Introduction to R in Econometrics

author: Oystein Myrland autosize: true width: 1920 height: 1080

Econometric software provide one of two alternative methods for users

– using a command language
– using a "point and click" approach

Avoid using point-and-click software in econometrics

- some find the command language complicated and chose to go with the point and click method
- replicating quantitative study findings is a necessary and desirable aspect of science
- replication in economics
- About 40% of economics experiments fail replication survey
- Replication, Replication
- Why so much science research is flawed - and what to do about it

Provide information for others to replicate your results

- 'raw data' are likely transformed during the data analysis period
- this is kept track of in a command language
- difficult to achieve using a point-and-click method
- once learned, it saves time
- change an element of the program, compared to an entire mouse sequence with every change
- once written it is easy to communicate code to other researchers and co-authors
- save for future studies and to use with different data sets ("future self")

Flexibility

- point-and-click methods are usually limited to 'precanned' applications
- what to do if 'button' for needed analysis is not made?
- many procedures are available using syntax which are not possible using point-and-click alone
- a program language provides more options for data analysis and manipulation
- program languages usually work across various versions of the software

Introducing R

- R is a open source programming environment

- R is a programming language/software
- allows for development of functions
- functions are distributed as packages/libraries
- any user can download and use packages to enhance the enviRonment

Base R and packages

- Base R and most R packages are available for download from the Comprehensive R Archive Network (CRAN)

- Base R comes with a number of basic data management, analysis, and graphical tools
- R's power and flexibility, lie in its array of packages, currently:

```
local({r <- getOption("repos")
r["CRAN"] <- "http://cran.r-project.org"
       options(repos=r) })
length(unique(rownames(available.packages())))
```

```
[1] 16049
```

Downloading and Installing R (locally)

Find files for your OS at:

http://www.r-project.org

Download

Interacting with R

- You can work directly in R, but most users prefer a graphical user interface (GUI)
- Recommend RStudio, an Integrated Development Environment (IDE)

http://www.rstudio.com

Download for local installation

- Through web-browser, use UiT username/password

http://rstudio.uit.no

The following packages are very useful

- **pacman** package for installing and attaching packages
- **mosaic** package for statistics and mathematics teaching utilities

- *tidyverse* (includes e.g.)
  - **dplyr** package for various data management tasks
  - **tidyr** package for making tidy data
  - **ggplot2** package for data visualization using the Grammar of Graphics

- **broom** package takes output and turns it into tidy data frames
- **stargazer** package for formatted output

R code

- R code can be entered into the command line (console) directly or saved to a script/snippet
- Commands are separated either by a ; or by a new line
- **R is case sensitive!**
- The # character means "a comment", and is not executed

Installing Packages (base R)

To use packages in R, we must first install them using the *install.packages()* function, which downloads and installs the package

```
install.packages("mosaic")
```

Note the use of: " " around the package name

Loading Packages (base R)

After installing (you only install once)
If you need a particular function in a package for your current R session, you must first load it into the R environment using the **library** or **require** function

```
library(mosaic)
# or
require(mosaic)
```

Note that the " " are gone!

Installing and loading packages using pacman

- **pacman** package has a joint install and library/require function in `p_load()`

```
install.packages("pacman") # first time
require(pacman)
```

once **pacman** is loaded, use only:

```
p_load(mosaic)
```

- for several packages, just separate them with commas

```
p_load(mosaic, dplyr, car)
```

Installing and loading packages using RStudio

RStudio has its own software panel with package management - install - library/require - update - search

R session info

R version and the attached packages used in the current session

```
sessionInfo()
```

```
R version 4.0.2 (2020-06-22)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18363)

Matrix products: default

locale:
[1] LC_COLLATE=English_Europe.1252  LC_CTYPE=English_Europe.1252
[3] LC_MONETARY=English_Europe.1252 LC_NUMERIC=C
[5] LC_TIME=English_Europe.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] knitr_1.28

loaded via a namespace (and not attached):
[1] compiler_4.0.2 magrittr_1.5   tools_4.0.2    stringi_1.4.6  stringr_1.4.0
[6] xfun_0.13      evaluate_0.14
```

Help

Help is accessed by preceding the name of the function with ? (e.g. ?c)

??keyword searches R documentation for keyword (e.g. ??regression)

- Or use RStudio's help pane

Objects

- R stores both data and output from data analysis (as well as everything else) in objects
- Things are assigned to and stored in objects using the <- or = operator

A list of all objects in the current session can be obtained with `ls()`

```
ls()
```

```
character(0)
```

- In RStudio, just look at the *Environment* panel

Example

assign the number 3 to a object called *a* using assignment <-

```
a <- 3
ls()
```

```
[1] "a"
```

```
a
```

```
[1] 3
```

```
a+2
```

```
[1] 5
```

Basic R function style

verb is a function name (what we would like to do)
it is enclosed with parenthesis
inside the () all options are separated by commas

```
verb(object, data=nameof, ...) # ... means options
verb(y ~ x, data=nameof, ...) # ~ means function of
verb(y ~ x | z, data=nameof, ...) # | means condition of
```

What do you want R to do? (verb)

- This determines the R function to use
- What must R know to do that?
- This determines the inputs to the function
- Must identify the variables and data frame

Example with built in data

Data on number of births in the US (1978)

```
require(pacman)
p_load(mosaic, mosaicData)
data(Births78, package="mosaicData")
names(Births78)
```

```
[1] "date"         "births"       "wday"         "year"         "month"
[6] "day_of_year"  "day_of_month" "day_of_week"
```

Births scatterplot

```
xyplot(births ~ date, data=Births78)
```

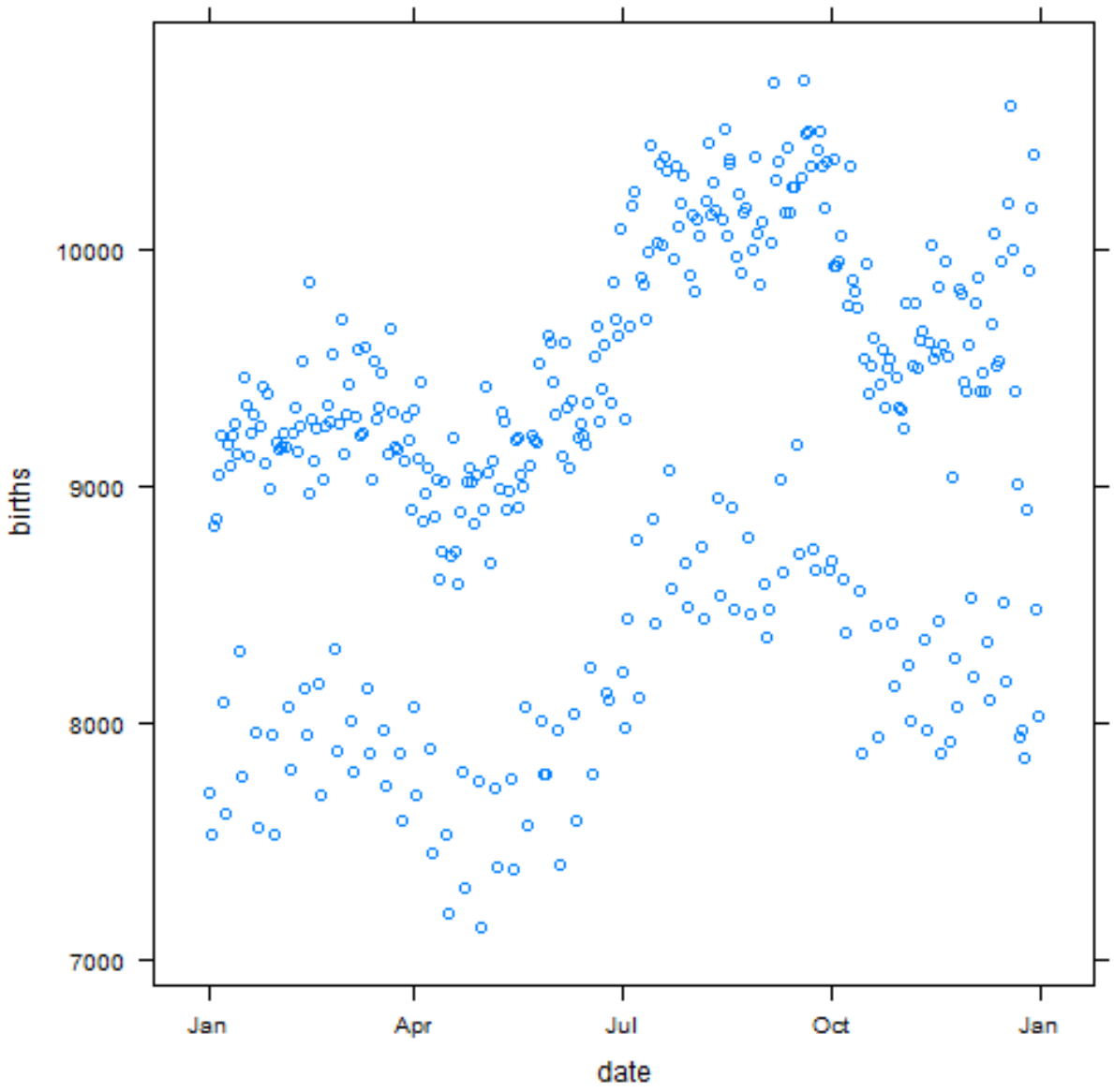


Figure 1: plot of chunk unnamed-chunk-12

Births conditioned scatterplot and mean

```
xyplot(births ~ date | wday, data=Births78)
```

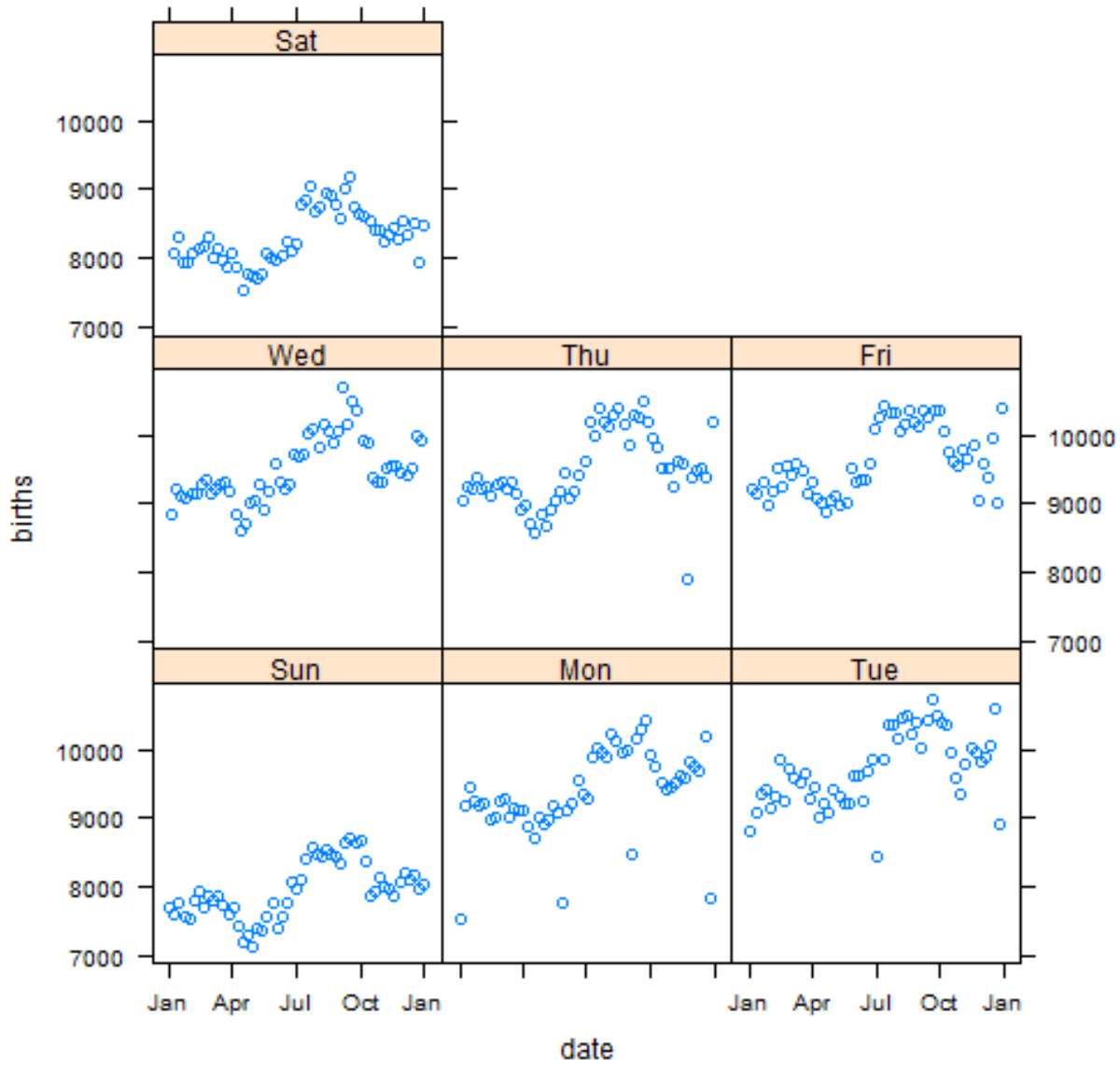


Figure 2: plot of chunk unnamed-chunk-13

```
mean(~births, data=Births78)
```

```
[1] 9132.162
```

```
mean(~births | wday, data=Births78)
```

```
     Sun      Mon      Tue      Wed      Thu      Fri      Sat
7950.943 9371.327 9708.808 9498.019 9483.635 9625.788 8309.327
```

```
mean(births ~ wday, data=Births78)
```

```
     Sun      Mon      Tue      Wed      Thu      Fri      Sat
7950.943 9371.327 9708.808 9498.019 9483.635 9625.788 8309.327
```

Entering Data

enter data into a vector x using the combine function `c()`

```
x <- c(1,2,4,6,7,9) ; x
```

```
[1] 1 2 4 6 7 9
```

Read Data

- R has several packages and interfaces to read all sorts of data
- In RStudio, use the Environment -> Import Dataset Tab

(tidy) Data file structure

R works most easily with data having the following structure:

- Each variable forms a column
- Each observation forms a row
- Each type of observational unit forms a table (matrix)

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

R can also retrieve files over the internet

Data definition file: http://www.principlesofeconometrics.com/poe5/data/def/food.def

```
load(url("http://www.principlesofeconometrics.com/poe5/data/rdata/food.rdata"))
head(food)
```

```
  food_exp income
1   115.22   3.69
2   135.98   4.39
3   119.34   4.75
4   114.96   6.03
5   187.05  12.47
6   243.92  12.98
```

Structure of objects

```
str(a)
```

```
 num 3
```

```
str(food)
```

```
'data.frame':   40 obs. of  2 variables:
 $ food_exp: num  115 136 119 115 187 ...
 $ income  : num  3.69 4.39 4.75 6.03 12.47 ...
 - attr(*, "var.labels")= chr [1:2] "household food expenditure per week" "weekly household income"
```

Viewing Data

R has several ways to look at a dataset at a glance

```
head(food, 2)
```

```
  food_exp income
1   115.22   3.69
2   135.98   4.39
```

```
tail(food, 2)
```

```
   food_exp income
39   257.95   29.4
40   375.73   33.4
```

Variable names

```
names(food)
```

```
[1] "food_exp" "income"
```

```
names(food) <- c("y","x")
names(food)
```

```
[1] "y" "x"
```

Data frame indexing

- individual rows, columns, and cells in a data frame can be accessed through many methods of indexing
- we most commonly use **object[row,column]** notation

```
head(food, 3) # first 3 rows
```

```
       y    x
1 115.22 3.69
2 135.98 4.39
3 119.34 4.75
```

```
food[2,1] # single cell value, 2nd row, 1st column
```

```
[1] 135.98
```

More variable indexing

We can also access variables directly by using their names, either with **object[, "variable"]** notation or **object$variable** notation

get first 7 rows of variable **x** using two methods

```
food[1:7, "x"]
```

```
[1]  3.69  4.39  4.75  6.03 12.47 12.98 14.20
```

```
food$x[1:7]
```

```
[1]  3.69  4.39  4.75  6.03 12.47 12.98 14.20
```

Combing values into a vector

The `c()` function is widely used to combine values of common type together to form a vector

For example, it can be used to access non-sequential rows and columns from a data frame.

```
# get rows 1, 3, 5 and 7-10 for column 1
food[c(1,3,5,7:10), 1]
```

```
[1] 115.22 119.34 187.05 267.43 238.71 295.94 317.78
```

Variable Names

to change one specific variable name, use indexing

```
names(food)[2] <- "var2"
names(food)
```

```
[1] "y"    "var2"
```

```
names(food) <- c("food_exp","income")
names(food)
```

```
[1] "food_exp" "income"
```

Avoid using special characters in variable names!

Working directory

```
getwd()
```

```
[1] "H:/poe5/h2020"
```

```
setwd("H:/data") # on laptop (local)
setwd("~/data") # on rstudio.uit.no (web)
```

use `dir()` to list files in directory

Saving Data

to save the data frame **df** in the current working directory as the file *data.rds*:

```
saveRDS(df, file="data.rds")
```

Then load it with:

```
df <- readRDS("data.rds")
```

Note that the data object now gets its original name `df`

Numerical Summaries

```
summary(food)
```

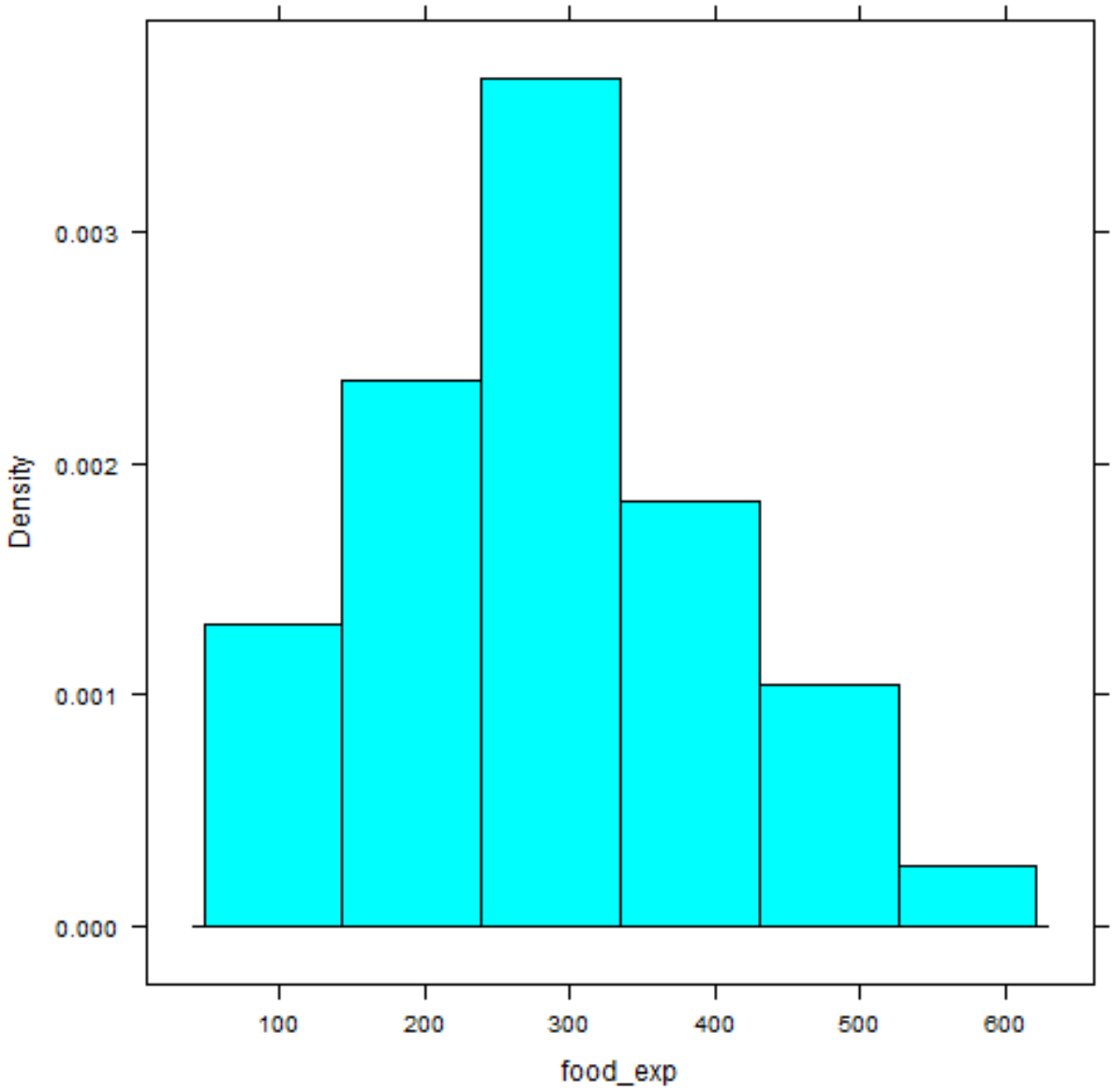
```
    food_exp         income
 Min.   :109.7   Min.   : 3.69
 1st Qu.:200.4   1st Qu.:17.11
 Median :264.5   Median :20.03
 Mean   :283.6   Mean   :19.60
 3rd Qu.:363.3   3rd Qu.:24.40
 Max.   :587.7   Max.   :33.40
```

Numerical Summaries: One Variable

```
mean(~food_exp, data=food)
```

```
[1] 283.5735
```

```
histogram(~food_exp, data=food)
```



Note that these functions require that you have loaded the `mosaic` package first.

Standard Normal probabilities

A random variable X is distributed normally as: $X \sim N(500, 100^2)$. Find $P(450 \leq X \leq 700)$:

```
xpnorm( c(450, 700), mean=500, sd=100)
```

```
[1] 0.3085375 0.9772499
```

Note that this functions require that you have loaded the `mosaic` package first.

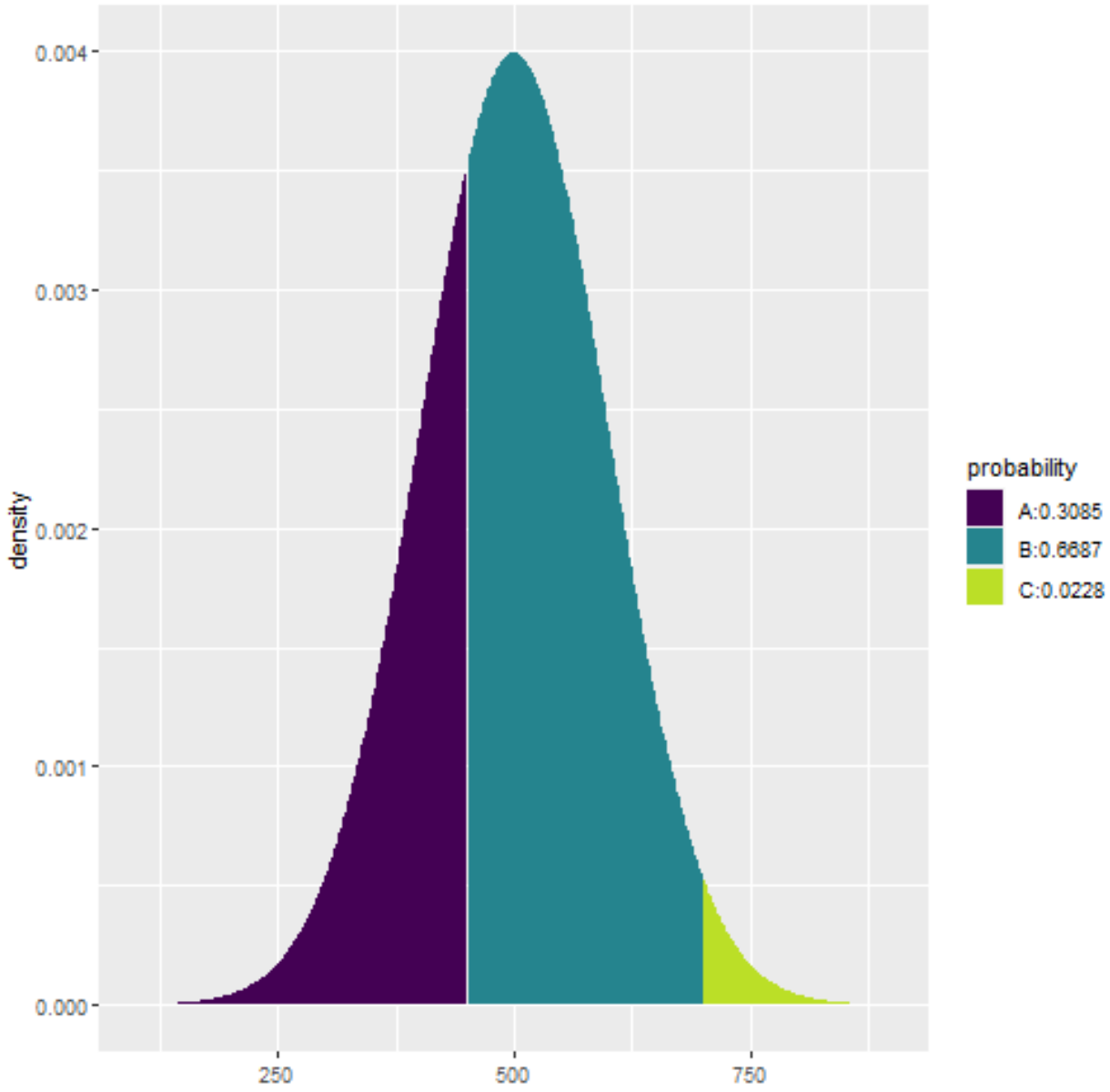


Figure 3: plot of chunk unnamed-chunk-29

Regression model (p. 65 in POE5)

```
m1 <- lm(food_exp~income, data=food)
p_load(stargazer)
stargazer(m1, type = "html", style = "aer", title="Output formatted as American Economic Review")
```

Output formatted as American Economic Review

food_exp

income

10.210***

(2.093)

Constant

83.416*

(43.410)

Observations

40

R2

0.385

Adjusted R2

0.369

Residual Std. Error

89.517 (df = 38)

F Statistic

23.789*** (df = 1; 38)

Notes:

***Significant at the 1 percent level.

**Significant at the 5 percent level.

*Significant at the 10 percent level.

Getting Help - Online

Google **r** & *what you would like to do in R*
e.g., r reference card
https://cran.r-project.org/doc/contrib/Short-refcard.pdf
rdocumentation.org
https://cran.r-project.org/web/packages/mosaic/vignettes/MinimalR.pdf http://stackoverflow.com/questions/tagged/r
Download: A Student's Guide to R

https://cran.r-project.org/other-docs.html
an extensive list of R books at http://www.r-project.org/doc/bib/R-books.html