is age coefficient and the second is the intercept

ErrorSquare <- t(error) %*% error/(length(Mind2$wage)-1) #-1 since we have one reg
ressor

VarBeta <- ErrorSquare %*% solve(t(Mind2$age) %*% Mind2$age)
VarBeta # 6.500652
```

```
         [,1]
[1,] 6.500652
```

Hide

```
#Using bootstrap 49
Results <- mat.or.vec(49, 2) #since we only have one coefficient to estimate

set.seed(99713)

for (i in 1:49){
  sam <- sample(1:length(Mind2$wage), length(Mind2$wage), rep = TRUE) #rep = TRUE
to give weight to random index
  boot <- Mind2[sam,] #randomly select observations
  X <- boot %>%
    select(age) %>%
    mutate(intercept = 1) %>%
    as.matrix()
  Beta <- solve(t(X) %*% X) %*% t(X) %*% boot$wage
  Results[i,] <- Beta
}
Results <- as.data.frame(Results)
lapply(Results, mean) #beta age: -178.5496, beta intercept: 21975.63  Note that V1
is the age coefficient and V2 is the intercept
```

```
$V1
[1] -178.5496

$V2
[1] 21975.63
```

Hide

```
lapply(Results, sd) #age sd: 5.249532, intercept sd: 291.9666
```

```
$V1
[1] 5.249532

$V2
[1] 291.9666
```

Hide

```
#Using bootstrap 499

Results2 <- mat.or.vec(499, 2)

for (i in 1:499){
  sam <- sample(1:length(Mind2$wage), length(Mind2$wage), rep = TRUE)
  boot <- Mind2[sam,]
  X <- boot %>%
    select(age) %>%
    mutate(intercept = 1) %>%
    as.matrix()
  Beta <- solve(t(X) %*% X) %*% t(X) %*% boot$wage
  Results2[i,] <- Beta
}
Results2 <- as.data.frame(Results2)
lapply(Results2, mean) # beta age: -180.5353, beta intercept: 22095.49. Note that V1 is the age coefficient and V2 is the intercept
```

```
$V1
[1] -180.5353

$V2
[1] 22095.49
```

Hide

```
lapply(Results2, sd) #age sd: 5.328217, intercept sd: 301.4035
```

```
$V1
[1] 5.328217

$V2
[1] 301.4035
```

Hide

```
#Using bootstrap gives me a higher standard error.
#While the coefficient (estimated by mean in bootstrap) is nearly the same,
#the result of doing 499 times is closer to matrix form solution than doing 49 tim
es
```

Hide

```
#Exercise 2
dind = list.files(pattern="datind")
for (i in 1:length(dind)) assign(dind[i], read_csv(dind[i]))
```

```
New names:
* `` -> ...1
Warning: One or more parsing issues, see `problems()` for details
```

```
Rows: 22144 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 24241 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (3): empstat, profession, gender
dbl (7): ...1, idind, idmen, year, respondent, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
Warning: One or more parsing issues, see `problems()` for details
```

```
Rows: 24940 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 25907 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 25510 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 25611 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26531 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 27071 Columns: 10
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 28534 Columns: 10
── Column specification ────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────

Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26353 Columns: 10
── Column specification ────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────

Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26787 Columns: 10
── Column specification ────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────

Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26644 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26647 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 25402 Columns: 10
── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 24698 Columns: 10
── Column specification ────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
New names:
* `` -> ...1
```

```
Rows: 26484 Columns: 10
── Column specification ────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
Delimiter: ","
chr (2): empstat, gender
dbl (8): ...1, idind, idmen, year, respondent, profession, age, wage

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
Mind3 <- do.call("rbind", list(datind2005.csv, datind2006.csv, datind2007.csv, dat
ind2008.csv, datind2009.csv,
                              datind2010.csv, datind2011.csv, datind2012.csv, dati
nd2013.csv, datind2014.csv, datind2015.csv,
                              datind2016.csv, datind2017.csv, datind2018.csv))
#2.1

Mind4 <- Mind3 %>%
  mutate(ag = case_when(age >= 18 & age <= 25 ~ 1,
         age >= 26 & age <= 30 ~ 2,
         age >= 31 & age <= 35 ~ 3,
         age >= 36 & age <= 40 ~ 4,
         age >= 41 & age <= 45 ~ 5,
         age >= 46 & age <= 50 ~ 6,
         age >= 51 & age <= 55 ~ 7,
         age >= 56 & age <= 60 ~ 8,
         age >  60  ~ 9,
         age <  18  ~ 0 ))
```

Hide

```
for (i in 0:9){
posiwage <- Mind4 %>%
  filter(wage != 0) %>%
  filter(ag == i)

plot<- ggplot(posiwage, aes(x = as.factor(year), y = wage, color = as.factor(year)
)) +
  geom_boxplot() + scale_y_continuous(name="wage", limits=c(0, 50000))
ggsave(paste("age_group_", i, ".pdf", sep = ""))
plot
}
```

```
Saving 7 x 7 in image
Warning: Removed 3 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 116 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 206 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 791 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 1333 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 2072 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 2214 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 2109 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 1748 rows containing non-finite values (stat_boxplot).
Saving 7 x 7 in image
Warning: Removed 867 rows containing non-finite values (stat_boxplot).
```

Hide

```
#clear NA
Mind5 <- Mind3[!is.na(Mind3$age) & !is.na(Mind3$wage) & !is.na(Mind3$year),]



#using lm
reg1 <- lm(wage ~ age + as.factor(year), data = Mind5)
summary(reg1)$coefficients
```

```
                      Estimate Std. Error     t value     Pr(>|t|)
(Intercept)         20675.05832 174.535606 118.4575389 0.000000e+00
age                  -186.87927   2.001569 -93.3663738 0.000000e+00
as.factor(year)2006    21.93723 206.900079   0.1060281 9.155601e-01
as.factor(year)2007   294.80257 204.758678   1.4397562 1.499375e-01
as.factor(year)2008  1425.19060 205.327803   6.9410502 3.900078e-12
as.factor(year)2009  1720.36049 205.075379   8.3889178 4.927979e-17
as.factor(year)2010  1869.52505 203.141747   9.2030569 3.501191e-20
as.factor(year)2011  2116.01760 202.051417  10.4726690 1.165692e-25
as.factor(year)2012  2601.22748 199.589059  13.0329162 8.153676e-39
as.factor(year)2013  2478.84340 203.356713  12.1896315 3.598927e-34
as.factor(year)2014  2749.67501 202.408468  13.5847825 5.078509e-42
as.factor(year)2015  3120.96921 202.710060  15.3962226 1.822024e-53
as.factor(year)2016  3410.11335 202.643067  16.8281768 1.626422e-63
as.factor(year)2017  3479.03189 204.645439  17.0002904 8.785992e-65
as.factor(year)2018  3636.15153 205.928835  17.6573210 9.724886e-70
```

Hide

```
#the age coefficient is  -186.87927

#using matrix
X <- Mind5%>%
  select(age, year)%>%
  mutate(intercept = 1) %>%
  as.matrix()

solve(t(X) %*% X) %*% t(X) %*% Mind5$wage #age coefficient: -186.8827
```

```
                  [,1]
age          -186.8827
year          290.9967
intercept -562591.9400
```

Hide

```
#using plm
install.packages("plm")
```

```
trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-arm64/contrib/4.1/plm_2.4-
3.tgz'
Content type 'application/x-gzip' length 2137849 bytes (2.0 MB)
==================================================
downloaded 2.0 MB
```

```
The downloaded binary packages are in
    /var/folders/3z/2_2kvxs57q38w9ppyslh0thm0000gn/T//RtmpBjMdLN/downloaded_packag
es
```

Hide

```
library(plm)
```

```
Attaching package: ‘plm’

The following objects are masked from ‘package:dplyr’:

    between, lag, lead
```

Hide

```
reg2 <- plm(wage ~ age, data = Mind5, model = "within", index = "year")
summary(reg2) #-186.8793
```

```
Oneway (individual) effect Within Model

Call:
plm(formula = wage ~ age, data = Mind5, model = "within", index = "year")

Unbalanced Panel: n = 14, T = 18767-22742, N = 289769

Residuals:
     Min.   1st Qu.    Median   3rd Qu.      Max.
 -21321.1  -11464.5   -7266.0    8496.3 1738140.9

Coefficients:
     Estimate Std. Error t-value  Pr(>|t|)
age -186.8793     2.0016 -93.366 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Total Sum of Squares:    1.2219e+14
Residual Sum of Squares: 1.1862e+14
R-Squared:      0.029206
Adj. R-Squared: 0.02916
F-statistic: 8717.28 on 1 and 289754 DF, p-value: < 2.22e-16
```

Hide

```
#Compare this to model without time fixed effect

compare <- lm(wage ~ age, data = Mind5)
summary(compare)$coefficients #age coefficient is -182.4896
```

```
              Estimate Std. Error   t value Pr(>|t|)
(Intercept) 22559.2993 104.318126 216.25484        0
age          -182.4896   1.999791 -91.25433        0
```

Hide

```
time <- 100
results3 <- mat.or.vec(time, 3)
#I only need to calculate the intercept and age coef, and I add one more column to
collect minimizing value

for (i in 1:time) {

  searchv = runif(2, -5, 5) #random starting search value
  result  = optim(searchv, fn = ProLike, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  age = age, empstat = empstat)
  results3[i,] = c(result$par, result$value) #I collect the estimated minimizing p
arameter and the minimizing value
}
```

```
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
final  value 14810.237786
converged
initial  value 14810.237786
```

Hide

```
# This time, I get the estimated intercept: 1.042158817, estimated age coefficient
: 0.00697, with the minimizing value 3555.8917
# Notes that the number of the coefficient can not be interpreted directly in prob
it model.
# We can only say age has a positive effect on market participation without contro
lling other factors.
```

Hide

```
#3.4
wage <- df$wage
age <- df$age
empstat <- as.numeric(df$empstat)

wage<- wage[which(!is.na(wage) ) ] #clear out NA
age <- age[which (!is.na(wage) ) ]
empstat <- empstat[which(!is.na(wage)) ]

length(wage) #check if I have the same dimension
```

```
[1] 11522
```

Hide

```
length(age)
```

```
[1] 11522
```

Hide

```
length(empstat)
```

```
[1] 11522
```

Hide

```
ProLike2 <- function(cf, age, wage, empstat){
  XB = cf[1] + cf[2]*age + cf[3]*wage
  Prob = pnorm(XB)
  Prob[Prob>0.999999] = 0.999999
  Prob[Prob<0.000001] = 0.000001
  p1 = log(Prob)   #represent the log prob of (y=1)
  p0 = log(1-Prob) #represent the log prob of (y=0)
  like = empstat * p1 + (1-empstat) * p0
  #By this method, if empstat = 1, then only the first term will be computed; othe
rwise, the second term will be computed
  return( -sum(like) ) #return negative for us to do maximization via minimizing t
he negative version
}
```

Hide

```
time <- 100
results4 <- mat.or.vec(time, 4) #I need to calculate the intercept, age and wage c
oefficients plus the minimizing value for later use

for (i in 1:time) {

  searchv = runif(3, -5, 5) #random starting search value
  result  = optim(searchv, fn = ProLike2, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  age = age, wage = wage, empstat = empstat)
  results4[i,] = c(result$par, result$value) #I collect the estimated minimizing p
arameter and the minimizing value
}
```

```
initial  value 127061.252927
final  value 127061.252927
converged
initial  value 144399.717422
final  value 144399.717422
converged
initial  value 126827.903975
final  value 14782.606749
converged
initial  value 14782.606749
final  value 14782.606749
converged
initial  value 144399.717422
final  value 144399.717422
converged
initial  value 32093.440225
final  value 32093.440225
converged
initial  value 14782.606749
```

```
converged
initial  value 14782.606749
final  value 14782.606749
converged
initial  value 31922.067914
final  value 31913.842120
converged
initial  value 127157.961494
final  value 127157.961494
converged
initial  value 127088.884144
final  value 127088.883946
converged
initial  value 31931.657569
final  value 14782.606749
converged
initial  value 32026.060531
final  value 14782.606749
converged
initial  value 14782.606749
final  value 14782.606749
converged
initial  value 14782.606749
final  value 14782.606749
converged
initial  value 126453.376090
final  value 126453.371099
converged
initial  value 144399.717422
final  value 144399.717422
converged
```

Hide

```
results4 <- format(results4, scientific = F) #By this I turn the scientific notati
on to full numbers
results4 <- as.data.frame(results4)

results4[ which(results4$V4 == min(results4$V4)), ]  #The age coefficient is 0.006
815962, and the wage coefficient is 2.475465594
```

	V1 <chr>	V2 <chr>	V3 <chr>	V4 <chr>
65	1.039405928	0.006849504	3.880669144	13268.748016375

1 row

Hide

```
# Exercise 4
M5 <- do.call("rbind", list(datind2005.csv, datind2006.csv, datind2007.csv,
                            datind2008.csv, datind2009.csv, datind2010.csv,
                            datind2011.csv, datind2012.csv, datind2013.csv, datind
2014.csv, datind2015.csv))

M5 <- M5 %>%
  filter(empstat != "Inactive", empstat != "Retired")


#4.2
M5$empstat[ M5$empstat == "Employed" ] = 1 #make dummies for empstat
M5$empstat[ M5$empstat == "Unemployed" ] = 0

testM5 <- M5 %>%
  mutate(dum = 1) %>%
  pivot_wider(names_from = year, values_from = dum, values_fill = 0) #make year du
mmies

M6 <- testM5 %>%
  select(-"2005") # I drop 2005 to avoid perfect collinearity


age <- M6$age

empstat <- as.numeric(M6$empstat)

which(is.na(age)) #no NA
```

```
integer(0)
```

Hide

```
which(is.na(empstat)) #no NA
```

```
integer(0)
```

Hide

```
y6 <- M6$`2006`
y7 <- M6$`2007`
y8 <- M6$`2008`
y9 <- M6$`2009`
y10 <- M6$`2010`
y11 <- M6$`2011`
y12 <- M6$`2012`
y13 <- M6$`2013`
y14 <- M6$`2014`
y15 <- M6$`2015`
```

Hide

```
# ========= Probit ==============

ProLike3 <- function(cf, age, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, empsta
t){
  XB = cf[1] + cf[2]*age + cf[3]*y6 + cf[4] *y7 + cf[5]*y8 + cf[6]*y9 + cf[7]*y10
+ cf[8]*y11 + cf[9]*y12 + cf[10]*y13 + cf[11]*y14 + cf[12]*y15
  Prob = pnorm(XB)
  Prob[Prob>0.999999] = 0.999999
  Prob[Prob<0.000001] = 0.000001
  p1 = log(Prob)
  p0 = log(1-Prob)
  like = empstat * p1 + (1-empstat) * p0
  return( -sum(like) )
}

time <- 100
results5 <- mat.or.vec(time, 12+1)
#I only need to calculate the intercept and age coef, and I add one more column to
collect minimizing value

for (i in 1:time) {

  searchv = runif(12, -5, 5) #random starting search value
  result  = optim(searchv, fn = ProLike3, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  age = age, y6 = y6, y7 = y7, y8 = y8, y9 = y9, y10 = y10,
                  y11 = y11, y12 = y12, y13 = y13, y14 = y14, y15 = y15,
                  empstat = empstat)

  results5[i,] = c(result$par, result$value)
}
```

```
initial  value 183608.250662
final  value 183608.250662
converged
initial  value 183608.250662
```

```
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 183608.250662
final  value 183608.250662
converged
initial  value 183608.250662
final  value 183608.250662
converged
initial  value 155048.482353
iter  10 value 87178.938979
iter  20 value 55772.307591
iter  30 value 42243.660141
iter  30 value 42243.660141
final  value 42243.658904
converged
```

Hide

```
results5 <- format(results5, scientific = F) #By this I turn the scientific notati
on to full numbers
results5 <- as.data.frame(results5)
results5[ which(results5$V13 == min(results5$V13)), ]
```

	V1 <chr>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>	V7 <chr>
100	0.74868244	0.01232911	0.01718713	0.08074294	0.10985885	0.02661328	0.02189773

1 row | 1-9 of 13 columns

Hide

```
ProRes <- results5[ which(results5$V13 == min(results5$V13)), ] #I store the resul
t
ProRes <- as.numeric( as.vector(t(ProRes)) )
ProRes <- ProRes[-length(ProRes)] #The last value is the minimizing negative logli
kelihood, and I don't need it afterward


# After controlling for year, I got  0.012329112 as the probit "age" coefficient.
The intercept is  0.748818912
```

Hide

```
#Here I tried to produce hessian but I failed
#searchv = runif(12) #random starting search value
#result  = optim(searchv, fn = ProLike3, method = "BFGS",
#                 control = list(trace = 6, REPORT = 1, maxit = 9000),
#                 age = age, y6 = y6, y7 = y7, y8 = y8, y9 = y9, y10 = y10,
#                 y11 = y11, y12 = y12, y13 = y13, y14 = y14, y15 = y15,
#                 empstat = empstat, hessian = TRUE)
#result$hessian

#!!!!!!!!!!!! return 0 hessian matrix
#result$value

#fisher = solve(result$hessian)
#se = sqrt( diag(fisher) )
#se[2] #Here I specify "2" to get the second standard error, which is my standard
error for age
```

Hide

```
LogitLike <- function(cf, age, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, empst
at){
  XB = cf[1] + cf[2]*age + cf[3]*y6 + cf[4] *y7 + cf[5]*y8 + cf[6]*y9 +
    cf[7]*y10 + cf[8]*y11 + cf[9]*y12 + cf[10]*y13 + cf[11]*y14 + cf[12]*y15
  Prob = exp(XB)/ (1 +exp(XB))
  Prob[Prob>0.999999] = 0.999999
  Prob[Prob<0.000001] = 0.000001
  p1 = log(Prob)
  p0 = log(1-Prob)
  like = empstat * p1 + (1-empstat) * p0
  return( -sum(like) )
}

time <- 100
results6 <- mat.or.vec(time, 12+1)


for (i in 1:time) {

  searchv = runif(12, -5, 5) #random starting search value
  result  = optim(searchv, fn = LogitLike, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  age = age, y6 = y6, y7 = y7, y8 = y8, y9 = y9, y10 = y10,
                  y11 = y11, y12 = y12, y13 = y13, y14 = y14, y15 = y15,
                  empstat = empstat)

  results6[i,] = c(result$par, result$value)
}
```

```
converged
initial  value 1265316.327249
final  value 183608.250662
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 183608.250662
final  value 183608.250662
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
initial  value 183608.250662
final  value 183608.250662
converged
initial  value 1593563.894109
final  value 1593563.894109
converged
```

Hide

```
results6 <- format(results6, scientific = F) #By this I turn the scientific notati
on to full numbers
results6 <- as.data.frame(results6)

results6[ which(results6$V13 == min(results6$V13)), ]
```

	V1 <chr>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>	V7 <chr>
21	1.120026111	0.025313596	0.031771019	0.157492273	0.212685190	0.045560425	0.0375

1 row | 1-8 of 13 columns

Hide

```
LogitRes <- t( results6[ which(results6$V13 == min(results6$V13)), ] )
LogitRes <- as.numeric ( as.vector(LogitRes) )
LogitRes <- LogitRes[-length(LogitRes)]


#Here the coefficient for age is  0.025307485, and for the intercept is 1.12044937
2.
#Same with probit, we cannot interpret the face value of coefficient. We can just
say that age is positively correlated with employment.
```

Hide

```
# ========= Linear ==============

LnLike <- function(cf, age, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, empstat)
{
  XB = cf[1] + cf[2]*age + cf[3]*y6 + cf[4] *y7 + cf[5]*y8 + cf[6]*y9 +
    cf[7]*y10 + cf[8]*y11 + cf[9]*y12 + cf[10]*y13 + cf[11]*y14 + cf[12]*y15
  Prob = XB
  Prob[Prob>0.999999] = 0.999999
  Prob[Prob<0.000001] = 0.000001
  # Note that I did  force the probability to be between [0,1], but Prob = XB can
be inherently bigger than 1 or smaller than 0
  p1 = log(Prob)
  p0 = log(1-Prob)
  like = empstat * p1 + (1-empstat) * p0
  return( -sum(like) )
}

time <- 100
results7 <- mat.or.vec(time, 12+1)


for (i in 1:time) {

  searchv = runif(12, -5, 5) #random starting search value
  result  = optim(searchv, fn = LnLike, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  age = age, y6 = y6, y7 = y7, y8 = y8, y9 = y9, y10 = y10,
                  y11 = y11, y12 = y12, y13 = y13, y14 = y14, y15 = y15,
                  empstat = empstat)

  results7[i,] = c(result$par, result$value)
}
```

```
initial  value 183608.250662
final  value 183608.250662
converged
```

```
results7 <- format(results7, scientific = F)
results7 <- as.data.frame(results7)

results7[ which(results7$V13 == min(results7$V13)), ]
```

	V1 <chr>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>
1	1.11468968	1.27951165	1.37091177	-4.34164052	0.31826173	-0.95300552
2	0.76827936	4.55536990	-1.97953293	-2.18581304	-3.01391464	3.85060381
6	-3.48389898	0.77382353	-0.25364508	-2.29617071	-2.81977456	-2.65077414
9	2.63522930	4.01696836	-4.90187781	0.38507133	-2.97030635	-4.74124418
14	-3.17104172	1.89144611	3.14392670	3.47678480	-3.29191688	-4.14932820
19	1.81601888	4.30128571	-3.02068016	0.79384470	3.29295399	-2.37529039
21	-1.26312901	4.31874610	-3.91388761	-4.27326023	1.37278161	-4.69478592
22	-4.38616076	1.82814226	4.91262506	-0.10939945	3.56308153	-3.19284005
23	140.49305508	8855.31830583	-2.68378733	-3.50614744	10.30327383	-3.20001794
24	-3.77451309	4.82216000	1.57069871	-3.95470109	2.61762258	0.80649350

1-10 of 49 rows | 1-8 of 13 columns Previous **1** 2 3 4 5 Next

Hide

```
LnRes <- t (results7[ which(results7$V13 == min(results7$V13)), ] )
LnRes <- as.vector(as.numeric(LnRes))
LnRes <- LnRes[-length(LnRes)]


# the estimate intercept is 1.971644518, and the estimated age coefficient is 0.10
5708194
#This means an age older is associated with 0.105 proportionate increase in employ
ment probability

#Unfortunately, because I always produce zero hessian matrix in optim(), I could n
ot report the standard error and the significance level
```

Hide

```
# 5.1

Intercept <- rep(1, length(age))
Xmat <- cbind(Intercept, age, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15)
dim(Xmat)
```

```
[1] 128636     12
```

Hide

```
#  ======= Probit ========

predictPro <- Xmat %*% ProRes # X matrix times beta
marginalPro <- mean( dnorm(predictPro)) * ProRes
marginalPro #note that the first is intercept, the second is coefficient of age
```

```
 [1]  0.133160220  0.002192848  0.003056893  0.014360892  0.019539431  0.004733422
0.003894717  0.009831213  0.001832989 -0.007060174
[11] -0.005882981 -0.009450188
```

Hide

```
#  ======= Logit ========

predictLog <- Xmat %*% LogitRes
#how to uncover marginal effect of logit function?

pdf <- exp(predictLog)/(1 + exp(predictLog))
marginalLog <- mean(pdf)* LogitRes #is this still cdf? Then, what is its pdf?
marginalLog
```

```
 [1]  1.00428816  0.02269781  0.02848796  0.14121780  0.19070736  0.04085244  0.03
369867  0.09071787  0.01073038 -0.07631406 -0.06395258
[12] -0.09985407
```

Hide

```
mean(dlogis(predictLog)) * LogitRes #this is closer, but not entirely the same
```

```
 [1]  0.102898611  0.002325601  0.002918855  0.014469070  0.019539732  0.004185710
0.003452741  0.009294885  0.001099426 -0.007819081
[11] -0.006552533 -0.010230973
```

Hide

```
#Failed attempt with probit boot

#for (i in 1:49){
#  sam <- sample(1:nrow(Fullmat), nrow(Fullmat), rep = TRUE) #rep = TRUE to give w
eight to random index
#  boot <- Fullmat[sam,]
#  age <- boot$age
#  empstat <- as.numeric(boot$empstat)

#  y6 <- boot$y6
#  y7 <- boot$y7
#  y8 <- boot$y8
#  y9 <- boot$y9
#  y10 <- boot$y10
#  y11 <- boot$y11
#  y12 <- boot$y12
#  y13 <- boot$y13
#  y14 <- boot$y14
#  y15 <- boot$y15

  #here I complete the data process

#  time <- 50 #here I only do 50 times
#  resBpro <- mat.or.vec(time, 12+1)

#  for (j in 1:time) {
#    searchv = runif(12, -5, 5) #random starting search value
#    result  = optim(searchv, fn = ProLike3, method = "BFGS",
#                    control = list(trace = 6, maxit = 1000),
#                    age = age, y6 = y6, y7 = y7, y8 = y8, y9 = y9, y10 = y10,
#                    y11 = y11, y12 = y12, y13 = y13, y14 = y14, y15 = y15,
#                    empstat = empstat)

#    resBpro[j,] = c(result$par, result$value)
#  }
#  resBpro <- format(resBpro, scientific = F)
#  resBpro <- as.data.frame(resBpro)

#  resBpro <- resBpro[ which(resBpro$V13 == min(resBpro$V13)), ]
#  resBpro <- as.numeric( as.vector(t(resBpro)) )
#  resBpro <- resBpro[-length(resBpro)]

#  Intercept <- rep(1, length(age))
#  Xmat <- cbind(Intercept, age, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15)

  #Error in Xmat %*% resBpro : non-conformable arguments
#  predictPro_b <- Xmat %*% as.matrix(resBpro)
#  marginalPro_b <- mean( dnorm(predictPro_b)) * resBpro

#  bootmarg[i,] <- marginalPro_b
```

```
#  }
```

Hide

```
apply(Bmix, MARGIN = 1, sd)
```

```
 [1] 0.0062172292 0.0001160997 0.2099995629 0.0036329970 0.1994521134 0.0040779691
0.1286183428 0.1122085614 0.0040960827 0.7712256396
[11] 0.0041474878 0.0032604732
```

Hide

```
trytimes <- 1:4
LogBootRes <-  sapply(trytimes, logtry)
LogBootRes2 <- sapply(trytimes, logtry)
LogBootRes3 <- sapply(trytimes, logtry)
LogBootRes4 <- sapply(trytimes, logtry)
LogBootRes5 <- sapply(trytimes, logtry)

Bmix2 <- do.call("cbind", list(LogBootRes, LogBootRes2, LogBootRes3, LogBootRes4,
LogBootRes5))
apply(Bmix2, MARGIN = 1, mean)
apply(Bmix2, MARGIN = 1, sd)
#Standard error of intercept: 4.809159e-03
#age: 9.572158e-05
#2006: 4.641618e-03
#2007: 5.000660e-03
#2008: 4.366556e-03
#2009: 4.576084e-03
#2010: 4.747205e-03
#2011: 3.896287e-03
#2012: 3.459685e-03
#2013: 455618e-03
#2014:3.790857e-03
#2015:281763e-03
```