Data Structures, Estimation and Optimization with R

Fan Wang

2021-06-29

Contents

Preface

This is a work-in-progress website consisting of R panel data and optimization examples for Statistics/Econometrics/Economic Analysis. Materials gathered from various projects in which R code is used. Files are from the R4Econ repository. This is not a R package, but a list of examples in PDF/HTML/Rmd formats. REconTools is a package that can be installed with tools used in projects involving R.

Bullet points show which base R, tidyverse or other functions/commands are used to achieve various objectives. An effort is made to use only base R (R Core Team, 2019) and tidyverse (Wickham, 2019) packages whenever possible to reduce dependencies. The goal of this repository is to make it easier to find/re-use codes produced for various projects. Some functions also rely on or correspond to functions from REconTools (Wang, 2020).

From other repositories: For dynamic borrowing and savings problems, see MEconTools and Dynamic Asset Repository; For code examples, see also Matlab Example Code, Stata Example Code, Python Example Code; For intro econ with Matlab, see Intro Mathematics for Economists, and for intro stat with R, see Intro Statistics for Undergraduates. See here for all of Fan's public repositories.

The site is built using Bookdown (Xie, 2020).

Please contact FanWangEcon for issues or problems.

Chapter 1

Array, Matrix, Dataframe

1.1 List

1.1.1 Lists

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

- r list tutorial
- r vector vs list
- r initialize empty multiple element list
- r name rows and columns of 2 dimensional list
- r row and colum names of list
- list dimnames
- r named list to string

1.1.1.1 Iteratively Build Up a List of Strings

Build up a list of strings, where the strings share common components. Iteratre over lists to generate variations in elements of the string list.

```
# common string components
st_base_name <- 'snwx_v_planner_docdense'
st_base_middle <- 'b1_xi0_manna_88'
# numeric values to loop over
ar_st_beta_val <- c('bt60', 'bt70', 'bt80', 'bt90')
ar_st_edu_type <- c('e1lm2', 'e2hm2')

# initialize string list
ls_snm <- vector(mode = "list", length = length(ar_st_beta_val)*length(ar_st_edu_type))

# generate list
it_ctr = 0
for (st_beta_val in ar_st_beta_val) {
  for (st_edu_type in ar_st_edu_type) {
    it_ctr = it_ctr + 1
    # snm_file_name <- 'snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt90'
    snm_file_name <- paste(st_base_name, st_edu_type, st_base_middle, st_beta_val, sep ='_')
    ls_snm[it_ctr] <- snm_file_name
  }
}

# print
for (snm in ls_snm) {
```

```
  print(snm)
}
```

```
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt60"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt60"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt70"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt70"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt80"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt80"
## [1] "snwx_v_planner_docdense_e1lm2_b1_xi0_manna_88_bt90"
## [1] "snwx_v_planner_docdense_e2hm2_b1_xi0_manna_88_bt90"
```

```
# if string in string
grepl('snwx_v_planner', snm)
```

```
## [1] TRUE
```

1.1.1.2 Named List of Matrixes

Save a list of matrixes. Retrieve Element of that list via loop.

```
# Define an array to loop over
ar_fl_mean <- c(10, 20, 30)

# store restuls in named list
ls_mt_res = vector(mode = "list", length = length(ar_fl_mean))
ar_st_names <- paste0('mean', ar_fl_mean)
names(ls_mt_res) <- ar_st_names

# Loop and generat a list of dataframes
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  fl_mean = ar_fl_mean[it_fl_mean]

  # dataframe
  set.seed(it_fl_mean)
  tb_combine <- as_tibble(
    matrix(rnorm(4,mean=fl_mean,sd=1), nrow=2, ncol=3)
    ) %>%
    rowid_to_column(var = "id") %>%
    rename_all(~c(c('id','var1','varb','vartheta')))

  ls_mt_res[[it_fl_mean]] = tb_combine
}

# Retrieve elements
print(ls_mt_res[[1]])
print(ls_mt_res$mean10)
print(ls_mt_res[['mean10']])

# Print via Loop
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  tb_combine = ls_mt_res[[it_fl_mean]]
  print(tb_combine)
}
```

1.1.1.3 One Dimensional Named List

1. define list
2. slice list
3. print r named list as a single line string

- R Unlist named list into one string with preserving list names

```
# Define Lists
ls_num <- list(1,2,3)
ls_str <- list('1','2','3')
ls_num_str <- list(1,2,'3')

# Named Lists
ar_st_names <- c('e1','e2','e3')
ls_num_str_named <- ls_num_str
names(ls_num_str_named) <- ar_st_names

# Add Element to Named List
ls_num_str_named$e4 <- 'this is added'
```

Initiate an empty list and add to it

```
# Initiate List
ls_abc <- vector(mode = "list", length = 0)
# Add Named Elements to List Sequentially
ls_abc$a = 1
ls_abc$b = 2
ls_abc$c = 'abc\'s third element'
# Get all Names Added to List
ar_st_list_names <- names(ls_abc)
# Print list in a loop
print(ls_abc)
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] "abc's third element"
```

```
for (it_list_ele_ctr in seq(1,length(ar_st_list_names))) {
  st_list_ele_name <- ar_st_list_names[it_list_ele_ctr]
  st_list_ele_val <- ls_abc[it_list_ele_ctr]
  print(paste0(st_list_ele_name,'=',st_list_ele_val))
}
```

```
## [1] "a=1"
## [1] "b=2"
## [1] "c=abc's third element"
```

1.1.1.4 Named List Print Function

- r print input as string
- r print parameter code as string
- How to convert variable (object) name into String

The function below ffi_lst2str is also a function in REconTools: ff_sup_lst2str.

```
# list to String printing function
ffi_lst2str <- function(ls_list, st_desc, bl_print=TRUE) {

  # string desc
  if(missing(st_desc)){
    st_desc <- deparse(substitute(ls_list))
  }
```

```
  # create string
  st_string_from_list = paste0(paste0(st_desc, ':'),
                               paste(names(ls_list), ls_list, sep="=", collapse=";" ))

  if (bl_print){
    print(st_string_from_list)
  }
}

# print full
ffi_lst2str(ls_num)
```

```
## [1] "ls_num:=1;=2;=3"
```

```
ffi_lst2str(ls_str)
```

```
## [1] "ls_str:=1;=2;=3"
```

```
ffi_lst2str(ls_num_str)
```

```
## [1] "ls_num_str:=1;=2;=3"
```

```
ffi_lst2str(ls_num_str_named)
```

```
## [1] "ls_num_str_named:e1=1;e2=2;e3=3;e4=this is added"
```

```
# print subset
ffi_lst2str(ls_num[2:3])
```

```
## [1] "ls_num[2:3]:=2;=3"
```

```
ffi_lst2str(ls_str[2:3])
```

```
## [1] "ls_str[2:3]:=2;=3"
```

```
ffi_lst2str(ls_num_str[2:4])
```

```
## [1] "ls_num_str[2:4]:=2;=3;=NULL"
```

```
ffi_lst2str(ls_num_str_named[c('e2','e3','e4')])
```

```
## [1] "ls_num_str_named[c(\"e2\", \"e3\", \"e4\")]:e2=2;e3=3;e4=this is added"
```

1.1.1.5 Two Dimensional Unnamed List

Generate a multiple dimensional list:

1. Initiate with an N element empty list
2. Reshape list to M by Q
3. Fill list elements
4. Get list element by row and column number

List allows for different data types to be stored together.

Note that element specific names in named list are not preserved when the list is reshaped to be two dimensional. Two dimensional list, however, could have row and column names.

```
# Dimensions
it_M <- 2
it_Q <- 3
it_N <- it_M*it_Q

# Initiate an Empty MxQ=N element list
ls_2d_flat <- vector(mode = "list", length = it_N)
ls_2d <- ls_2d_flat
```

```
# Named flat
ls_2d_flat_named <- ls_2d_flat
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))
ls_2d_named <- ls_2d_flat_named

# Reshape
dim(ls_2d) <- c(it_M, it_Q)
# named 2d list can not carry 1d name after reshape
dim(ls_2d_named) <- c(it_M, it_Q)
```

Print Various objects generated above, print list flattened.

```
# display
ffi_lst2str(ls_2d_flat_named)
```

```
## [1] "ls_2d_flat_named:e1=NULL;e2=NULL;e3=NULL;e4=NULL;e5=NULL;e6=NULL"
```

```
# print(ls_2d_flat_named)
ffi_lst2str(ls_2d_named)
```

```
## [1] "ls_2d_named:=NULL;=NULL;=NULL;=NULL;=NULL;=NULL"
```

```
print(ls_2d_named)
```

```
##      [,1] [,2] [,3]
## [1,] NULL NULL NULL
## [2,] NULL NULL NULL
```

Select element from list:

```
# Select Values, double bracket to select from 2dim list
print('ls_2d[[1,2]]')
```

```
## [1] "ls_2d[[1,2]]"
```

```
print(ls_2d[[1,2]])
```

```
## NULL
```

1.1.1.6 Define Two Dimensional Named LIst

For naming two dimensional lists, *rowname* and *colname* does not work. Rather, we need to use *dimnames*. Note that in addition to dimnames, we can continue to have element specific names. Both can co-exist. But note that the element specific names are not preserved after dimension transform, so need to be redefined afterwards.

How to select an element of a two dimensional list:

1. row and column names: *dimnames*, *ls_2d_flat_named[['row2','col2']]*
2. named elements: *names*, *ls_2d_flat_named[['e5']]*
3. select by index: *index*, *ls_2d_flat_named[[5]]*
4. converted two dimensional named list to tibble/matrix

Neither *dimnames* nor *names* are required, but both can be used to select elements.

```
# Dimensions
it_M <- 3
it_Q <- 4
it_N <- it_M*it_Q

# Initiate an Empty MxQ=N element list
ls_2d_flat_named <- vector(mode = "list", length = it_N)
dim(ls_2d_flat_named) <- c(it_M, it_Q)
```

```
# Fill with values
for (it_Q_ctr in seq(1,it_Q)) {
  for (it_M_ctr in seq(1,it_M)) {
    # linear index
    ls_2d_flat_named[[it_M_ctr, it_Q_ctr]] <- (it_Q_ctr-1)*it_M+it_M_ctr
  }
}

# Replace row names, note rownames does not work
dimnames(ls_2d_flat_named)[[1]] <- paste0('row',seq(1,it_M))
dimnames(ls_2d_flat_named)[[2]] <- paste0('col',seq(1,it_Q))

# Element Specific Names
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))

# Convert to Matrix
tb_2d_flat_named <- as_tibble(ls_2d_flat_named) %>% unnest()
mt_2d_flat_named <- as.matrix(tb_2d_flat_named)
```

Print various objects generated above:

```
# These are not element names, can still name each element
# display
print('ls_2d_flat_named')
```

```
## [1] "ls_2d_flat_named"
```

```
print(ls_2d_flat_named)
```

```
##      col1 col2 col3 col4
## row1 1    4    7    10
## row2 2    5    8    11
## row3 3    6    9    12
## attr(,"names")
##  [1] "e1"  "e2"  "e3"  "e4"  "e5"  "e6"  "e7"  "e8"  "e9"  "e10" "e11" "e12"
```

```
print('tb_2d_flat_named')
```

```
## [1] "tb_2d_flat_named"
```

```
print(tb_2d_flat_named)
print('mt_2d_flat_named')
```

```
## [1] "mt_2d_flat_named"
```

```
print(mt_2d_flat_named)
```

```
##      col1 col2 col3 col4
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Select elements from list:

```
# Select elements with with dimnames
ffi_lst2str(ls_2d_flat_named[['row2','col2']])
```

```
## [1] "ls_2d_flat_named[[\"row2\", \"col2\"]]:=5"
```

```
# Select elements with element names
ffi_lst2str(ls_2d_flat_named[['e5']])
```

```
## [1] "ls_2d_flat_named[[\"e5\"]]:=5"
```

```
# Select elements with index
ffi_lst2str(ls_2d_flat_named[[5]])
```

```
## [1] "ls_2d_flat_named[[5]]:=5"
```

1.1.1.7 Two-Dimensioanl Named List for Joint Probability Mass

There are two discrete random variables, generate some random discrete probability mass, name the columns and rows, and then convert to matrix.

```
set.seed(123)

# Generate prob list
it_Q <- 2
it_M <- 2
ls_2d <- vector(mode = "list", length = it_Q*it_M)
dim(ls_2d) <- c(it_Q, it_M)
# Random joint mass
ar_rand <- runif(it_Q*it_M)
ar_rand <- ar_rand/sum(ar_rand)
# Fill with values
it_ctr <- 0
for (it_Q_ctr in seq(1,it_Q)) {
  for (it_M_ctr in seq(1,it_M)) {
    # linear index
    ls_2d[[it_M_ctr, it_Q_ctr]] <- ar_rand[(it_Q_ctr-1)*it_M+it_M_ctr]
  }
}
# Replace row names, note rownames does not work
dimnames(ls_2d)[[1]] <- paste0('E',seq(1,it_M))
dimnames(ls_2d)[[2]] <- paste0('A',seq(1,it_Q))
# rename
ls_prob_joint_E_A <- ls_2d
mt_prob_joint_E_A <- matrix(unlist(ls_prob_joint_E_A), ncol=it_M, byrow=F)
print('ls_prob_joint_E_A')
```

```
## [1] "ls_prob_joint_E_A"
```

```
print(ls_prob_joint_E_A)
```

```
##    A1        A2
## E1 0.1214495 0.1727188
## E2 0.3329164 0.3729152
```

```
print(mt_prob_joint_E_A)
```

```
##           [,1]      [,2]
## [1,] 0.1214495 0.1727188
## [2,] 0.3329164 0.3729152
```

Create conditional probabilities: $F = P(A_1|E_1)$, $B = P(A_1|E_2)$, $C = P(E_1|A_1)$, $D = P(E_1|A_2)$

```
fl_F <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[1,])
fl_B <- mt_prob_joint_E_A[2,1]/sum(mt_prob_joint_E_A[2,])
fl_C <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[,1])
fl_D <- mt_prob_joint_E_A[1,2]/sum(mt_prob_joint_E_A[,2])
print(paste0('fl_F=', fl_F, ',fl_B=',fl_B,',fl_C=',fl_C,',fl_D=',fl_D))
```

```
## [1] "fl_F=0.412857205138471,fl_B=0.471665472604598,fl_C=0.267294503388642,fl_D=0.316546995323062"
```

1.2 Array

1.2.1 Array Basics

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.2.1.1 Multidimesional Arrays

```
ar_a <- c(1,2,3)
ar_b <- c(1,2,3/1,2,3)
rep(0, length(ar_a))
```

1.2.1.1.1 Repeat one Number by the Size of an Array

```
## [1] 0 0 0
```

```
# Multidimensional Array
# 1 is r1c1t1, 1.5 in r2c1t1, 0 in r1c2t1, etc.
# Three dimensions, row first, column second, and tensor third
x <- array(c(1, 1.5, 0, 2, 0, 4, 0, 3), dim=c(2, 2, 2))
dim(x)
```

1.2.1.1.2 Generate 2 Dimensional Array

```
## [1] 2 2 2
```

```
print(x)
```

```
## , , 1
##
##      [,1] [,2]
## [1,]  1.0    0
## [2,]  1.5    2
##
## , , 2
##
##      [,1] [,2]
## [1,]    0    0
## [2,]    4    3
```

1.2.1.2 Array Slicing

1.2.1.2.1 Get a Subset of Array Elements, N Cuts from M Points There is an array with M elements, get N elements from the M elements.

First cut including the starting and ending points.

```
it_M <- 5
it_N <- 4
ar_all_elements = seq(1,10,10)
```

1.2.1.2.2 Remove Elements of Array Select elements with direct indexing, or with head and tail functions. Get the first two elements of three elements array.

```
# Remove last element of array
vars.group.bydf <- c('23','dfa', 'wer')
vars.group.bydf[-length(vars.group.bydf)]
```

```
## [1] "23"  "dfa"
```

```
# Use the head function to remove last element
head(vars.group.bydf, -1)
```

```
## [1] "23"  "dfa"
```

```
head(vars.group.bydf, 2)
```

```
## [1] "23"  "dfa"
```

Get last two elements of array.

```
# Remove first element of array
vars.group.bydf <- c('23','dfa', 'wer')
vars.group.bydf[2:length(vars.group.bydf)]
```

```
## [1] "dfa" "wer"
```

```
# Use Tail function
tail(vars.group.bydf, -1)
```

```
## [1] "dfa" "wer"
```

```
tail(vars.group.bydf, 2)
```

```
## [1] "dfa" "wer"
```

Select all except for the first and the last element of an array.

```
# define array
ar_amin <- c(0, 0.25, 0.50, 0.75, 1)
# select without head and tail
tail(head(ar_amin, -1), -1)
```

```
## [1] 0.25 0.50 0.75
```

Select the first and the last element of an array. The extreme values.

```
# define array
ar_amin <- c(0, 0.25, 0.50, 0.75, 1)
# select head and tail
c(head(ar_amin, 1), tail(ar_amin, 1))
```

```
## [1] 0 1
```

1.2.1.3 NA in Array

```
# Convert Inf and -Inf to NA
x <- c(1, -1, Inf, 10, -Inf)
na_if(na_if(x, -Inf), Inf)
```

1.2.1.3.1 Check if NA is in Array

```
## [1]  1 -1 NA 10 NA
```

1.2.1.4 Complex Number

Handling numbers with real and imaginary components. Two separate issues, given an array of numbers that includes real as well as imaginary numbers, keep subset that only has real components. Additionally, for the same array, generate an equal length version of the array that includes the real components of all numbers.

Define complex numbers.

```
# Define a complex number
cx_number_a <- 0+0.0460246857561777i
# Define another complex number
```

```
cx_number_b <- complex(real = 0.02560982, imaginary = 0.0460246857561777)
# An array of numbers some of which are complex
ar_cx_number <- c(0.02560982+0.000000000i, 0.00000000+0.044895305i,
                  0.00000000+0.009153429i, 0.05462045+0.000000000i,
                  0.00000000+0.001198538i, 0.00000000+0.019267050i)
```

Extract real components from a complex array.

```
# equi-length real component
ar_fl_number_re <- Re(ar_cx_number)
print(ar_fl_number_re)
```

```
## [1] 0.02560982 0.00000000 0.00000000 0.05462045 0.00000000 0.00000000
```

```
# equi-length img component
ar_fl_number_im <- Im(ar_cx_number)
print(ar_fl_number_im)
```

```
## [1] 0.000000000 0.044895305 0.009153429 0.000000000 0.001198538 0.019267050
```

Keep only real elements of array.

```
# subset of array that is real
ar_fl_number_re_subset <- Re(ar_cx_number[Re(ar_cx_number)!=0])
print(ar_fl_number_re_subset)
```

```
## [1] 0.02560982 0.05462045
```

1.2.1.5 Number Formatting

1.2.1.5.1 e notation

1. Case one: *1.149946e+00*
   - this is approximately: 1.14995
2. Case two: *9.048038e-01*
   - this is approximately: 0.90480
3. Case three: *9.048038e-01*
   - this is approximately: 0.90480

1.2.2 Generate Arrays

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.2.2.1 Generate Often Used Arrays

1.2.2.1.1 Equi-distance Array with Bound Consider multiple income groups in income bins that are equal-width, for the final income group, consider all individuals above some final bin minimum bound. Below the code generates this array of numbers: $0, 20000, 40000, 60000, 80000, 100000, 100000000$.

```
# generate income cut-offs
fl_bin_start <- 0
# width equal to 20,000
fl_bin_width <- 2e4
# final point is 100 million
fl_bin_final_end <- 1e8
# final segment starting point is 100,000 dollars
fl_bin_final_start <- 1e5
# generate tincome bins
ar_income_bins <- c(seq(fl_bin_start, fl_bin_final_start, by=fl_bin_width),
                    fl_bin_final_end)
# Display
print(ar_income_bins)
```

```
## [1] 0e+00 2e+04 4e+04 6e+04 8e+04 1e+05 1e+08
```

Generate finer bins, at 5000 USD intervals, and stopping at 200 thousand dollars.

```
fl_bin_start <- 0
fl_bin_width <- 5e3
fl_bin_final_end <- 1e8
fl_bin_final_start <- 2e5
ar_income_bins <- c(seq(fl_bin_start, fl_bin_final_start, by=fl_bin_width),
                    fl_bin_final_end)
print(ar_income_bins)
```

```
##  [1] 0.00e+00 5.00e+03 1.00e+04 1.50e+04 2.00e+04 2.50e+04 3.00e+04 3.50e+04 4.00e+04 4.50e+04 5.
## [12] 5.50e+04 6.00e+04 6.50e+04 7.00e+04 7.50e+04 8.00e+04 8.50e+04 9.00e+04 9.50e+04 1.00e+05 1.
## [23] 1.10e+05 1.15e+05 1.20e+05 1.25e+05 1.30e+05 1.35e+05 1.40e+05 1.45e+05 1.50e+05 1.55e+05 1.
## [34] 1.65e+05 1.70e+05 1.75e+05 1.80e+05 1.85e+05 1.90e+05 1.95e+05 2.00e+05 1.00e+08
```

1.2.2.1.2 Log Space Arrays Often need to generate arrays on log rather than linear scale, below is log 10 scaled grid.

```
# Parameters
it.lower.bd.inc.cnt <- 3
fl.log.lower <- -10
fl.log.higher <- -9
fl.min.rescale <- 0.01
it.log.count <- 4
# Generate
ar.fl.log.rescaled <- exp(log(10)*seq(log10(fl.min.rescale),
                                      log10(fl.min.rescale +
                                              (fl.log.higher-fl.log.lower)),
                                      length.out=it.log.count))
ar.fl.log <- ar.fl.log.rescaled + fl.log.lower - fl.min.rescale
# Print
ar.fl.log
```

```
## [1] -10.000000  -9.963430  -9.793123  -9.000000
```

1.2.2.2 Generate Arrays Based on Existing Arrays

1.2.2.2.1 Probability Mass Array and Discrete Value Array There are two arrays, an array of values, and an array of probabilities. The probability array sums to 1. The array of values, however, might not be unique.

First, generate some array of numbers not sorted and some proability mass for each non-sorted, non-unique element of the array.

```
set.seed(123)
it_len <- 10
ar_x <- ceiling(runif(it_len)*5+10)
ar_prob <- dbinom(seq(0,it_len-1,length.out = it_len), it_len-1, prob=0.5)
print(cbind(ar_x,ar_prob))
```

```
##       ar_x     ar_prob
##  [1,]   12 0.001953125
##  [2,]   14 0.017578125
##  [3,]   13 0.070312500
##  [4,]   15 0.164062500
##  [5,]   15 0.246093750
##  [6,]   11 0.246093750
##  [7,]   13 0.164062500
```

```
##  [8,]   15 0.070312500
##  [9,]   13 0.017578125
## [10,]   13 0.001953125
```

```
print(paste0('sum(ar_prob)=',sum(ar_prob)))
```

```
## [1] "sum(ar_prob)=1"
```

Second, sorting index for ar_x, and resort ar_prob with the same index:

```
ls_sorted_res <- sort(ar_x, decreasing = FALSE, index.return=TRUE)
ar_idx_increasing_x <- ls_sorted_res$ix
ar_x_sorted <- ls_sorted_res$x
ar_prob_sorted <- ar_prob[ar_idx_increasing_x]
print(cbind(ar_x_sorted,ar_prob_sorted))
```

```
##       ar_x_sorted ar_prob_sorted
##  [1,]          11    0.246093750
##  [2,]          12    0.001953125
##  [3,]          13    0.070312500
##  [4,]          13    0.164062500
##  [5,]          13    0.017578125
##  [6,]          13    0.001953125
##  [7,]          14    0.017578125
##  [8,]          15    0.164062500
##  [9,]          15    0.246093750
## [10,]          15    0.070312500
```

Third, sum within group and generate unique, using the aggregate function. Then we have a column of unique values and associated probabilities.

```
ar_x_unique <- unique(ar_x_sorted)
mt_prob_unique <- aggregate(ar_prob_sorted, by=list(ar_x_sorted), FUN=sum)
ar_x_unique_prob <- mt_prob_unique$x
print(cbind(ar_x_unique, ar_x_unique_prob))
```

```
##      ar_x_unique ar_x_unique_prob
## [1,]          11      0.246093750
## [2,]          12      0.001953125
## [3,]          13      0.253906250
## [4,]          14      0.017578125
## [5,]          15      0.480468750
```

Finally, the several steps together.

```
# data
set.seed(123)
it_len <- 30
ar_x <- ceiling(runif(it_len)*20+10)
ar_prob <- runif(it_len)
ar_prob <- ar_prob/sum(ar_prob)
# step 1, sort
ls_sorted_res <- sort(ar_x, decreasing = FALSE, index.return=TRUE)
# step 2, unique sorted
ar_x_unique <- unique(ls_sorted_res$x)
# step 3, mass for each unique
mt_prob_unique <- aggregate(ar_prob[ls_sorted_res$ix], by=list(ls_sorted_res$x), FUN=sum)
ar_x_unique_prob <- mt_prob_unique$x
# results
print(cbind(ar_x_unique, ar_x_unique_prob))
```

```
##       ar_x_unique ar_x_unique_prob
##  [1,]          11      0.071718383
```

```
##  [2,]          13      0.040040920
##  [3,]          15      0.017708800
##  [4,]          16      0.141199002
##  [5,]          17      0.020211876
##  [6,]          19      0.052488290
##  [7,]          20      0.049104113
##  [8,]          21      0.067328518
##  [9,]          22      0.109454333
## [10,]          23      0.060712145
## [11,]          24      0.107671406
## [12,]          25      0.015694798
## [13,]          26      0.068567789
## [14,]          28      0.090925756
## [15,]          29      0.001870451
## [16,]          30      0.085303420
```

1.2.3 String Arrays

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.2.3.1 Positive or Negative Floating Number to String

There is a number, that contains decimal and possibly negative sign and has some decimals, convert this to a string that is more easily used as a file or folder name.

```
ls_fl_rho <- c(1, -1, -1.5 -100, 0.5, 0.11111111, -199.22123)
for (fl_rho in ls_fl_rho) {
  st_rho <- paste0(round(fl_rho, 4))
  st_rho <- gsub(x = st_rho,  pattern = "-", replacement = "n")
  st_rho <- gsub(x = st_rho,  pattern = "\\.", replacement = "p")
  print(paste0('st_rho=', st_rho))
}
```

```
## [1] "st_rho=1"
## [1] "st_rho=n1"
## [1] "st_rho=n101p5"
## [1] "st_rho=0p5"
## [1] "st_rho=0p1111"
## [1] "st_rho=n199p2212"
```

1.2.3.2 String Replace

- r string wildcard replace between regex
- R - replace part of a string using wildcards

```
# String replacement
gsub(x = paste0(unique(df.slds.stats.perc$it.inner.counter), ':',
                unique(df.slds.stats.perc$z_n_a_n), collapse = ';'),
     pattern = "\n",
     replacement = "")
gsub(x = var,  pattern = "\n", replacement = "")
gsub(x = var.input,  pattern = "\\.", replacement = "_")
```

String replaces a segment, search by wildcard. Given the string below, delete all text between carriage return and pound sign:

```
st_tex_text <- "\n% Lat2ex Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha + \\be
st_clean_a1 <- gsub("\\%.*?\\\n", "", st_tex_text)
st_clean_a2 <- gsub("L.*?x", "[LATEX]", st_tex_text)
print(paste0('st_tex_text:', st_tex_text))
```

```
## [1] "st_tex_text:\n% Lat2ex Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha +
```

```
print(paste0('st_clean_a1:', st_clean_a1))
```

```
## [1] "st_clean_a1:\n\\newcommand{\\exa}{\\text{from external file: } \\alpha + \\beta}\n"
```

```
print(paste0('st_clean_a2:', st_clean_a2))
```

```
## [1] "st_clean_a2:\n% [LATEX] Comments\n\\newcommand{\\exa}{\\text{from external file: } \\alpha +
```

String delete after a particular string:

```
st_tex_text <- "\\end{equation}\n}\n% Even more comments from Latex preamble"
st_clean_a1 <- gsub("\\\n%.*","", st_tex_text)
print(paste0('st_tex_text:', st_tex_text))
```

```
## [1] "st_tex_text:\\end{equation}\n}\n% Even more comments from Latex preamble"
```

```
print(paste0('st_clean_a1:', st_clean_a1))
```

```
## [1] "st_clean_a1:\\end{equation}\n}"
```

1.2.3.3 Search If and Which String Contains

- r if string contains
- r if string contains either or grepl
- Use grepl to search either of multiple substrings in a text

Search for a single substring in a single string:

```
st_example_a <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
st_example_b <- 'C:/Users/fan/R4Econ/amto/tibble/_main.html'
grepl('_main', st_example_a)
```

```
## [1] FALSE
```

```
grepl('_main', st_example_b)
```

```
## [1] TRUE
```

Search for if one of a set of substring exists in a set of strings. In particular which one of the elements of *ls_spn* contains at least one of the elements of *ls_str_if_contains*. In the example below, only the first path does not contain either the word *aggregate* or *index* in the path. This can be used after all paths have been found recursively in some folder to select only desired paths from the full set of possibilities:

```
ls_spn <- c("C:/Users/fan/R4Econ//panel/basic/fs_genpanel.Rmd",
            "C:/Users/fan/R4Econ//summarize/aggregate/main.Rmd",
            "C:/Users/fan/R4Econ//summarize/index/fs_index_populate.Rmd")
ls_str_if_contains <- c("aggregate", "index")
str_if_contains <- paste(ls_str_if_contains, collapse = "|")
grepl(str_if_contains, ls_spn)
```

```
## [1] FALSE  TRUE  TRUE
```

1.2.3.4 String Split

Given some string, generated for example by cut, get the lower cut starting points, and also the higher end point

```
# Extract 0.216 and 0.500 as lower and upper bounds
st_cut_cate <- '(0.216,0.500]'
# Extract Lower Part
substring(strsplit(st_cut_cate, ",")[[1]][1], 2)
```

```
## [1] "0.216"
```

```
# Extract second part except final bracket Option 1
intToUtf8(rev(utf8ToInt(substring(intToUtf8(rev(utf8ToInt(strsplit(st_cut_cate, ",")[[1]][2]))), 2))
```

```
## [1] "0.500"
```

```
# Extract second part except final bracket Option 2
gsub(strsplit(st_cut_cate, ",")[[1]][2],  pattern = "]", replacement = "")
```

```
## [1] "0.500"
```

1.2.3.5 String Concatenate

```
# Simple Collapse
vars.group.by <- c('abc', 'efg')
paste0(vars.group.by, collapse='|')
```

```
## [1] "abc|efg"
```

1.2.3.6 String Add Leading Zero

```
# Add Leading zero for integer values to allow for sorting when
# integers are combined into strings
it_z_n <- 1
it_a_n <- 192
print(sprintf("%02d", it_z_n))
```

```
## [1] "01"
```

```
print(sprintf("%04d", it_a_n))
```

```
## [1] "0192"
```

1.2.3.7 Substring Components

Given a string, with certain structure, get components.

- r time string get month and year and day

```
snm_full <- "20100701"
snm_year <-substr(snm_full,0,4)
snm_month <-substr(snm_full,5,6)
snm_day <-substr(snm_full,7,8)
print(paste0('full:', snm_full,
             ', year:', snm_year,
             ', month:', snm_month,
             ', day:', snm_day))
```

```
## [1] "full:20100701, year:2010, month:07, day:01"
```

1.2.4 Mesh Matrices, Arrays and Scalars

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

- r expand.grid meshed array to matrix
- r meshgrid
- r array to matrix
- r reshape array to matrix
- dplyr permuations rows of matrix and element of array
- tidyr expand_grid mesh matrix and vector

1.2.4.1 Mesh Two or More Vectors with expand_grid

In the example below, we have a matrix that is 2 by 2 (endogenous states), a vector that is 3 by 1 (choices), and another matrix that is 4 by 3 (exogenous states shocks).

We want to generate a tibble dataset that meshes the matrix and the vector, so that all combinations show up. Additionally, we want to add some additional values that are common across all rows to the meshed dataframe.

Note *expand_grid* is a from tidyr 1.0.0.

```
# A. Generate the 5 by 2 Matrix (ENDO STATES)
# it_child_count = N, the number of children
it_N_child_cnt = 2
# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
fl_rho = 0.1
fl_lambda = 1.1
mt_nP_A_alpha = cbind(ar_nN_A, ar_nN_alpha, fl_rho, fl_lambda)
ar_st_varnames <- c('s_A', 's_alpha', 'p_rho', 'p_lambda')
tb_states_endo <- as_tibble(mt_nP_A_alpha) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "state_id")

# B. Choice Grid
it_N_choice_cnt = 3
fl_max = 10
fl_min = 0
ar_nN_d = seq(fl_min, fl_max, length.out = it_N_choice_cnt)
ar_st_varnames <- c('c_food')
tb_choices <- as_tibble(ar_nN_d) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "choice_id")

# C. Shock Grid
set.seed(123)
it_N_shock_cnt = 4
ar_nQ_shocks = exp(rnorm(it_N_shock_cnt, mean=0, sd=1))
ar_st_varnames <- c('s_eps')
tb_states_exo <- as_tibble(ar_nQ_shocks) %>%
  rename_all(~c(ar_st_varnames)) %>%
  rowid_to_column(var = "shock_id")

# dataframe expand with other non expanded variables
ar_st_varnames <-
tb_states_shk_choices <- tb_states_endo %>%
  expand_grid(tb_choices) %>%
  expand_grid(tb_states_exo) %>%
  select(state_id, choice_id, shock_id,
         s_A, s_alpha, s_eps, c_food,
         p_rho, p_lambda)

# display
kable(tb_states_shk_choices) %>% kable_styling_fc()
```

Using expand_grid directly over arrays

```
# expand grid with dplyr
expand_grid(x = 1:3, y = 1:2, z = -3:-1)
```

state_id	choice_id	shock_id	s_A	s_alpha	s_eps	c_food	p_rho	p_lambda
1	1	1	-2	0.1	0.5709374	0	0.1	1.1
1	1	2	-2	0.1	0.7943926	0	0.1	1.1
1	1	3	-2	0.1	4.7526783	0	0.1	1.1
1	1	4	-2	0.1	1.0730536	0	0.1	1.1
1	2	1	-2	0.1	0.5709374	5	0.1	1.1
1	2	2	-2	0.1	0.7943926	5	0.1	1.1
1	2	3	-2	0.1	4.7526783	5	0.1	1.1
1	2	4	-2	0.1	1.0730536	5	0.1	1.1
1	3	1	-2	0.1	0.5709374	10	0.1	1.1
1	3	2	-2	0.1	0.7943926	10	0.1	1.1
1	3	3	-2	0.1	4.7526783	10	0.1	1.1
1	3	4	-2	0.1	1.0730536	10	0.1	1.1
2	1	1	2	0.9	0.5709374	0	0.1	1.1
2	1	2	2	0.9	0.7943926	0	0.1	1.1
2	1	3	2	0.9	4.7526783	0	0.1	1.1
2	1	4	2	0.9	1.0730536	0	0.1	1.1
2	2	1	2	0.9	0.5709374	5	0.1	1.1
2	2	2	2	0.9	0.7943926	5	0.1	1.1
2	2	3	2	0.9	4.7526783	5	0.1	1.1
2	2	4	2	0.9	1.0730536	5	0.1	1.1
2	3	1	2	0.9	0.5709374	10	0.1	1.1
2	3	2	2	0.9	0.7943926	10	0.1	1.1
2	3	3	2	0.9	4.7526783	10	0.1	1.1
2	3	4	2	0.9	1.0730536	10	0.1	1.1

1.2.4.2 Mesh Arrays with expand.grid

Given two arrays, mesh the two arrays together.

```
# use expand.grid to generate all combinations of two arrays

it_ar_A = 5
it_ar_alpha = 10

ar_A = seq(-2, 2, length.out=it_ar_A)
ar_alpha = seq(0.1, 0.9, length.out=it_ar_alpha)

mt_A_alpha = expand.grid(A = ar_A, alpha = ar_alpha)

mt_A_meshed = mt_A_alpha[,1]
dim(mt_A_meshed) = c(it_ar_A, it_ar_alpha)

mt_alpha_meshed = mt_A_alpha[,2]
dim(mt_alpha_meshed) = c(it_ar_A, it_ar_alpha)

# display
kable(mt_A_meshed) %>%
  kable_styling_fc()
```

-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2

```
kable(mt_alpha_meshed) %>%
  kable_styling_fc_wide()
```

0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9
0.1	0.1888889	0.2777778	0.3666667	0.4555556	0.5444444	0.6333333	0.7222222	0.8111111	0.9

Two Identical Arrays, individual attributes, each column is an individual for a matrix, and each row is also an individual.

```
# use expand.grid to generate all combinations of two arrays

it_ar_A = 5

ar_A = seq(-2, 2, length.out=it_ar_A)
mt_A_A = expand.grid(Arow = ar_A, Arow = ar_A)
mt_Arow = mt_A_A[,1]
dim(mt_Arow) = c(it_ar_A, it_ar_A)
mt_Acol = mt_A_A[,2]
dim(mt_Acol) = c(it_ar_A, it_ar_A)

# display
kable(mt_Arow) %>%
  kable_styling_fc()
```

-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

```
kable(mt_Acol) %>%
  kable_styling_fc()
```

-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2
-2	-1	0	1	2

1.3 Matrix

1.3.1 Generate Matrixes

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.3.1.1 Create a N by 2 Matrix from 3 arrays

Names of each array become row names automatically.

```
ar_row_one <- c(-1,+1)
ar_row_two <- c(-3,-2)
ar_row_three <- c(0.35,0.75)
```

```
mt_n_by_2 <- rbind(ar_row_one, ar_row_two, ar_row_three)
kable(mt_n_by_2) %>%
  kable_styling_fc()
```

ar_row_one	-1.00	1.00
ar_row_two	-3.00	-2.00
ar_row_three	0.35	0.75

1.3.1.2 Name Matrix Columns and Rows

```
# An empty matrix with Logical NA
mt_named <- matrix(data=NA, nrow=2, ncol=2)
colnames(mt_named) <- paste0('c', seq(1,2))
rownames(mt_named) <- paste0('r', seq(1,2))
mt_named
```

```
##    c1 c2
## r1 NA NA
## r2 NA NA
```

1.3.1.3 Generate NA Matrix

- Best way to allocate matrix in R, NULL vs NA?

Allocate with NA or NA_real_ or NA_int_. Clarity in type definition is preferred.

```
# An empty matrix with Logical NA
mt_na <- matrix(data=NA, nrow=2, ncol=2)
str(mt_na)
```

```
##  logi [1:2, 1:2] NA NA NA NA
```

```
# An empty matrix with numerica NA
mt_fl_na <- matrix(data=NA_real_, nrow=2, ncol=2)
mt_it_na <- matrix(data=NA_integer_, nrow=2, ncol=2)

str(mt_fl_na)
```

```
##  num [1:2, 1:2] NA NA NA NA
```

```
str(mt_fl_na)
```

```
##  num [1:2, 1:2] NA NA NA NA
```

1.3.1.4 Generate Random Matrixes

Random draw from the normal distribution, random draw from the uniform distribution, and combine resulting matrixes.

```
# Generate 15 random normal, put in 5 rows, and 3 columns
mt_rnorm <- matrix(rnorm(15,mean=0,sd=1), nrow=5, ncol=3)

# Generate 15 random normal, put in 5 rows, and 3 columns
mt_runif <- matrix(runif(15,min=0,max=1), nrow=5, ncol=5)

# Combine
mt_rnorm_runif <- cbind(mt_rnorm, mt_runif)

# Display
kable(mt_rnorm_runif) %>% kable_styling_fc_wide()
```

0.1292877	-0.4456620	-0.5558411	0.3181810	0.3688455	0.2659726	0.3181810	0.3688455
1.7150650	1.2240818	1.7869131	0.2316258	0.1524447	0.8578277	0.2316258	0.1524447
0.4609162	0.3598138	0.4978505	0.1428000	0.1388061	0.0458312	0.1428000	0.1388061
-1.2650612	0.4007715	-1.9666172	0.4145463	0.2330341	0.4422001	0.4145463	0.2330341
-0.6868529	0.1106827	0.7013559	0.4137243	0.4659625	0.7989248	0.4137243	0.4659625

1.3.1.5 Add Column to Matrix with Common Scalar Value

Given some matrix of information, add a column, where all rows of the column have the same numerical value. Use the matrix created prior. - R add column to matrix - r append column to matrix constant value

```
fl_new_first_col_val <- 111
fl_new_last_col_val <- 999
mt_with_more_columns <- cbind(rep(fl_new_first_col_val, dim(mt_rnorm_runif)[1]),
                              mt_rnorm_runif,
                              rep(fl_new_last_col_val, dim(mt_rnorm_runif)[1]))
# Display
kable(mt_with_more_columns) %>% kable_styling_fc_wide()
```

111	0.1292877	-0.4456620	-0.5558411	0.3181810	0.3688455	0.2659726	0.3181810	0.3688455	999
111	1.7150650	1.2240818	1.7869131	0.2316258	0.1524447	0.8578277	0.2316258	0.1524447	999
111	0.4609162	0.3598138	0.4978505	0.1428000	0.1388061	0.0458312	0.1428000	0.1388061	999
111	-1.2650612	0.4007715	-1.9666172	0.4145463	0.2330341	0.4422001	0.4145463	0.2330341	999
111	-0.6868529	0.1106827	0.7013559	0.4137243	0.4659625	0.7989248	0.4137243	0.4659625	999

1.3.2 Linear Algebra

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.3.2.1 Matrix Multiplication

Multiply Together a 3 by 2 matrix and a 2 by 1 vector

```
ar_row_one <- c(-1,+1)
ar_row_two <- c(-3,-2)
ar_row_three <- c(0.35,0.75)
mt_n_by_2 <- rbind(ar_row_one, ar_row_two, ar_row_three)

ar_row_four <- c(3,4)

# Matrix Multiplication
mt_out <- mt_n_by_2 %*% ar_row_four
print(mt_n_by_2)
```

```
##               [,1]  [,2]
## ar_row_one   -1.00  1.00
## ar_row_two   -3.00 -2.00
## ar_row_three  0.35  0.75
```

```
print(ar_row_four)
```

```
## [1] 3 4
```

```
print(mt_out)
```

```
##                [,1]
## ar_row_one     1.00
## ar_row_two   -17.00
## ar_row_three   4.05
```

1.4 Variables in Dataframes

1.4.1 Generate Dataframe

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.4.1.1 Simple Meshed Dataframe Name Columns

```
# 5 by 3 matrix
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=3)

# Column Names
ar_st_varnames <- c('id','var1','varb','vartheta')

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_rnorm_a) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_combine) %>% kable_styling_fc()
```

id	var1	varb	vartheta
1	-1.1655448	-0.8185157	0.6849361
2	-0.8185157	0.6849361	-0.3200564
3	0.6849361	-0.3200564	-1.1655448
4	-0.3200564	-1.1655448	-0.8185157
5	-1.1655448	-0.8185157	0.6849361

1.4.1.2 Generate Tibble given Matrixes and Arrays

Given Arrays and Matrixes, Generate Tibble and Name Variables/Columns

- naming tibble columns
- tibble variable names
- dplyr rename tibble
- dplyr rename tibble all variables
- dplyr rename all columns by index
- dplyr tibble add index column
- see also: SO-51205520

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=2)
mt_rnorm_b <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=4)

# Combine Matrix
mt_combine <- cbind(ar_col, mt_rnorm_a, mt_rnorm_b)
colnames(mt_combine) <- c('ar_col',
                          paste0('matcolvar_grpa_', seq(1,dim(mt_rnorm_a)[2])),
                          paste0('matcolvar_grpb_', seq(1,dim(mt_rnorm_b)[2])))

# Variable Names
ar_st_varnames <- c('var_one',
                    paste0('tibcolvar_ga_', c(1,2)),
                    paste0('tibcolvar_gb_', c(1,2,3,4)))

# Combine to tibble, add name col1, col2, etc.
```

```
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine %>%
                  rename_at(vars(starts_with("tibcolvar_gb_")),
                            funs(str_replace(., "_gb_", "_gbrenamed_")))

# Tibble back to matrix
mt_tb_combine_back <- data.matrix(tb_combine)

# Display
kable(mt_combine) %>% kable_styling_fc_wide()
```

ar_col	matcolvar_grpa_1	matcolvar_grpa_2	matcolvar_grpb_1	matcolvar_grpb_2	matcolvar_grpb_3	matcolvar_grpb_4
-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
2	1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

```
kable(mt_tb_combine_back) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	-1.3115224	-0.1294107	-0.1513960	-3.2273228	-0.1513960	-3.2273228
2	1	-0.5996083	0.8867361	0.3297912	-0.7717918	0.3297912	-0.7717918

1.4.1.3 Rename Tibble with Numeric Column Names

After reshaping, often could end up with variable names that are all numeric, intgers for example, how to rename these variables to add a common prefix for example.

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_c <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=10)
mt_combine <- cbind(ar_col, mt_rnorm_c)

# Variable Names
ar_it_cols_ctr <- seq(1, dim(mt_rnorm_c)[2])
ar_st_varnames <- c('var_one', ar_it_cols_ctr)

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine_ori <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine_ori %>%
                  rename_at(
                    vars(num_range('',ar_it_cols_ctr)),
                    funs(paste0("rho", . , 'var'))
                    )

# Display
kable(tb_combine_ori) %>% kable_styling_fc_wide()
```

ID	var_one	1	2	3	4	5	6	7	8	9	10
1	-1	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120
2	1	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504
3	-1	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769
4	1	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486
5	-1	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	rho1var	rho2var	rho3var	rho4var	rho5var	rho6var	rho7var	rho8var	rho9var	rho10var
1	-1	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120
2	1	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504
3	-1	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769
4	1	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486
5	-1	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120	0.4345504	0.8001769	0.2865486	-1.2205120

1.4.1.4 Tibble Row and Column and Summarize

Show what is in the table: 1, column and row names; 2, contents inside table.

```
tb_iris <- as_tibble(iris)
print(rownames(tb_iris))
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"  "13"  "14"  "15"  "
##  [17] "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "
##  [33] "33"  "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "
##  [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "
##  [65] "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"  "79"  "
##  [81] "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "
##  [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110" "111" "
## [113] "113" "114" "115" "116" "117" "118" "119" "120" "121" "122" "123" "124" "125" "126" "127" "
## [129] "129" "130" "131" "132" "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "
## [145] "145" "146" "147" "148" "149" "150"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
summary(tb_iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

1.4.1.5 Tibble Sorting

- dplyr arrange desc reverse
- dplyr sort

```
# Sort in Ascending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(Species, Sepal.Length) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	4.3	3.0	1.1	0.1
setosa	4.4	2.9	1.4	0.2
setosa	4.4	3.0	1.3	0.2
setosa	4.4	3.2	1.3	0.2
setosa	4.5	2.3	1.3	0.3
setosa	4.6	3.1	1.5	0.2
setosa	4.6	3.4	1.4	0.3
setosa	4.6	3.6	1.0	0.2
setosa	4.6	3.2	1.4	0.2
setosa	4.7	3.2	1.3	0.2

```
# Sort in Descending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(desc(Species), desc(Sepal.Length)) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica	7.9	3.8	6.4	2.0
virginica	7.7	3.8	6.7	2.2
virginica	7.7	2.6	6.9	2.3
virginica	7.7	2.8	6.7	2.0
virginica	7.7	3.0	6.1	2.3
virginica	7.6	3.0	6.6	2.1
virginica	7.4	2.8	6.1	1.9
virginica	7.3	2.9	6.3	1.8
virginica	7.2	3.6	6.1	2.5
virginica	7.2	3.2	6.0	1.8

1.4.1.6 REconTools Summarize over Tible

Use R4Econ's summary tool.

```
df_summ_stats <- ff_summ_percentiles(tb_iris)
kable(t(df_summ_stats)) %>% kable_styling_fc_wide()
```

stats	n	unique	NAobs	ZERObs	mean	sd	cv	min	p01	p05	p10	p25	p50	p75	p90	p95	p99	max
Petal.Length	150	43	0	0	3.758000	1.7652982	0.4697441	1.0	1.149	1.300	1.4	1.6	4.35	5.1	5.80	6.100	6.700	6.9
Petal.Width	150	22	0	0	1.199333	0.7622377	0.6355511	0.1	0.100	0.200	0.2	0.3	1.30	1.8	2.20	2.300	2.500	2.5
Sepal.Length	150	35	0	0	5.843333	0.8280661	0.1417113	4.3	4.400	4.600	4.8	5.1	5.80	6.4	6.90	7.255	7.700	7.9
Sepal.Width	150	23	0	0	3.057333	0.4358663	0.1425642	2.0	2.200	2.345	2.5	2.8	3.00	3.3	3.61	3.800	4.151	4.4

1.4.2 Factor Label and Combine

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.4.2.1 Factor, Label, Cross and Graph

Generate a Scatter plot with different colors representing different categories. There are multiple underlying factor/categorical variables, for example two binary variables. Generate scatter plot with colors for the combinations of these two binary variables.

We combine here the *vs* and *am* variables from the *mtcars* dataset. *vs* is engine shape, *am* is auto or manual shift. We will generate a scatter plot of *mpg* and *qsec* over four categories with different colors.

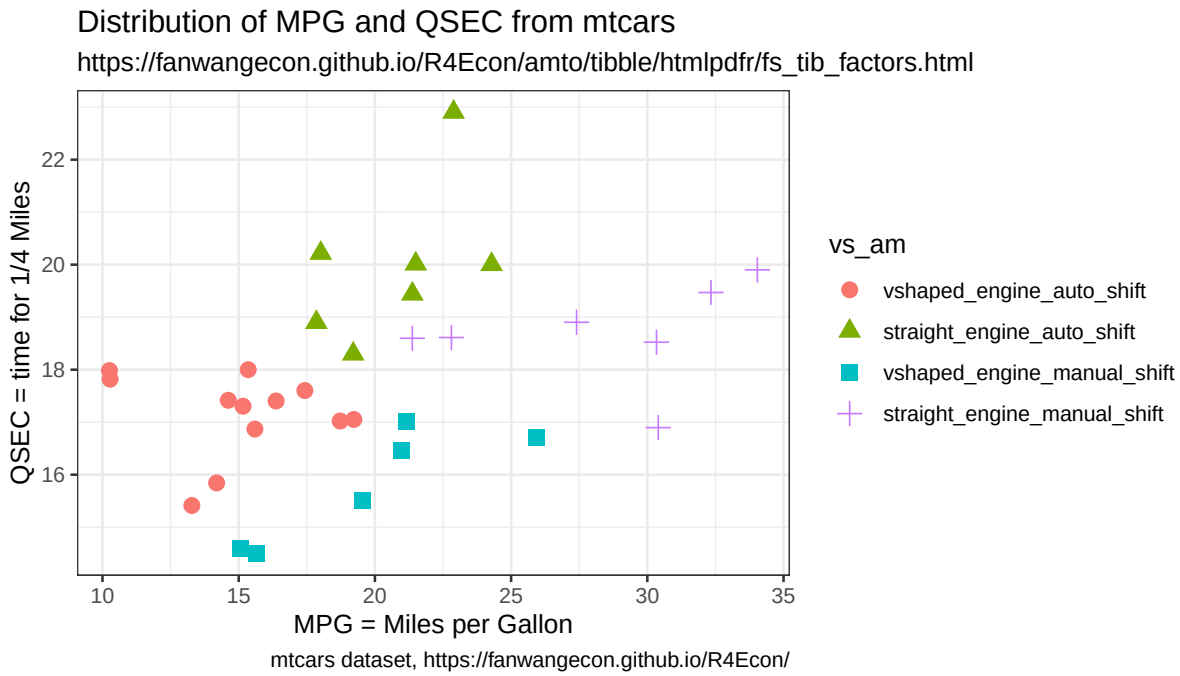
- *am*: Transmission (0 = automatic, 1 = manual)
- *vs*: Engine (0 = V-shaped, 1 = straight)
- *mpg*: miles per galon

- *qsec*: 1/4 mile time

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
  mutate(vs = as_factor(vs), am = as_factor(am))

# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c(vshaped_engine = "0", straight_engine = "1")
tb_mtcars <- tb_mtcars %>%
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))

# Third Combine Factors
tb_mtcars_selected <- tb_mtcars %>%
  mutate(vs_am = fct_cross(vs, am, sep='_', keep_empty = FALSE)) %>%
  select(mpg, qsec, vs_am)
print(tb_mtcars_selected)
```

Now we generate scatter plot based on the combined factors

```
# Labeling
st_title <- paste0('Distribution of MPG and QSEC from mtcars')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/amto/tibble/htmlpdfr/fs_tib_factors.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars_selected,
         aes(x=mpg, y=qsec, colour=vs_am, shape=vs_am)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_scatter)
```



1.4.3 Drawly Random Rows

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.4.3.1 Draw Random Subset of Sample

- r random discrete

We have a sample of N individuals in some dataframe. Draw without replacement a subset $M < N$ of rows.

```
# parameters, it_M < it_N
it_N <- 10
it_M <- 5

# Draw it_m from indexed list of it_N
set.seed(123)
ar_it_rand_idx <- sample(it_N, it_M, replace=FALSE)

# dataframe
df_full <- as_tibble(matrix(rnorm(4,mean=0,sd=1), nrow=it_N, ncol=4)) %>% rowid_to_column(var = "ID"

# random Subset
df_rand_sub_a <- df_full[ar_it_rand_idx,]

# Random subset also
df_rand_sub_b <- df_full[sample(dim(df_full)[1], it_M, replace=FALSE),]

# Print
# Display
kable(df_full) %>% kable_styling_fc()
```

```
kable(df_rand_sub_a) %>% kable_styling_fc()
```

```
kable(df_rand_sub_b) %>% kable_styling_fc()
```

ID	V1	V2	V3	V4
1	0.1292877	0.4609162	0.1292877	0.4609162
2	1.7150650	-1.2650612	1.7150650	-1.2650612
3	0.4609162	0.1292877	0.4609162	0.1292877
4	-1.2650612	1.7150650	-1.2650612	1.7150650
5	0.1292877	0.4609162	0.1292877	0.4609162
6	1.7150650	-1.2650612	1.7150650	-1.2650612
7	0.4609162	0.1292877	0.4609162	0.1292877
8	-1.2650612	1.7150650	-1.2650612	1.7150650
9	0.1292877	0.4609162	0.1292877	0.4609162
10	1.7150650	-1.2650612	1.7150650	-1.2650612

ID	V1	V2	V3	V4
3	0.4609162	0.1292877	0.4609162	0.1292877
10	1.7150650	-1.2650612	1.7150650	-1.2650612
2	1.7150650	-1.2650612	1.7150650	-1.2650612
8	-1.2650612	1.7150650	-1.2650612	1.7150650
6	1.7150650	-1.2650612	1.7150650	-1.2650612

ID	V1	V2	V3	V4
5	0.1292877	0.4609162	0.1292877	0.4609162
3	0.4609162	0.1292877	0.4609162	0.1292877
9	0.1292877	0.4609162	0.1292877	0.4609162
1	0.1292877	0.4609162	0.1292877	0.4609162
4	-1.2650612	1.7150650	-1.2650612	1.7150650

1.4.3.2 Random Subset of Panel

There are N individuals, each could be observed M times, but then select a subset of rows only, so each person is randomly observed only a subset of times. Specifically, there there are 3 unique students with student ids, and the second variable shows the random dates in which the student showed up in class, out of the 10 classes available.

```
# Define
it_N <- 3
it_M <- 10
svr_id <- 'student_id'

# dataframe
set.seed(123)
df_panel_rand <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(date = row_number()) %>%
  ungroup() %>% mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  filter(in_class == 1) %>% select(!!sym(svr_id), date) %>%
  rename(date_in_class = date)

# Print
kable(df_panel_rand) %>% kable_styling_fc()
```

1.4.4 Generate Variables Conditional On Others

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

student_id	date_in_class
1	1
1	2
1	8
1	9
1	10
2	5
2	8
2	10
3	1
3	2
3	3
3	4
3	5
3	6
3	9

1.4.4.1 case_when Basic Example

Given several other variables, and generate a new variable when these varaibles satisfy conditions. Note that case_when are ifelse type statements. So below

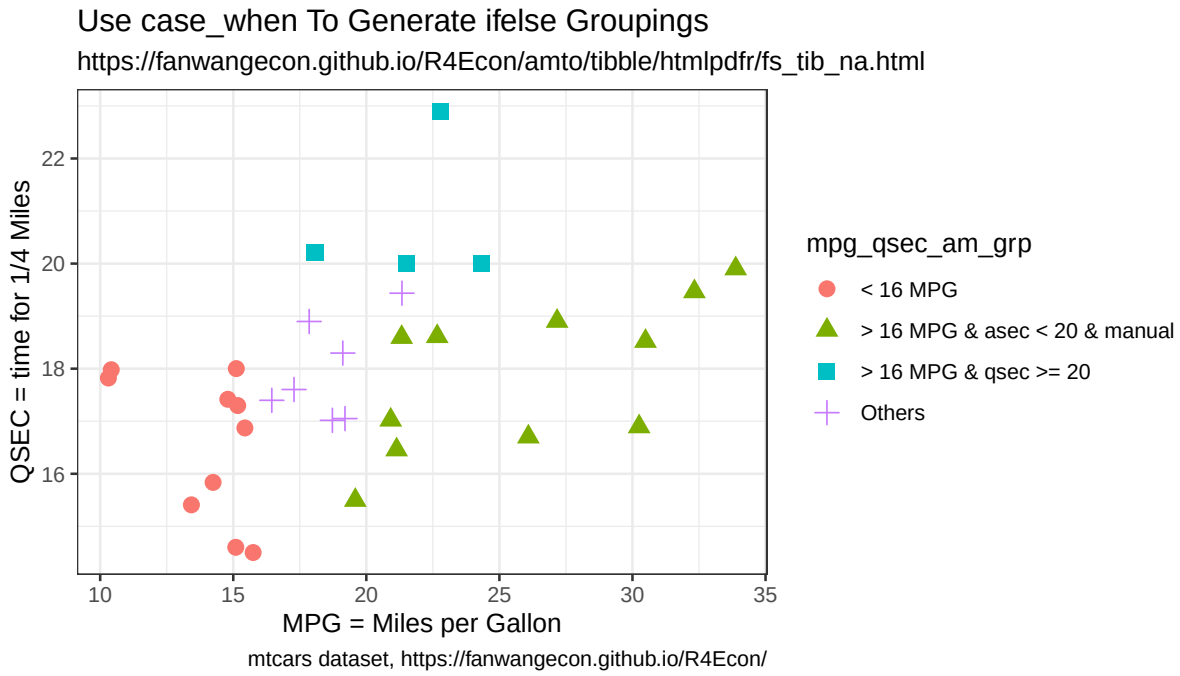
1. group one is below 16 MPG
2. when do *qsec >= 20* second line that is elseif, only those that are *>=16* are considered here
3. then think about two dimensional mpg and qsec grid, the lower-right area, give another category to manual cars in that group

```
# Get mtcars
df_mtcars <- mtcars

# case_when with mtcars
df_mtcars <- df_mtcars %>%
  mutate(mpg_qsec_am_grp =
           case_when(mpg < 16 ~ "< 16 MPG",
                     qsec >= 20 ~ "> 16 MPG & qsec >= 20",
                     am == 1 ~ "> 16 MPG & asec < 20 & manual",
                     TRUE ~ "Others"))

# # For dataframe
# df.reg <-df.reg %>% na_if(-Inf) %>% na_if(Inf)
# # For a specific variable in dataframe
# df.reg.use %>% mutate(!!(var.input) := na_if(!!sym(var.input), 0))
#
# # Setting to NA
# df.reg.use <- df.reg.guat %>% filter(!!sym(var.mth) != 0)
# df.reg.use.log <- df.reg.use
# df.reg.use.log[which(is.nan(df.reg.use$prot.imputed.log)),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==-Inf),] = NA
# df.reg.use.log <- df.reg.use.log %>% drop_na(prot.imputed.log)
# # df.reg.use.log$prot.imputed.log
```

Now we generate scatter plot based on the combined factors

```
# Labeling
st_title <- paste0('Use case_when To Generate ifelse Groupings')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/amto/tibble/htmlpdfr/fs_tib_na.html')
st_caption <- paste0('mtcars dataset, ',
```

```
                      'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_casewhen_scatter <-
  ggplot(df_mtcars,
         aes(x=mpg, y=qsec,
             colour=mpg_qsec_am_grp,
             shape=mpg_qsec_am_grp)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_casewhen_scatter)
```



1.4.4.2 Generate NA values if Variables have Certain Value

In the example below, in one line:

1. generate a random standard normal vector
2. two set na methods:
   - if the value of the standard normal is negative, set value to -999, otherwise MPG, replace the value *-999* with *NA*
   - case_when only with type specific NA values
   - Assigning NA yields error in case_when
   - note we need to conform NA to type
3. generate new categorical variable based on NA condition using is.na with both string and numeric NAs jointly considered.
   - fake NA string to be printed on chart

```
# Get mtcars
df_mtcars <- mtcars
```

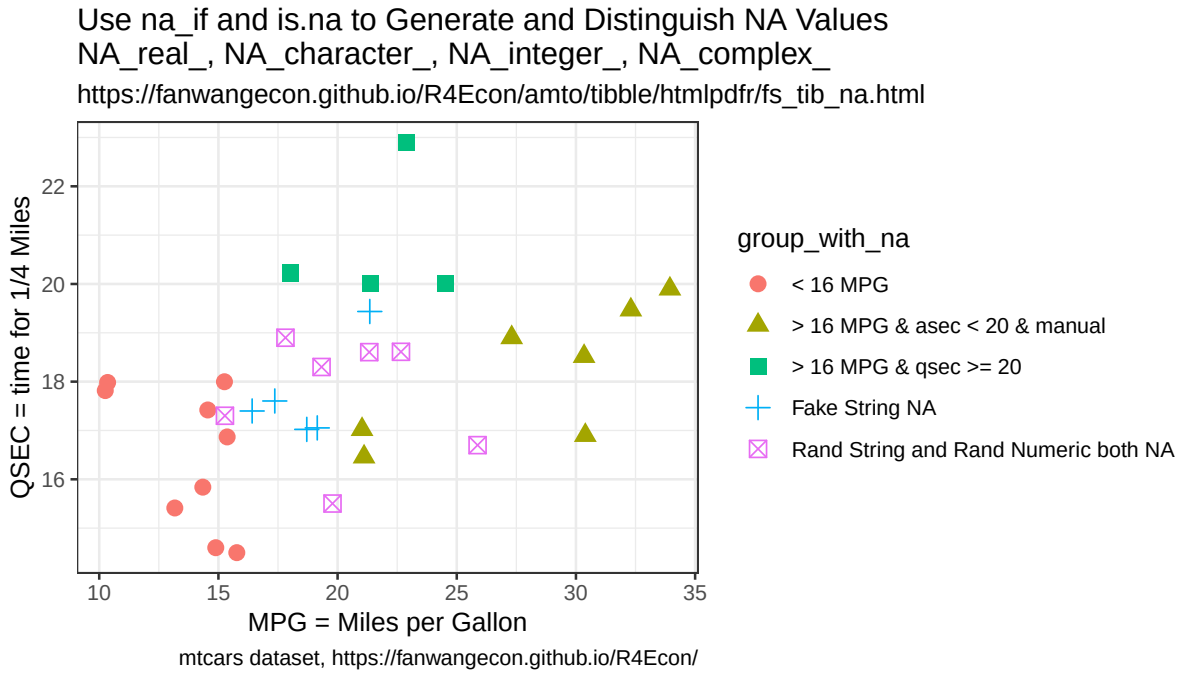
```
# Make some values of mpg randomly NA
# the NA has to conform to the type of the remaining values for the new variable
# NA_real_, NA_character_, NA_integer_, NA_complex_
set.seed(2341)
df_mtcars <- df_mtcars %>%
  mutate(mpg_wth_NA1 = na_if(
    case_when(
      rnorm(n(),mean=0,sd=1) < 0 ~ -999,
      TRUE ~ mpg),
    -999)) %>%
  mutate(mpg_wth_NA2 = case_when(
    rnorm(n(),mean=0,sd=1) < 0 ~ NA_real_,
    TRUE ~ mpg)) %>%
  mutate(mpg_wth_NA3 = case_when(
    rnorm(n(),mean=0,sd=1) < 0 ~ NA_character_,
    TRUE ~ "shock > 0 string"))

# Generate New Variables based on if mpg_wth_NA is NA or not
# same variable as above, but now first a category based on if NA
# And we generate a fake string "NA" variable, this is not NA
# the String NA allows for it to be printed on figure
df_mtcars <- df_mtcars %>%
  mutate(group_with_na =
           case_when(is.na(mpg_wth_NA2) & is.na(mpg_wth_NA3) ~
                       "Rand String and Rand Numeric both NA",
                     mpg < 16 ~ "< 16 MPG",
                     qsec >= 20 ~ "> 16 MPG & qsec >= 20",
                     am == 1 ~ "> 16 MPG & asec < 20 & manual",
                     TRUE ~ "Fake String NA"))

# show
kable(head(df_mtcars %>% select(starts_with('mpg')),13)) %>%
  kable_styling_fc()
```

mpg	mpg_wth_NA1	mpg_wth_NA2	mpg_wth_NA3
21.0	NA	NA	shock > 0 string
21.0	21.0	21.0	NA
22.8	NA	NA	NA
21.4	NA	21.4	NA
18.7	NA	18.7	NA
18.1	18.1	NA	shock > 0 string
14.3	14.3	NA	shock > 0 string
24.4	NA	24.4	NA
22.8	22.8	22.8	NA
19.2	19.2	NA	NA
17.8	NA	NA	NA
16.4	16.4	16.4	NA
17.3	NA	NA	shock > 0 string

```
# # Setting to NA
# df.reg.use <- df.reg.guat %>% filter(!!sym(var.mth) != 0)
# df.reg.use.log <- df.reg.use
# df.reg.use.log[which(is.nan(df.reg.use$prot.imputed.log)),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==-Inf),] = NA
# df.reg.use.log <- df.reg.use.log %>% drop_na(prot.imputed.log)
# # df.reg.use.log$prot.imputed.log
```

Now we generate scatter plot based on the combined factors, but now with the NA category

```
# Labeling
st_title <- paste0('Use na_if and is.na to Generate and Distinguish NA Values\n',
                   'NA_real_, NA_character_, NA_integer_, NA_complex_')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/amto/tibble/htmlpdfr/fs_tib_na.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_ifisna_scatter <-
  ggplot(df_mtcars,
         aes(x=mpg, y=qsec,
             colour=group_with_na,
             shape=group_with_na)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_ifisna_scatter)
```



1.4.4.3 Approximate Values Comparison

- r values almost the same
- all.equal

From numeric approximation, often values are very close, and should be set to equal. Use *isTRUE(all.equal)*. In the example below, we randomly generates four arrays. Two of the arrays have slightly higher variance, two arrays have slightly lower variance. They sd are to be 10 times below or 10 times above the tolerance comparison level. The values are not the same in any of the columns,

but by allowing for almost true given some tolerance level, in the low standard deviation case, the values differences are within tolerance, so they are equal.

This is an essential issue when dealing with optimization results.

```
# Set tolerance
tol_lvl = 1.5e-3
sd_lower_than_tol = tol_lvl/10
sd_higher_than_tol = tol_lvl*10

# larger SD
set.seed(123)
mt_runif_standard <- matrix(rnorm(10,mean=0,sd=sd_higher_than_tol), nrow=5, ncol=2)

# small SD
set.seed(123)
mt_rnorm_small_sd <- matrix(rnorm(10,mean=0,sd=sd_lower_than_tol), nrow=5, ncol=2)

# Generates Random Matirx
tb_rnorm_runif <- as_tibble(cbind(mt_rnorm_small_sd, mt_runif_standard))

# Are Variables the same, not for strict comparison
tb_rnorm_runif_approxi_same <- tb_rnorm_runif %>%
  mutate(V1_V2_ALMOST_SAME =
           case_when(isTRUE(all.equal(V1, V2, tolerance=tol_lvl)) ~
                       paste0('TOL=',sd_lower_than_tol,', SAME ALMOST'),
                     TRUE ~
                       paste0('TOL=',sd_lower_than_tol,', NOT SAME ALMOST'))) %>%
  mutate(V3_V4_ALMOST_SAME =
           case_when(isTRUE(all.equal(V3, V4, tolerance=tol_lvl)) ~
                       paste0('TOL=',sd_higher_than_tol,', SAME ALMOST'),
                     TRUE ~
                       paste0('TOL=',sd_higher_than_tol,', NOT SAME ALMOST')))

# Pring
kable(tb_rnorm_runif_approxi_same) %>% kable_styling_fc_wide()
```

V1	V2	V3	V4	V1_V2_ALMOST_SAME	V3_V4_ALMOST_SAME
-0.0000841	0.0002573	-0.0084071	0.0257260	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
-0.0000345	0.0000691	-0.0034527	0.0069137	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0002338	-0.0001898	0.0233806	-0.0189759	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000106	-0.0001030	0.0010576	-0.0103028	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000194	-0.0000668	0.0019393	-0.0066849	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST

1.4.5 String Dataframes

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

1.4.5.1 List of Strings to Tibble Datfare

There are several lists of strings, store them as variables in a dataframe.

```
# Sting data inputs
ls_st_abc <- c('a', 'b', 'c')
ls_st_efg <- c('e', 'f', 'g')
ls_st_opq <- c('o', 'p', 'q')
mt_str = cbind(ls_st_abc, ls_st_efg, ls_st_opq)

# Column Names
```

```
ar_st_varnames <- c('id','var1','var2','var3')

# Combine to tibble, add name col1, col2, etc.
tb_st_combine <- as_tibble(mt_str) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_st_combine) %>% kable_styling_fc()
```

id	var1	var2	var3
1	a	e	o
2	b	f	p
3	c	g	q

1.4.5.2 Find and Replace

Find and Replace in Dataframe.

```
# if string value is contained in variable
("bridex.B" %in% (df.reg.out.all$vars_var.y))
# if string value is not contained in variable:
# 1. type is variable name
# 2. Toyota|Mazda are strings to be excluded
filter(mtcars, !grepl('Toyota|Mazda', type))

# filter does not contain string
rs_hgt_prot_log_tidy %>% filter(!str_detect(term, 'prot'))
```

Chapter 2

Summarize Data

2.1 Counting Observation

2.1.1 Uncount

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

In some panel, there are N individuals, each observed for Y_i years. Given a dataset with two variables, the individual index, and the Y_i variable, expand the dataframe so that there is a row for each individual index's each unique year in the survey.

Search:

- r duplicate row by variable

Links:

- see: Create duplicate rows based on a variable

Algorithm:

1. generate testing frame, the individual attribute dataset with invariant information over panel
2. uncount, duplicate rows by years in survey
3. group and generate sorted index
4. add indiviual specific stat year to index

```
# 1. Array of Years in the Survey
ar_years_in_survey <- c(2,3,1,10,2,5)
ar_start_yaer <- c(1,2,3,1,1,1)
ar_end_year <- c(2,4,3,10,2,5)
mt_combine <- cbind(ar_years_in_survey, ar_start_yaer, ar_end_year)

# This is the individual attribute dataset, attributes that are invariant acrosss years
tb_indi_attributes <- as_tibble(mt_combine) %>% rowid_to_column(var = "ID")

# 2. Sort and generate variable equal to sorted index
tb_indi_panel <- tb_indi_attributes %>% uncount(ar_years_in_survey)

# 3. Panel now construct exactly which year in survey, note that all needed is sort index
# Note sorting not needed, all rows identical now
tb_indi_panel <- tb_indi_panel %>%
                    group_by(ID) %>%
                    mutate(yr_in_survey = row_number())

tb_indi_panel <- tb_indi_panel %>%
                    mutate(calendar_year = yr_in_survey + ar_start_yaer - 1)
```

```
# Show results Head 10
tb_indi_panel %>% head(10) %>%
  kable() %>%
  kable_styling_fc()
```

ID	ar_start_yaer	ar_end_year	yr_in_survey	calendar_year
1	1	2	1	1
1	1	2	2	2
2	2	4	1	2
2	2	4	2	3
2	2	4	3	4
3	3	3	1	3
4	1	10	1	1
4	1	10	2	2
4	1	10	3	3
4	1	10	4	4

2.2 Sorting, Indexing, Slicing

2.2.1 Sorting

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

2.2.1.1 Generate Sorted Index within Group with Repeating Values

There is a variable, sort by this variable, then generate index from 1 to N representing sorted values of this index. If there are repeating values, still assign index, different index each value.

- r generate index sort
- dplyr mutate equals index

```
# Sort and generate variable equal to sorted index
df_iris <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              select(Sepal.Length, Sepal.Len.Index, everything())

# Show results Head 10
df_iris %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Index	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	1	3.0	1.1	0.1	setosa
4.4	2	2.9	1.4	0.2	setosa
4.4	3	3.0	1.3	0.2	setosa
4.4	4	3.2	1.3	0.2	setosa
4.5	5	2.3	1.3	0.3	setosa
4.6	6	3.1	1.5	0.2	setosa
4.6	7	3.4	1.4	0.3	setosa
4.6	8	3.6	1.0	0.2	setosa
4.6	9	3.2	1.4	0.2	setosa
4.7	10	3.2	1.3	0.2	setosa

2.2.1.2 Populate Value from Lowest Index to All other Rows

We would like to calculate for example the ratio of each individual's highest to the the person with the lowest height in a dataset. We first need to generated sorted index from lowest to highest, and then populate the lowest height to all rows, and then divide.

Search Terms:

- r spread value to all rows from one row
- r other rows equal to the value of one row
- Conditional assignment of one variable to the value of one of two other variables
- dplyr mutate conditional
- dplyr value from one row to all rows
- dplyr mutate equal to value in another cell

Links:

- see: dplyr rank
- see: dplyr case_when

2.2.1.2.1 Short Method: mutate and min

We just want the lowest value to be in its own column, so that we can compute various statistics using the lowest value variable and the original variable.

```
# 1. Sort
df_iris_m1 <- iris %>% mutate(Sepal.Len.Lowest.all = min(Sepal.Length)) %>%
                select(Sepal.Length, Sepal.Len.Lowest.all, everything())


# Show results Head 10
df_iris_m1 %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Lowest.all	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	4.3	3.5	1.4	0.2	setosa
4.9	4.3	3.0	1.4	0.2	setosa
4.7	4.3	3.2	1.3	0.2	setosa
4.6	4.3	3.1	1.5	0.2	setosa
5.0	4.3	3.6	1.4	0.2	setosa
5.4	4.3	3.9	1.7	0.4	setosa
4.6	4.3	3.4	1.4	0.3	setosa
5.0	4.3	3.4	1.5	0.2	setosa
4.4	4.3	2.9	1.4	0.2	setosa
4.9	4.3	3.1	1.5	0.1	setosa

2.2.1.2.2 Long Method: row_number and case_when

This is the long method, using row_number, and case_when. The benefit of this method is that it generates several intermediate variables that might be useful. And the key final step is to set a new variable (A=*Sepal.Len.Lowest.all*) equal to another variable's (B=*Sepal.Length*'s) value at the index that satisfies condition based a third variable (C=*Sepal.Len.Index*).

```
# 1. Sort
# 2. generate index
# 3. value at lowest index (case_when)
# 4. spread value from lowest index to other rows
# Note step 4 does not require step 3
df_iris_m2 <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              mutate(Sepal.Len.Lowest.one =
```

```
                    case_when(row_number()==1 ~ Sepal.Length)) %>%
             mutate(Sepal.Len.Lowest.all =
                    Sepal.Length[Sepal.Len.Index==1]) %>%
             select(Sepal.Length, Sepal.Len.Index,
                   Sepal.Len.Lowest.one, Sepal.Len.Lowest.all)


# Show results Head 10
df_iris_m2 %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Index	Sepal.Len.Lowest.one	Sepal.Len.Lowest.all
4.3	1	4.3	4.3
4.4	2	NA	4.3
4.4	3	NA	4.3
4.4	4	NA	4.3
4.5	5	NA	4.3
4.6	6	NA	4.3
4.6	7	NA	4.3
4.6	8	NA	4.3
4.6	9	NA	4.3
4.7	10	NA	4.3

2.2.1.3 Generate Sorted Index based on Deviations

Generate Positive and Negative Index based on Ordered Deviation from some Number.

There is a variable that is continuous, substract a number from this variable, and generate index based on deviations. Think of the index as generating intervals indicating where the value lies. 0th index indicates the largest value in sequence that is smaller than or equal to number x, 1st index indicates the smallest value in sequence that is larger than number x.

The solution below is a little bit convoluated and long, there is likely a much quicker way. The process below shows various intermediary outputs that help arrive at deviation index *Sepal.Len.Devi.Index* from initial sorted index *Sepal.Len.Index*.

search:

- dplyr arrange ignore na
- dplyr index deviation from order number sequence
- dplyr index below above
- dplyr index order below above value

```
# 1. Sort and generate variable equal to sorted index
# 2. Plus or minus deviations from some value
# 3. Find the zero, which means, the number closests to zero including zero from the negative side
# 4. Find the index at the highest zero and below deviation point
# 5. Difference of zero index and original sorted index
sc_val_x <- 4.65
df_iris_deviate <- iris %>% arrange(Sepal.Length) %>%
             mutate(Sepal.Len.Index = row_number()) %>%
             mutate(Sepal.Len.Devi = (Sepal.Length - sc_val_x)) %>%
             mutate(Sepal.Len.Devi.Neg =
                    case_when(Sepal.Len.Devi <= 0 ~ (-1)*(Sepal.Len.Devi))) %>%
             arrange((Sepal.Len.Devi.Neg), desc(Sepal.Len.Index)) %>%
```

```
              mutate(Sepal.Len.Index.Zero =
                       case_when(row_number() == 1 ~ Sepal.Len.Index)) %>%
              mutate(Sepal.Len.Devi.Index =
                       Sepal.Len.Index - Sepal.Len.Index.Zero[row_number() == 1]) %>%
              arrange(Sepal.Len.Index) %>%
              select(Sepal.Length, Sepal.Len.Index, Sepal.Len.Devi,
                     Sepal.Len.Devi.Neg, Sepal.Len.Index.Zero, Sepal.Len.Devi.Index)


# Show results Head 10
df_iris_deviate %>% head(20) %>%
  kable() %>%
  kable_styling_fc_wide()
```

Sepal.Length	Sepal.Len.Index	Sepal.Len.Devi	Sepal.Len.Devi.Neg	Sepal.Len.Index.Zero	Sepal.Len.Devi.Index
4.3	1	-0.35	0.35	NA	-8
4.4	2	-0.25	0.25	NA	-7
4.4	3	-0.25	0.25	NA	-6
4.4	4	-0.25	0.25	NA	-5
4.5	5	-0.15	0.15	NA	-4
4.6	6	-0.05	0.05	NA	-3
4.6	7	-0.05	0.05	NA	-2
4.6	8	-0.05	0.05	NA	-1
4.6	9	-0.05	0.05	9	0
4.7	10	0.05	NA	NA	1
4.7	11	0.05	NA	NA	2
4.8	12	0.15	NA	NA	3
4.8	13	0.15	NA	NA	4
4.8	14	0.15	NA	NA	5
4.8	15	0.15	NA	NA	6
4.8	16	0.15	NA	NA	7
4.9	17	0.25	NA	NA	8
4.9	18	0.25	NA	NA	9
4.9	19	0.25	NA	NA	10
4.9	20	0.25	NA	NA	11

2.2.2 Group, Sort and Slice

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

2.2.2.1 Get Highest Values from Groups

There is a dataframe with a grouping variable. Get N rows that have the highest sorted value for another numeric variable. In the example below, group by *cyl* and get the cars with the lowest *mpg* in each *cyl* group.

Show all values.

```
kable(mtcars %>% arrange(cyl, mpg)) %>% kable_styling_fc()
```

Three groups min mpg each group:

```
# use mtcars: slice_head gets the lowest sorted value
df_groupby_top_mpg <- mtcars %>%
  arrange(cyl, mpg) %>%
  group_by(cyl) %>%
  slice_head(n=1) %>%
  select(cyl, mpg)

# display
kable(df_groupby_top_mpg) %>% kable_styling_fc()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2

cyl	mpg
4	21.4
6	17.8
8	10.4

2.3 Group Statistics

2.3.1 Cumulative Statistics within Group

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

2.3.1.1 Cumulative Mean

There is a dataset where there are different types of individuals, perhaps household size, that is the grouping variable. Within each group, we compute the incremental marginal propensity to consume for each additional check. We now also want to know the average propensity to consume up to each check considering all allocated checks. We needed to calculatet this for Nygaard, Sørensen and Wang (2021). This can be dealt with by using the cumall function.

Use the df_hgt_wgt as the testing dataset. In the example below, group by individual id, sort by survey month, and cumulative mean over the protein variable.

In the protein example

First select the testing dataset and variables.

```
# Load the REconTools Dataset df_hgt_wgt
data("df_hgt_wgt")
# str(df_hgt_wgt)

# Select several rows
df_hgt_wgt_sel <- df_hgt_wgt %>%
  filter(S.country == "Cebu") %>%
  select(indi.id, svymthRound, prot)
```

Second, arrange, groupby, and cumulative mean. The protein variable is protein for each survey month, from month 2 to higher as babies grow. The protein intake observed is increasing quickly, hence, the cumulative mean is lower than the observed value for the survey month of the baby.

```
# Group by indi.id and sort by protein
df_hgt_wgt_sel_cummean <- df_hgt_wgt_sel %>%
  arrange(indi.id, svymthRound) %>%
  group_by(indi.id) %>%
  mutate(prot_cummean = cummean(prot))

# display results
REconTools::ff_summ_percentiles(df_hgt_wgt_sel_cummean)
# display results
df_hgt_wgt_sel_cummean %>% filter(indi.id %in% c(17, 18)) %>%
  kable() %>% kable_styling_fc()
```

Third, in the basic implementation above, if an incremental month has NA, no values computed at that point or after. This is the case for individual 18 above. To ignore NA, we have, from this. Note how results for individual 18 changes.

```
# https://stackoverflow.com/a/49906718/8280804
# Group by indi.id and sort by protein
df_hgt_wgt_sel_cummean_noNA <- df_hgt_wgt_sel %>%
  arrange(indi.id, svymthRound) %>%
  group_by(indi.id, isna = is.na(prot)) %>%
  mutate(prot_cummean = ifelse(isna, NA, cummean(prot)))

# display results
df_hgt_wgt_sel_cummean_noNA %>% filter(indi.id %in% c(17, 18)) %>%
  kable() %>% kable_styling_fc()
```

2.3.2 Groups Statistics

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

2.3.2.1 Aggrgate Groups only Unique Group and Count

There are two variables that are numeric, we want to find all the unique groups of these two variables in a dataset and count how many times each unique group occurs

- r unique occurrence of numeric groups
- How to add count of unique values by group to R data.frame

```
# Numeric value combinations unique Groups
vars.group <- c('hgt0', 'wgt0')

# dataset subsetting
df_use <- df_hgt_wgt %>% select(!!!syms(c(vars.group))) %>%
            mutate(hgt0 = round(hgt0/5)*5, wgt0 = round(wgt0/2000)*2000) %>%
```

indi.id	svymthRound	prot	prot_cummean
17	0	0.5	0.5000000
17	2	0.7	0.6000000
17	4	0.5	0.5666667
17	6	0.5	0.5500000
17	8	6.1	1.6600000
17	10	5.0	2.2166667
17	12	6.4	2.8142857
17	14	20.1	4.9750000
17	16	20.1	6.6555556
17	18	23.0	8.2900000
17	20	24.9	9.8000000
17	22	20.1	10.6583333
17	24	10.1	10.6153846
17	102	NA	NA
17	138	NA	NA
17	187	NA	NA
17	224	NA	NA
17	258	NA	NA
18	0	1.2	1.2000000
18	2	4.7	2.9500000
18	4	17.2	7.7000000
18	6	18.6	10.4250000
18	8	NA	NA
18	10	16.8	NA
18	12	NA	NA
18	14	NA	NA
18	16	NA	NA
18	18	NA	NA
18	20	NA	NA
18	22	15.7	NA
18	24	22.5	NA
18	102	NA	NA
18	138	NA	NA
18	187	NA	NA
18	224	NA	NA
18	258	NA	NA

```
            drop_na()

# Group, count and generate means for each numeric variables
# mutate_at(vars.group, funs(as.factor(.))) %>%
df.group.count <- df_use %>% group_by(!!!syms(vars.group)) %>%
                    arrange(!!!syms(vars.group)) %>%
                    summarise(n_obs_group=n())

# Show results Head 10
df.group.count %>% kable() %>% kable_styling_fc()
```

2.3.2.2 Aggrgate Groups only Unique Group Show up With Means

Several variables that are grouping identifiers. Several variables that are values which mean be unique for each group members. For example, a Panel of income for N households over T years with also household education information that is invariant over time. Want to generate a dataset where the unit of observation are households, rather than household years. Take average of all numeric variables that are household and year specific.

indi.id	svymthRound	prot	isna	prot_cummean
17	0	0.5	FALSE	0.5000000
17	2	0.7	FALSE	0.6000000
17	4	0.5	FALSE	0.5666667
17	6	0.5	FALSE	0.5500000
17	8	6.1	FALSE	1.6600000
17	10	5.0	FALSE	2.2166667
17	12	6.4	FALSE	2.8142857
17	14	20.1	FALSE	4.9750000
17	16	20.1	FALSE	6.6555556
17	18	23.0	FALSE	8.2900000
17	20	24.9	FALSE	9.8000000
17	22	20.1	FALSE	10.6583333
17	24	10.1	FALSE	10.6153846
17	102	NA	TRUE	NA
17	138	NA	TRUE	NA
17	187	NA	TRUE	NA
17	224	NA	TRUE	NA
17	258	NA	TRUE	NA
18	0	1.2	FALSE	1.2000000
18	2	4.7	FALSE	2.9500000
18	4	17.2	FALSE	7.7000000
18	6	18.6	FALSE	10.4250000
18	8	NA	TRUE	NA
18	10	16.8	FALSE	11.7000000
18	12	NA	TRUE	NA
18	14	NA	TRUE	NA
18	16	NA	TRUE	NA
18	18	NA	TRUE	NA
18	20	NA	TRUE	NA
18	22	15.7	FALSE	12.3666667
18	24	22.5	FALSE	13.8142857
18	102	NA	TRUE	NA
18	138	NA	TRUE	NA
18	187	NA	TRUE	NA
18	224	NA	TRUE	NA
18	258	NA	TRUE	NA

hgt0	wgt0	n_obs_group
40	2000	122
45	2000	4586
45	4000	470
50	2000	9691
50	4000	13106
55	2000	126
55	4000	1900
60	6000	18

A complicating factor potentially is that the number of observations differ within group, for example, income might be observed for all years for some households but not for other households.

- r dplyr aggregate group average
- Aggregating and analyzing data with dplyr
- column can't be modified because it is a grouping variable
- see also: Aggregating and analyzing data with dplyr

```
# In the df_hgt_wgt from R4Econ, there is a country id, village id,
# and individual id, and various other statistics
vars.group <- c('S.country', 'vil.id', 'indi.id')
vars.values <- c('hgt', 'momEdu')

# dataset subsetting
df_use <- df_hgt_wgt %>% select(!!!syms(c(vars.group, vars.values)))

# Group, count and generate means for each numeric variables
df.group <- df_use %>% group_by(!!!syms(vars.group)) %>%
            arrange(!!!syms(vars.group)) %>%
            summarise_if(is.numeric,
                         funs(mean = mean(., na.rm = TRUE),
                              sd = sd(., na.rm = TRUE),
                              n = sum(is.na(.)==0)))

# Show results Head 10
df.group %>% head(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	hgt_mean	momEdu_mean	hgt_sd	momEdu_sd	hgt_n	momEdu_n
Cebu	1	1	61.80000	5.3	9.520504	0	7	18
Cebu	1	2	68.86154	7.1	9.058931	0	13	18
Cebu	1	3	80.45882	9.4	29.894231	0	17	18
Cebu	1	4	88.10000	13.9	35.533166	0	18	18
Cebu	1	5	97.70556	11.3	41.090366	0	18	18
Cebu	1	6	87.49444	7.3	35.586439	0	18	18
Cebu	1	7	90.79412	10.4	38.722385	0	17	18
Cebu	1	8	68.45385	13.5	10.011961	0	13	18
Cebu	1	9	86.21111	10.4	35.126057	0	18	18
Cebu	1	10	87.67222	10.5	36.508127	0	18	18

```
# Show results Head 10
df.group %>% tail(10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	hgt_mean	momEdu_mean	hgt_sd	momEdu_sd	hgt_n	momEdu_n
Guatemala	14	2014	66.97000	NaN	8.967974	NA	10	0
Guatemala	14	2015	71.71818	NaN	11.399984	NA	11	0
Guatemala	14	2016	66.33000	NaN	9.490352	NA	10	0
Guatemala	14	2017	76.40769	NaN	14.827871	NA	13	0
Guatemala	14	2018	74.55385	NaN	12.707846	NA	13	0
Guatemala	14	2019	70.47500	NaN	11.797390	NA	12	0
Guatemala	14	2020	60.28750	NaN	7.060036	NA	8	0
Guatemala	14	2021	84.96000	NaN	15.446193	NA	10	0
Guatemala	14	2022	79.38667	NaN	15.824749	NA	15	0
Guatemala	14	2023	66.50000	NaN	8.613113	NA	8	0

2.3.3 One Variable Group Summary

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

There is a categorical variable (based on one or the interaction of multiple variables), there is a continuous variable, obtain statistics for the continuous variable conditional on the categorical variable, but also unconditionally.

Store results in a matrix, but also flatten results wide to row with appropriate keys/variable-names for all group statistics.

Pick which statistics to be included in final wide row

2.3.3.1 Build Program

```
# Single Variable Group Statistics (also generate overall statistics)
ff_summ_by_group_summ_one <- function(
  df, vars.group, var.numeric, str.stats.group = 'main',
  str.stats.specify = NULL, boo.overall.stats = TRUE){

  # List of statistics
  # https://rdrr.io/cran/dplyr/man/summarise.html
  strs.center <- c('mean', 'median')
  strs.spread <- c('sd', 'IQR', 'mad')
  strs.range <- c('min', 'max')
  strs.pos <- c('first', 'last')
  strs.count <- c('n_distinct')

  # Grouping of Statistics
  if (missing(str.stats.specify)) {
    if (str.stats.group == 'main') {
      strs.all <- c('mean', 'min', 'max', 'sd')
    }
    if (str.stats.group == 'all') {
      strs.all <- c(strs.center, strs.spread, strs.range, strs.pos, strs.count)
    }
  } else {
    strs.all <- str.stats.specify
  }

  # Start Transform
  df <- df %>% drop_na() %>%
    mutate(!!(var.numeric) := as.numeric(!!sym(var.numeric)))

  # Overall Statistics
  if (boo.overall.stats) {
    df.overall.stats <- df %>%
      summarize_at(vars(var.numeric), funs(!!!strs.all))
    if (length(strs.all) == 1) {
      # give it a name, otherwise if only one stat, name of stat not saved
      df.overall.stats <- df.overall.stats %>%
        rename(!!strs.all := !!sym(var.numeric))
    }
    names(df.overall.stats) <-
      paste0(var.numeric, '.', names(df.overall.stats))
  }

  # Group Sort
  df.select <- df %>%
    group_by(!!!syms(vars.group)) %>%
    arrange(!!!syms(c(vars.group, var.numeric)))

  # Table of Statistics
  df.table.grp.stats <- df.select %>%
    summarize_at(vars(var.numeric), funs(!!!strs.all))
```

```
  # Add Stat Name
  if (length(strs.all) == 1) {
    # give it a name, otherwise if only one stat, name of stat not saved
    df.table.grp.stats <- df.table.grp.stats %>%
      rename(!!strs.all := !!sym(var.numeric))
  }


  # Row of Statistics
  str.vars.group.combine <- paste0(vars.group, collapse='_')
  if (length(vars.group) == 1) {
    df.row.grp.stats <- df.table.grp.stats %>%
      mutate(!!(str.vars.group.combine) :=
               paste0(var.numeric, '.',
                      vars.group, '.g',
                      (!!!syms(vars.group)))) %>%
      gather(variable, value, -one_of(vars.group)) %>%
      unite(str.vars.group.combine, c(str.vars.group.combine, 'variable')) %>%
      spread(str.vars.group.combine, value)
  } else {
    df.row.grp.stats <- df.table.grp.stats %>%
      mutate(vars.groups.combine :=
               paste0(paste0(vars.group, collapse='.')),
             !!(str.vars.group.combine) :=
               paste0(interaction(!!!(syms(vars.group))))) %>%
      mutate(!!(str.vars.group.combine) :=
               paste0(var.numeric, '.', vars.groups.combine, '.',
                      (!!sym(str.vars.group.combine)))) %>%
      ungroup() %>%
      select(-vars.groups.combine, -one_of(vars.group)) %>%
      gather(variable, value, -one_of(str.vars.group.combine)) %>%
      unite(str.vars.group.combine, c(str.vars.group.combine, 'variable')) %>%
      spread(str.vars.group.combine, value)
  }

  # Clean up name strings
  names(df.table.grp.stats) <-
    gsub(x = names(df.table.grp.stats),pattern = "_", replacement = "\\.")
  names(df.row.grp.stats) <-
    gsub(x = names(df.row.grp.stats),pattern = "_", replacement = "\\.")

  # Return
  list.return <-
    list(df_table_grp_stats = df.table.grp.stats,
         df_row_grp_stats = df.row.grp.stats)

  # Overall Statistics, without grouping
  if (boo.overall.stats) {
    df.row.stats.all <- c(df.row.grp.stats, df.overall.stats)
    list.return <- append(list.return,
                          list(df_overall_stats = df.overall.stats,
                               df_row_stats_all = df.row.stats.all))
  }

  # Return
  return(list.return)

}
```

2.3.3.2 Test

Load data and test

```
# Library
library(tidyverse)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

2.3.3.2.1 Function Testing By Gender Groups Need two variables, a group variable that is a factor, and a numeric

```
vars.group <- 'sex'
var.numeric <- 'hgt'
```

```
df.select <- df %>% select(one_of(vars.group, var.numeric)) %>% drop_na()
```

Main Statistics:

```
# Single Variable Group Statistics
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.group = 'main')$df_table_grp_stats
```

Specify Two Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean', 'sd'))$df_table_grp_stats
```

Specify One Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean'))$df_table_grp_stats
```

2.3.3.2.2 Function Testing By Country and Gender Groups Need two variables, a group variable that is a factor, and a numeric. Now joint grouping variables.

```
vars.group <- c('S.country', 'sex')
var.numeric <- 'hgt'
```

```
df.select <- df %>% select(one_of(vars.group, var.numeric)) %>% drop_na()
```

Main Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.group = 'main')$df_table_grp_stats
```

Specify Two Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean', 'sd'))$df_table_grp_stats
```

Specify One Specific Statistics:

```
ff_summ_by_group_summ_one(
  df.select, vars.group = vars.group, var.numeric = var.numeric,
  str.stats.specify = c('mean'))$df_table_grp_stats
```

2.3.4 Nested within Group Stats

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

By Multiple within Individual Groups Variables, Averages for All Numeric Variables within All Groups of All Group Variables (Long to very Wide). Suppose you have an individual level final outcome. The individual is observed for N periods, where each period the inputs differ. What inputs impacted the final outcome?

Suppose we can divide N periods in which the individual is in the data into a number of years, a number of semi-years, a number of quarters, or uneven-staggered lengths. We might want to generate averages across individuals and within each of these different possible groups averages of inputs.

Then we want to version of the data where each row is an individual, one of the variables is the final outcome, and the other variables are these different averages: averages for the 1st, 2nd, 3rd year in which indivdiual is in data, averages for 1st, ..., final quarter in which indivdiual is in data.

2.3.4.1 Build Function

This function takes as inputs:

1. **vars.not.groups2avg**: a list of variables that are not the within-indivdiual or across-individual grouping variables, but the variables we want to average over. Withnin indivdiual grouping averages will be calculated for these variables using the not-listed variables as within indivdiual groups (excluding vars.indi.grp groups).
2. **vars.indi.grp**: a list or individual variables, and also perhaps villages, province, etc id variables that are higher than individual ID. Note the groups are are ACROSS individual higher level group variables.
3. the remaining variables are all within individual grouping variables.

the function output is a dataframe:

1. each row is an individual
2. initial variables individual ID and across individual groups from *vars.indi.grp*.
3. other variables are all averages for the variables in *vars.not.groups2avg*
   - if there are 2 within individual group variables, and the first has 3 groups (years), the second has 6 groups (semi-years), then there would be 9 average variables.
   - each average variables has the original variable name from vars.not.groups2avg plus the name of the within individual grouping variable, and at the end 'c_x', where x is a integer representing the category within the group (if 3 years, x=1, 2, 3)

```
# Data Function
# https://fanwangecon.github.io/R4Econ/summarize/summ/ByGroupsSummWide.html
f.by.groups.summ.wide <- function(df.groups.to.average,
                                  vars.not.groups2avg,
                                  vars.indi.grp = c('S.country','ID'),
                                  display=TRUE) {

# 1. generate categoricals for full year (m.12), half year (m.6), quarter year (m.4)
# 2. generate categoricals also for uneven years (m12t14) using
#  stagger (+2 rather than -1)
# 3. reshape wide to long, so that all categorical date groups appear in var=value,
    # and categories in var=variable
# 4. calculate mean for all numeric variables for all date groups
# 5. combine date categorical variable and value, single var:
    # m.12.c1= first year average from m.12 averaging
```

```
######## ######## ######## ######## #######
# Step 1
######## ######## ######## ######## #######
# 1. generate categoricals for full year (m.12), half year (m.6), quarter year (m.4)
# 2. generate categoricals also for uneven years (m12t14) using stagger
#  (+2 rather than -1)

######## ######## ######## ######## #######
# S2: reshape wide to long, so that all categorical date groups appear in var=value,
# and categories in var=variable; calculate mean for all
# numeric variables for all date groups
######## ######## ######## ######## #######
df.avg.long <- df.groups.to.average %>%
       gather(variable, value, -one_of(c(vars.indi.grp,
                                         vars.not.groups2avg))) %>%
       group_by(!!!syms(vars.indi.grp), variable, value) %>%
       summarise_if(is.numeric, funs(mean(., na.rm = TRUE)))

if (display){
  dim(df.avg.long)
  options(repr.matrix.max.rows=10, repr.matrix.max.cols=20)
  print(df.avg.long)
}

######## ######## ######## ######## #######
# S3 combine date categorical variable and value, single var:
# m.12.c1= first year average from m.12 averaging; to do this make
# data even longer first
######## ######## ######## ######## #######

# We already have the averages, but we want them to show up as variables,
    # mean for each group of each variable.
df.avg.allvars.wide <- df.avg.long %>%
   ungroup() %>%
   mutate(all_m_cate = paste0(variable, '_c', value)) %>%
   select(all_m_cate, everything(), -variable, -value) %>%
   gather(variable, value, -one_of(vars.indi.grp), -all_m_cate) %>%
   unite('var_mcate', variable, all_m_cate) %>%
   spread(var_mcate, value)

if (display){
  dim(df.avg.allvars.wide)
  options(repr.matrix.max.rows=10, repr.matrix.max.cols=10)
  print(df.avg.allvars.wide)
}

return(df.avg.allvars.wide)
}
```

2.3.4.2 Test Program

In our sample dataset, the number of nutrition/height/income etc information observed within each country and month of age group are different. We have a panel dataset for children observed over different months of age.

We have two key grouping variables: 1. country: data are observed for guatemala and cebu 2. month-age (survey month round=svymthRound): different months of age at which each individual child is observed

A child could be observed for many months, or just a few months. A child's height information could be observed for more months-of-age than nutritional intake information. We eventually want to run regressions where the outcome is height/weight and the input is nutrition. The regressions will be at the month-of-age level. We need to know how many times different variables are observed at the month-of-age level.

```
# Library
library(tidyverse)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

2.3.4.2.1 Generate Within Individual Groups In the data, children are observed for different number of months since birth. We want to calculate quarterly, semi-year, annual, etc average nutritional intakes. First generate these within-individual grouping variables. We can also generate uneven-staggered calendar groups as shown below.

```
mth.var <- 'svymthRound'
df.groups.to.average<- df %>%
        filter(!!sym(mth.var) >= 0 & !!sym(mth.var) <= 24)  %>%
        mutate(m12t24=(floor((!!sym(mth.var) - 12) %/% 14) + 1),
               m8t24=(floor((!!sym(mth.var) - 8) %/% 18) + 1),
               m12 = pmax((floor((!!sym(mth.var)-1) %/% 12) + 1), 1),
               m6 = pmax((floor((!!sym(mth.var)-1) %/% 6) + 1), 1),
               m3 = pmax((floor((!!sym(mth.var)-1) %/% 3) + 1), 1))
```

```
# Show Results
options(repr.matrix.max.rows=30, repr.matrix.max.cols=20)
vars.arrange <- c('S.country','indi.id','svymthRound')
vars.groups.within.indi <- c('m12t24', 'm8t24', 'm12', 'm6', 'm3')
as.tibble(df.groups.to.average %>%
          group_by(!!!syms(vars.arrange)) %>%
          arrange(!!!syms(vars.arrange)) %>%
          select(!!!syms(vars.arrange), !!!syms(vars.groups.within.indi)))
```

2.3.4.2.2 Within Group Averages With the within-group averages created, we can generate averages for all variables within these groups.

```
vars.not.groups2avg <- c('prot', 'cal')
vars.indi.grp <- c('S.country', 'indi.id')
vars.groups.within.indi <- c('m12t24', 'm8t24', 'm12', 'm6', 'm3')

df.groups.to.average.select <- df.groups.to.average %>%
                        select(one_of(c(vars.indi.grp,
                                        vars.not.groups2avg,
                                        vars.groups.within.indi)))
df.avg.allvars.wide <- f.by.groups.summ.wide(df.groups.to.average.select,
                                             vars.not.groups2avg,
                                             vars.indi.grp, display=FALSE)
```

This is the tabular version of results

```
dim(df.avg.allvars.wide)
```

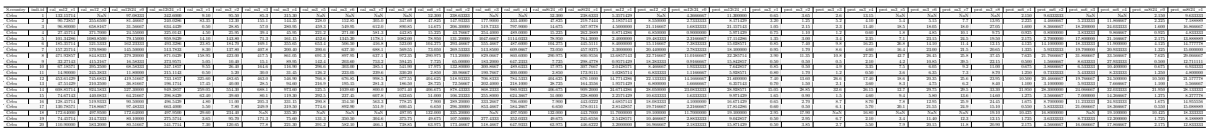
```
## [1] 2023   38
```

```
names(df.avg.allvars.wide)
```

```
##  [1] "S.country"      "indi.id"        "cal_m12_c1"     "cal_m12_c2"     "cal_m12t24_c0"
##  [6] "cal_m12t24_c1"  "cal_m3_c1"      "cal_m3_c2"      "cal_m3_c3"      "cal_m3_c4"
```

```
## [11] "cal_m3_c5"      "cal_m3_c6"      "cal_m3_c7"      "cal_m3_c8"      "cal_m6_c1"
## [16] "cal_m6_c2"      "cal_m6_c3"      "cal_m6_c4"      "cal_m8t24_c0"   "cal_m8t24_c1"
## [21] "prot_m12_c1"    "prot_m12_c2"    "prot_m12t24_c0" "prot_m12t24_c1" "prot_m3_c1"
## [26] "prot_m3_c2"     "prot_m3_c3"     "prot_m3_c4"     "prot_m3_c5"     "prot_m3_c6"
## [31] "prot_m3_c7"     "prot_m3_c8"     "prot_m6_c1"     "prot_m6_c2"     "prot_m6_c3"
## [36] "prot_m6_c4"     "prot_m8t24_c0"  "prot_m8t24_c1"
```

```
df.avg.allvars.wide[1:20,] %>% kable() %>% kable_styling_fc_wide()
```



2.4 Distributional Statistics

2.4.1 Histogram

2.4.1.1 Generate Test Score Dataset

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

- r generate text string as csv
- r tibble matrix hand input

First, we will generate a test score dataset, directly from string. Below we type line by line a dataset with four variables in comma separated (csv) format, where the first row includes the variables names. These texts could be stored in a separate file, or they could be directly included in code and read in as csv

```
ar_test_scores_ec3 <- c(107.72,101.28,105.92,109.31,104.27,110.27,91.92846154,81.8,109.0071429,103.0
ar_test_scores_ec1 <- c(101.72,101.28,99.92,103.31,100.27,104.27,90.23615385,77.8,103.4357143,97.07,
mt_test_scores <- cbind(ar_test_scores_ec1, ar_test_scores_ec3)
ar_st_varnames <- c('course_total_ec1p','course_total_ec3p')
tb_final_twovar <- as_tibble(mt_test_scores) %>% rename_all(~c(ar_st_varnames))
summary(tb_final_twovar)
```

2.4.1.1.1 A Dataset with only Two Continuous Variable

```
##  course_total_ec1p course_total_ec3p
##  Min.   : 40.48    Min.   : 44.23
##  1st Qu.: 76.46    1st Qu.: 79.91
##  Median : 86.35    Median : 89.28
##  Mean   : 83.88    Mean   : 87.90
##  3rd Qu.: 95.89    3rd Qu.:100.75
##  Max.   :104.27    Max.   :112.22
```

```
ff_summ_percentiles(df = tb_final_twovar, bl_statsasrows = TRUE, col2varname = FALSE)
```

```
ar_final_scores <- c(94.28442509,95.68817475,97.25219512,77.89268293,95.08795497,93.27380863,92.3,84
mt_test_scores <- cbind(seq(1,length(ar_final_scores)), ar_final_scores)
ar_st_varnames <- c('index', 'course_final')
tb_onevar <- as_tibble(mt_test_scores) %>% rename_all(~c(ar_st_varnames))
summary(tb_onevar)
```

2.4.1.1.2 A Dataset with one Continuous Variable and Histogram

```
##       index       course_final
```

```
##  Min.   : 1.0   Min.   :  2.293
##  1st Qu.:12.5   1st Qu.: 76.372
##  Median :24.0   Median : 86.959
##  Mean   :24.0   Mean   : 82.415
##  3rd Qu.:35.5   3rd Qu.: 94.686
##  Max.   :47.0   Max.   :100.898
```

```
ff_summ_percentiles(df = tb_onevar, bl_statsasrows = TRUE, col2varname = FALSE)
```

```
#load in data empirically by hand
txt_test_data <- "init_prof, later_prof, class_id, exam_score
 'SW', 'SW', 1, 102
 'SW', 'SW', 1, 102
 'SW', 'SW', 1, 101
 'SW', 'SW', 1, 100
 'SW', 'SW', 1, 100
 'SW', 'SW', 1, 99
 'SW', 'SW', 1, 98.5
 'SW', 'SW', 1, 98.5
 'SW', 'SW', 1, 97
 'SW', 'SW', 1, 95
 'SW', 'SW', 1, 94
 'SW', 'SW', 1, 91
 'SW', 'SW', 1, 91
 'SW', 'SW', 1, 90
 'SW', 'SW', 1, 89
 'SW', 'SW', 1, 88.5
 'SW', 'SW', 1, 88
 'SW', 'SW', 1, 87
 'SW', 'SW', 1, 87
 'SW', 'SW', 1, 87
 'SW', 'SW', 1, 86
 'SW', 'SW', 1, 86
 'SW', 'SW', 1, 84
 'SW', 'SW', 1, 82
 'SW', 'SW', 1, 78.5
 'SW', 'SW', 1, 76
 'SW', 'SW', 1, 72
 'SW', 'SW', 1, 70.5
 'SW', 'SW', 1, 67.5
 'SW', 'SW', 1, 67.5
 'SW', 'SW', 1, 67
 'SW', 'SW', 1, 63.5
 'SW', 'SW', 1, 60
 'SW', 'SW', 1, 59
 'SW', 'SW', 1, 44.5
 'SW', 'SW', 1, 44
 'SW', 'SW', 1, 42.5
 'SW', 'SW', 1, 40.5
 'SW', 'SW', 1, 40.5
 'SW', 'SW', 1, 36.5
 'SW', 'SW', 1, 35.5
 'SW', 'SW', 1, 21.5
 'SW', 'SW', 1, 4
 'MP', 'MP', 2, 105
 'MP', 'MP', 2, 103
 'MP', 'MP', 2, 102
```

```
'MP', 'MP', 2, 101
'MP', 'MP', 2, 101
'MP', 'MP', 2, 100.5
'MP', 'MP', 2, 100
'MP', 'MP', 2, 99
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 97
'MP', 'MP', 2, 96
'MP', 'MP', 2, 95
'MP', 'MP', 2, 91
'MP', 'MP', 2, 89
'MP', 'MP', 2, 85
'MP', 'MP', 2, 84
'MP', 'MP', 2, 84
'MP', 'MP', 2, 84
'MP', 'MP', 2, 83.5
'MP', 'MP', 2, 82.5
'MP', 'MP', 2, 81.5
'MP', 'MP', 2, 80.5
'MP', 'MP', 2, 80
'MP', 'MP', 2, 77
'MP', 'MP', 2, 77
'MP', 'MP', 2, 75
'MP', 'MP', 2, 75
'MP', 'MP', 2, 71
'MP', 'MP', 2, 70
'MP', 'MP', 2, 68
'MP', 'MP', 2, 63
'MP', 'MP', 2, 56
'MP', 'MP', 2, 56
'MP', 'MP', 2, 55.5
'MP', 'MP', 2, 49.5
'MP', 'MP', 2, 48.5
'MP', 'MP', 2, 47.5
'MP', 'MP', 2, 44.5
'MP', 'MP', 2, 34.5
'MP', 'MP', 2, 29.5
'CA', 'MP', 3, 103
'CA', 'MP', 3, 103
'CA', 'MP', 3, 101
'CA', 'MP', 3, 96.5
'CA', 'MP', 3, 93.5
'CA', 'MP', 3, 93
'CA', 'MP', 3, 93
'CA', 'MP', 3, 92
'CA', 'MP', 3, 90
'CA', 'MP', 3, 90
'CA', 'MP', 3, 89
'CA', 'MP', 3, 86.5
'CA', 'MP', 3, 84.5
'CA', 'MP', 3, 83
'CA', 'MP', 3, 83
'CA', 'MP', 3, 82
'CA', 'MP', 3, 78
'CA', 'MP', 3, 75
'CA', 'MP', 3, 74.5
```

```
 'CA', 'MP', 3, 70
 'CA', 'MP', 3, 54.5
 'CA', 'MP', 3, 52
 'CA', 'MP', 3, 50
 'CA', 'MP', 3, 42
 'CA', 'MP', 3, 36.5
 'CA', 'MP', 3, 28
 'CA', 'MP', 3, 26
 'CA', 'MP', 3, 11
 'CA', 'SN', 4, 103
 'CA', 'SN', 4, 103
 'CA', 'SN', 4, 102
 'CA', 'SN', 4, 102
 'CA', 'SN', 4, 101
 'CA', 'SN', 4, 100
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 98
 'CA', 'SN', 4, 95
 'CA', 'SN', 4, 95
 'CA', 'SN', 4, 92.5
 'CA', 'SN', 4, 92
 'CA', 'SN', 4, 91
 'CA', 'SN', 4, 90
 'CA', 'SN', 4, 85.5
 'CA', 'SN', 4, 84
 'CA', 'SN', 4, 82.5
 'CA', 'SN', 4, 81
 'CA', 'SN', 4, 77.5
 'CA', 'SN', 4, 77
 'CA', 'SN', 4, 72
 'CA', 'SN', 4, 71.5
 'CA', 'SN', 4, 69
 'CA', 'SN', 4, 68.5
 'CA', 'SN', 4, 68
 'CA', 'SN', 4, 67
 'CA', 'SN', 4, 65.5
 'CA', 'SN', 4, 62.5
 'CA', 'SN', 4, 62
 'CA', 'SN', 4, 61.5
 'CA', 'SN', 4, 61
 'CA', 'SN', 4, 57.5
 'CA', 'SN', 4, 54
 'CA', 'SN', 4, 52.5
 'CA', 'SN', 4, 51
 'CA', 'SN', 4, 50.5
 'CA', 'SN', 4, 50
 'CA', 'SN', 4, 49
 'CA', 'SN', 4, 43
 'CA', 'SN', 4, 39.5
 'CA', 'SN', 4, 32.5
 'CA', 'SN', 4, 25.5
 'CA', 'SN', 4, 18"

csv_test_data = read.csv(text=txt_test_data, header=TRUE)
ar_st_varnames <- c('first_half_professor',
                    'second_half_professor',
                    'course_id', 'exam_score')
```

```
tb_test_data <- as_tibble(csv_test_data) %>%
  rename_all(~c(ar_st_varnames))
summary(tb_test_data)
```

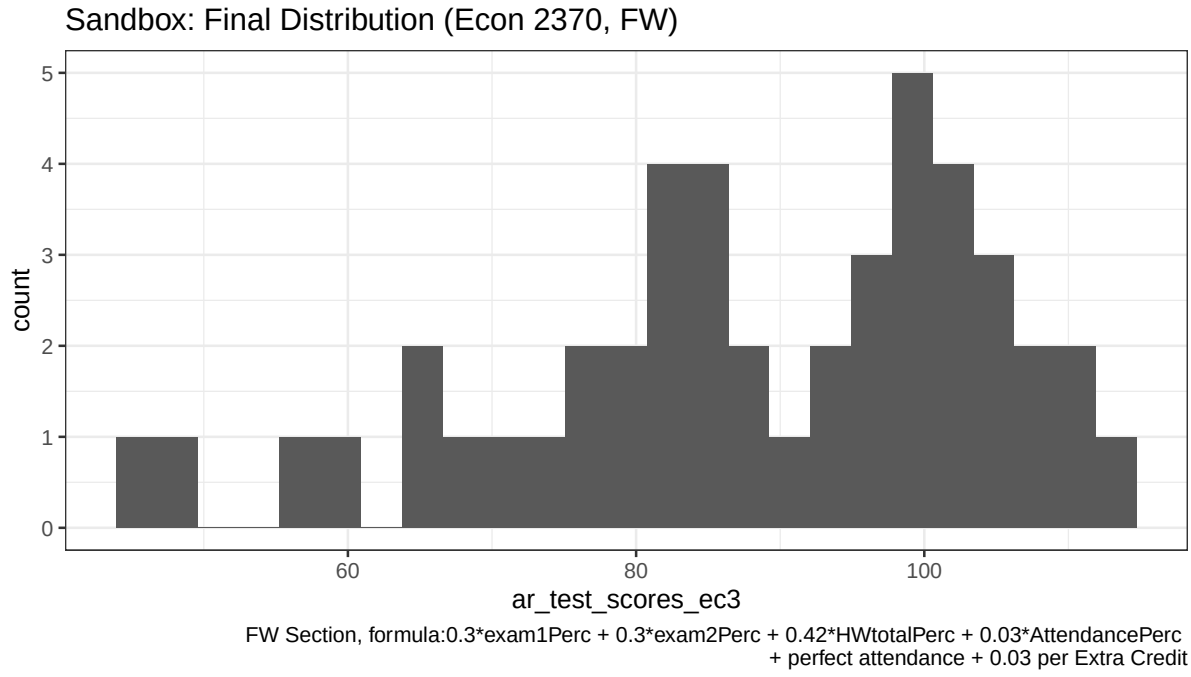
2.4.1.1.3 A Dataset with Multiple Variables

```
##  first_half_professor second_half_professor   course_id       exam_score
##  Length:157           Length:157            Min.   :1.000   Min.   :  4.00
##  Class :character     Class :character      1st Qu.:1.000   1st Qu.: 60.00
##  Mode  :character     Mode  :character      Median :2.000   Median : 82.00
##                                             Mean   :2.465   Mean   : 75.08
##                                             3rd Qu.:4.000   3rd Qu.: 94.00
##                                             Max.   :4.000   Max.   :105.00
```

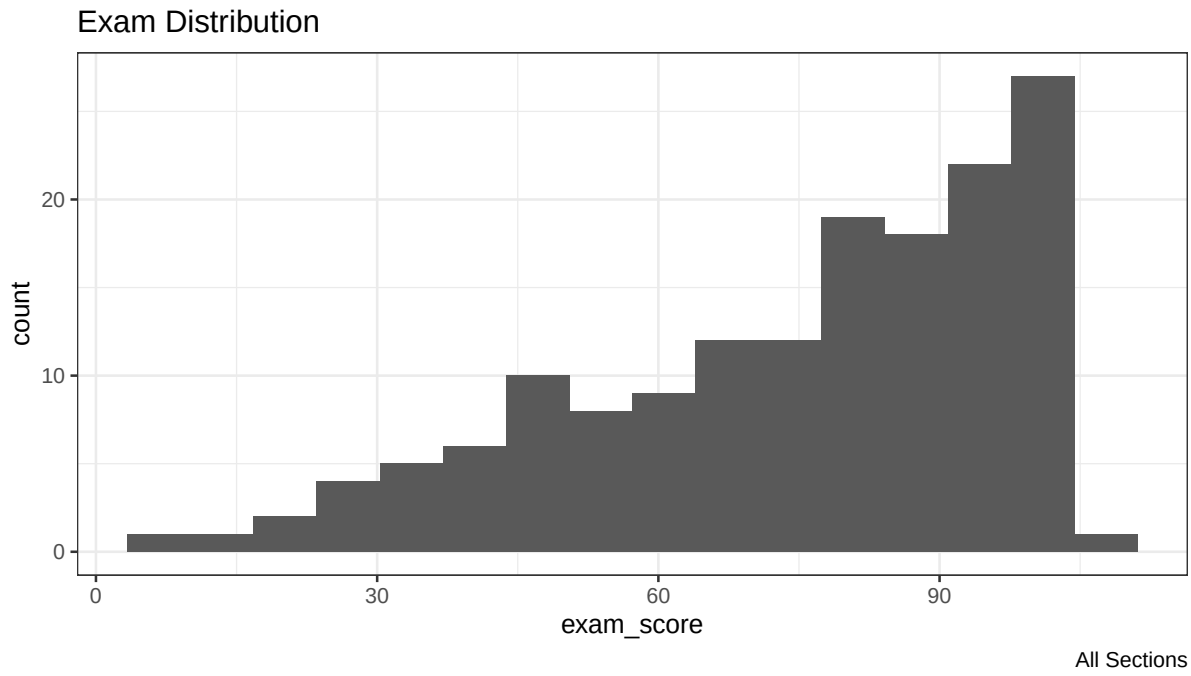
2.4.1.2 Test Score Distributions

```
ggplot(tb_final_twovar, aes(x=ar_test_scores_ec3)) +
  geom_histogram(bins=25) +
  labs(title = paste0('Sandbox: Final Distribution (Econ 2370, FW)'),
       caption = paste0('FW Section, formula:',
                        '0.3*exam1Perc + 0.3*exam2Perc + ',
                        '0.42*HWtotalPerc + 0.03*AttendancePerc \n',
                        '+ perfect attendance + 0.03 per Extra Credit')) +
  theme_bw()
```

2.4.1.2.1 Histogram




```
ggplot(tb_test_data, aes(x=exam_score)) +
  geom_histogram(bins=16) +
  labs(title = paste0('Exam Distribution'),
       caption = 'All Sections') +
  theme_bw()
```



2.5 Summarize Multiple Variables

2.5.1 Generate Replace Variables

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

2.5.1.1 Replace NA for Multiple Variables

Replace some variables NA by some values, and other variables' NAs by other values.

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'date'

# NA dataframe
df_NA <- as_tibble(matrix(NA, nrow=it_N, ncol=it_M)) %>%
  rowid_to_column(var = svr_id) %>%
  rename_at(vars(starts_with("V")),
            funs(str_replace(., "V", "var")))
kable(df_NA) %>%
  kable_styling_fc()
```

date	var1	var2	var3	var4	var5
1	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA

```
# Replace NA
df_NA_replace <- df_NA %>%
  mutate_at(vars(one_of(c('var1', 'var2'))), list(~replace_na(., 0))) %>%
  mutate_at(vars(one_of(c('var3', 'var5'))), list(~replace_na(., 99)))
kable(df_NA_replace) %>%
  kable_styling_fc()
```

date	var1	var2	var3	var4	var5
1	0	0	99	NA	99
2	0	0	99	NA	99
3	0	0	99	NA	99

2.5.1.2 Cumulative Sum Multiple Variables

Each row is a different date, each column is the profit a firms earns on a date, we want to compute cumulatively how much a person is earning. Also renames variable names below jointly.

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'date'

# random dataframe, daily profit of firms
# dp_fx: daily profit firm ID something
set.seed(123)
df_daily_profit <- as_tibble(matrix(rnorm(it_N*it_M), nrow=it_N, ncol=it_M)) %>%
  rowid_to_column(var = svr_id) %>%
  rename_at(vars(starts_with("V")),
            funs(str_replace(., "V", "dp_f")))
kable(df_daily_profit) %>%
  kable_styling_fc()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411

```
# cumulative sum with suffix
df_cumu_profit_suffix <- df_daily_profit %>%
  mutate_at(vars(contains('dp_f')), .funs = list(cumu = ~cumsum(.)))
kable(df_cumu_profit_suffix) %>%
  kable_styling_fc_wide()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5	dp_f1_cumu	dp_f2_cumu	dp_f3_cumu	dp_f4_cumu	dp_f5_cumu
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827	-0.7906531	0.1997961	-0.8041450	0.7784198	0.5114542
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411	0.7680552	1.9148611	-1.4909979	1.1382337	-0.0443870

```
# cumulative sum variables naming to prefix
df_cumu_profit <- df_cumu_profit_suffix %>%
  rename_at(vars(contains( "_cumu") ), list(~paste("cp_f", gsub("_cumu", "", .), sep = ""))) %>%
  rename_at(vars(contains( "cp_f") ), list(~gsub("dp_f", "", .)))
kable(df_cumu_profit) %>%
  kable_styling_fc_wide()
```

date	dp_f1	dp_f2	dp_f3	dp_f4	dp_f5	cp_f1	cp_f2	cp_f3	cp_f4	cp_f5
1	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715	-0.5604756	0.0705084	0.4609162	-0.4456620	0.4007715
2	-0.2301775	0.1292877	-1.2650612	1.2240818	0.1106827	-0.7906531	0.1997961	-0.8041450	0.7784198	0.5114542
3	1.5587083	1.7150650	-0.6868529	0.3598138	-0.5558411	0.7680552	1.9148611	-1.4909979	1.1382337	-0.0443870

Chapter 3

Functions

3.1 Dataframe Mutate

3.1.1 Row Input Functions

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

We want evaluate nonlinear function f(Q_i, y_i, ar_x, ar_y, c, d), where c and d are constants, and ar_x and ar_y are arrays, both fixed. x_i and y_i vary over each row of matrix. We would like to evaluate this nonlinear function concurrently across N individuals. The eventual goal is to find the i specific Q that solves the nonlinear equations.

This is a continuation of R use Apply, Sapply and dplyr Mutate to Evaluate one Function Across Rows of a Matrix

3.1.1.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)
ar_nN_N_choice = seq(1,it_N_child_cnt)/sum(seq(1,it_N_child_cnt))

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha, ar_nN_N_choice)

# Convert Matrix to Tibble
ar_st_col_names = c('fl_A', 'fl_alpha', 'fl_N')
tb_nN_by_nQ_A_alpha <- as_tibble(mt_nN_by_nQ_A_alpha) %>% rename_all(~c(ar_st_col_names))

# Show
```

```
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

fl_A	fl_alpha	fl_N
-2	0.1	0.0666667
-1	0.3	0.1333333
0	0.5	0.2000000
1	0.7	0.2666667
2	0.9	0.3333333

3.1.1.2 Mutate over Simple Function

For this example, use a very simple function with only one type of input, all inputs are scalars.

```
# Define Implicit Function
ffi_nonlinear <- function(fl_A, fl_alpha){

  fl_out <- (fl_A + fl_alpha*fl_A)/(fl_A)^2

  return(fl_out)
}
```

Apply the function over the dataframe, note five different ways below, the third way allows for parameters to be strings.

```
# variable names
svr_fl_A <- 'fl_A'
svr_fl_alpha <- 'fl_alpha'

# Evaluate
tb_nN_by_nQ_A_alpha_mutate_rows <- tb_nN_by_nQ_A_alpha %>%
  mutate(fl_out_m1 = ffi_nonlinear(fl_A=.$fl_A, fl_alpha=.$fl_alpha),
         fl_out_m2 = ffi_nonlinear(fl_A=`$`(., 'fl_A'), fl_alpha=`$`(., 'fl_alpha')),
         fl_out_m3 = ffi_nonlinear(fl_A=.[[svr_fl_A]], fl_alpha=.[[svr_fl_alpha]]),
         fl_out_m4 = ffi_nonlinear(fl_A=fl_A, fl_alpha=fl_alpha),
         fl_out_m5 = ffi_nonlinear(fl_A, fl_alpha))

# print
kable(tb_nN_by_nQ_A_alpha_mutate_rows) %>% kable_styling_fc()
```

fl_A	fl_alpha	fl_N	fl_out_m1	fl_out_m2	fl_out_m3	fl_out_m4	fl_out_m5
-2	0.1	0.0666667	-0.55	-0.55	-0.55	-0.55	-0.55
-1	0.3	0.1333333	-1.30	-1.30	-1.30	-1.30	-1.30
0	0.5	0.2000000	NaN	NaN	NaN	NaN	NaN
1	0.7	0.2666667	1.70	1.70	1.70	1.70	1.70
2	0.9	0.3333333	0.95	0.95	0.95	0.95	0.95

3.1.1.3 Testing Function with Scalar and Arrays

Test non-linear Equation.

```
# Test Parameters
fl_N_agg = 100
fl_rho = -1
fl_N_q = ar_nN_N_choice[4]*fl_N_agg
ar_A_alpha = mt_nN_by_nQ_A_alpha[4,]
# Apply Function
ar_p1_s1 = exp((ar_A_alpha[1] - ar_nN_A)*fl_rho)
```

```
ar_p1_s2 = (ar_A_alpha[2]/ar_nN_alpha)
ar_p1_s3 = (1/(ar_nN_alpha*fl_rho - 1))
ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
ar_p2 = fl_N_q^((ar_A_alpha[2]*fl_rho-1)/(ar_nN_alpha*fl_rho-1))
ar_overall = ar_p1*ar_p2
fl_overall = fl_N_agg - sum(ar_overall)
print(fl_overall)
```

```
## [1] -598.2559
```

Implement the non-linear problem's evaluation using apply over all N individuals.

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  # # Test Parameters
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}

# Parameters
fl_rho = -1

# Evaluate Function
print(ffi_nonlin_dplyrdo(mt_nN_by_nQ_A_alpha[1,1],
                         mt_nN_by_nQ_A_alpha[1,2],
                         mt_nN_by_nQ_A_alpha[1,3]*fl_N_agg,
                         ar_nN_A, ar_nN_alpha, fl_N_agg, fl_rho))
```

```
## [1] 81.86645
```

```
for (i in seq(1,dim(mt_nN_by_nQ_A_alpha)[1])){
  fl_eval = ffi_nonlin_dplyrdo(mt_nN_by_nQ_A_alpha[i,1],
                               mt_nN_by_nQ_A_alpha[i,2],
                               mt_nN_by_nQ_A_alpha[i,3]*fl_N_agg,
                               ar_nN_A, ar_nN_alpha, fl_N_agg, fl_rho)
  print(fl_eval)
}
```

```
## [1] 81.86645
## [1] 54.48885
## [1] -65.5619
## [1] -598.2559
## [1] -3154.072
```

3.1.1.4 Evaluate Nonlinear Function using dplyr mutate

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){

  # Test Parameters
  # ar_A = ar_nN_A
  # ar_alpha = ar_nN_alpha
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = (fl_N*fl_N_agg)^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}

# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_nN_by_nQ_A_alpha = tb_nN_by_nQ_A_alpha %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
# Show
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

fl_A	fl_alpha	fl_N	dplyr_eval
-2	0.1	0.0666667	81.86645
-1	0.3	0.1333333	54.48885
0	0.5	0.2000000	-65.56190
1	0.7	0.2666667	-598.25595
2	0.9	0.3333333	-3154.07226

3.1.2 Evaluate Choices Across States

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

See the ff_opti_bisect_pmap_multi function from Fan's *REconTools* Package, which provides a resuable function based on the algorithm worked out here.

We want evaluate linear function $0 = f(z_{ij}, x_i, y_i, \mathbf{X}, \mathbf{Y}, c, d)$. There are i functions that have i specific x and y. For each i function, we evaluate along a grid of feasible values for z, over $j \in J$ grid points, potentially looking for the j that is closest to the root. $\mathbf{X}$ and $\mathbf{Y}$ are arrays common across the i equations, and c and d are constants.

The evaluation strategy is the following, given min and max for z that are specific for each j, and given common number of grid points, generate a matrix of z_{ij}. Suppose there the number of i is I, and the number of grid points for j is J.

1. Generate a $J \cdot I$ by 3 matrix where the columns are z, x, y as tibble
2. Follow this Mutate to evaluate the $f(\cdot)$ function.

3. Add two categorical columns for grid levels and wich i, i and j index. Plot Mutate output evaluated column categorized by i as color and j as x-axis.

3.1.2.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

Now we need to expand this by the number of choice grid. Each row, representing one equation, is expanded by the number of choice grids. We are graphically searching, or rather brute force searching, which means if we have 100 individuals, we want to plot out the nonlinear equation for each of these lines, and show graphically where each line crosses zero. We achieve this, by evaluating the equation for each of the 100 individuals along a grid of feasible choices.

In this problem here, the feasible choices are shared across individuals.

```
# Parameters
fl_rho = 0.20
svr_id_var = 'INDI_ID'

# it_child_count = N, the number of children
it_N_child_cnt = 4
# it_heter_param = Q, number of parameters that are heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# Choice Grid for nutritional feasible choices for each
fl_N_agg = 100
fl_N_min = 0
it_N_choice_cnt_ttest = 3
it_N_choice_cnt_dense = 100
ar_N_choices_ttest = seq(fl_N_min, fl_N_agg, length.out = it_N_choice_cnt_ttest)
ar_N_choices_dense = seq(fl_N_min, fl_N_agg, length.out = it_N_choice_cnt_dense)

# Mesh Expand
tb_states_choices <- as_tibble(mt_nN_by_nQ_A_alpha) %>% rowid_to_column(var=svr_id_var)
tb_states_choices_ttest <- tb_states_choices %>% expand_grid(choices = ar_N_choices_ttest)
tb_states_choices_dense <- tb_states_choices %>% expand_grid(choices = ar_N_choices_dense)

# display
summary(tb_states_choices_dense)
```

```
##     INDI_ID        ar_nN_A    ar_nN_alpha     choices
##  Min.   :1.00   Min.   :-2   Min.   :0.1   Min.   :  0
##  1st Qu.:1.75   1st Qu.:-1   1st Qu.:0.3   1st Qu.: 25
##  Median :2.50   Median : 0   Median :0.5   Median : 50
##  Mean   :2.50   Mean   : 0   Mean   :0.5   Mean   : 50
##  3rd Qu.:3.25   3rd Qu.: 1   3rd Qu.:0.7   3rd Qu.: 75
##  Max.   :4.00   Max.   : 2   Max.   :0.9   Max.   :100
```

```
kable(tb_states_choices_ttest) %>%
  kable_styling_fc()
```

INDI_ID	ar_nN_A	ar_nN_alpha	choices
1	-2.0000000	0.1000000	0
1	-2.0000000	0.1000000	50
1	-2.0000000	0.1000000	100
2	-0.6666667	0.3666667	0
2	-0.6666667	0.3666667	50
2	-0.6666667	0.3666667	100
3	0.6666667	0.6333333	0
3	0.6666667	0.6333333	50
3	0.6666667	0.6333333	100
4	2.0000000	0.9000000	0
4	2.0000000	0.9000000	50
4	2.0000000	0.9000000	100

3.1.2.2 Apply Same Function all Rows, Some Inputs Row-specific, other Shared

There are two types of inputs, row-specific inputs, and inputs that should be applied for each row. The Function just requires all of these inputs, it does not know what is row-specific and what is common for all row. Dplyr recognizes which parameter inputs already existing in the piped dataframe/tibble, given rowwise, those will be row-specific inputs. Additional function parameters that do not exist in dataframe as variable names, but that are pre-defined scalars or arrays will be applied to all rows.

- **?** string variable name of input where functions are evaluated, these are already contained in the dataframe, existing variable names, row specific, rowwise computation over these, each rowwise calculation using different rows: *fl_A*, *fl_alpha*, *fl_N*
- **?** scalar and array values that are applied to every rowwise calculation, all rowwise calculations using the same scalars and arrays:*ar_A*, *ar_alpha*, *fl_N_agg*, *fl_rho*
- **?** string output variable name

The function looks within group, finds min/max etc that are relevant.

```
# Convert Matrix to Tibble
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha')
tb_states_choices <- tb_states_choices %>% rename_all(~c(ar_st_col_names))
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha', 'fl_N')
tb_states_choices_ttest <- tb_states_choices_ttest %>% rename_all(~c(ar_st_col_names))
tb_states_choices_dense <- tb_states_choices_dense %>% rename_all(~c(ar_st_col_names))

# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){
  # scalar value that are row-specific, in dataframe already: *fl_A*, *fl_alpha*, *fl_N*
  # array and scalars not in dataframe, common all rows: *ar_A*, *ar_alpha*, *fl_N_agg*, *fl_rho*

  # Test Parameters
  # ar_A = ar_nN_A
  # ar_alpha = ar_nN_alpha
  # fl_N = 100
  # fl_rho = -1
  # fl_N_q = 10

  # Apply Function
  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
```

```
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}
```

3.1.2.2.1 3 Points and Denser Dataframs and Define Function

3.1.2.2.2 Evaluate at Three Choice Points and Show Table In the example below, just show results evaluating over three choice points and show table.

```
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_states_choices_ttest_eval = tb_states_choices_ttest %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
# Show
kable(tb_states_choices_ttest_eval) %>%
  kable_styling_fc()
```

INDI_ID	fl_A	fl_alpha	fl_N	dplyr_eval
1	-2.0000000	0.1000000	0	100.00000
1	-2.0000000	0.1000000	50	-5666.95576
1	-2.0000000	0.1000000	100	-12880.28392
2	-0.6666667	0.3666667	0	100.00000
2	-0.6666667	0.3666667	50	-595.73454
2	-0.6666667	0.3666667	100	-1394.70698
3	0.6666667	0.6333333	0	100.00000
3	0.6666667	0.6333333	50	-106.51058
3	0.6666667	0.6333333	100	-323.94216
4	2.0000000	0.9000000	0	100.00000
4	2.0000000	0.9000000	50	22.55577
4	2.0000000	0.9000000	100	-51.97161

3.1.2.2.3 Evaluate at Many Choice Points and Show Graphically Same as above, but now we evaluate the function over the individuals at many choice points so that we can graph things out.

```
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_states_choices_dense_eval = tb_states_choices_dense %>% rowwise() %>%
                        mutate(dplyr_eval = ffi_nonlin_dplyrdo(fl_A, fl_alpha, fl_N,
                                                               ar_nN_A, ar_nN_alpha,
                                                               fl_N_agg, fl_rho))
```

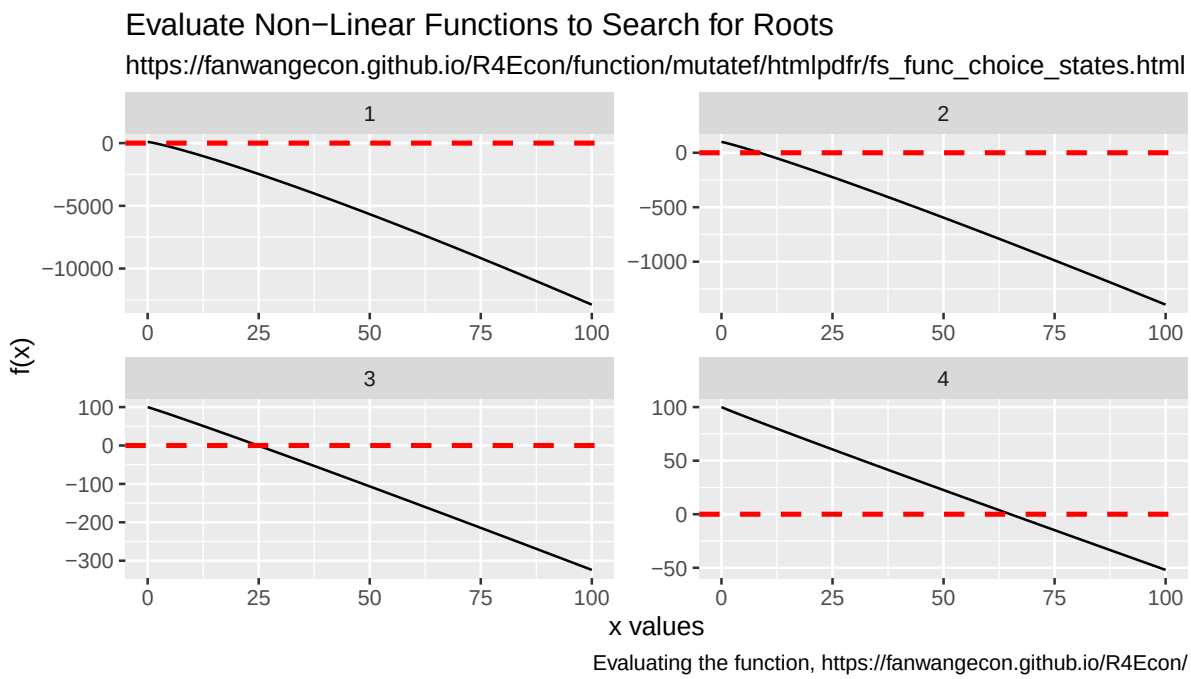
```
# Labeling
st_title <- paste0('Evaluate Non-Linear Functions to Search for Roots')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/function/mutatef/htmlpdfr/fs_func_choice_states.html')
st_caption <- paste0('Evaluating the function, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'x values'
st_y_label <- 'f(x)'

# Show
dim(tb_states_choices_dense_eval)
```

```
## [1] 400   5
```

```
summary(tb_states_choices_dense_eval)
```

```
##     INDI_ID          fl_A       fl_alpha        fl_N       dplyr_eval
##  Min.   :1.00   Min.   :-2   Min.   :0.1   Min.   :  0   Min.   :-12880.28
##  1st Qu.:1.75   1st Qu.:-1   1st Qu.:0.3   1st Qu.: 25   1st Qu.: -1167.29
##  Median :2.50   Median : 0   Median :0.5   Median : 50   Median :  -202.42
##  Mean   :2.50   Mean   : 0   Mean   :0.5   Mean   : 50   Mean   : -1645.65
##  3rd Qu.:3.25   3rd Qu.: 1   3rd Qu.:0.7   3rd Qu.: 75   3rd Qu.:     0.96
##  Max.   :4.00   Max.   : 2   Max.   :0.9   Max.   :100   Max.   :   100.00
```

```
lineplot <- tb_states_choices_dense_eval %>%
    ggplot(aes(x=fl_N, y=dplyr_eval)) +
        geom_line() +
        facet_wrap( . ~ INDI_ID, scales = "free") +
        geom_hline(yintercept=0, linetype="dashed",
                color = "red", size=1) +
        labs(title = st_title,
             subtitle = st_subtitle,
             x = st_x_label,
             y = st_y_label,
             caption = st_caption)
print(lineplot)
```



3.2 Dataframe Do Anything

3.2.1 (Mx1 by N) to (MxQ by N+1)

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Case One: There is a dataframe with M rows, based on these m specific information, generate dataframes for each m. Stack these indivdiual dataframes together and merge original m specific information in as well. The number of rows for each m is Q_m, each m could have different number of expansion rows.

Generate a panel with M individuals, each individual is observed for different spans of times (*uncount*).

Before expanding, generate individual specific normal distribution standard deviation. All individuals share the same mean, but have increasing standard deviations.

3.2.1.1 Generate Dataframe with M Rows.

This is the first step, generate M rows of data, to be expanded. Each row contains the number of normal draws to make and the mean and the standard deviation for normal daraws that are m specific.

```
# Parameter Setups
it_M <- 3
it_Q_max <- 5
fl_rnorm_mu <- 1000
ar_rnorm_sd <- seq(0.01, 200, length.out=it_M)
ar_it_q <- sample.int(it_Q_max, it_M, replace=TRUE)

# N by Q varying parameters
mt_data = cbind(ar_it_q, ar_rnorm_sd)
tb_M <- as_tibble(mt_data) %>% rowid_to_column(var = "ID") %>%
                rename(sd = ar_rnorm_sd, Q = ar_it_q) %>%
                mutate(mean = fl_rnorm_mu)

# display
kable(tb_M) %>%
  kable_styling_fc()
```

ID	Q	sd	mean
1	1	0.010	1000
2	3	100.005	1000
3	4	200.000	1000

3.2.1.2 Random Normal Draw Expansion

The steps are:

1. do anything
2. use ".$" sign to refer to variable names, or [['name']]
3. unnest
4. left_join expanded and original

Note these all give the same results

Use dot dollar to get variables

```
# Generate $Q_m$ individual specific incomes, expanded different number of times for each m
tb_income <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.$Q, mean=.$mean, sd=.$sd)) %>%
  unnest(c(income))

# Merge back with tb_M
tb_income_full_dd <- tb_income %>%
  left_join(tb_M)

# display
kable(tb_income) %>%
  kable_styling_fc()
```

```
kable(tb_income_full_dd) %>%
  kable_styling_fc()
```

ID	income
1	999.9803
2	1070.1391
2	952.7185
2	893.2123
3	956.4050
3	794.7991
3	854.2218
3	874.9921

ID	income	Q	sd	mean
1	999.9803	1	0.010	1000
2	1070.1391	3	100.005	1000
2	952.7185	3	100.005	1000
2	893.2123	3	100.005	1000
3	956.4050	4	200.000	1000
3	794.7991	4	200.000	1000
3	854.2218	4	200.000	1000
3	874.9921	4	200.000	1000

3.2.2 (MxP by N) to (Mx1 by 1)

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

There is a Panel with M individuals and each individual has Q records/rows. A function generate an individual specific outcome given the Q individual specific inputs, along with shared parameters and arrays across the M individuals.

For example, suppose we have a dataframe of individual wage information from different countries, each row is an individual from one country. We want to generate country specific gini based on the individual data for each country in the dataframe. But additionally, perhaps the gini formula requires not just individual income but some additional parameters or shared dataframes as inputs.

Given the within m income observations, we can compute gini statistics that are individual specific based on the observed distribution of incomes. For this, we will use the ff_dist_gini_vector_pos.html function from REconTools.

To make this more interesting, we will generate large dataframe with more M and more Q each m.

3.2.2.1 Income Rows for Individuals in Many Groups

There are up to ten thousand income observation per person. And there are ten people.

```
# Parameter Setups
it_M <- 10
it_Q_max <- 10000
fl_rnorm_mu <- 1
ar_rnorm_sd <- seq(0.01, 0.2, length.out=it_M)
ar_it_q <- sample.int(it_Q_max, it_M, replace=TRUE)

# N by Q varying parameters
mt_data = cbind(ar_it_q, ar_rnorm_sd)
tb_M <- as_tibble(mt_data) %>% rowid_to_column(var = "ID") %>%
                rename(sd = ar_rnorm_sd, Q = ar_it_q) %>%
                mutate(mean = fl_rnorm_mu)
```

3.2.2.2 Compute Group Specific Gini

There is only one input for the gini function *ar_pos*. Note that the gini are not very large even with large SD, because these are normal distributions. By Construction, most peple are in the middle. So with almost zero standard deviation, we have perfect equality, as standard deviation increases, inequality increases, but still pretty equal overall, there is no fat upper tail.

Note that there are three ways of referring to variable names with dot, which are all shown below:

1. We can explicitly refer to names
2. We can use the dollar dot structure to use string variable names in do anything.
3. We can use dot bracket, this is the only option that works with string variable names

First: Generate individual group all incomes:

```
# A. Normal Draw Expansion, Explicitly Name
set.seed('123')
tb_income_norm_dot_dollar <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.$Q,
                    mean=.$mean,
                    sd=.$sd)) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot dollar differently with string variable name
set.seed('123')
tb_income_norm_dollar_dot <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(`$`(., 'Q'),
                    mean = `$`(., 'mean'),
                    sd = `$`(., 'sd'))) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot double bracket
set.seed('123')
svr_mean <- 'mean'
svr_sd <- 'sd'
svr_Q <- 'Q'
tb_income_norm_dot_bracket_db <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.[[svr_Q]],
                    mean = .[[svr_mean]],
                    sd = .[[svr_sd]])) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# display
print(dim(tb_income_norm_dot_bracket_db))
```

```
## [1] 59429     5
```

```
kable(head(tb_income_norm_dot_bracket_db, 20)) %>% kable_styling_fc()
```

Second, compute gini:

```
# Gini by Group
tb_gini_norm <- tb_income_norm_dot_bracket_db %>% group_by(ID) %>%
  do(inc_gini_norm = ff_dist_gini_vector_pos(.$income)) %>%
  unnest(c(inc_gini_norm)) %>%
  left_join(tb_M, by="ID")

# display
kable(tb_gini_norm) %>% kable_styling_fc()
```

ID	income	Q	sd	mean
1	0.9943952	3004	0.01	1
1	0.9976982	3004	0.01	1
1	1.0155871	3004	0.01	1
1	1.0007051	3004	0.01	1
1	1.0012929	3004	0.01	1
1	1.0171506	3004	0.01	1
1	1.0046092	3004	0.01	1
1	0.9873494	3004	0.01	1
1	0.9931315	3004	0.01	1
1	0.9955434	3004	0.01	1
1	1.0122408	3004	0.01	1
1	1.0035981	3004	0.01	1
1	1.0040077	3004	0.01	1
1	1.0011068	3004	0.01	1
1	0.9944416	3004	0.01	1
1	1.0178691	3004	0.01	1
1	1.0049785	3004	0.01	1
1	0.9803338	3004	0.01	1
1	1.0070136	3004	0.01	1
1	0.9952721	3004	0.01	1

ID	inc_gini_norm	Q	sd	mean
1	0.0056006	3004	0.0100000	1
2	0.0174893	3207	0.0311111	1
3	0.0295527	7989	0.0522222	1
4	0.0412807	3995	0.0733333	1
5	0.0537107	8358	0.0944444	1
6	0.0650354	217	0.1155556	1
7	0.0766718	9506	0.1366667	1
8	0.0891009	8157	0.1577778	1
9	0.1014251	6216	0.1788889	1
10	0.1135054	8780	0.2000000	1

3.2.3 (MxP by N) to (MxQ by N+Z)

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

There is a dataframe composed of M mini-dataframes. Group by a variable that identifies each unique sub-dataframe, and use the sub-dataframes with P rows as inputs to a function.

The function outputs Q *by* Z rows and columns of results, stack the results. The output file has MxQ rows and the Z columns of additional results should be appended.

3.2.3.1 Generate the MxP by N Dataframe

M Grouping characteristics, P rows for each group, and N Variables.

1. M are individuals
2. P are dates
3. A wage variable for individual wage at each date. And a savings varaible as well.

```
# Define
it_M <- 3
it_P <- 5
svr_m <- 'group_m'
svr_mp <- 'info_mp'
```

```
# dataframe
set.seed(123)
df_panel_skeleton <- as_tibble(matrix(it_P, nrow=it_M, ncol=1)) %>%
  rowid_to_column(var = svr_m) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_m)) %>% mutate(!!sym(svr_mp) := row_number()) %>%
  ungroup() %>%
  rowwise() %>% mutate(wage = rnorm(1, 100, 10),
                       savings = rnorm(1, 200, 30)) %>%
  ungroup() %>%
  rowid_to_column(var = "id_ji")

# Print
kable(df_panel_skeleton) %>% kable_styling_fc()
```

id_ji	group_m	info_mp	wage	savings
1	1	1	94.39524	253.6074
2	1	2	97.69823	214.9355
3	1	3	115.58708	141.0015
4	1	4	100.70508	221.0407
5	1	5	101.29288	185.8163
6	2	1	117.15065	167.9653
7	2	2	104.60916	193.4608
8	2	3	87.34939	169.2199
9	2	4	93.13147	178.1333
10	2	5	95.54338	181.2488
11	3	1	112.24082	149.3992
12	3	2	103.59814	225.1336
13	3	3	104.00771	204.6012
14	3	4	101.10683	165.8559
15	3	5	94.44159	237.6144

3.2.3.2 Subgroup Compute and Expand

Use the M sub-dataframes, generate Q by Z result for each of the M groups. Stack all results together.

Base on all the wages for each individual, generate individual specific mean and standard deviations. Do this for three things, the wage variable, the savings variable, and the sum of wage and savings:

1. Z=*2*: 2 columns, mean and standard deviation
2. Q=*3*: 3 rows, statistics based on wage, savings, and the sum of both

First, here is the processing function that takes the dataframe as input, with a parameter for rounding:

```
# define function
ffi_subset_mean_sd <- function(df_sub, it_round=1) {
  #' A function that generates mean and sd for several variables
  #'
  #' @description
  #' Assume there are two variables in df_sub wage and savings
  #'
  #' @param df_sub dataframe where each individual row is a different
  #' data point, over which we compute mean and sd, Assum there are two
  #' variables, savings and wage
  #' @param it_round integer rounding for resulting dataframe
  #' @return a dataframe where each row is aggregate for a different type
  #' of variablea and each column is a different statistics

  fl_wage_mn = mean(df_sub$wage)
```

```
  fl_wage_sd = sd(df_sub$wage)

  fl_save_mn = mean(df_sub$savings)
  fl_save_sd = sd(df_sub$savings)

  fl_wgsv_mn = mean(df_sub$wage + df_sub$savings)
  fl_wgsv_sd = sd(df_sub$wage + df_sub$savings)

  ar_mn <- c(fl_wage_mn, fl_save_mn, fl_wgsv_mn)
  ar_sd <- c(fl_wage_sd, fl_save_sd, fl_wgsv_sd)
  ar_st_row_lab <- c('wage', 'savings', 'wage_and_savings')

  mt_stats <- cbind(ar_mn, ar_sd)
  mt_stats <- round(mt_stats, it_round)

  ar_st_varnames <- c('mean', 'sd', 'variables')
  df_combine <- as_tibble(mt_stats) %>%
    add_column(ar_st_row_lab) %>%
    rename_all(~c(ar_st_varnames)) %>%
    select(variables, 'mean', 'sd') %>%
    rowid_to_column(var = "id_q")

  return(df_combine)
}
# testing function
ffi_subset_mean_sd(df_panel_skeleton %>% filter(!!sym(svr_m)==1))
```

Second, call *ffi_subset_mean_sd* function for each of the groups indexed by j and stack results together with j index:

1. group by
2. call function
3. unnest

```
# run group stats and stack dataframes
df_outputs <- df_panel_skeleton %>% group_by(!!sym(svr_m)) %>%
  do(df_stats = ffi_subset_mean_sd(., it_round=2)) %>%
  unnest() %>%
  rowid_to_column(var = "id_mq")
# print
kable(df_outputs) %>% kable_styling_fc()
```

id_mq	group_m	id_q	variables	mean	sd
1	1	1	wage	101.94	8.11
2	1	2	savings	203.28	42.33
3	1	3	wage_and_savings	305.22	34.83
4	2	1	wage	99.56	11.63
5	2	2	savings	178.01	10.34
6	2	3	wage_and_savings	277.56	15.48
7	3	1	wage	103.08	6.39
8	3	2	savings	196.52	37.86
9	3	3	wage_and_savings	299.60	33.50

In the resulting file, we went from a matrix with MxP rows to a matrix with MxQ Rows.

3.3 Apply and pmap

3.3.1 Apply and Sapply

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

- r apply matrix to function row by row
- r evaluate function on grid
- Apply a function to every row of a matrix or a data frame
- r apply
- r sapply
- sapply over matrix row by row
- function as parameters using formulas
- do

We want evaluate linear function f(x_i, y_i, ar_x, ar_y, c, d), where c and d are constants, and ar_x and ar_y are arrays, both fixed. x_i and y_i vary over each row of matrix. More specifically, we have a functions, this function takes inputs that are individual specific. We would like to evaluate this function concurrently across N individuals.

The function is such that across the N individuals, some of the function parameter inputs are the same, but others are different. If we are looking at demand for a particular product, the prices of all products enter the demand equation for each product, but the product's own price enters also in a different way.

The objective is either to just evaluate this function across N individuals, or this is a part of a nonlinear solution system.

What is the relationship between apply, lapply and vectorization? see Is the "*apply" family really not vectorized?.

3.3.1.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are
# heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# display
kable(mt_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.3.1.2 Using apply

3.3.1.2.1 Named Function First we use the apply function, we have to hard-code the arrays that are fixed for each of the N individuals. Then apply allows us to loop over the matrix that is N by Q,

ar_nN_A	ar_nN_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

each row one at a time, from 1 to N.

```
# Define Implicit Function
ffi_linear_hardcode <- function(ar_A_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  fl_out = sum(ar_A_alpha[1]*ar_nN_A +
                 1/(ar_A_alpha[2] + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row
ar_func_apply = apply(mt_nN_by_nQ_A_alpha, 1, ffi_linear_hardcode)
```

3.3.1.2.2 Anonymous Function

- apply over matrix

Apply with anonymous function generating a list of arrays of different lengths. In the example below, we want to drawn N sets of random uniform numbers, but for each set the number of draws we want to have is Q_i. Furthermore, we want to rescale the random uniform draws so that they all become proportions that sum u pto one for each i, but then we multply each row's values by the row specific aggregates.

The anonymous function has hard coded parameters. Using an anonymous function here allows for parameters to be provided inside the function that are shared across each looped evaluation. This is perhaps more convenient than sapply with additional parameters.

```
set.seed(1039)

# Define the number of draws each row and total amount
it_N <- 4
fl_unif_min <- 1
fl_unif_max <- 2
mt_draw_define <- cbind(sample(it_N, it_N, replace=TRUE),
                        runif(it_N, min=1, max=10))
tb_draw_define <- as_tibble(mt_draw_define) %>%
  rowid_to_column(var = "draw_group")
print(tb_draw_define)

# apply row by row, anonymous function has hard
# coded min and max
ls_ar_draws_shares_lvls =
  apply(tb_draw_define,
        1,
        function(row) {
          it_draw <- row[2]
          fl_sum <- row[3]
          ar_unif <- runif(it_draw,
                           min=fl_unif_min,
                           max=fl_unif_max)
```

```
          ar_share <- ar_unif/sum(ar_unif)
          ar_levels <- ar_share*fl_sum
          return(list(ar_share=ar_share,
                      ar_levels=ar_levels))
        })

# Show Results
print(ls_ar_draws_shares_lvls)
```

```
## [[1]]
## [[1]]$ar_share
## [1] 0.2783638 0.2224140 0.2797840 0.2194381
##
## [[1]]$ar_levels
## [1] 1.492414 1.192446 1.500028 1.176491
##
##
## [[2]]
## [[2]]$ar_share
## [1] 0.5052919 0.4947081
##
## [[2]]$ar_levels
## [1] 3.866528 3.785541
##
##
## [[3]]
## [[3]]$ar_share
## [1] 1
##
## [[3]]$ar_levels
##       V2
## 9.572211
##
##
## [[4]]
## [[4]]$ar_share
## [1] 0.4211426 0.2909812 0.2878762
##
## [[4]]$ar_levels
## [1] 4.051971 2.799640 2.769765
```

We will try to do the same thing as above, but now the output will be a stacked dataframe. Note that within each element of the apply row by row loop, we are generating two variables *ar_share* and *ar_levels*. We will not generate a dataframe with multiple columns, storing *ar_share*, *ar_levels* as well as information on *min*, *max*, number of draws and rescale total sum.

```
set.seed(1039)
# apply row by row, anonymous function has hard coded min and max
ls_mt_draws_shares_lvls =
  apply(tb_draw_define, 1, function(row) {

    it_draw_group <- row[1]
    it_draw <- row[2]
    fl_sum <- row[3]

    ar_unif <- runif(it_draw,
                     min=fl_unif_min,
                     max=fl_unif_max)
    ar_share <- ar_unif/sum(ar_unif)
```

```
    ar_levels <- ar_share*fl_sum

    mt_all_res <- cbind(it_draw_group, it_draw, fl_sum,
                        ar_unif, ar_share, ar_levels)
    colnames(mt_all_res) <-
      c('draw_group', 'draw_count', 'sum',
        'unif_draw', 'share', 'rescale')
    rownames(mt_all_res) <- NULL

    return(mt_all_res)
  })
mt_draws_shares_lvls_all <- do.call(rbind, ls_mt_draws_shares_lvls)
# Show Results
kable(mt_draws_shares_lvls_all) %>% kable_styling_fc()
```

draw_group	draw_count	sum	unif_draw	share	rescale
1	4	5.361378	1.125668	0.1988606	1.066167
1	4	5.361378	1.668536	0.2947638	1.580340
1	4	5.361378	1.419382	0.2507483	1.344356
1	4	5.361378	1.447001	0.2556274	1.370515
2	2	7.652069	1.484598	0.4605236	3.523959
2	2	7.652069	1.739119	0.5394764	4.128110
3	1	9.572211	1.952468	1.0000000	9.572211
4	3	9.621375	1.957931	0.3609352	3.472693
4	3	9.621375	1.926995	0.3552324	3.417824
4	3	9.621375	1.539678	0.2838324	2.730858

3.3.1.3 Using sapply

3.3.1.3.1 Named Function

- r convert matrix to list
- Convert a matrix to a list of vectors in R

Sapply allows us to not have tohard code in the A and alpha arrays. But Sapply works over List or Vector, not Matrix. So we have to convert the N by Q matrix to a N element list Now update the function with sapply.

```
ls_ar_nN_by_nQ_A_alpha = as.list(data.frame(t(mt_nN_by_nQ_A_alpha)))

# Define Implicit Function
ffi_linear_sapply <- function(ar_A_alpha, ar_A, ar_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  fl_out = sum(ar_A_alpha[1]*ar_nN_A +
                 1/(ar_A_alpha[2] + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row
ar_func_sapply = sapply(ls_ar_nN_by_nQ_A_alpha, ffi_linear_sapply,
                        ar_A=ar_nN_A, ar_alpha=ar_nN_alpha)
```

3.3.1.3.2 Anonymous Function

- sapply anonymous function
- r anoymous function multiple lines

Sapply with anonymous function generating a list of arrays of different lengths. In the example below, we want to drawn N sets of random uniform numbers, but for each set the number of draws we want to have is Q_i. Furthermore, we want to rescale the random uniform draws so that they all become proportions that sum u pto one for each i.

```
it_N <- 4
fl_unif_min <- 1
fl_unif_max <- 2

# Generate using runif without anonymous function
set.seed(1039)
ls_ar_draws = sapply(seq(it_N),
                     runif,
                     min=fl_unif_min, max=fl_unif_max)
print(ls_ar_draws)
```

```
## [[1]]
## [1] 1.125668
##
## [[2]]
## [1] 1.668536 1.419382
##
## [[3]]
## [1] 1.447001 1.484598 1.739119
##
## [[4]]
## [1] 1.952468 1.957931 1.926995 1.539678
```

```
# Generate Using Anonymous Function
set.seed(1039)
ls_ar_draws_shares = sapply(seq(it_N),
                            function(n, min, max) {
                              ar_unif <- runif(n,min,max)
                              ar_share <- ar_unif/sum(ar_unif)
                              return(ar_share)
                            },
                            min=fl_unif_min, max=fl_unif_max)
# Print Share
print(ls_ar_draws_shares)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 0.5403432 0.4596568
##
## [[3]]
## [1] 0.3098027 0.3178522 0.3723451
##
## [[4]]
## [1] 0.2646671 0.2654076 0.2612141 0.2087113
```

```
# Sapply with anonymous function to check sums
sapply(seq(it_N), function(x) {sum(ls_ar_draws[[x]])})
```

```
## [1] 1.125668 3.087918 4.670717 7.377071
```

```
sapply(seq(it_N), function(x) {sum(ls_ar_draws_shares[[x]])})
```

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

3.3.1.4 Compare Results

```
# Show overall Results
mt_results <- cbind(ar_func_apply, ar_func_sapply)
colnames(mt_results) <- c('eval_lin_apply', 'eval_lin_sapply')
kable(mt_results) %>% kable_styling_fc()
```

	eval_lin_apply	eval_lin_sapply
X1	2.346356	2.346356
X2	2.094273	2.094273
X3	1.895316	1.895316
X4	1.733708	1.733708
X5	1.599477	1.599477

3.3.2 Mutate Evaluate Functions

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Apply a function over rows of a matrix using mutate, rowwise, etc.

3.3.2.1 Set up Input Arrays

There is a function that takes $M = Q + P$ inputs, we want to evaluate this function N times. Each time, there are M inputs, where all but Q of the M inputs, meaning P of the M inputs, are the same. In particular, $P = Q * N$.

$$M = Q + P = Q + Q * N$$

```
# it_child_count = N, the number of children
it_N_child_cnt = 5
# it_heter_param = Q, number of parameters that are
# heterogeneous across children
it_Q_hetpa_cnt = 2

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = it_N_child_cnt)
ar_nN_alpha = seq(0.1, 0.9, length.out = it_N_child_cnt)
ar_nP_A_alpha = c(ar_nN_A, ar_nN_alpha)

# N by Q varying parameters
mt_nN_by_nQ_A_alpha = cbind(ar_nN_A, ar_nN_alpha)

# display
kable(mt_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

ar_nN_A	ar_nN_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

```
# Convert Matrix to Tibble
ar_st_col_names = c('fl_A', 'fl_alpha')
tb_nN_by_nQ_A_alpha <- as_tibble(mt_nN_by_nQ_A_alpha) %>%
```

```
  rename_all(~c(ar_st_col_names))
# Show
kable(tb_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

fl_A	fl_alpha
-2	0.1
-1	0.3
0	0.5
1	0.7
2	0.9

3.3.2.2 mutate rowwise

- dplyr mutate own function
- dplyr all row function
- dplyr do function
- apply function each row dplyr
- applying a function to every row of a table using dplyr
- dplyr rowwise

```
# Define Implicit Function
ffi_linear_dplyrdo <- function(fl_A, fl_alpha, ar_nN_A, ar_nN_alpha){
  # ar_A_alpha[1] is A
  # ar_A_alpha[2] is alpha

  print(paste0('cur row, fl_A=', fl_A, ', fl_alpha=', fl_alpha))
  fl_out = sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))

  return(fl_out)
}

# Evaluate function row by row of tibble
# fl_A, fl_alpha are from columns of tb_nN_by_nQ_A_alpha
tb_nN_by_nQ_A_alpha_show <- tb_nN_by_nQ_A_alpha %>%
  rowwise() %>%
  mutate(dplyr_eval =
           ffi_linear_dplyrdo(fl_A, fl_alpha, ar_nN_A, ar_nN_alpha))
```

```
## [1] "cur row, fl_A=-2, fl_alpha=0.1"
## [1] "cur row, fl_A=-1, fl_alpha=0.3"
## [1] "cur row, fl_A=0, fl_alpha=0.5"
## [1] "cur row, fl_A=1, fl_alpha=0.7"
## [1] "cur row, fl_A=2, fl_alpha=0.9"
```

```
# Show
kable(tb_nN_by_nQ_A_alpha_show) %>%
  kable_styling_fc()
```

fl_A	fl_alpha	dplyr_eval
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

same as before, still rowwise, but hard code some inputs:

```
# Define function, fixed inputs are not parameters, but
# defined earlier as a part of the function
# ar_nN_A, ar_nN_alpha are fixed, not parameters
ffi_linear_dplyrdo_func <- function(fl_A, fl_alpha){
  fl_out <- sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))
  return(fl_out)
}

# Evaluate function row by row of tibble
tbfunc_A_nN_by_nQ_A_alpha_rowwise = tb_nN_by_nQ_A_alpha %>% rowwise() %>%
  mutate(dplyr_eval = ffi_linear_dplyrdo_func(fl_A, fl_alpha))
# Show
kable(tbfunc_A_nN_by_nQ_A_alpha_rowwise) %>%
  kable_styling_fc()
```

fl_A	fl_alpha	dplyr_eval
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

3.3.2.3 mutate with pmap

Apparantly *rowwise()* is not a good idea, and *pmap* should be used, below is the *pmap* solution to the problem. Which does seem nicer. Crucially, don't have to define input parameter names, automatically I think they are matching up to the names in the function

- dplyr mutate pass function
- r function quosure string multiple
- r function multiple parameters as one string
- dplyr mutate anonymous function
- quosure style lambda
- pmap tibble rows
- dplyr pwalk

```
# Define function, fixed inputs are not parameters, but defined
# earlier as a part of the function Rorate fl_alpha and fl_A name
# compared to before to make sure pmap tracks by names
ffi_linear_dplyrdo_func <- function(fl_alpha, fl_A){
  fl_out <- sum(fl_A*ar_nN_A + 1/(fl_alpha + 1/ar_nN_alpha))
  return(fl_out)
}

# Evaluate a function row by row of dataframe, generate list,
# then to vector
tb_nN_by_nQ_A_alpha %>% pmap(ffi_linear_dplyrdo_func) %>% unlist()
```

```
## [1] 2.346356 2.094273 1.895316 1.733708 1.599477
```

```
# Same as above, but in line line and save output as new column
# in dataframe note this ONLY works if the tibble only has variables
# that are inputs for the function if tibble contains additional
#  variables, those should be droppd, or only the ones needed selected,
# inside the pmap call below.
tbfunc_A_nN_by_nQ_A_alpha_pmap <- tb_nN_by_nQ_A_alpha %>%
  mutate(dplyr_eval_pmap =
           unlist(
             pmap(tb_nN_by_nQ_A_alpha, ffi_linear_dplyrdo_func)
```

```
            )
  )

# Show
kable(tbfunc_A_nN_by_nQ_A_alpha_pmap) %>%
  kable_styling_fc()
```

fl_A	fl_alpha	dplyr_eval_pmap
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

3.3.2.4 rowwise and do

Now, we have three types of parameters, for something like a bisection type calculation. We will supply the program with a function with some hard-coded value inside, and as parameters, we will have one parameter which is a row in the current matrix, and another parameter which is a sclar values. The three types of parameters are dealt with sparately:

1. parameters that are fixed for all bisection iterations, but differ for each row
- these are hard-coded into the function
2. parameters that are fixed for all bisection iterations, but are shared across rows
- these are the first parameter of the function, a list
3. parameters that differ for each iteration, but differ acoss iterations
- second scalar value parameter for the function
- dplyr mutate function applow to each row dot notation
- note rowwise might be bad according to Hadley, should use pmap?

```
ffi_linear_dplyrdo_fdot <- function(ls_row, fl_param){
  # Type 1 Param = ar_nN_A, ar_nN_alpha
  # Type 2 Param = ls_row$fl_A, ls_row$fl_alpha
  # Type 3 Param = fl_param

  fl_out <- (sum(ls_row$fl_A*ar_nN_A +
                   1/(ls_row$fl_alpha + 1/ar_nN_alpha))) + fl_param
  return(fl_out)
}

cur_func <- ffi_linear_dplyrdo_fdot
fl_param <- 0
dplyr_eval_flex <- tb_nN_by_nQ_A_alpha %>% rowwise() %>%
  do(dplyr_eval_flex = cur_func(., fl_param)) %>%
  unnest(dplyr_eval_flex)
tbfunc_B_nN_by_nQ_A_alpha <- tb_nN_by_nQ_A_alpha %>% add_column(dplyr_eval_flex)
# Show
kable(tbfunc_B_nN_by_nQ_A_alpha) %>%
  kable_styling_fc()
```

3.3.2.5 Compare Apply and Mutate Results

```
# Show overall Results
mt_results <- cbind(tb_nN_by_nQ_A_alpha_show['dplyr_eval'],
```

fl_A	fl_alpha	dplyr_eval_flex
-2	0.1	2.346356
-1	0.3	2.094273
0	0.5	1.895316
1	0.7	1.733708
2	0.9	1.599477

```
                    tbfunc_A_nN_by_nQ_A_alpha_rowwise['dplyr_eval'],
                    tbfunc_A_nN_by_nQ_A_alpha_pmap['dplyr_eval_pmap'],
                    tbfunc_B_nN_by_nQ_A_alpha['dplyr_eval_flex'],
                    mt_nN_by_nQ_A_alpha)
colnames(mt_results) <- c('eval_dplyr_mutate',
                          'eval_dplyr_mutate_hcode',
                          'eval_dplyr_mutate_pmap',
                          'eval_dplyr_mutate_flex',
                          'A_child', 'alpha_child')
kable(mt_results) %>%
  kable_styling_fc_wide()
```

eval_dplyr_mutate	eval_dplyr_mutate_hcode	eval_dplyr_mutate_pmap	eval_dplyr_mutate_flex	A_child	alpha_child
2.346356	2.346356	2.346356	2.346356	-2	0.1
2.094273	2.094273	2.094273	2.094273	-1	0.3
1.895316	1.895316	1.895316	1.895316	0	0.5
1.733708	1.733708	1.733708	1.733708	1	0.7
1.599477	1.599477	1.599477	1.599477	2	0.9

Chapter 4

Multi-dimensional Data Structures

4.1 Generate, Gather, Bind and Join

4.1.1 Generate Panel Structure

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

4.1.1.1 Balanced Panel Skeleton

There are N individuals, each could be observed M times. In the example below, there are 3 students, each observed over 4 dates. This just uses the uncount function from *tidyr*.

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'student_id'
svr_date <- 'class_day'

# dataframe
df_panel_skeleton <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup()

# Print
kable(df_panel_skeleton) %>%
  kable_styling_fc()
```

4.1.1.2 Panel of Children with Height Growth

Given N individuals, each with G observations. There is an initial height variable and height grows every year. There are growth variables, variables for cumulative growth and variables for height at each age for each child.

Individuals are defined by gender (1 = female), race (1=asian), and birth height. Within individual yearly information includes height at each year of age.

```
# Define
it_N <- 5
it_M <- 3
svr_id <- 'indi_id'
svr_gender <- 'female'
svr_asian <- 'asian'
```

student_id	class_day
1	1
1	2
1	3
1	4
1	5
2	1
2	2
2	3
2	4
2	5
3	1
3	2
3	3
3	4
3	5

```
svr_age <- 'year_of_age'
# Define Height Related Variables
svr_brthgt <- 'birth_height'
svr_hgtgrow <- 'hgt_growth'
svr_hgtgrow_cumu <- 'hgt_growcumu'
svr_height <- 'height'

# panel dataframe following
set.seed(123)
df_panel_indiage <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  mutate(!!sym(svr_gender) := rbinom(n(), 1, 0.5),
         !!sym(svr_asian) := rbinom(n(), 1, 0.5),
         !!sym(svr_brthgt) := rnorm(n(), mean=60,sd=3)) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_gender), !!sym(svr_asian), !!sym(svr_brthgt)) %>%
  mutate(!!sym(svr_age) := row_number(),
         !!sym(svr_hgtgrow) := runif(n(), min=5, max=15),
         !!sym(svr_hgtgrow_cumu) := cumsum(!!sym(svr_hgtgrow)),
         !!sym(svr_height) := !!sym(svr_brthgt) + !!sym(svr_hgtgrow_cumu))  %>%
  ungroup()

# Add Height Index
kable(df_panel_indiage) %>% kable_styling_fc()
```

4.1.1.3 Create Group IDs

Given the dataframe just created, generate group IDs for each Gender and Race Groups. Given that both are binary, there can only be 4 unique groups.

```
# group id
svr_group_id <- 'female_asian_id'
# Define
ls_svr_group_vars <- c('female', 'asian')

# panel dataframe following
df_panel_indiage_id <- df_panel_indiage %>%
  arrange(!!!syms(ls_svr_group_vars)) %>%
  group_by(!!!syms(ls_svr_group_vars)) %>%
  mutate(!!sym(svr_group_id) := (row_number()==1)*1) %>%
  ungroup() %>%
```

female	asian	birth_height	year_of_age	hgt_growth	hgt_growcumu	height
0	0	65.14520	1	13.895393	13.895393	79.04059
0	0	65.14520	2	11.928034	25.823427	90.96862
0	0	65.14520	3	11.405068	37.228495	102.37369
1	1	61.38275	1	11.907053	11.907053	73.28980
1	1	61.38275	2	12.954674	24.861727	86.24448
1	1	61.38275	3	5.246137	30.107864	91.49061
0	1	56.20482	1	14.942698	14.942698	71.14751
0	1	56.20482	2	11.557058	26.499756	82.70457
0	1	56.20482	3	12.085305	38.585060	94.78988
1	1	57.93944	1	6.471137	6.471137	64.41058
1	1	57.93944	2	14.630242	21.101379	79.04082
1	1	57.93944	3	14.022991	35.124369	93.06381
1	0	58.66301	1	10.440660	10.440660	69.10367
1	0	58.66301	2	10.941420	21.382081	80.04509
1	0	58.66301	3	7.891597	29.273678	87.93669

```
  mutate(!!sym(svr_group_id) := cumsum(!!sym(svr_group_id))) %>%
  select(one_of(svr_group_id, ls_svr_group_vars), everything())

# Add Height Index
kable(df_panel_indiage_id) %>%
  kable_styling_fc_wide()
```

female_asian_id	female	asian	birth_height	year_of_age	hgt_growth	hgt_growcumu	height
1	0	0	65.14520	1	13.895393	13.895393	79.04059
1	0	0	65.14520	2	11.928034	25.823427	90.96862
1	0	0	65.14520	3	11.405068	37.228495	102.37369
2	0	1	56.20482	1	14.942698	14.942698	71.14751
2	0	1	56.20482	2	11.557058	26.499756	82.70457
2	0	1	56.20482	3	12.085305	38.585060	94.78988
3	1	0	58.66301	1	10.440660	10.440660	69.10367
3	1	0	58.66301	2	10.941420	21.382081	80.04509
3	1	0	58.66301	3	7.891597	29.273678	87.93669
4	1	1	61.38275	1	11.907053	11.907053	73.28980
4	1	1	61.38275	2	12.954674	24.861727	86.24448
4	1	1	61.38275	3	5.246137	30.107864	91.49061
4	1	1	57.93944	1	6.471137	6.471137	64.41058
4	1	1	57.93944	2	14.630242	21.101379	79.04082
4	1	1	57.93944	3	14.022991	35.124369	93.06381

4.1.2 Join Datasets

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

4.1.2.1 Join Panel with Multiple Keys

We have two datasets, one for student enrollment, panel over time, but some students do not show up on some dates. The other is a skeleton panel with all student ID and all dates. Often we need to join dataframes together, and we need to join by the student ID and the panel time Key at the same time. When students show up, there is a quiz score for that day, so the joined panel should have as data column quiz score

Student count is N, total dates are M. First we generate two panels below, then we join by both keys using *left_join*. First, define dataframes:

```
# Define
it_N <- 4
it_M <- 3
svr_id <- 'sid'
svr_date <- 'classday'
svr_attend <- 'date_in_class'

# Panel Skeleton
df_panel_balanced_skeleton <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup()
# Print
kable(df_panel_balanced_skeleton) %>%
  kable_styling_fc()
```

sid	classday
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3
4	1
4	2
4	3

```
# Smaller Panel of Random Days in School
set.seed(456)
df_panel_attend <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
  ungroup() %>% mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  filter(in_class == 1) %>% select(!!sym(svr_id), !!sym(svr_date)) %>%
  rename(!!sym(svr_attend) := !!sym(svr_date)) %>%
  mutate(dayquizscore = rnorm(n(),mean=80,sd=10))
# Print
kable(df_panel_attend) %>%
  kable_styling_fc()
```

sid	date_in_class	dayquizscore
1	1	89.88726
2	1	96.53929
2	2	65.59195
2	3	99.47356
4	2	97.36936

Second, now join dataframes:

```
# Join with explicit names
df_quiz_joined_multikey <- df_panel_balanced_skeleton %>%
  left_join(df_panel_attend,
```

```
            by=(c('sid'='sid', 'classday'='date_in_class')))

# Join with setname strings
df_quiz_joined_multikey_setnames <- df_panel_balanced_skeleton %>%
  left_join(df_panel_attend, by=setNames(c('sid', 'date_in_class'), c('sid', 'classday')))

# Print
kable(df_quiz_joined_multikey) %>%
  kable_styling_fc()
```

sid	classday	dayquizscore
1	1	89.88726
1	2	NA
1	3	NA
2	1	96.53929
2	2	65.59195
2	3	99.47356
3	1	NA
3	2	NA
3	3	NA
4	1	NA
4	2	97.36936
4	3	NA

```
kable(df_quiz_joined_multikey_setnames) %>%
  kable_styling_fc()
```

sid	classday	dayquizscore
1	1	89.88726
1	2	NA
1	3	NA
2	1	96.53929
2	2	65.59195
2	3	99.47356
3	1	NA
3	2	NA
3	3	NA
4	1	NA
4	2	97.36936
4	3	NA

4.1.2.2 Stack Panel Frames Together

There are multiple panel dataframe, each for different subsets of dates. All variable names and units of observations are identical. Use DPLYR bind_rows.

```
# Define
it_N <- 2 # Number of individuals
it_M <- 3 # Number of Months
svr_id <- 'sid'
svr_date <- 'date'

# Panel First Half of Year
df_panel_m1tom3 <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number()) %>%
```

```
  ungroup()

# Panel Second Half of Year
df_panel_m4tom6 <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(!!sym(svr_date) := row_number() + 3) %>%
  ungroup()

# Bind Rows
df_panel_m1tm6 <- bind_rows(df_panel_m1tom3, df_panel_m4tom6) %>% arrange(!!!syms(c(svr_id, svr_date

# Print
kable(df_panel_m1tom3) %>%
  kable_styling_fc()
```

sid	date
1	1
1	2
1	3
2	1
2	2
2	3

```
kable(df_panel_m4tom6) %>%
  kable_styling_fc()
```

sid	date
1	4
1	5
1	6
2	4
2	5
2	6

```
kable(df_panel_m1tm6) %>%
  kable_styling_fc()
```

sid	date
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
2	3
2	4
2	5
2	6

4.1.3 Gather Files

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

4.1.3.1 Stack CSV Files Together Extract and Select Variables

There are multiple csv files, each was simulated with a different combination of parameters, each file has the same columns and perhaps even the same number of rows. We want to combine the files together, and provide correct attributes to rows from each table stacked, based on each underlying csv file's file name.

This is necessary, for example, when running computational exercises across EC2 instances in batch array and files are saved to different S3 folders. Need to gather parallel computational results together in a single file after syncing files locally with S3.

In the csv folder under this section, there are four subfolder, each containing 3 files with identical file structures.

We want to find the relevant csv files from these directories, and stack the results together.

1. File search search string, search in all subfolders, the search string contains file prefix that is common across files that need to be gathered.
2. Extract path folder hierarchy, each layer of folder is a different variable
3. Stack files together, with variables for file name and folder name
4. Extract from file name the component that is not in the search string, keep as separate variable
5. Follow specific rules about how file suffix is constructed to obtain additional variables.
6. Keep only a subset of columns of interest.

First, search and find all files with certain prefix.

```
# can search in multiple paths, second path here has no relevant contents
spt_roots <- c('C:/Users/fan/R4Econ/panel/basic/_file/csv',
               'C:/Users/fan/R4Econ/panel/basic/_file/tex')
# can skip file names with certain strings
spn_skip <- c('A3420')
# prefix search patther,
st_search_str <- 'solu_19E1NEp99r99x_ITG_PE_cev_*'

# Search and get all Path
ls_sfls <- list.files(path=spt_roots,
                      recursive=T,
                      pattern=st_search_str,
                      full.names=T)

# Skip path if contains words in skip list
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}
```

Second, show all the files found, show their full path, the file name and the two folder names above the file name.

```
# Loop and print found files
it_folders_names_to_keep = 2
for (spt_file in ls_sfls) {
    ls_srt_folders_name_keep <- tail(strsplit(spt_file, "/")[[1]], n=it_folders_names_to_keep+1)
    snm_file_name <- tail(ls_srt_folders_name_keep, 1)
    ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)
    print(paste0('path:', spt_file))
    print(snm_file_name)
    print(ls_srt_folders_keep)
}
```

```
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-2000/solu_19E1NEp99r99x_ITG_PE_cev_c0_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0.csv"
## [1] "csv"      "cev-2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-2000/solu_19E1NEp99r99x_ITG_PE_cev_c0_cev
```

```
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840.csv"
## [1] "csv"      "cev-2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-947/solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0.csv"
## [1] "csv"     "cev-947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev-947/solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840.csv"
## [1] "csv"     "cev-947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev2000/solu_19E1NEp99r99x_ITG_PE_cev_c19_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0.csv"
## [1] "csv"     "cev2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev2000/solu_19E1NEp99r99x_ITG_PE_cev_c19_cev
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840.csv"
## [1] "csv"     "cev2000"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev947/solu_19E1NEp99r99x_ITG_PE_cev_c14_cev9
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0.csv"
## [1] "csv"    "cev947"
## [1] "path:C:/Users/fan/R4Econ/panel/basic/_file/csv/cev947/solu_19E1NEp99r99x_ITG_PE_cev_c14_cev9
## [1] "solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A6840.csv"
## [1] "csv"    "cev947"
```

Third, create a dataframe with the folder and file names:

```
# String matrix empty
mt_st_paths_names <- matrix(data=NA, nrow=length(ls_sfls), ncol=4)

# Loop and print found files
it_folders_names_to_keep = 2
it_file_counter = 0
for (spt_file in ls_sfls) {
    # row counter
    it_file_counter = it_file_counter + 1

    # get file paths
    ls_srt_folders_name_keep <- tail(strsplit(spt_file, "/")[[1]], n=it_folders_names_to_keep+1)
    snm_file_name <- tail(ls_srt_folders_name_keep, 1)
    ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)

    # store
    # tools::file_path_sans_ext to drop suffix
    mt_st_paths_names[it_file_counter,1] = tools::file_path_sans_ext(snm_file_name)
    mt_st_paths_names[it_file_counter,2] = ls_srt_folders_keep[1]
    mt_st_paths_names[it_file_counter,3] = ls_srt_folders_keep[2]
    mt_st_paths_names[it_file_counter,4] = spt_file
}

# Column Names
ar_st_varnames <- c('fileid','name','folder1','folder2', 'fullpath')

# Combine to tibble, add name col1, col2, etc.
tb_csv_info <- as_tibble(mt_st_paths_names) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_csv_info[,1:4]) %>% kable_styling_fc()
```

Fourth, create a dataframe by expanding each row with the datafile loaded in, use apply with anonymous function.

fileid	name	folder1	folder2
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	csv	cev-2000
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	csv	cev-2000
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	csv	cev-947
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	csv	cev-947
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	csv	cev2000
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	csv	cev2000
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	csv	cev947
8	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A6840	csv	cev947

```
# Generate a list of dataframes
ls_df_loaded_files =
  apply(tb_csv_info,
        1,
        function(row) {
          # Loading file
          spn_full_path <- row[5]
          mt_csv = read.csv(file = spn_full_path)
          # dataframe
          it_fileid <- row[1]
          snm_filename <- row[2]
          srt_folder_level2 <- row[3]
          srt_folder_level1 <- row[4]

          tb_combine = as_tibble(mt_csv) %>%
            na.omit %>%
            rowid_to_column(var = "statesid") %>%
            mutate(fileid = it_fileid,
                   filename = snm_filename,
                   folder_lvl1 = srt_folder_level1,
                   folder_lvl2 = srt_folder_level2) %>%
            select(fileid, filename, folder_lvl1, folder_lvl2,
                   statesid, everything())
          # return
          return(tb_combine)
        })

# Stack dataframes together
df_all_files = do.call(bind_rows, ls_df_loaded_files)

# show stacked table
kable(df_all_files[seq(1,601,50),1:6]) %>% kable_styling_fc_wide()
```

fileid	filename	folder_lvl1	folder_lvl2	statesid	EjV
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	cev-2000	csv	1	-28.8586860
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	cev-2000	csv	51	-0.2106603
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	cev-2000	csv	3	-28.8586860
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	cev-2000	csv	53	-0.0642997
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	cev-947	csv	5	-5.8826609
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	cev-947	csv	55	0.0353187
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	cev-947	csv	7	-2.7046907
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	cev-947	csv	57	0.1094474
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	cev2000	csv	9	-2.9782236
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	cev2000	csv	59	0.3389275
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	cev2000	csv	11	-1.7229647
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	cev2000	csv	61	-14.6880377
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	cev947	csv	13	-1.7623279

Fifth, get additional information from the file name and file folder. Extract those as separate variables.

The file names is dash connected, with various information. First, split just the final element of the string file name out, which is *A###*. Then, also extract the number next to N as a separate numeric column. Additional *folder_lvl1* separate out the numeric number from the initial word *cev*.

Split "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A###" to "solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000" and "A###":

```
# separate last eleemtnafter underscore
df_all_files_finalA <- df_all_files %>%
   separate(filename, into = c("filename_main", "prod_type_st"),
            sep="_(?=[^_]+$)",
            remove = FALSE) %>%
   select(fileid, filename, filename_main, prod_type_st, folder_lvl1, folder_lvl2,
          statesid, everything())
# show stacked table
kable(df_all_files_finalA[seq(1,601,50),1:10]) %>% kable_styling_fc_wide()
```

fileid	filename	filename_main	prod_type_st	folder_lvl1	folder_lvl2	statesid	EjV	k_tt	b_tt
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A0	cev-2000	csv	1	-28.8586860	0.000000	0.000000
1	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A0	cev-2000	csv	51	-0.2106603	0.000000	78.005300
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A6840	cev-2000	csv	3	-28.8586860	0.000000	0.000000
2	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c0_cev-2000	A6840	cev-2000	csv	53	-0.0642997	0.000000	84.215368
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A0	cev-947	csv	5	-5.8826609	0.000000	3.869909
3	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A0	cev-947	csv	55	0.0353187	0.000000	90.611739
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A6840	cev-947	csv	7	-2.7046907	0.000000	7.855916
4	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c5_cev-947	A6840	cev-947	csv	57	0.1094474	0.000000	90.611739
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A0	cev2000	csv	9	-2.9782236	0.000000	7.855916
5	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A0	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A0	cev2000	csv	59	0.3389275	0.000000	97.200000
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A6840	cev2000	csv	11	-1.7229647	0.000000	11.961502
6	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000_A6840	solu_19E1NEp99r99x_ITG_PE_cev_c19_cev2000	A6840	cev2000	csv	61	-14.6880377	1.990694	-1.879215
7	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947_A0	solu_19E1NEp99r99x_ITG_PE_cev_c14_cev947	A0	cev947	csv	13	-1.7623279	0.000000	16.190257

Split "A###" to "A" and "A###". Additionally, also split *cev####* to *cev* and *####*, allow for positive and negative numbers. See regular expression 101 helper

```
# string and number separation
df_all_files_finalB <- df_all_files_finalA %>%
  separate(prod_type_st,
           into = c("prod_type_st_prefix", "prod_type_lvl"),
           sep="(?<=[A-Za-z])(?=[-0-9])", # positive or negative numbers
           remove=FALSE) %>%
  separate(folder_lvl1,
           into = c("cev_prefix", "cev_lvl"),
           sep="(?<=[A-Za-z])(?=[-0-9])", # positive or negative numbers
           remove=FALSE) %>%
  mutate(cev_st = folder_lvl1,
         prod_type_lvl = as.numeric(prod_type_lvl),
         cev_lvl = as.numeric(cev_lvl)/10000) %>%
  select(fileid,
         prod_type_st, prod_type_lvl,
         cev_st, cev_lvl,
         statesid, EjV,
         filename, folder_lvl1, folder_lvl2)
# Ordering, sort by cev_lvl, then prod_type_lvl, then stateid
df_all_files_finalB <- df_all_files_finalB %>%
  arrange(cev_lvl, prod_type_lvl, statesid)
# show stacked table
kable(df_all_files_finalB[seq(1,49*16,49),1:7]) %>% kable_styling_fc_wide()
```

4.2 Wide and Long

4.2.1 Long to Wide

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Using the pivot_wider function in tidyr to reshape panel or other data structures

fileid	prod_type_st	prod_type_lvl	cev_st	cev_lvl	statesid	EjV
1	A0	0	cev-2000	-0.2000	1	-28.8586860
1	A0	0	cev-2000	-0.2000	50	-0.2106603
2	A6840	6840	cev-2000	-0.2000	1	-28.8586860
2	A6840	6840	cev-2000	-0.2000	50	-0.1311749
3	A0	0	cev-947	-0.0947	1	-28.0399281
3	A0	0	cev-947	-0.0947	50	-0.0911499
4	A6840	6840	cev-947	-0.0947	1	-28.0399281
4	A6840	6840	cev-947	-0.0947	50	-0.0134719
7	A0	0	cev947	0.0947	1	-26.8243673
7	A0	0	cev947	0.0947	50	0.0857474
8	A6840	6840	cev947	0.0947	1	-26.8243673
8	A6840	6840	cev947	0.0947	50	0.1608382
5	A0	0	cev2000	0.2000	1	-26.2512036
5	A0	0	cev2000	0.2000	50	0.1694524
6	A6840	6840	cev2000	0.2000	1	-26.2512036
6	A6840	6840	cev2000	0.2000	50	0.2432677

4.2.1.1 Panel Long Attendance Roster to Wide

There are N students in class, but only a subset of them attend class each day. If student id_i is in class on day Q, the teacher records on a sheet the date and the student ID. So if the student has been in class 10 times, the teacher has ten rows of recorded data for the student with two columns: column one is the student ID, and column two is the date on which the student was in class. Suppose there were 50 students, who on average attended exactly 10 classes each during the semester, this means we have $10 \cdot 50$ rows of data, with differing numbers of rows for each student. This is shown as *df_panel_attend_date* generated below.

Now we want to generate a new dataframe, where each row is a date, and each column is a student. The values in the new dataframe shows, at the Q^{th} day, how many classes student i has attended so far. The following results is also in a REconTools Function. This is shown as *df_attend_cumu_by_day* generated below.

First, generate the raw data structure, *df_panel_attend_date*:

```
# Define
it_N <- 3
it_M <- 5
svr_id <- 'student_id'

# from : support/rand/fs_rand_draws.Rmd
set.seed(222)
df_panel_attend_date <- as_tibble(matrix(it_M, nrow=it_N, ncol=1)) %>%
  rowid_to_column(var = svr_id) %>%
  uncount(V1) %>%
  group_by(!!sym(svr_id)) %>% mutate(date = row_number()) %>%
  ungroup() %>% mutate(in_class = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>%
  filter(in_class == 1) %>% select(!!sym(svr_id), date) %>%
  rename(date_in_class = date)

# Print
kable(df_panel_attend_date) %>%
  kable_styling_fc()
```

Second, generate wider data structure, *df_attend_cumu_by_day*:

student_id	date_in_class
1	2
1	4
2	1
2	2
2	5
3	2
3	3
3	5

```
# Define
svr_id <- 'student_id'
svr_date <- 'date_in_class'
st_idcol_prefix <- 'sid_'

# Generate cumulative enrollment counts by date
df_panel_attend_date_addone <- df_panel_attend_date %>% mutate(attended = 1)
kable(df_panel_attend_date_addone) %>%
  kable_styling_fc()
```

student_id	date_in_class	attended
1	2	1
1	4	1
2	1	1
2	2	1
2	5	1
3	2	1
3	3	1
3	5	1

```
# Pivot Wide
df_panel_attend_date_wider <- df_panel_attend_date_addone %>%
  pivot_wider(names_from = svr_id,
              values_from = attended)
kable(df_panel_attend_date_wider) %>%
  kable_styling_fc()
```

date_in_class	1	2	3
2	1	1	1
4	1	NA	NA
1	NA	1	NA
5	NA	1	1
3	NA	NA	1

```
# Sort and rename
# rename see: https://fanwangecon.github.io/R4Econ/amto/tibble/fs_tib_basics.html
ar_unique_ids <- sort(unique(df_panel_attend_date %>% pull(!!sym(svr_id))))
df_panel_attend_date_wider_sort <- df_panel_attend_date_wider %>%
    arrange(!!sym(svr_date)) %>%
    rename_at(vars(num_range('',ar_unique_ids))
              , list(~paste0(st_idcol_prefix, . , ''))
              )
kable(df_panel_attend_date_wider_sort) %>%
  kable_styling_fc()
```

date_in_class	sid_1	sid_2	sid_3
1	NA	1	NA
2	1	1	1
3	NA	NA	1
4	1	NA	NA
5	NA	1	1

```
# replace NA and cumusum again
# see: R4Econ/support/function/fs_func_multivar for renaming and replacing
df_attend_cumu_by_day <- df_panel_attend_date_wider_sort %>%
  mutate_at(vars(contains(st_idcol_prefix)), list(~replace_na(., 0))) %>%
  mutate_at(vars(contains(st_idcol_prefix)), list(~cumsum(.)))

kable(df_attend_cumu_by_day) %>%
  kable_styling_fc()
```

date_in_class	sid_1	sid_2	sid_3
1	0	1	0
2	1	2	1
3	1	2	2
4	2	2	2
5	2	3	3

The structure above is also a function in Fan's REconTools Package, here the function is tested:

```
# Parameters
df <- df_panel_attend_date
svr_id_i <- 'student_id'
svr_id_t <- 'date_in_class'
st_idcol_prefix <- 'sid_'

# Invoke Function
ls_df_rosterwide <- ff_panel_expand_longrosterwide(df, svr_id_t, svr_id_i, st_idcol_prefix)
df_roster_wide_func <- ls_df_rosterwide$df_roster_wide
df_roster_wide_cumu_func <- ls_df_rosterwide$df_roster_wide_cumu

# Print
print(df_roster_wide_func)
print(df_roster_wide_cumu_func)
```

4.2.2 Wide to Long

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Using the pivot_wider function in tidyr to reshape panel or other data structures

4.2.2.1 Generated Matrix by States to Long Table

A matrix of ev given states, rows are states and cols are shocks. Convert to Long table with shock and state values and ev.

Generated Matrix by States to Long Table where state values are stored as variables, with correct value labels for states:

1. Generate a matrix
2. Convert matrix to tibble
3. Tibble make longer, and store column and row id var names

```
# Generate A Matrix
set.seed(123)
ar_a <- c(1.1,5.1)
ar_z <- seq(-2.5, 2.53, length.out=11)
mt_ev = matrix(rnorm(ar_a*ar_z), nrow=length(ar_a), ncol=length(ar_z))

# Name Matrix
rownames(mt_ev) <- paste0('ai', seq(1:length(ar_a)))
colnames(mt_ev) <- paste0('zi', seq(1:length(ar_z)))

# to tibble
tb_ev <- as_tibble(mt_ev) %>% rowid_to_column(var = "ai")

# longer
tb_ev_long <- tb_ev %>%
  pivot_longer(cols = starts_with('zi'),
               names_to = c('zi'),
               names_pattern = paste0("zi(.*)"),
               values_to = "ev") %>%
  mutate(zi = as.numeric(zi))

# Merge with a and z values
tb_ev_long <- tb_ev_long %>%
  left_join(as_tibble(ar_a) %>%
              rowid_to_column(var = "ai") %>%
              rename(a = value)
              , by = 'ai') %>%
  left_join(as_tibble(ar_z) %>%
              rowid_to_column(var = "zi") %>%
              rename(z = value),
            by = 'zi') %>%
  select(a,ai,z,zi,ev)

# Display
kable(tb_ev) %>% kable_styling_fc_wide()
```

ai	zi1	zi2	zi3	zi4	zi5	zi6	zi7	zi8	zi9	zi10	zi11
1	-0.5604756	1.5587083	0.1292877	0.4609162	-0.6868529	1.2240818	-0.2301775	0.0705084	1.7150650	-1.2650612	-0.445662
2	-0.2301775	0.0705084	1.7150650	-1.2650612	-0.4456620	-0.5604756	1.5587083	0.1292877	0.4609162	-0.6868529	1.224082

```
kable(tb_ev_long) %>% kable_styling_fc()
```

4.3 Join and Compare

4.3.1 Find Closest Neighbor on Grid

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Using the pivot_wider function in tidyr to reshape panel or other data structures

4.3.1.1 Closest Neighbor on Grid

There is a dataframe that provides $V(coh, a, cev)$ levels. There is another dataframe with $\hat{V}(coh, a)$, for each coh, a, find the cev that such that the difference between $\hat{V}(coh, a)$ and $V(coh, a, cev)$ is minimized.

V and $\hat{V}$ information are stored in a dataframe in the csv folder in the current directory. In fact, we have one V surface, but multiple $\hat{V}$ files, so we want to do the find closest neighbor exercise for each one of the several $\hat{V}$ files.

a	ai	z	zi	ev
1.1	1	-2.500	1	-0.5604756
1.1	1	-1.997	2	1.5587083
1.1	1	-1.494	3	0.1292877
1.1	1	-0.991	4	0.4609162
1.1	1	-0.488	5	-0.6868529
1.1	1	0.015	6	1.2240818
1.1	1	0.518	7	-0.2301775
1.1	1	1.021	8	0.0705084
1.1	1	1.524	9	1.7150650
1.1	1	2.027	10	-1.2650612
1.1	1	2.530	11	-0.4456620
5.1	2	-2.500	1	-0.2301775
5.1	2	-1.997	2	0.0705084
5.1	2	-1.494	3	1.7150650
5.1	2	-0.991	4	-1.2650612
5.1	2	-0.488	5	-0.4456620
5.1	2	0.015	6	-0.5604756
5.1	2	0.518	7	1.5587083
5.1	2	1.021	8	0.1292877
5.1	2	1.524	9	0.4609162
5.1	2	2.027	10	-0.6868529
5.1	2	2.530	11	1.2240818

The structure is as follows: (1) Load in the V file, where coh, a, cev are all variable attributes. (2) Merge with one $\hat{V}$ file. (3) Take the difference between the V and $\hat{V}$ columns, and take the absolute value of the difference. (4) Group by coh, a, and sort to get the smallest absolute difference among the cev possibilities, and slice out the row for the smallest. (5) Now We have $V(coh, a, cev^{\star}(coh, a))$. (6) Do this for each of the several $\hat{V}$ files. (7) Stack the results from 1 through 6 together, generate a column that identifies which simulation/exercise/counterfactual each of the $\hat{V}$ file comes from. (8) Visualize by plotting as subplot different a, coh is x-axis, different $\hat{V}$ outcome are different lines, and $cev^{\star}\left(coh, a, \hat{V}\right)$ is the y-axis outcome.

First, load the CEV file.

```
# folder
spt_root <- c('C:/Users/fan/R4Econ/panel/join/_file/csv')
# cev surface file, the V file
snm_cev_surface <- 'e_19E1NEp99r99_ITG_PE_cev_subsettest.csv'
mt_cev_surface <- read.csv(file = file.path(spt_root, snm_cev_surface))
tb_cev_surface <- as_tibble(mt_cev_surface) %>%
  rename(EjVcev = EjV)
```

Second, loop over the V hat files, join V with V hat:

```
ls_tb_cev_surfhat = vector(mode = "list", length = 4)
for (it_simu_counter in c(1,2,3,4)) {

    # conditionally change file names
    if (it_simu_counter == 1) {
        st_counter <- '19E1NEp99r99'
    } else if (it_simu_counter == 2) {
        st_counter <- '19E1NEp02r99'
    } else if (it_simu_counter == 3) {
        st_counter <- '19E1NEp02per02ger99'
    } else if (it_simu_counter == 4) {
        st_counter <- '19E1NEp02r02'
    }
```

```
    snm_v_hat <- paste0('e_', st_counter, '_ITG_PE_subsettest.csv')

    # Overall path to files
    mt_v_hat <- read.csv(file = file.path(spt_root, snm_v_hat))
    tb_v_hat <- as_tibble(mt_v_hat) %>%
      select(prod_type_lvl, statesid, EjV)

    # Merge file using key
    tb_cev_surfhat <- tb_cev_surface %>%
      left_join(tb_v_hat, by=(c('prod_type_lvl'='prod_type_lvl',
                                'statesid'='statesid'))) %>%
      arrange(statesid, prod_type_lvl, cev_lvl) %>%
      mutate(counter_policy = st_counter)

    # Store to list
    ls_tb_cev_surfhat[[it_simu_counter]] <- tb_cev_surfhat
}

# Display
kable(ls_tb_cev_surfhat[[1]][seq(1, 40, 5),]) %>% kable_styling_fc_wide()
```

X	cev_st	cev_lvl	prod_type_st	prod_type_lvl	statesid	cash_tt	EjVcev	EjV	counter_policy
1	cev-2000	-0.2000	A0	0	526	32.84747	-1.0479929	-0.7957419	19E1NEp99r99
1501	cev-947	-0.0947	A0	0	526	32.84747	-0.9079859	-0.7957419	19E1NEp99r99
3001	cev105	0.0105	A0	0	526	32.84747	-0.7880156	-0.7957419	19E1NEp99r99
4501	cev1157	0.1157	A0	0	526	32.84747	-0.6803586	-0.7957419	19E1NEp99r99
51	cev-2000	-0.2000	A2504	2504	526	32.90371	-1.0002921	-0.7504785	19E1NEp99r99
1551	cev-947	-0.0947	A2504	2504	526	32.90371	-0.8613743	-0.7504785	19E1NEp99r99
3051	cev105	0.0105	A2504	2504	526	32.90371	-0.7423281	-0.7504785	19E1NEp99r99
4551	cev1157	0.1157	A2504	2504	526	32.90371	-0.6354620	-0.7504785	19E1NEp99r99

Third, sort each file, and keep only the best match rows that minimize the absolute distance between *EjV* and *EjVcev*.

```
ls_tb_cev_matched = vector(mode = "list", length = 4)
for (it_simu_counter in c(1,2,3,4)) {

    # Load merged file
    tb_cev_surfhat <- ls_tb_cev_surfhat[[it_simu_counter]]

    # Difference Column
    tb_cev_surfhat <- tb_cev_surfhat %>%
      mutate(EjVcev_gap = abs(EjVcev - EjV))

    # Group by, Arrange and Slice, get lowest gap
    tb_cev_matched <- tb_cev_surfhat %>%
      arrange(statesid, prod_type_lvl, EjVcev_gap) %>%
      group_by(statesid, prod_type_lvl) %>%
      slice_head(n=1)

    # Store to list
    ls_tb_cev_matched[[it_simu_counter]] <- tb_cev_matched
}

# Display
kable(ls_tb_cev_matched[[2]][seq(1, 30, 1),]) %>% kable_styling_fc_wide()
```

Fourth, row_bind results together.

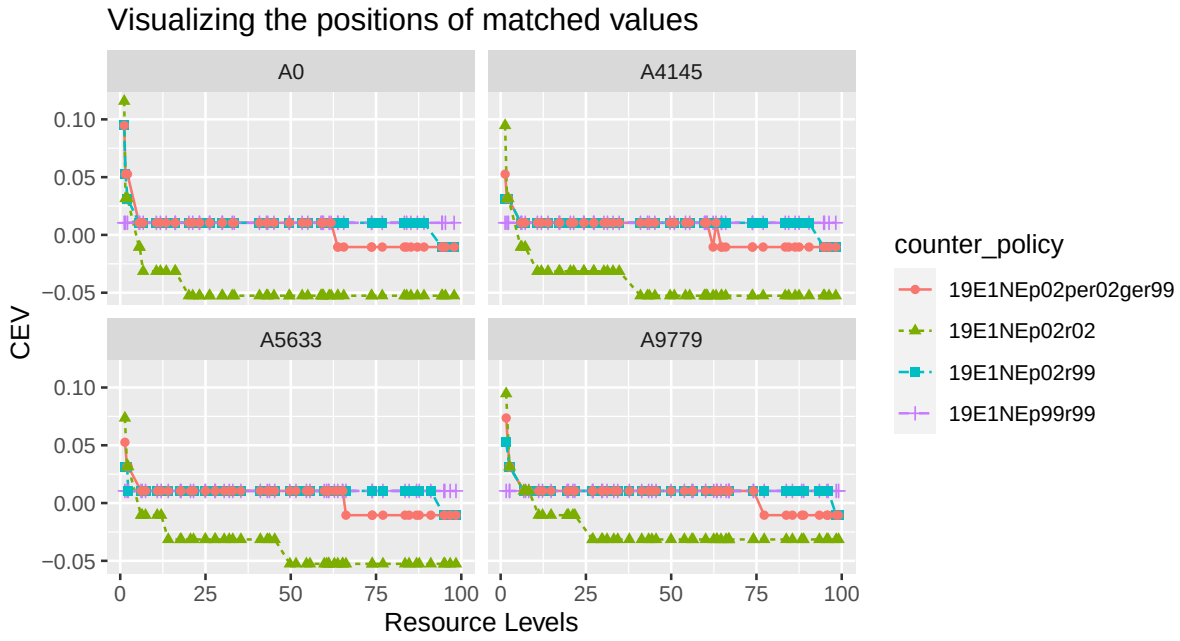
X	cev_st	cev_lvl	prod_type_st	prod_type_lvl	statesid	cash_tt	EjVcev	EjV	counter_policy	EjVcev_gap
3001	cev105	0.0105	A0	0	526	32.847471	-0.7880156	-0.7928034	19E1NEp02r99	0.0047878
3051	cev105	0.0105	A2504	2504	526	32.903714	-0.7423281	-0.7480617	19E1NEp02r99	0.0057336
3101	cev105	0.0105	A4145	4145	526	32.948970	-0.7082006	-0.7145418	19E1NEp02r99	0.0063412
3151	cev105	0.0105	A5633	5633	526	32.996952	-0.6753576	-0.6818996	19E1NEp02r99	0.0065420
3201	cev105	0.0105	A7274	7274	526	33.058832	-0.6368297	-0.6431710	19E1NEp02r99	0.0063413
3251	cev105	0.0105	A9779	9779	526	33.175241	-0.5711706	-0.5774648	19E1NEp02r99	0.0062942
3002	cev105	0.0105	A0	0	555	53.346587	-0.2985944	-0.3041922	19E1NEp02r99	0.0055978
3052	cev105	0.0105	A2504	2504	555	53.815772	-0.2617572	-0.2680026	19E1NEp02r99	0.0062454
3102	cev105	0.0105	A4145	4145	555	54.193302	-0.2340822	-0.2406142	19E1NEp02r99	0.0065320
3152	cev105	0.0105	A5633	5633	555	54.593579	-0.2067964	-0.2134634	19E1NEp02r99	0.0066670
3202	cev105	0.0105	A7274	7274	555	55.109790	-0.1740126	-0.1806320	19E1NEp02r99	0.0066194
3252	cev105	0.0105	A9779	9779	555	56.080888	-0.1169470	-0.1236111	19E1NEp02r99	0.0066641
3603	cev526	0.0526	A0	0	905	1.533025	-5.2530406	-5.2486887	19E1NEp02r99	0.0043519
3353	cev315	0.0315	A2504	2504	905	1.714498	-4.5517474	-4.5408560	19E1NEp02r99	0.0108913
3403	cev315	0.0315	A4145	4145	905	1.860521	-4.1039608	-4.1072736	19E1NEp02r99	0.0033128
3453	cev315	0.0315	A5633	5633	905	2.015341	-3.7465733	-3.7611842	19E1NEp02r99	0.0146109
3503	cev315	0.0315	A7274	7274	905	2.215003	-3.4101025	-3.4235413	19E1NEp02r99	0.0134388
3553	cev315	0.0315	A9779	9779	905	2.590608	-2.9413469	-2.9535570	19E1NEp02r99	0.0122101
3004	cev105	0.0105	A0	0	953	20.125381	-1.3249909	-1.3290865	19E1NEp02r99	0.0040957
3054	cev105	0.0105	A2504	2504	953	20.306854	-1.2476021	-1.2531860	19E1NEp02r99	0.0055839
3104	cev105	0.0105	A4145	4145	953	20.452876	-1.1916003	-1.1975215	19E1NEp02r99	0.0059211
3154	cev105	0.0105	A5633	5633	953	20.607697	-1.1383665	-1.1444048	19E1NEp02r99	0.0060383
3204	cev105	0.0105	A7274	7274	953	20.807359	-1.0766095	-1.0823344	19E1NEp02r99	0.0057250
3254	cev105	0.0105	A9779	9779	953	21.182964	-0.9729832	-0.9781408	19E1NEp02r99	0.0051576
3005	cev105	0.0105	A0	0	1017	63.774766	-0.1284542	-0.1342653	19E1NEp02r99	0.0058110
3055	cev105	0.0105	A2504	2504	1017	64.298911	-0.0967695	-0.1031112	19E1NEp02r99	0.0063417
3105	cev105	0.0105	A4145	4145	1017	64.720664	-0.0728485	-0.0793940	19E1NEp02r99	0.0065454
3155	cev105	0.0105	A5633	5633	1017	65.167829	-0.0490898	-0.0557238	19E1NEp02r99	0.0066341
3205	cev105	0.0105	A7274	7274	1017	65.744507	-0.0203378	-0.0269149	19E1NEp02r99	0.0065772
3255	cev105	0.0105	A9779	9779	1017	66.829359	0.0299397	0.0233507	19E1NEp02r99	0.0065890

```
# Single dataframe with all results
tb_cev_matched_all_counter <- do.call(bind_rows, ls_tb_cev_matched)
# check size
print(dim(tb_cev_matched_all_counter))
```

```
## [1] 1200   11
```

Fifth, visualize results

```
# select four from the productivity types
ar_prod_type_lvl_unique <- unique(tb_cev_matched_all_counter %>% pull(prod_type_lvl))
ar_prod_type_lvl_selected <- ar_prod_type_lvl_unique[round(seq(1, length(ar_prod_type_lvl_unique), l
# graph
lineplot <- tb_cev_matched_all_counter %>%
    filter(prod_type_lvl %in% ar_prod_type_lvl_selected) %>%
    group_by(prod_type_st, cash_tt) %>%
    ggplot(aes(x=cash_tt, y=cev_lvl,
               colour=counter_policy, linetype=counter_policy, shape=counter_policy)) +
        facet_wrap( ~ prod_type_st) +
        geom_line() +
        geom_point() +
        labs(title = 'Visualizing the positions of matched values',
             x = 'Resource Levels',
             y = 'CEV',
             caption = paste0('https://fanwangecon.github.io/',
                              'R4Econ/panel/join/htmlpdfr/fs_join_compare.html'))
print(lineplot)
```



https://fanwangecon.github.io/R4Econ/panel/join/htmlpdfr/fs_join_compare.html

Chapter 5

Linear Regression

5.1 OLS and IV

Back to **Fan**'s R4Econ Homepage **Table of Content**

5.1.1 OLS and IV Regression

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

IV regression using AER package. Option to store all results in dataframe row for combining results from other estimations together. Produce Row Statistics.

5.1.1.1 Construct Program

```
# IV regression function
# The code below uses the AER library's regresison function
# All results are stored in a single row as data_frame
# This functoin could work with dplyr do
# var.y is single outcome, vars.x, vars.c and vars.z are vectors of endogenous variables, controls a
regf.iv <- function(var.y, vars.x,
                    vars.c, vars.z, df, transpose=TRUE) {

  # A. Set-Up Equation
  str.vars.x <- paste(vars.x, collapse='+')
  str.vars.c <- paste(vars.c, collapse='+')

  df <- df %>%
    select(one_of(var.y, vars.x, vars.c, vars.z)) %>%
    drop_na() %>% filter_all(all_vars(!is.infinite(.)))

  if (length(vars.z) >= 1) {
    #     library(AER)
    str.vars.z <- paste(vars.z, collapse='+')
    equa.iv <- paste(var.y,
                     paste(paste(str.vars.x, str.vars.c, sep='+'),
                           paste(str.vars.z, str.vars.c, sep='+'),
                           sep='|'),
                     sep='~')
    #     print(equa.iv)

    # B. IV Regression
    ivreg.summ <- summary(ivreg(as.formula(equa.iv), data=df),
```

```
                        vcov = sandwich, df = Inf, diagnostics = TRUE)

  # C. Statistics from IV Regression
  #     ivreg.summ$coef
  #     ivreg.summ$diagnostics

  # D. Combine Regression Results into a Matrix
  df.results <- suppressWarnings(suppressMessages(
    as_tibble(ivreg.summ$coef, rownames='rownames') %>%
      full_join(as_tibble(ivreg.summ$diagnostics, rownames='rownames')) %>%
      full_join(tibble(rownames=c('vars'),
                       var.y=var.y,
                       vars.x=str.vars.x,
                       vars.z=str.vars.z,
                       vars.c=str.vars.c))))
} else {

  # OLS regression
  equa.ols <- paste(var.y,
                    paste(paste(vars.x, collapse='+'),
                          paste(vars.c, collapse='+'), sep='+'),
                    sep='~')

  lmreg.summ <- summary(lm(as.formula(equa.ols), data=df))

  lm.diagnostics <- as_tibble(
    list(df1=lmreg.summ$df[[1]],
         df2=lmreg.summ$df[[2]],
         df3=lmreg.summ$df[[3]],
         sigma=lmreg.summ$sigma,
         r.squared=lmreg.summ$r.squared,
         adj.r.squared=lmreg.summ$adj.r.squared)) %>%
    gather(variable, value) %>%
    rename(rownames = variable) %>%
    rename(v = value)

  df.results <- suppressWarnings(suppressMessages(
    as_tibble(lmreg.summ$coef, rownames='rownames') %>%
      full_join(lm.diagnostics) %>%
      full_join(tibble(rownames=c('vars'),
                       var.y=var.y,
                       vars.x=str.vars.x,
                       vars.c=str.vars.c))))
}

# E. Flatten Matrix, All IV results as a single tibble
# row to be combined with other IV results
df.row.results <- df.results %>%
  gather(variable, value, -rownames) %>%
  drop_na() %>%
  unite(esti.val, rownames, variable) %>%
  mutate(esti.val = gsub(' ', '', esti.val))

if (transpose) {
  df.row.results <- df.row.results %>% spread(esti.val, value)
}
```

```
  # F. Return
  return(data.frame(df.row.results))
}
```

5.1.1.2 Program Testing

Load Data

```
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

```
# One Instrucments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- NULL
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

esti.val	value
(Intercept)_Estimate	52.1186286658651
prot_Estimate	0.374472386357917
sexMale_Estimate	0.611043720578292
hgt0_Estimate	0.148513781160842
wgt0_Estimate	0.00150560230505631
(Intercept)_Std.Error	1.57770483608693
prot_Std.Error	0.00418121191133815
sexMale_Std.Error	0.118396259120659
hgt0_Std.Error	0.0393807494783186
wgt0_Std.Error	0.000187123663624397
(Intercept)_tvalue	33.0344608660332
prot_tvalue	89.5607288744356
sexMale_tvalue	5.16100529794248
hgt0_tvalue	3.77122790013449
wgt0_tvalue	8.04602836377991
(Intercept)_Pr(>|t|)	9.92126150975783e-233
prot_Pr(>|t|)	0
sexMale_Pr(>|t|)	2.48105505495642e-07
hgt0_Pr(>|t|)	0.000162939618371183
wgt0_Pr(>|t|)	9.05257561534111e-16
df1_v	5
df2_v	18958
df3_v	5
sigma_v	8.06197784622979
r.squared_v	0.319078711001325
adj.r.squared_v	0.318935041565942
vars_var.y	hgt
vars_vars.x	prot
vars_vars.c	sex+hgt0+wgt0

5.1.1.2.1 Example No Instrument, OLS

```
# One Instrucments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- c('momEdu')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

esti.val	value
(Intercept)_Estimate	43.4301969117558
prot_Estimate	0.130833343849446
sexMale_Estimate	0.868121847262411
hgt0_Estimate	0.412093881817148
wgt0_Estimate	0.000858630042617921
(Intercept)_Std.Error	1.82489550971182
prot_Std.Error	0.0192036220809189
sexMale_Std.Error	0.13373016700542
hgt0_Std.Error	0.0459431912927002
wgt0_Std.Error	0.00022691057702563
(Intercept)_zvalue	23.798730766023
prot_zvalue	6.81295139521853
sexMale_zvalue	6.49159323361366
hgt0_zvalue	8.96963990141069
wgt0_zvalue	3.7840018472164
(Intercept)_Pr(>|z|)	3.4423766196876e-125
prot_Pr(>|z|)	9.56164541643828e-12
sexMale_Pr(>|z|)	8.49333228172763e-11
hgt0_Pr(>|z|)	2.97485394526792e-19
wgt0_Pr(>|z|)	0.000154326676608523
Weakinstruments_df1	1
Wu-Hausman_df1	1
Sargan_df1	0
Weakinstruments_df2	16394
Wu-Hausman_df2	16393
Weakinstruments_statistic	935.817456612075
Wu-Hausman_statistic	123.595856606729
Weakinstruments_p-value	6.39714929178024e-200
Wu-Hausman_p-value	1.30703637796748e-28
vars_var.y	hgt
vars_vars.x	prot
vars_vars.z	momEdu
vars_vars.c	sex+hgt0+wgt0

5.1.1.2.2 Example 1 Insturment

```
# Multiple Instrucments
var.y <- c('hgt')
vars.x <- c('prot')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
```

```
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

esti.val	value
(Intercept)_Estimate	42.2437613555242
prot_Estimate	0.26699945194704
sexMale_Estimate	0.695548488812932
hgt0_Estimate	0.424954881263031
wgt0_Estimate	0.000486951420329484
(Intercept)_Std.Error	1.85356686789642
prot_Std.Error	0.0154939347964083
sexMale_Std.Error	0.133157977814374
hgt0_Std.Error	0.0463195803786233
wgt0_Std.Error	0.000224867994873235
(Intercept)_zvalue	22.7905246296649
prot_zvalue	17.2325142357597
sexMale_zvalue	5.22348341593581
hgt0_zvalue	9.17441129192849
wgt0_zvalue	2.16549901022595
(Intercept)_Pr(>|z|)	5.69294074735747e-115
prot_Pr(>|z|)	1.51424021931607e-66
sexMale_Pr(>|z|)	1.75588197502565e-07
hgt0_Pr(>|z|)	4.54048595587756e-20
wgt0_Pr(>|z|)	0.030349491114332
Weakinstruments_df1	4
Wu-Hausman_df1	1
Sargan_df1	3
Weakinstruments_df2	14914
Wu-Hausman_df2	14916
Weakinstruments_statistic	274.147084958343
Wu-Hausman_statistic	17.7562545747101
Sargan_statistic	463.729664547249
Weakinstruments_p-value	8.61731956233366e-228
Wu-Hausman_p-value	2.52567249124181e-05
Sargan_p-value	3.45452874915475e-100
vars_var.y	hgt
vars_vars.x	prot
vars_vars.z	momEdu+wealthIdx+p.A.prot+p.A.nProt
vars_vars.c	sex+hgt0+wgt0

5.1.1.2.3 Example Multiple Instrucments

```
# Multiple Instrucments
var.y <- c('hgt')
vars.x <- c('prot', 'cal')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
# Regression
regf.iv(var.y, vars.x, vars.c, vars.z, df, transpose=FALSE) %>%
  kable() %>%
  kable_styling_fc()
```

5.1.1.2.4 Example Multiple Endogenous Variables

esti.val	value
(Intercept)_Estimate	44.0243196254297
prot_Estimate	-1.4025623247106
cal_Estimate	0.065104895750151
sexMale_Estimate	0.120832787571818
hgt0_Estimate	0.286525437984517
wgt0_Estimate	0.000850481389651033
(Intercept)_Std.Error	2.75354847244082
prot_Std.Error	0.198640060273635
cal_Std.Error	0.00758881298880996
sexMale_Std.Error	0.209984580636303
hgt0_Std.Error	0.0707828182888255
wgt0_Std.Error	0.00033711210444429
(Intercept)_zvalue	15.9882130516502
prot_zvalue	-7.06082309267581
cal_zvalue	8.57906181719737
sexMale_zvalue	0.575436478267434
hgt0_zvalue	4.04795181812859
wgt0_zvalue	2.52284441418383
(Intercept)_Pr(>|z|)	1.54396598126854e-57
prot_Pr(>|z|)	1.65519210848649e-12
cal_Pr(>|z|)	9.56500648203187e-18
sexMale_Pr(>|z|)	0.564996139463599
hgt0_Pr(>|z|)	5.16677787108928e-05
wgt0_Pr(>|z|)	0.0116409892837831
Weakinstruments(prot)_df1	4
Weakinstruments(cal)_df1	4
Wu-Hausman_df1	2
Sargan_df1	2
Weakinstruments(prot)_df2	14914
Weakinstruments(cal)_df2	14914
Wu-Hausman_df2	14914
Weakinstruments(prot)_statistic	274.147084958343
Weakinstruments(cal)_statistic	315.036848606231
Wu-Hausman_statistic	94.7020085425169
Sargan_statistic	122.081979628898
Weakinstruments(prot)_p-value	8.61731956233366e-228
Weakinstruments(cal)_p-value	1.18918641220866e-260
Wu-Hausman_p-value	1.35024050408262e-41
Sargan_p-value	3.09196773720398e-27
vars_var.y	hgt
vars_vars.x	prot+cal
vars_vars.z	momEdu+wealthIdx+p.A.prot+p.A.nProt
vars_vars.c	sex+hgt0+wgt0

5.1.1.2.5 Examples Line by Line The examples are just to test the code with different types of variables.

```
# Selecting Variables
var.y <- c('hgt')
vars.x <- c('prot', 'cal')
vars.z <- c('momEdu', 'wealthIdx', 'p.A.prot', 'p.A.nProt')
vars.c <- c('sex', 'hgt0', 'wgt0')
```

```
# A. create Equation
str.vars.x <- paste(vars.x, collapse='+')
str.vars.c <- paste(vars.c, collapse='+')
```

```
str.vars.z <- paste(vars.z, collapse='+')
print(str.vars.x)
```

```
## [1] "prot+cal"
```

```
print(str.vars.c)
```

```
## [1] "sex+hgt0+wgt0"
```

```
print(str.vars.z)
```

```
## [1] "momEdu+wealthIdx+p.A.prot+p.A.nProt"
```

```
equa.iv <- paste(var.y,
                 paste(paste(str.vars.x, str.vars.c, sep='+'),
                       paste(str.vars.z, str.vars.c, sep='+'),
                       sep='|'),
                 sep='~')
print(equa.iv)
```

```
## [1] "hgt~prot+cal+sex+hgt0+wgt0|momEdu+wealthIdx+p.A.prot+p.A.nProt+sex+hgt0+wgt0"
```

```
# B. regression
res.ivreg <- ivreg(as.formula(equa.iv), data=df)
coef(res.ivreg)
```

```
##   (Intercept)          prot           cal       sexMale          hgt0          wgt0
## 44.0243196254 -1.4025623247  0.0651048958  0.1208327876  0.2865254380  0.0008504814
```

```
# C. Regression Summary
ivreg.summ <- summary(res.ivreg, vcov = sandwich, df = Inf, diagnostics = TRUE)

ivreg.summ$coef
```

```
##                  Estimate   Std. Error    z value     Pr(>|z|)
## (Intercept) 44.0243196254 2.7535484724 15.9882131 1.543966e-57
## prot        -1.4025623247 0.1986400603 -7.0608231 1.655192e-12
## cal          0.0651048958 0.0075888130  8.5790618 9.565006e-18
## sexMale      0.1208327876 0.2099845806  0.5754365 5.649961e-01
## hgt0         0.2865254380 0.0707828183  4.0479518 5.166778e-05
## wgt0         0.0008504814 0.0003371121  2.5228444 1.164099e-02
## attr(,"df")
## [1] 0
## attr(,"nobs")
## [1] 14922
```

```
ivreg.summ$diagnostics
```

```
##                          df1   df2 statistic       p-value
## Weak instruments (prot)    4 14914 274.14708 8.617320e-228
## Weak instruments (cal)     4 14914 315.03685 1.189186e-260
## Wu-Hausman                 2 14914  94.70201  1.350241e-41
## Sargan                     2    NA 122.08198  3.091968e-27
```

```
# D. Combine Regression Results into a Matrix
df.results <- suppressMessages(as_tibble(ivreg.summ$coef, rownames='rownames') %>%
    full_join(as_tibble(ivreg.summ$diagnostics, rownames='rownames')) %>%
    full_join(tibble(rownames=c('vars'),
                     var.y=var.y,
                     vars.x=str.vars.x,
                     vars.z=str.vars.z,
                     vars.c=str.vars.c)))
# E. Flatten Matrix, All IV results as a single tibble row to be combined with other IV results
```

```
df.row.results <- df.results %>%
    gather(variable, value, -rownames) %>%
    drop_na() %>%
    unite(esti.val, rownames, variable) %>%
    mutate(esti.val = gsub(' ', '', esti.val))
```

```
# F. Results as Single Colum
# df.row.results
```

```
# G. Results as Single Row
# df.row.results
```

```
# t(df.row.results %>% spread(esti.val, value)) %>%
#   kable() %>%
#   kable_styling_fc_wide()
```

5.1.2 IV Loop over RHS

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Regression with a Variety of Outcome Variables and Right Hand Side Variables. There are M outcome variables, and there are N alternative right hand side variables. Regress each M outcome variable and each N alternative right hand side variable, with some common sets of controls and perhaps shared instruments. The output file is a M by N matrix of coefficients, with proper variable names and row names. The matrix stores coefficients for this key endogenous variable.

- Dependency: *R4Econ/linreg/ivreg/ivregdfrow.R*

5.1.2.1 Construct Program

The program relies on double lapply. lapply is used for convenience, not speed.

```
ff_reg_mbyn <- function(list.vars.y, list.vars.x,
                        vars.c, vars.z, df,
                        return_all = FALSE,
                        stats_ends = 'value', time = FALSE) {

  # regf.iv() function is from C:\Users\fan\R4Econ\linreg\ivreg\ivregdfrow.R
  if (time) {
    start_time <- Sys.time()
  }

  if (return_all) {
    df.reg.out.all <-
      bind_rows(lapply(list.vars.x,
                       function(x) (
                         bind_rows(
                           lapply(list.vars.y, regf.iv,
                                  vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df))
                       )))

  } else {
    df.reg.out.all <-
      (lapply(list.vars.x,
              function(x) (
                bind_rows(
                  lapply(list.vars.y, regf.iv,
                         vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df)) %>%
                  select(vars_var.y, starts_with(x)) %>%
```

```
                    select(vars_var.y, ends_with(stats_ends))
                ))) %>% reduce(full_join)
  }

  if (time) {
    end_time <- Sys.time()
    print(paste0('Estimation for all ys and xs took (seconds):',
                 end_time - start_time))
  }

  return(df.reg.out.all)
}
```

5.1.2.2 Prepare Data

```
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')

# Source Dependency
source('C:/Users/fan/R4Econ/linreg/ivreg/ivregdfrow.R')

# Setting
options(repr.matrix.max.rows=50, repr.matrix.max.cols=50)
```

Parameters.

```
var.y1 <- c('hgt')
var.y2 <- c('wgt')
var.y3 <- c('vil.id')
list.vars.y <- c(var.y1, var.y2, var.y3)

var.x1 <- c('prot')
var.x2 <- c('cal')
var.x3 <- c('wealthIdx')
var.x4 <- c('p.A.prot')
var.x5 <- c('p.A.nProt')
list.vars.x <- c(var.x1, var.x2, var.x3, var.x4, var.x5)

vars.z <- c('indi.id')
vars.c <- c('sex', 'wgt0', 'hgt0', 'svymthRound')
```

5.1.2.3 Program Testing

```
vars.z <- NULL
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'value'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_tvalue	cal_tvalue	wealthIdx_tvalue	p.A.prot_tvalue	p.A.nProt_tvalue
hgt	18.8756010031786	23.4421863484661	13.508899618216	3.83682180045518	32.5448257554855
wgt	16.3591125056062	17.3686031309332	14.1390521528113	1.36958319982295	12.0961557911467
vil.id	-14.9385580468907	-19.6150110809452	34.0972558327347	8.45943342783186	17.7801422421419

5.1.2.3.1 Test Program OLS Z-Stat

```
vars.z <- c('indi.id')
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'value'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_zvalue	cal_zvalue	wealthIdx_zvalue	p.A.prot_zvalue	p.A.nProt_zvalue
hgt	8.87674929300964	12.0739764947235	4.62589553677969	26.6373587567312	32.1162192385744
wgt	5.60385871756365	6.1225187008946	5.17869536991717	11.9295584469998	12.3509307017263
vil.id	-9.22106223347162	-13.0586007975839	-51.5866689219593	-29.9627476577329	-38.3528894620707

5.1.2.3.2 Test Program IV T-stat

```
vars.z <- NULL
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'Estimate'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_Estimate	cal_Estimate	wealthIdx_Estimate	p.A.prot_Estimate	p.A.nProt_Estimate
hgt	0.049431093806755	0.00243408846205622	0.21045655488185	3.86952250259526e-05	0.00542428867316449
wgt	16.5557424523585	0.699072500364623	106.678721085969	0.00521731297924587	0.779514232050632
vil.id	-0.0758835879205584	-0.00395676177098486	0.451733304543324	0.000149388430455142	0.00526237555581024

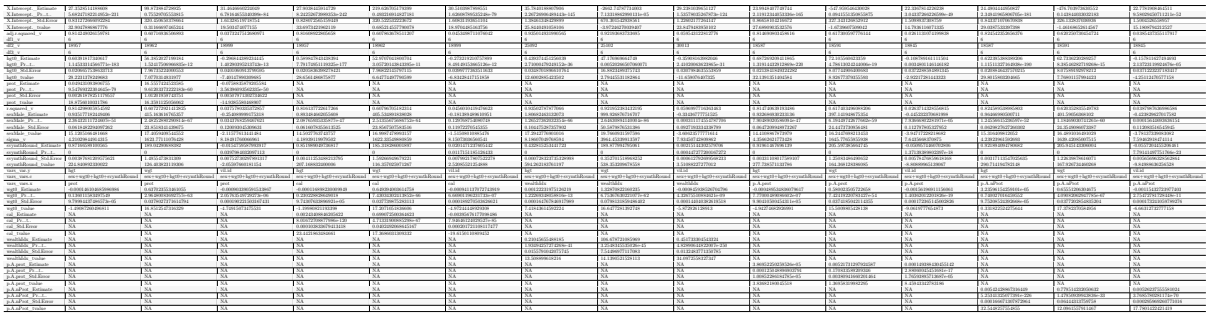
5.1.2.3.3 Test Program OLS Coefficient

```
vars.z <- c('indi.id')
suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = FALSE,
              stats_ends = 'Estimate'))) %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_Estimate	cal_Estimate	wealthIdx_Estimate	p.A.prot_Estimate	p.A.nProt_Estimate
hgt	0.859205733632614	0.0238724384575419	0.144503490136948	0.00148073028434642	0.0141317656200726
wgt	98.9428234201406	2.71948246216953	69.1816142883022	0.221916473012486	2.11856940494335
vil.id	-6.02451379136132	-0.168054407187466	-1.91414470908345	-0.00520794333267238	-0.0494468877742109

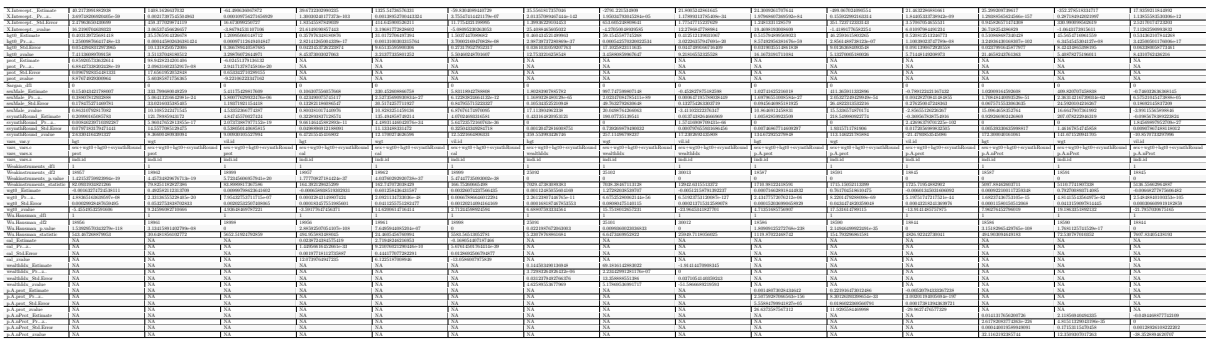
5.1.2.3.4 Test Program IV coefficient

```
vars.z <- NULL
t(suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = TRUE,
              stats_ends = 'Estimate')))) %>%
  kable() %>%
  kable_styling_fc_wide()
```



5.1.2.3.5 Test Program OLS Return All

```
vars.z <- c('indi.id')
t(suppressWarnings(suppressMessages(
  ff_reg_mbyn(list.vars.y, list.vars.x,
              vars.c, vars.z, df,
              return_all = TRUE,
              stats_ends = 'Estimate')))) %>%
  kable() %>%
  kable_styling_fc_wide()
```



5.1.2.3.6 Test Program IV Return All

5.1.2.4 Program Line by Line

Set Up Parameters

```
vars.z <- c('indi.id')
vars.z <- NULL
vars.c <- c('sex', 'wgt0', 'hgt0', 'svymthRound')
```

```
df.reg.out <- as_tibble(
  bind_rows(lapply(list.vars.y, regf.iv,
                   vars.x=var.x1, vars.c=vars.c, vars.z=vars.z, df=df)))
```

5.1.2.4.1 Lapply

```
lapply(list.vars.y, function(y) (mean(df[[var.x1]], na.rm=TRUE) +
                                   mean(df[[y]], na.rm=TRUE)))
```

5.1.2.4.2 Nested Lapply Test

```
## [[1]]
## [1] 98.3272
##
## [[2]]
## [1] 13626.51
##
## [[3]]
## [1] 26.11226
```

```
lapplytwice <- lapply(
  list.vars.x, function(x) (
    lapply(list.vars.y, function(y) (mean(df[[x]], na.rm=TRUE) +
                                       mean(df[[y]], na.rm=TRUE)))))
# lapplytwice
```

```
df.reg.out.all <- bind_rows(
  lapply(list.vars.x,
         function(x) (
           bind_rows(
             lapply(list.vars.y, regf.iv,
                    vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df))
         )))
```

```
# df.reg.out.all %>%
#   kable() %>%
#   kable_styling_fc_wide()
```

5.1.2.4.3 Nested Lapply All

```
df.reg.out.all <-
  (lapply(list.vars.x,
          function(x) (
            bind_rows(lapply(list.vars.y, regf.iv,
                             vars.x=x, vars.c=vars.c, vars.z=vars.z, df=df)) %>%
              select(vars_var.y, starts_with(x)) %>%
              select(vars_var.y, ends_with('value'))
          ))) %>% reduce(full_join)
```

```
df.reg.out.all %>%
  kable() %>%
  kable_styling_fc_wide()
```

vars_var.y	prot_tvalue	cal_tvalue	wealthIdx_tvalue	p.A.prot_tvalue	p.A.nProt_tvalue
hgt	18.8756010031786	23.4421863484661	13.508899618216	3.83682180045518	32.5448257554855
wgt	16.3591125056062	17.3686031309332	14.1390521528113	1.36958319982295	12.0961557911467
vil.id	-14.9385580468907	-19.6150110809452	34.0972558327347	8.45943342783186	17.7801422421419

5.1.2.4.4 Nested Lapply Select

5.2 Decomposition

5.2.1 Decompose RHS

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

One runs a number of regressions. With different outcomes, and various right hand side variables.

What is the remaining variation in the left hand side variable if right hand side variable one by one is set to the average of the observed values.

- Dependency: *R4Econ/linreg/ivreg/ivregdfrow.R*

The code below does not work with categorical variables (except for dummies). Dummy variable inputs need to be converted to zero/one first. The examples are just to test the code with different types of variables.

```
# Library
library(tidyverse)
library(AER)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')

# Source Dependency
source('C:/Users/fan/R4Econ/linreg/ivreg/ivregdfrow.R')
```

Data Cleaning.

```
# Convert Variable for Sex which is categorical to Numeric
df <- df
df$male <- (as.numeric(factor(df$sex)) - 1)
summary(factor(df$sex))
```

```
## Female   Male
##  16446  18619
```

```
summary(df$male)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   0.531   1.000   1.000
```

```
df.use <- df %>% filter(S.country == 'Guatemala') %>%
  filter(svymthRound %in% c(12, 18, 24))
dim(df.use)
```

```
## [1] 2022   16
```

Setting Up Parameters.

```
# Define Left Hand Side Variab les
var.y1 <- c('hgt')
var.y2 <- c('wgt')
vars.y <- c(var.y1, var.y2)
# Define Right Hand Side Variables
vars.x <- c('prot')
vars.c <- c('male', 'wgt0', 'hgt0', 'svymthRound')
# vars.z <- c('p.A.prot')
vars.z <- c('vil.id')
# vars.z <- NULL
vars.xc <- c(vars.x, vars.c)

# Other variables to keep
```

```
vars.other.keep <- c('S.country', 'vil.id', 'indi.id', 'svymthRound')

# Decompose sequence
vars.tomean.first <- c('male', 'hgt0')
var.tomean.first.name.suffix <- '_mh02m'
vars.tomean.second <- c(vars.tomean.first, 'hgt0', 'wgt0')
var.tomean.second.name.suffix <- '_mh0me2m'
vars.tomean.third <- c(vars.tomean.second, 'prot')
var.tomean.third.name.suffix <- '_mh0mep2m'
vars.tomean.fourth <- c(vars.tomean.third, 'svymthRound')
var.tomean.fourth.name.suffix <- '_mh0mepm2m'
list.vars.tomean = list(
#                          vars.tomean.first,
                        vars.tomean.second,
                        vars.tomean.third,
                        vars.tomean.fourth
                        )
list.vars.tomean.name.suffix <- list(
#                                     var.tomean.first.name.suffix,
                                    var.tomean.second.name.suffix,
                                    var.tomean.third.name.suffix,
                                    var.tomean.fourth.name.suffix
                                   )
```

5.2.1.1 Obtain Regression Coefficients from somewhere

```
# Regressions
# regf.iv from C:\Users\fan\R4Econ\linreg\ivreg\ivregdfrow.R
df.reg.out <- as_tibble(
  bind_rows(lapply(vars.y, regf.iv,
                   vars.x=vars.x, vars.c=vars.c, vars.z=vars.z, df=df)))
# Regressions
# reg1 <- regf.iv(var.y = var.y1, vars.x, vars.c, vars.z, df.use)
# reg2 <- regf.iv(var.y = var.y2, vars.x, vars.c, vars.z, df.use)
# df.reg.out <- as_tibble(bind_rows(reg1, reg2))
```

```
# df.reg.out
```

```
# Select Variables
str.esti.suffix <- '_Estimate'
arr.esti.name <- paste0(vars.xc, str.esti.suffix)
str.outcome.name <- 'vars_var.y'
arr.columns2select <- c(arr.esti.name, str.outcome.name)
arr.columns2select
```

```
## [1] "prot_Estimate"        "male_Estimate"        "wgt0_Estimate"        "hgt0_Estimate"
## [5] "svymthRound_Estimate" "vars_var.y"
```

```
# Generate dataframe for coefficients
df.coef <- df.reg.out[,c(arr.columns2select)] %>%
  mutate_at(vars(arr.esti.name), as.numeric) %>% column_to_rownames(str.outcome.name)
df.coef %>%
  kable() %>%
  kable_styling_fc()
```

	prot_Estimate	male_Estimate	wgt0_Estimate	hgt0_Estimate	svymthRound_Estimate
hgt	-0.2714772	1.244735	0.0004430	0.6834853	1.133919
wgt	-59.0727542	489.852902	0.7696158	75.4867897	250.778883

```
str(df.coef)
```

```
## 'data.frame':    2 obs. of  5 variables:
##  $ prot_Estimate       : num  -0.271 -59.073
##  $ male_Estimate       : num  1.24 489.85
##  $ wgt0_Estimate       : num  0.000443 0.769616
##  $ hgt0_Estimate       : num  0.683 75.487
##  $ svymthRound_Estimate: num  1.13 250.78
```

5.2.1.2 Decomposition Step 1

```
# Decomposition Step 1: gather
df.decompose_step1 <- df.use %>%
                        filter(svymthRound %in% c(12, 18, 24)) %>%
                        select(one_of(c(vars.other.keep, vars.xc, vars.y))) %>%
                        drop_na() %>%
                        gather(variable, value, -one_of(c(vars.other.keep, vars.xc)))
options(repr.matrix.max.rows=20, repr.matrix.max.cols=20)
dim(df.decompose_step1)
```

```
## [1] 1382   10
```

```
head(df.decompose_step1, 10) %>%
  kable() %>%
  kable_styling_fc()
```

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8

5.2.1.3 Decomposition Step 2

```
# Decomposition Step 2: mutate_at(vars, funs(mean = mean(.)))
# the xc averaging could have taken place earlier, no difference in mean across variables
df.decompose_step2 <- df.decompose_step1 %>%
                        group_by(variable) %>%
                        mutate_at(vars(c(vars.xc, 'value')), funs(mean = mean(.))) %>%
                        ungroup()

options(repr.matrix.max.rows=20, repr.matrix.max.cols=20)
dim(df.decompose_step2)
```

```
## [1] 1382   16
```

```
head(df.decompose_step2,10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value	prot_mean	male_mean	wgt0_mean	hgt0_mean	svymthRound_mean	value_mean
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216

5.2.1.4 Decomposition Step 3 Non-Loop

```
ff_lr_decompose_valadj <- function(df, df.coef, vars.tomean, str.esti.suffix) {
  new_value <- (df$value +
                  rowSums((df[paste0(vars.tomean, '_mean')] - df[vars.tomean])
                          *df.coef[df$variable, paste0(vars.tomean, str.esti.suffix)]))
  return(new_value)
}
```

5.2.1.5 Decomposition Step 3 With Loop

```
df.decompose_step3 <- df.decompose_step2
for (i in 1:length(list.vars.tomean)) {
    var.decomp.cur <- (paste0('value', list.vars.tomean.name.suffix[[i]]))
    vars.tomean <- list.vars.tomean[[i]]
    var.decomp.cur
    df.decompose_step3 <- df.decompose_step3 %>%
      mutate((!!var.decomp.cur) :=
               ff_lr_decompose_valadj(., df.coef, vars.tomean, str.esti.suffix))

}

dim(df.decompose_step3)
```

```
## [1] 1382   19
```

```
head(df.decompose_step3, 10) %>%
  kable() %>%
  kable_styling_fc_wide()
```

S.country	vil.id	indi.id	svymthRound	prot	male	wgt0	hgt0	variable	value	prot_mean	male_mean	wgt0_mean	hgt0_mean	svymthRound_mean	value_mean	value_mh0me2m	value_mh0mep2m	value_mh0mepm2m
Guatemala	3	1352	18	13.3	1	2545.2	47.4	hgt	70.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.19390	71.19903	71.68148
Guatemala	3	1352	24	46.3	1	2545.2	47.4	hgt	75.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	78.79390	85.75778	79.43671
Guatemala	3	1354	12	1.0	1	3634.3	51.2	hgt	66.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	63.61689	58.28285	65.56882
Guatemala	3	1354	18	9.8	1	3634.3	51.2	hgt	69.2	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	66.51689	63.57185	64.05430
Guatemala	3	1354	24	15.4	1	3634.3	51.2	hgt	75.3	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	72.61689	71.19213	64.87106
Guatemala	3	1356	12	8.6	1	3911.8	51.9	hgt	68.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	64.33707	61.06626	68.35222
Guatemala	3	1356	18	17.8	1	3911.8	51.9	hgt	74.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	70.33707	69.56385	70.04630
Guatemala	3	1356	24	30.5	1	3911.8	51.9	hgt	77.1	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.33707	76.01161	69.69055
Guatemala	3	1357	12	1.0	1	3791.4	52.6	hgt	71.5	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	66.83353	61.49949	68.78545
Guatemala	3	1357	18	12.7	1	3791.4	52.6	hgt	77.8	20.64819	0.5499276	3312.297	49.75137	18.42547	73.41216	73.13353	70.97578	71.45823

5.2.1.6 Decomposition Step 4 Variance

```
df.decompose_step3 %>%
        select(variable, contains('value')) %>%
        group_by(variable) %>%
        summarize_all(funs(mean = mean, var = var)) %>%
        select(matches('value')) %>% select(ends_with("_var")) %>%
        mutate_if(is.numeric, funs( frac = (./value_var))) %>%
        mutate_if(is.numeric, round, 3) %>%
  kable() %>%
  kable_styling_fc_wide()
```

value_var	value_mean_var	value_mh0me2m_var	value_mh0mep2m_var	value_mh0mepm2m_var	value_var_frac	value_mean_var_frac	value_mh0me2m_var_frac	value_mh0mep2m_var_frac	value_mh0mepm2m_var_frac
21.864	NA	25.35	49.047	23.06	1	NA	1.159	2.243	1.055
2965693.245	NA	2949187.64	4192769.518	3147506.60	1	NA	0.994	1.414	1.061

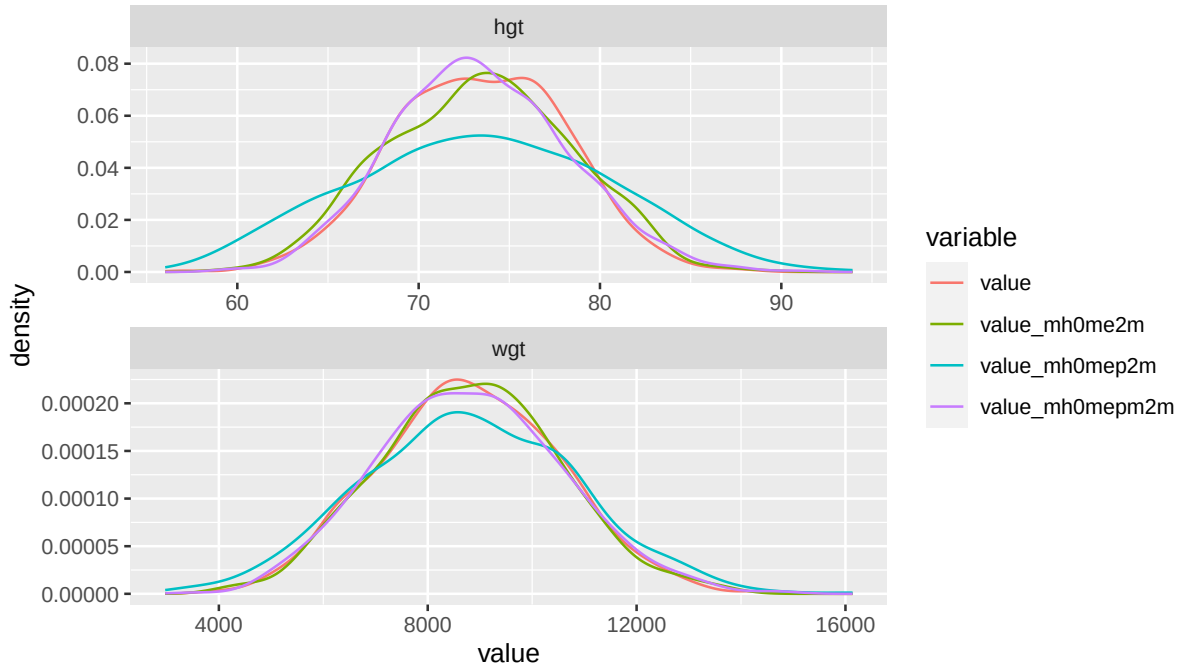
5.2.1.7 Graphical Results

Graphically, difficult to pick up exact differences in variance, a 50 percent reduction in variance visually does not look like 50 percent. Intuitively, we are kind of seeing standard deviation, not variance on the graph if we think abou the x-scale.

```
head(df.decompose_step3 %>%
    select(variable, contains('value'), -value_mean), 10) %>%
  kable() %>%
  kable_styling_fc()
```

variable	value	value_mh0me2m	value_mh0mep2m	value_mh0mepm2m
hgt	70.2	73.19390	71.19903	71.68148
hgt	75.8	78.79390	85.75778	79.43671
hgt	66.3	63.61689	58.28285	65.56882
hgt	69.2	66.51689	63.57185	64.05430
hgt	75.3	72.61689	71.19213	64.87106
hgt	68.1	64.33707	61.06626	68.35222
hgt	74.1	70.33707	69.56385	70.04630
hgt	77.1	73.33707	76.01161	69.69055
hgt	71.5	66.83353	61.49949	68.78545
hgt	77.8	73.13353	70.97578	71.45823

```
df.decompose_step3 %>%
    select(variable, contains('value'), -value_mean) %>%
    rename(outcome = variable) %>%
    gather(variable, value, -outcome) %>%
    ggplot(aes(x=value, color = variable, fill = variable)) +
        geom_line(stat = "density") +
        facet_wrap(~ outcome, scales='free', nrow=2)
```



5.2.1.8 Additional Decomposition Testings

```
head(df.decompose_step2[vars.tomean.first],3)
head(df.decompose_step2[paste0(vars.tomean.first, '_mean')], 3)
```

```
head(df.coef[df.decompose_step2$variable,
             paste0(vars.tomean.first, str.esti.suffix)], 3)
df.decompose.tomean.first <- df.decompose_step2 %>%
    mutate(pred_new = df.decompose_step2$value +
        rowSums((df.decompose_step2[paste0(vars.tomean.first, '_mean')]
                 - df.decompose_step2[vars.tomean.first])
            *df.coef[df.decompose_step2$variable,
                     paste0(vars.tomean.first, str.esti.suffix)])) %>%
        select(variable, value, pred_new)
head(df.decompose.tomean.first, 10)
df.decompose.tomean.first %>%
        group_by(variable) %>%
        summarize_all(funs(mean = mean, sd = sd))  %>%
  kable() %>%
  kable_styling_fc()
```

variable	value_mean	pred_new_mean	value_sd	pred_new_sd
hgt	73.41216	73.41216	4.675867	4.534947
wgt	8807.87656	8807.87656	1722.118824	1695.221845

Note the r-square from regression above matches up with the 1 - ratio below. This is the proper decomposition method that is equivalent to r2.

```
df.decompose_step2 %>%
    mutate(pred_new = df.decompose_step2$value +
        rowSums((df.decompose_step2[paste0(vars.tomean.second, '_mean')]
                 - df.decompose_step2[vars.tomean.second])
            *df.coef[df.decompose_step2$variable,
                     paste0(vars.tomean.second, str.esti.suffix)])) %>%
        select(variable, value, pred_new) %>%
        group_by(variable) %>%
        summarize_all(funs(mean = mean, var = var)) %>%
        mutate(ratio = (pred_new_var/value_var))  %>%
  kable() %>%
  kable_styling_fc()
```

variable	value_mean	pred_new_mean	value_var	pred_new_var	ratio
hgt	73.41216	73.41216	2.186374e+01	25.3504	1.1594724
wgt	8807.87656	8807.87656	2.965693e+06	2949187.6357	0.9944345

Chapter 6

Nonlinear and Other Regressions

6.1 Logit Regression

6.1.1 Binary Logit

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Data Preparation

```
df_mtcars <- mtcars

# X-variables to use on RHS
ls_st_xs <- c('mpg', 'qsec')
ls_st_xs <- c('mpg')
ls_st_xs <- c('qsec')
ls_st_xs <- c('wt')
ls_st_xs <- c('mpg', 'wt', 'vs')

svr_binary <- 'hpLowHigh'
svr_binary_lb0 <- 'LowHP'
svr_binary_lb1 <- 'HighHP'
svr_outcome <- 'am'
sdt_name <- 'mtcars'

# Discretize hp
df_mtcars <- df_mtcars %>%
    mutate(!!sym(svr_binary) := cut(hp,
                            breaks=c(-Inf, 210, Inf),
                            labels=c(svr_binary_lb0, svr_binary_lb1)))
```

6.1.1.1 Logit Regresion and Prediction

logit regression with glm, and predict using estimation data. Prediction and estimation with one variable.

- LOGIT REGRESSION R DATA ANALYSIS EXAMPLES
- Generalized Linear Models

```
# Regress
rs_logit <- glm(as.formula(paste(svr_outcome, "~", paste(ls_st_xs, collapse="+")))
                ,data = df_mtcars, family = "binomial")
summary(rs_logit)
```

```
##
## Call:
```

```
## glm(formula = as.formula(paste(svr_outcome, "~", paste(ls_st_xs,
##     collapse = "+"))), family = "binomial", data = df_mtcars)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73603  -0.25477  -0.04891   0.13402   1.90321
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 22.69008   13.95112   1.626   0.1039
## mpg         -0.01786    0.33957  -0.053   0.9581
## wt          -6.73804    3.01400  -2.236   0.0254 *
## vs          -4.44046    2.84247  -1.562   0.1182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.230  on 31  degrees of freedom
## Residual deviance: 13.092  on 28  degrees of freedom
## AIC: 21.092
##
## Number of Fisher Scoring iterations: 7
```

```
# Predcit Using Regression Data
df_mtcars$p_mpg <- predict(rs_logit, newdata = df_mtcars, type = "response")
```

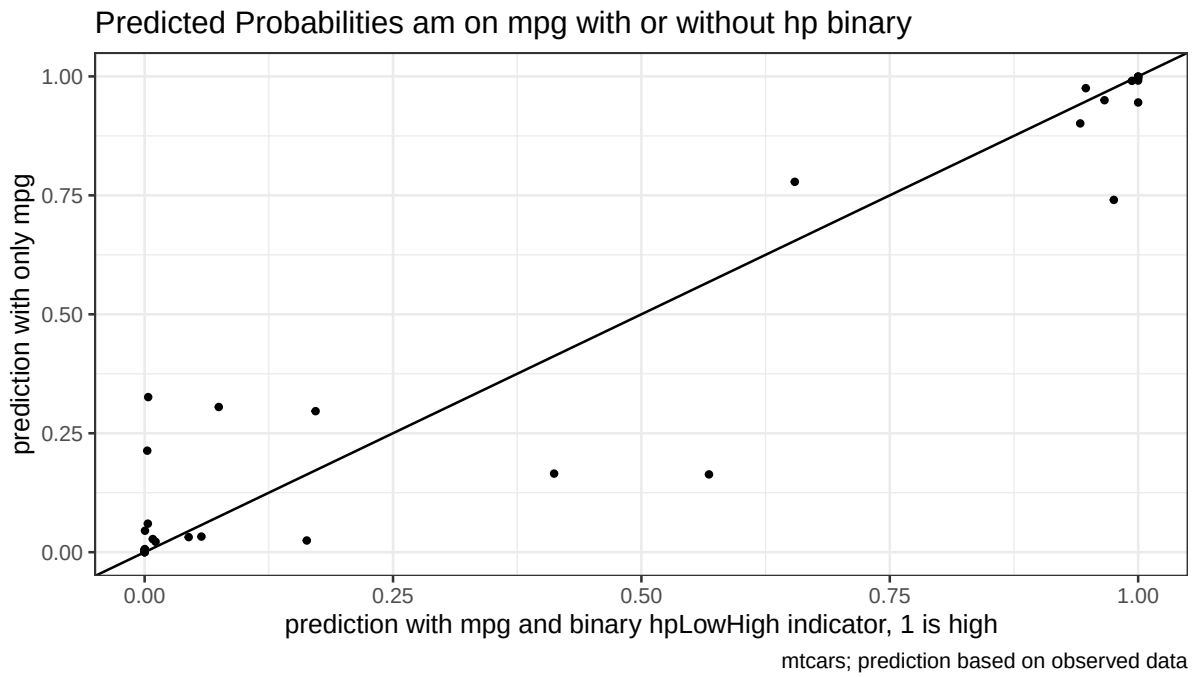
6.1.1.1.1 Prediction with Observed Binary Input Logit regression with a continuous variable and a binary variable. Predict outcome with observed continuous variable as well as observed binary input variable.

```
# Regress
rs_logit_bi <- glm(as.formula(paste(svr_outcome,
                                    "~ factor(", svr_binary,") + ",
                                    paste(ls_st_xs, collapse="+")))
                   , data = df_mtcars, family = "binomial")
summary(rs_logit_bi)
```

```
##
## Call:
## glm(formula = as.formula(paste(svr_outcome, "~ factor(", svr_binary,
##     ") + ", paste(ls_st_xs, collapse = "+"))), family = "binomial",
##     data = df_mtcars)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.45771  -0.09563  -0.00875   0.00555   1.87612
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 3.8285    18.0390   0.212   0.8319
## factor(hpLowHigh)HighHP     6.9907     5.5176   1.267   0.2052
## mpg                         0.8985     0.8906   1.009   0.3131
## wt                         -6.7291     3.3166  -2.029   0.0425 *
## vs                         -5.9206     4.1908  -1.413   0.1577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 43.2297  on 31  degrees of freedom
## Residual deviance:  8.9777  on 27  degrees of freedom
## AIC: 18.978
##
## Number of Fisher Scoring iterations: 9
```

```
# Predcit Using Regresion Data
df_mtcars$p_mpg_hp <- predict(rs_logit_bi, newdata = df_mtcars, type = "response")

# Predicted Probabilities am on mgp with or without hp binary
scatter <- ggplot(df_mtcars, aes(x=p_mpg_hp, y=p_mpg)) +
      geom_point(size=1) +
      # geom_smooth(method=lm) + # Trend line
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Predicted Probabilities ', svr_outcome, ' on ', ls_st_xs, ' with or witho
           x = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' indicator, 1 is hi
           y = paste0('prediction with only ', ls_st_xs),
           caption = 'mtcars; prediction based on observed data') +
      theme_bw()
print(scatter)
```



6.1.1.1.2 Prediction with Binary set to 0 and 1 Now generate two predictions. One set where binary input is equal to 0, and another where the binary inputs are equal to 1. Ignore whether in data binary input is equal to 0 or 1. Use the same regression results as what was just derived.

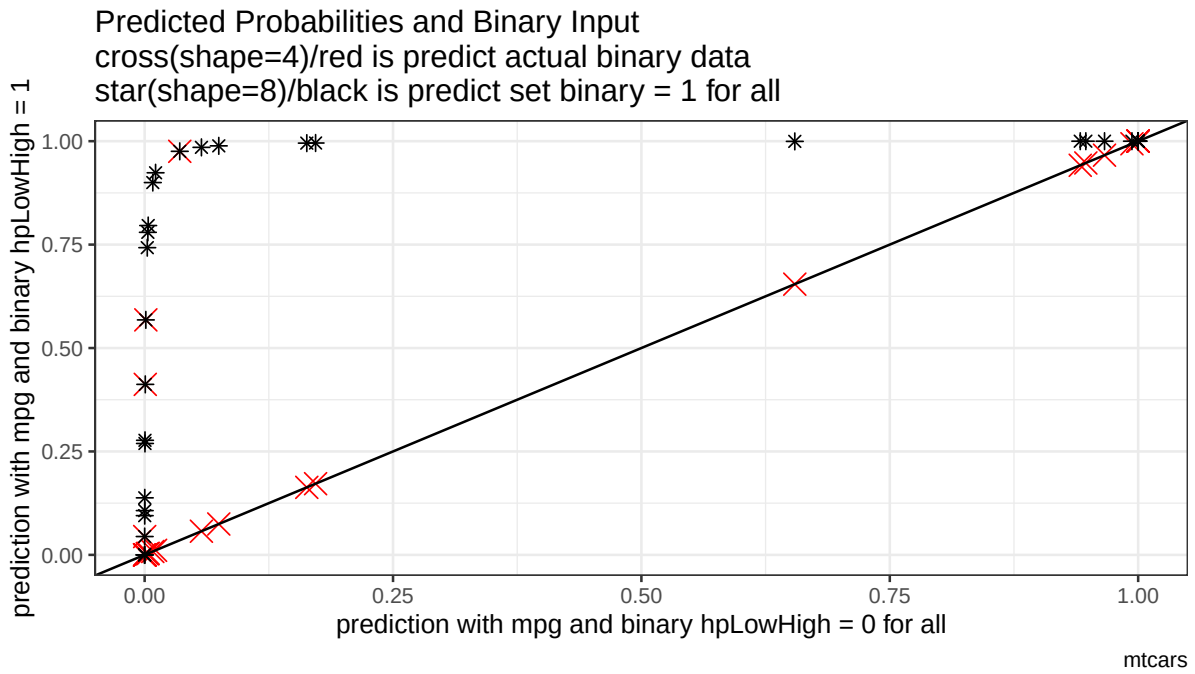
Note that given the example here, the probability changes a lot when we

```
# Previous regression results
summary(rs_logit_bi)
```

```
##
## Call:
## glm(formula = as.formula(paste(svr_outcome, "~ factor(", svr_binary,
##     ") + ", paste(ls_st_xs, collapse = "+"))), family = "binomial",
##     data = df_mtcars)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.45771  -0.09563  -0.00875   0.00555   1.87612
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               3.8285    18.0390   0.212   0.8319
## factor(hpLowHigh)HighHP   6.9907     5.5176   1.267   0.2052
## mpg                       0.8985     0.8906   1.009   0.3131
## wt                       -6.7291     3.3166  -2.029   0.0425 *
## vs                       -5.9206     4.1908  -1.413   0.1577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.2297  on 31  degrees of freedom
## Residual deviance:  8.9777  on 27  degrees of freedom
## AIC: 18.978
##
## Number of Fisher Scoring iterations: 9
```

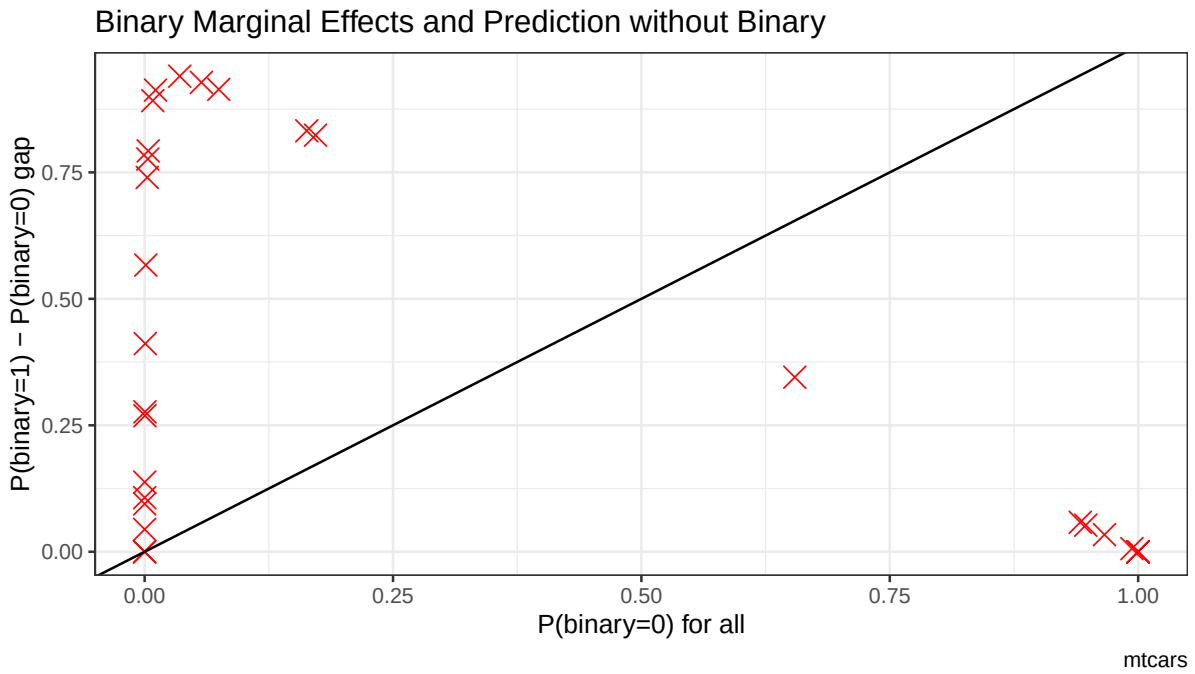
```
# Two different dataframes, mutate the binary regressor
df_mtcars_bi0 <- df_mtcars %>% mutate(!!sym(svr_binary) := svr_binary_lb0)
df_mtcars_bi1 <- df_mtcars %>% mutate(!!sym(svr_binary) := svr_binary_lb1)

# Predcit Using Regresion Data
df_mtcars$p_mpg_hp_bi0 <- predict(rs_logit_bi, newdata = df_mtcars_bi0, type = "response")
df_mtcars$p_mpg_hp_bi1 <- predict(rs_logit_bi, newdata = df_mtcars_bi1, type = "response")

# Predicted Probabilities and Binary Input
scatter <- ggplot(df_mtcars, aes(x=p_mpg_hp_bi0)) +
      geom_point(aes(y=p_mpg_hp), size=4, shape=4, color="red") +
      geom_point(aes(y=p_mpg_hp_bi1), size=2, shape=8) +
      # geom_smooth(method=lm) + # Trend line
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Predicted Probabilities and Binary Input',
                          '\ncross(shape=4)/red is predict actual binary data',
                          '\nstar(shape=8)/black is predict set binary = 1 for all'),
           x = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' = 0 for all'),
           y = paste0('prediction with ', ls_st_xs, ' and binary ', svr_binary, ' = 1'),
          caption = paste0(sdt_name)) +
      theme_bw()
print(scatter)
```



6.1.1.1.3 Prediction with Binary set to 0 and 1 Difference What is the difference in probability between binary = 0 vs binary = 1. How does that relate to the probability of outcome of interest when binary = 0 for all.

In the binary logit case, the relationship will be hump–shaped by construction between A_i and α_i. In the exponential wage cases, the relationship is convex upwards.

```
# Generate Gap Variable
df_mtcars <- df_mtcars %>% mutate(alpha_i = p_mpg_hp_bi1 - p_mpg_hp_bi0) %>%
                mutate(A_i = p_mpg_hp_bi0)

# Binary Marginal Effects and Prediction without Binary
scatter <- ggplot(df_mtcars, aes(x=A_i)) +
      geom_point(aes(y=alpha_i), size=4, shape=4, color="red") +
      geom_abline(intercept = 0, slope = 1) + # 45 degree line
      labs(title = paste0('Binary Marginal Effects and Prediction without Binary'),
           x = 'P(binary=0) for all',
           y = 'P(binary=1) - P(binary=0) gap',
           caption = paste0(sdt_name)) +
      theme_bw()
print(scatter)
```

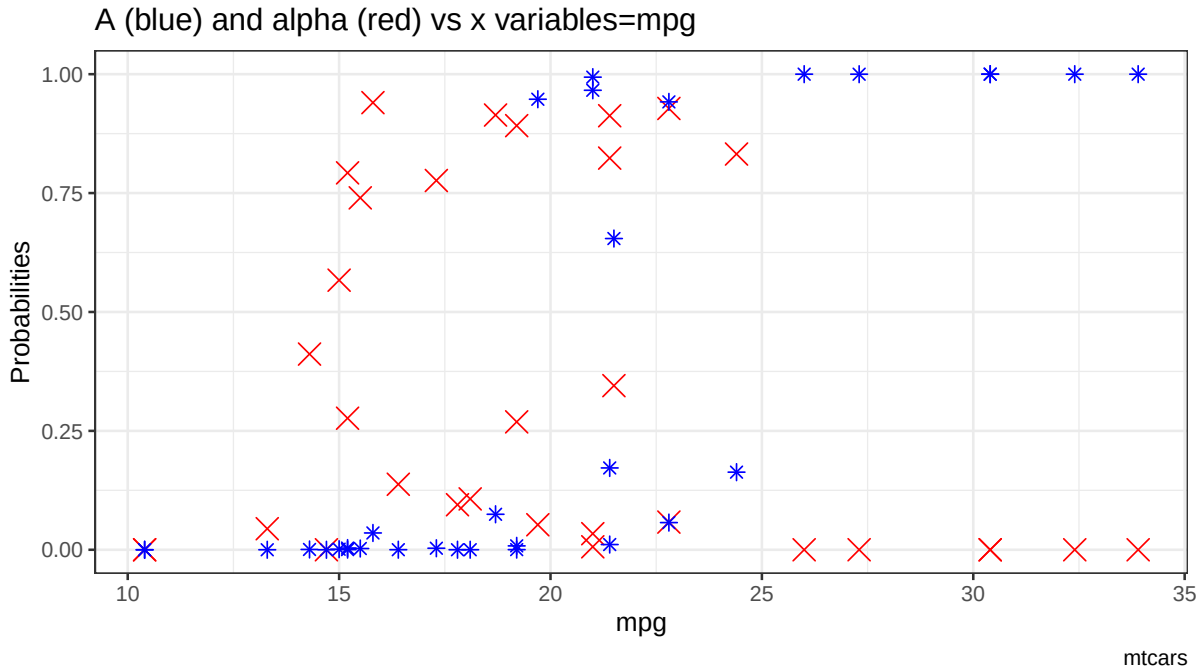


6.1.1.1.4 X variables and A and alpha

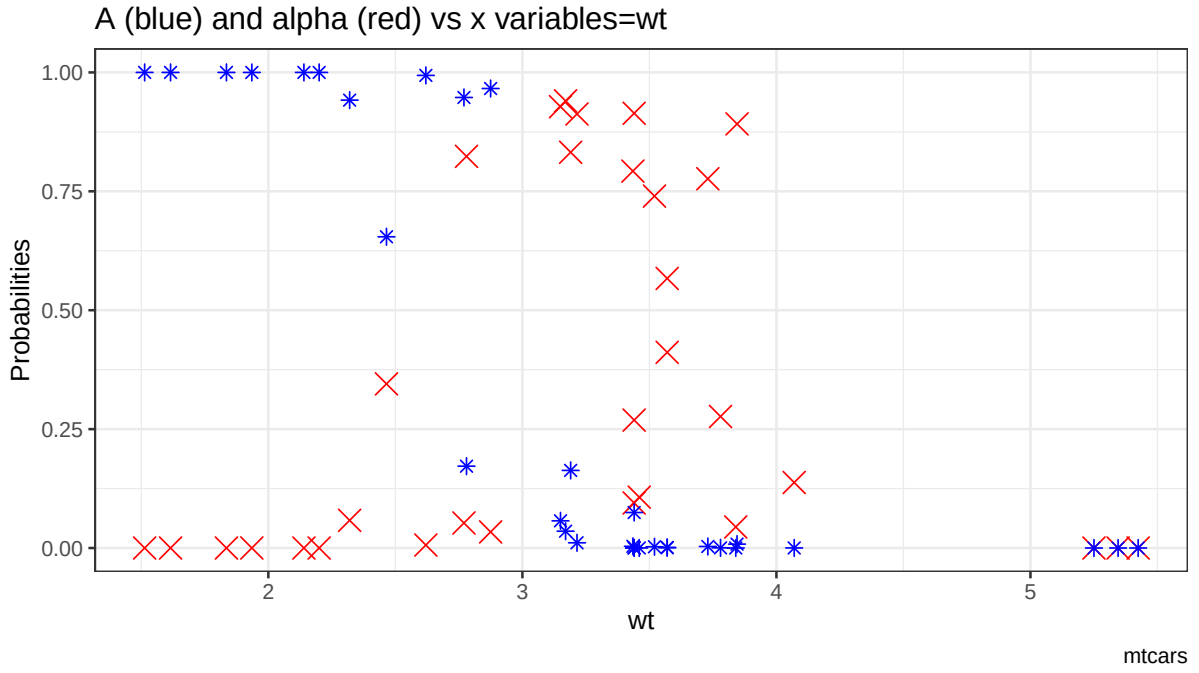
Given the x-variables included in the logit regression, how do they relate to A_i and alpha_i

```
# Generate Gap Variable
df_mtcars <- df_mtcars %>% mutate(alpha_i = p_mpg_hp_bi1 - p_mpg_hp_bi0) %>%
                mutate(A_i = p_mpg_hp_bi0)

# Binary Marginal Effects and Prediction without Binary
ggplot.A.alpha.x <- function(svr_x, df,
                             svr_alpha = 'alpha_i', svr_A = "A_i"){

  scatter <- ggplot(df, aes(x=!!sym(svr_x))) +
        geom_point(aes(y=alpha_i), size=4, shape=4, color="red") +
        geom_point(aes(y=A_i), size=2, shape=8, color="blue") +
        geom_abline(intercept = 0, slope = 1) + # 45 degree line
        labs(title = paste0('A (blue) and alpha (red) vs x variables=', svr_x),
             x = svr_x,
             y = 'Probabilities',
             caption = paste0(sdt_name)) +
        theme_bw()

return(scatter)
}

# Plot over multiple
lapply(ls_st_xs,
       ggplot.A.alpha.x,
       df = df_mtcars)
```

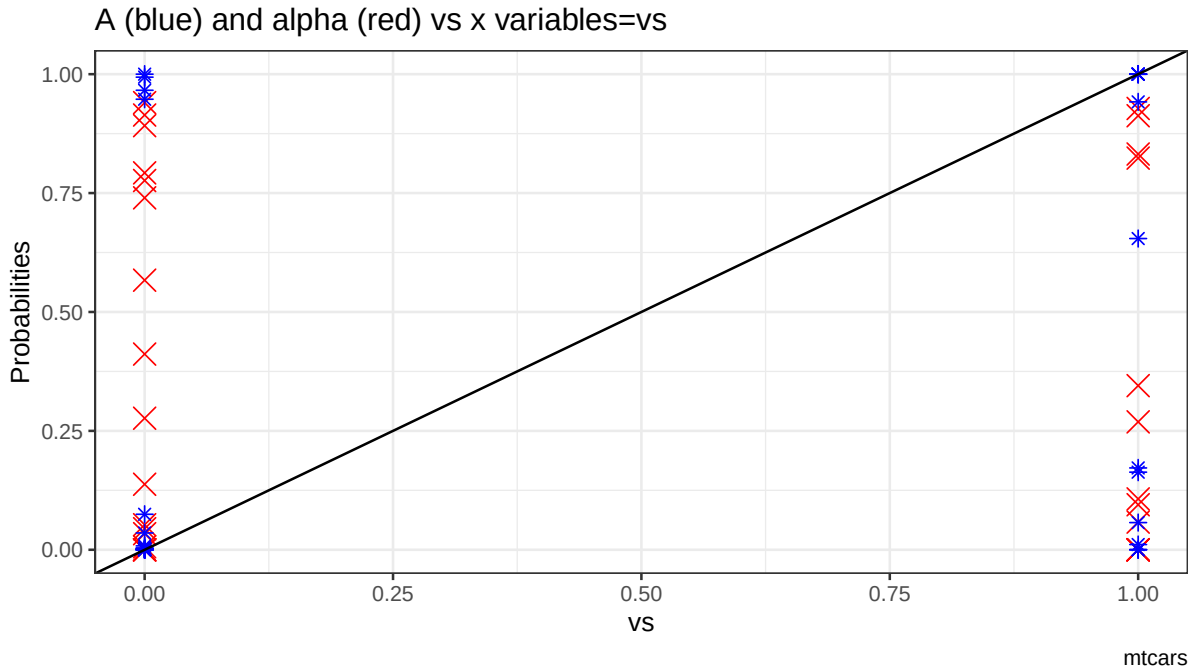
```
## [[1]]
```


A (blue) and alpha (red) vs x variables=mpg
Probabilities
1.00
0.75
0.50
0.25
0.00
10
15
20
25
30
35
mpg
mtcars

```
##
## [[2]]
```


A (blue) and alpha (red) vs x variables=wt
Probabilities
1.00
0.75
0.50
0.25
0.00
2
3
4
5
wt
mtcars

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



6.1.2 Logistic Choice Model with Aggregate Shares

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

6.1.2.1 Logistic Choices, Wages and Market Shares

6.1.2.1.1 Wage and Aggregate Share of Workers

Note: See Fit Prices Given Quantities Logistic Choice with Aggregate Data for solving/estimating for wages given aggregate shares.

Individual i can choose among $M+1$ options, the M options, are for example, alternative job possibilities. The $+1$ refers to a leisure category. The value associated with each one of the choice alternatives is:

$$V_{itm} = \widehat{V}_{tm} + \epsilon_{itm} = \alpha_m + \beta \cdot \text{WAGE}_{tm} + \epsilon_{itm}$$

Note that β captures the effect of occupation-specific wage, β is the same across occupational groups. Each occupational group has its own intercept α_m, and ϵ_{im} is the individual and occupation specific extreme value error. The non-error component of the value of leisure is normalized to 0 ($\exp(0) = 1$).

The discrete choice problem is solved by a comparison across alternatives. o_i is the individual optimal choice $o_i = \arg\max_m \left(V_{i,m}\right)$

Choice probabilities are functions of wages. The probability that individual i chooses occupation m is (ignoring the t-subscripts):

$$P(o = m) = \frac{\exp\left(\widehat{V}_m\right)}{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)}$$

The log ratio of probability of choosing any of the M occupation alternatives and leisure is:

$$\log P\left(o = m\right) - \log P\left(o = 0\right) = \log\left(\frac{P\left(o = m\right)}{P\left(o = 0\right)}\right) = \log\left(\frac{\exp\left(\widehat{V}_m\right)}{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)} \cdot \frac{1 + \sum_{\hat{m}=1}^{M} \exp\left(\widehat{V}_{\hat{m}}\right)}{1}\right)$$

This means that the log of the probability ratio is linear in wages:

$$\log\left(\frac{P\left(o=m\right)}{P\left(o=0\right)}\right)=\alpha_m+\beta\cdot\text{WAGE}_m$$

Supose we have $M-1$, either work or leisure. Suppose we have aggregate data from 1 time period, with relative work to leisure share (aggregate-share data, "market-share" data) and a single measure of wage, then we have 1 equation with two parameters, α and β can not be jointly identified. α and β, neither of which is time-varying, are exactly identified if we have data from two periods with variations in wage across the periods.

6.1.2.1.2 Simulate Market Share In this section, we now simulate the above model, with $M=2$ and data over three periods, and estimate α and β via OLS. Note that $m=0$ is leisure.

First, simulate the data.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure)
it_T <- 3
it_M <- 2
# define alpha and beta parameters
ar_alpha <- runif(it_M) + 0
fl_beta <- runif(1) + 0
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_T*it_M) + 1, nrow=it_T, ncol=it_M)
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('t', seq(1,it_T))
# Define a probability assignment function
ffi_logit_prob_alpha_beta <- function(ar_alpha, fl_beta, mt_wages) {
  # Dimensions
  it_T <- dim(mt_wages)[1]
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_T, ncol=it_M+1)
  colnames(mt_prob_o) <- paste0('m', seq(0,it_M))
  rownames(mt_prob_o) <- paste0('t', seq(1,it_T))
  # Generate Probabilities
  for (it_t in seq(1, it_T)) {
    # get current period wages
    ar_wage_at_t <- mt_wages[it_t, ]
    # Value without shocks/errors, M+1
    ar_V_hat_at_t <- c(0, ar_alpha + fl_beta*ar_wage_at_t)
    # Probabilities across M+1
    fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
    ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
    # Fill in
    mt_prob_o[it_t,] <- ar_prob_at_t
  }
  return(mt_prob_o)
}
# Show probabilities
mt_prob_o <- ffi_logit_prob_alpha_beta(ar_alpha, fl_beta, mt_wages)
st_caption <- 'Occupation aggregate participation probabilities across time'
kable(mt_prob_o, caption=st_caption) %>% kable_styling_fc()
```

```
# mt_prob_o
```

6.1.2.1.3 Create Regression Data Inputs Second, generate relative market shares of various work occupations, columns 2 through $M+1$, with respect to leisure, column 1. If there are M categories and

Occupation aggregate participation probabilities across time

	m0	m1	m2
t1	0.1251712	0.3604563	0.5143725
t2	0.1147084	0.3381795	0.5471121
t3	0.1390180	0.2842316	0.5767504

T time periods, there are $M \times T$ observations (rows). Flatten the structure so that row-groups are the M categories, and we have time-specific information within row groups.

```
# A Matrix with share from 1:M columns
mt_prob_rela_m2leisure <-
  matrix(data=mt_prob_o[1:it_T, 2:(it_M+1)], nrow=it_T, ncol=it_M)
colnames(mt_prob_rela_m2leisure) <- paste0('m', seq(1,it_M))
rownames(mt_prob_rela_m2leisure) <- paste0('t', seq(1,it_T))
# Divide 1:M by leisure
mt_prob_rela_m2leisure <- mt_prob_rela_m2leisure/mt_prob_o[1:it_T, 1]
# Take Logs, log(Pm/Po)
mt_prob_rela_m2leisure_log <- log(mt_prob_rela_m2leisure)
# Flatten to single column
ar_prob_ols_output <- matrix(data=mt_prob_rela_m2leisure_log, nrow=it_T*it_M, ncol=1)
# Show probabilities
st_caption <- 'Log of relative participation probabilities against leisure'
kable(mt_prob_rela_m2leisure_log, caption=st_caption) %>% kable_styling_fc()
```

Log of relative participation probabilities against leisure

	m1	m2
t1	1.057688	1.413265
t2	1.081184	1.562261
t3	0.715186	1.422806

Third, construct the estimation input matrices. If there are M categories and T time periods, there are $M \times T$ observations (rows). Flatten in the same way as for outcomes. There are M indicator vectors and 1 wage vector for data inputs:

- α is a fixed effect that is m-specific, so we have as Data, M indicator vectors.
- The wage variable is shared by all, so a single data column

```
# Regression input matrix
mt_prob_ols_input <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M+1)
colnames(mt_prob_ols_input) <- paste0('m', seq(1,dim(mt_prob_ols_input)[2]))
rownames(mt_prob_ols_input) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input)[1]))
# Generate index position in ols input matrix for M indicators
mt_m_indicators <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M)
colnames(mt_prob_ols_input) <- paste0('m', seq(1,dim(mt_prob_ols_input)[2]))
rownames(mt_prob_ols_input) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input)[1]))
# loop over columns
for (it_indix in seq(1, it_M)) {
  # loop over rows
  for (it_rowMcrsT in seq(1, it_T*it_M)) {
    if ((it_rowMcrsT >= (it_indix-1)*(it_T) + 1) && it_rowMcrsT <= it_indix*(it_T)){
      mt_m_indicators[it_rowMcrsT, it_indix] <- 1
    } else {
      mt_m_indicators[it_rowMcrsT, it_indix] <- 0
    }
  }
}
# Indicators are the earlier columns
```

```
mt_prob_ols_input[, 1:it_M] <- mt_m_indicators
# Wage is the last column
mt_prob_ols_input[, it_M+1] <- matrix(data=mt_wages, nrow=it_T*it_M, ncol=1)
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```
# Construct data structure
mt_all_inputs <- cbind(ar_prob_ols_output, mt_prob_ols_input)
colnames(mt_all_inputs)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs)[dim(mt_all_inputs)[2]] <- 'wages'
tb_all_inputs <- as_tibble(mt_all_inputs)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and wages, OLS i
kable(tb_all_inputs, caption=st_caption) %>% kable_styling_fc()
```

LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and wages, OLS inputs (other columns)

log_pm_over_po	m1	m2	wages
1.057688	1	0	1.883017
1.081184	1	0	1.940467
0.715186	1	0	1.045556
1.413265	0	1	1.528105
1.562261	0	1	1.892419
1.422806	0	1	1.551435

6.1.2.1.4 Estimate Wage Coefficients Fifth, estimate the OLS equation, and compare estimate to true parameters.

```
# Regression
fit_ols_agg_prob <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs)
summary(fit_ols_agg_prob)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs)
##
## Residuals:
##          1          2          3          4          5          6
##  4.027e-17  2.276e-17 -6.303e-17 -6.481e-17 -1.631e-16  2.279e-16
##
## Coefficients:
##        Estimate Std. Error   t value Pr(>|t|)
## m1    2.876e-01  3.784e-16 7.599e+14   <2e-16 ***
## m2    7.883e-01  3.859e-16 2.043e+15   <2e-16 ***
## wages 4.090e-01  2.250e-16 1.818e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.721e-16 on 3 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.043e+32 on 3 and 3 DF,  p-value: < 2.2e-16
```

```
# alpha estimates
ar_coefficients <- fit_ols_agg_prob$coefficients
ar_alpha_esti <- ar_coefficients[1:(length(ar_coefficients)-1)]
fl_beta_esti <- ar_coefficients[length(ar_coefficients)]
# Compare estimates and true
print(paste0('ar_alpha_esti=', ar_alpha_esti))
```

```
## [1] "ar_alpha_esti=0.287577520124614" "ar_alpha_esti=0.788305135443806"
```

```
print(paste0('ar_alpha=', ar_alpha))
```

```
## [1] "ar_alpha=0.287577520124614" "ar_alpha=0.788305135443807"
```

```
print(paste0('fl_beta_esti=', fl_beta_esti))
```

```
## [1] "fl_beta_esti=0.4089769218117"
```

```
print(paste0('fl_beta=', fl_beta))
```

```
## [1] "fl_beta=0.4089769218117"
```

```
# Simulate given estimated parameters using earlier function
mt_prob_o_esti <- ffi_logit_prob_alpha_beta(ar_alpha_esti, fl_beta_esti, mt_wages)
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o_esti, caption=st_caption) %>% kable_styling_fc()
```

Predicted probabilities based on estimates

	m0	m1	m2
t1	0.1251712	0.3604563	0.5143725
t2	0.1147084	0.3381795	0.5471121
t3	0.1390180	0.2842316	0.5767504

```
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
kable(mt_prob_o_esti-mt_prob_o, caption=st_caption) %>% kable_styling_fc()
```

Compare differences in probability predictions based on estimates and true probabilities

	m0	m1	m2
t1	0	0	0
t2	0	0	0
t3	0	0	0

6.1.2.2 Logistic Choices with Time-specific Observables

In our example here, there are choice alternatives (m) and time periods (t). In terms of data inputs, in the prior example, we had on the Right

In the example in the prior section, the data structure included:

1. m-specific variables: indicators
2. m- and t-specific variables: wages

In the new data structure for this section, we have:

1. m-specific variables: indicators
2. m- and t-specific variables: wages
3. t-specific variables: characteristics that are homogeneous across groups, but varying over time. This will include both a trend variable capturing time patterns, as well as an observable that is different over time.

Suppose there is some information that impacts individual's willingness to stay at home. Note that leisure is the same as work from home. For example, there could be technological improvements that makes home-production less time-consuming, and we might have a variable that captures the efficiency of home-technology. Captured by a time trend, there might also be changes in social attitudes, which could be harder to measure.

Suppose we still have:

$$V_{itm} = \widehat{V}_{tm} + \epsilon_{itm} = \alpha_m + \beta \cdot \text{WAGE}_{tm} + \epsilon_{itm}$$

But now, for the leisure category we have:

$$V_{it0} = \widehat{V}_{t0} + \epsilon_{it0} = \alpha_0 + \phi \cdot t + \theta \cdot \text{HOMETECH}_t + \epsilon_{it0}$$

Because only the differences in value across choices matter, so we can continue with the prior normalization, in difference, we have:

$$V_{itm} - V_{it0} = (\alpha_m - \alpha_0) + (\beta \cdot \text{WAGE}_{tm} - \phi \cdot t - \theta \cdot \text{HOMETECH}_t) + (\epsilon_{itm} - \epsilon_{it0})$$

Note that:

- α_0 is not identifiable.
- If the HOMETECH or time variables are multiplied by negative one, the signs changes.
- Since only the differences in value matter, the coefficients for these variables represent the net changes on alternative value difference due to changes in time-trend or the HOMETECH variable.

6.1.2.2.1 Simulate Market Share We simulate market share data now, with information over several more periods, and the HOMETECH variable, along with the time trend.

First, simulate the data.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure/home)
it_T <- 5
it_M <- 4
# define alpha and beta parameters
ar_alpha <- runif(it_M)*0.5
fl_beta <- runif(1)*0.25
# also negative time-trend from home category perspective, culturally increasingly accepting of work
fl_phi <- -runif(1)*0.50
# home-tech is negative from home category perspective, higher home-tech, more chance to work
fl_theta <- -runif(1)*1
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_T*it_M) + 1, nrow=it_T, ncol=it_M)
# HOMETECH changes, random and sorted in ascending order, increasing over time
ar_hometech <- sort(runif(it_T))
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('t', seq(1,it_T))
# Define a probability assignment function
ffi_logit_prob2_alpha_beta <- function(ar_alpha, fl_beta, fl_phi, fl_theta, ar_hometech, mt_wages) {
  # Dimensions
  it_T <- dim(mt_wages)[1]
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_T, ncol=it_M+1)
  colnames(mt_prob_o) <- paste0('m', seq(0,it_M))
  rownames(mt_prob_o) <- paste0('t', seq(1,it_T))
  # Generate Probabilities
  for (it_t in seq(1, it_T)) {
    # get current period wages
    ar_wage_at_t <- mt_wages[it_t, ]
    # Value without shocks/errors, M+1
    ar_V_hat_at_t <- c(0, ar_alpha + fl_beta*ar_wage_at_t - fl_phi*it_t - fl_theta*ar_hometech[it_t]
    # Probabilities across M+1
    fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
```

```
    ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
    # Fill in
    mt_prob_o[it_t,] <- ar_prob_at_t
  }
  return(mt_prob_o)
}
# Show probabilities
mt_prob_o2 <- ffi_logit_prob2_alpha_beta(ar_alpha, fl_beta, fl_phi, fl_theta, ar_hometech, mt_wages)
st_caption <- 'Occupation aggregate participation probabilities across time: time-varying observable
kable(mt_prob_o2, caption=st_caption) %>% kable_styling_fc()
```

Occupation aggregate participation probabilities across time: time-varying observables, time- and occupation-specific wages, occupation-specific intercepts; note reduction in m0, home-activity share over time

	m0	m1	m2	m3	m4
t1	0.1032335	0.2056532	0.2511476	0.1789233	0.2610423
t2	0.0932896	0.1891478	0.2441729	0.1906952	0.2826945
t3	0.0805830	0.1920307	0.2269798	0.2293885	0.2710180
t4	0.0633282	0.2043484	0.2589892	0.2137280	0.2596062
t5	0.0653460	0.1978795	0.2420864	0.2224408	0.2722473

```
# mt_prob_o
```

6.1.2.2.2 Create Regression Data Inputs

Second, generate relative market shares of various work occupations as prior

```
mt_prob_rela_m2leisure2 <- matrix(data=mt_prob_o2[1:it_T, 2:(it_M+1)], nrow=it_T, ncol=it_M)
ar_prob_ols_output2 <- matrix(data=log(mt_prob_rela_m2leisure2/mt_prob_o2[1:it_T, 1]), nrow=it_T*it_
```

Third, construct the estimation input matrices as before. There are M indicator vectors, a time variable, a HOMETECH variable, and 1 wage vector for data inputs:

```
# Regression input matrix
mt_prob_ols_input2 <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M+2+1)
colnames(mt_prob_ols_input2) <- paste0('m', seq(1,dim(mt_prob_ols_input2)[2]))
rownames(mt_prob_ols_input2) <- paste0('rowMcrsT', seq(1,dim(mt_prob_ols_input2)[1]))
# Generate index position in ols input matrix for M indicators
mt_m_indicators <- matrix(data=NA, nrow=it_T*it_M, ncol=it_M)
for (it_indix in seq(1, it_M)) {
  for (it_rowMcrsT in seq(1, it_T*it_M)) {
    if ((it_rowMcrsT >= (it_indix-1)*(it_T) + 1) && it_rowMcrsT <= it_indix*(it_T)){
      mt_m_indicators[it_rowMcrsT, it_indix] <- 1
    } else {
      mt_m_indicators[it_rowMcrsT, it_indix] <- 0
    }
  }
}
# Indicators are the earlier columns
mt_prob_ols_input2[, 1:it_M] <- mt_m_indicators
# Time variable column
ar_time <- matrix(seq(1,it_T), nrow=it_T*it_M, ncol=1)
mt_prob_ols_input2[, it_M+1] <- ar_time
# Home Tech Column
ar_hometech_mesh <- matrix(ar_hometech, nrow=it_T*it_M, ncol=1)
mt_prob_ols_input2[, it_M+2] <- ar_hometech_mesh
# Wage is the last column
mt_prob_ols_input2[, it_M+3] <- matrix(data=mt_wages, nrow=it_T*it_M, ncol=1)
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```
# Construct data structure
mt_all_inputs2 <- cbind(ar_prob_ols_output2, mt_prob_ols_input2)
colnames(mt_all_inputs2)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]-2] <- 'time'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]-1] <- 'hometech'
colnames(mt_all_inputs2)[dim(mt_all_inputs2)[2]] <- 'wages'
tb_all_inputs2 <- as_tibble(mt_all_inputs2)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and other variab
kable(tb_all_inputs2, caption=st_caption) %>% kable_styling_fc()
```

LHS=Log Probability Ratios (column 1); RHS=Indicators of occupations and other variables, OLS inputs (other columns)

log_pm_over_po	m1	m2	m3	m4	time	hometech	wages
0.6891981	1	0	0	0	1	0.1471136	1.892419
0.7068206	1	0	0	0	2	0.2891597	1.551435
0.8683678	1	0	0	0	3	0.5941420	1.456615
1.1714953	1	0	0	0	4	0.9022990	1.956833
1.1079617	1	0	0	0	5	0.9630242	1.453334
0.8890474	0	1	0	0	1	0.1471136	1.677571
0.9621685	0	1	0	0	2	0.2891597	1.572633
1.0355731	0	1	0	0	3	0.5941420	1.102925
1.4084555	0	1	0	0	4	0.9022990	1.899825
1.3095984	0	1	0	0	5	0.9630242	1.246088
0.5499640	0	0	1	0	1	0.1471136	1.042059
0.7149683	0	0	1	0	2	0.2891597	1.327921
1.0461296	0	0	1	0	3	0.5941420	1.954504
1.2163730	0	0	1	0	4	0.9022990	1.889539
1.2249646	0	0	1	0	5	0.9630242	1.692803
0.9276892	0	0	0	1	1	0.1471136	1.640507
1.1086584	0	0	0	1	2	0.2891597	1.994270
1.2128974	0	0	0	1	3	0.5941420	1.655706
1.4108350	0	0	0	1	4	0.9022990	1.708530
1.4270142	0	0	0	1	5	0.9630242	1.544066

6.1.2.2.3 Estimate Wage Coefficients, HOMETECH Effects and Time Trend Similar to before, estimate with OLS.

```
# Regression
fit_ols_agg_prob2 <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs2)
summary(fit_ols_agg_prob2)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs2)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.670e-16 -8.439e-17 -2.664e-17  9.403e-17  2.704e-16
##
## Coefficients:
##           Estimate Std. Error   t value Pr(>|t|)
## m1       1.438e-01  3.165e-16 4.543e+14   <2e-16 ***
## m2       3.942e-01  2.927e-16 1.347e+15   <2e-16 ***
## m3       2.045e-01  3.046e-16 6.713e+14   <2e-16 ***
```

```
## m4       4.415e-01  3.234e-16 1.365e+15   <2e-16 ***
## time     2.278e-02  1.568e-16 1.453e+14   <2e-16 ***
## hometech 5.281e-01  6.900e-16 7.653e+14   <2e-16 ***
## wages    2.351e-01  1.615e-16 1.456e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.741e-16 on 13 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.098e+32 on 7 and 13 DF,  p-value: < 2.2e-16
```

```
# alpha estimates
ar_coefficients2 <- fit_ols_agg_prob2$coefficients
ar_alpha_esti2 <- ar_coefficients2[1:(length(ar_coefficients2)-3)]
# time -1 because of the sign structure
fl_phi_esti2 <- -1*ar_coefficients2[length(ar_coefficients2)-2]
# time -1 because of the sign structure
fl_theta_esti2 <- -1*ar_coefficients2[length(ar_coefficients2)-1]
fl_beta_esti2 <- ar_coefficients2[length(ar_coefficients2)]
# Compare estimates and true
print(paste0('ar_alpha_esti2=', ar_alpha_esti2))
```

```
## [1] "ar_alpha_esti2=0.143788760062308" "ar_alpha_esti2=0.394152567721904"
## [3] "ar_alpha_esti2=0.20448846090585"  "ar_alpha_esti2=0.441508702002466"
```

```
print(paste0('ar_alpha=', ar_alpha))
```

```
## [1] "ar_alpha=0.143788760062307" "ar_alpha=0.394152567721903" "ar_alpha=0.20448846090585"
## [4] "ar_alpha=0.441508702002466"
```

```
print(paste0('fl_theta_esti2=', fl_theta_esti2))
```

```
## [1] "fl_theta_esti2=-0.528105488047005"
```

```
print(paste0('fl_theta=', fl_theta))
```

```
## [1] "fl_theta=-0.528105488047004"
```

```
print(paste0('fl_phi_esti2=', fl_phi_esti2))
```

```
## [1] "fl_phi_esti2=-0.0227782496949655"
```

```
print(paste0('fl_phi=', fl_phi))
```

```
## [1] "fl_phi=-0.0227782496949658"
```

```
print(paste0('fl_beta_esti2=', fl_beta_esti2))
```

```
## [1] "fl_beta_esti2=0.235116821073461"
```

```
print(paste0('fl_beta=', fl_beta))
```

```
## [1] "fl_beta=0.235116821073461"
```

```
# Simulate given estimated parameters using earlier function
mt_prob_o2_esti <- ffi_logit_prob2_alpha_beta(ar_alpha_esti2, fl_beta_esti2, fl_phi_esti2, fl_theta_
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o2_esti, caption=st_caption) %>% kable_styling_fc()
```

```
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
kable(mt_prob_o2_esti-mt_prob_o2, caption=st_caption) %>% kable_styling_fc()
```

Predicted probabilities based on estimates

	m0	m1	m2	m3	m4
t1	0.1032335	0.2056532	0.2511476	0.1789233	0.2610423
t2	0.0932896	0.1891478	0.2441729	0.1906952	0.2826945
t3	0.0805830	0.1920307	0.2269798	0.2293885	0.2710180
t4	0.0633282	0.2043484	0.2589892	0.2137280	0.2596062
t5	0.0653460	0.1978795	0.2420864	0.2224408	0.2722473

Compare differences in probability predictions based on estimates and true probabilities

	m0	m1	m2	m3	m4
t1	0	0	0	0	0
t2	0	0	0	0	0
t3	0	0	0	0	0
t4	0	0	0	0	0
t5	0	0	0	0	0

6.1.2.3 Logistic Choices and Multiple Types of Workers

In the prior two examples, we had the same type of worker, choosing among multiple occupations. There could be different types of workers, unskilled and skilled workers, both of whom can choose among the same set of job possibilities. If the two types of workers do not share any parameters in the occupation-choice problems, then we can estimate their problems separately. However, perhaps on the stay-at-home front, they share the same time trend, which captures overall cultural changes to work vs not-work. And perhaps the wage parameter is the same. On other fronts, they are different with different parameters as well as data inputs.

So we have:

$$V_{istm} - V_{ist0} = (\alpha_{sm} - \alpha_{s0}) + (\beta \cdot \text{WAGE}_{stm} - \phi \cdot t - \theta_s \cdot \text{HOMETECH}_{st}) + (\epsilon_{istm} - \epsilon_{ist0})$$

In the new data structure for this section, we have:

1. k- and m-specific variables: indicators
2. k- and m- and t-specific variable: wages
3. t-specific variable: dates
4. k- and t-specific variable: hometech

6.1.2.3.1 Simulate Market Share

In the simulation example, store all K types and T time periods as rows, and the M occupational types as columns.

```
# set seed
set.seed(123)
# K skill levels, T periods, and M occupations (+1 leisure/home)
it_K <- 2
it_T <- 3
it_M <- 4
# define alpha and beta parameters
mt_alpha <- matrix(runif(it_K*it_M)*0.5, nrow=it_K, ncol=it_M)
colnames(mt_alpha) <- paste0('m', seq(1,dim(mt_alpha)[2]))
rownames(mt_alpha) <- paste0('k', seq(1,dim(mt_alpha)[1]))
fl_beta <- runif(1)*0.25
# also negative time-trend from home category perspective, culturally increasingly accepting of work
fl_phi <- -runif(1)*0.50
# home-tech is negative from home category perspective, higher home-tech, more chance to work
ar_theta <- -runif(it_K)*1
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(it_K*it_T*it_M) + 1, nrow=it_K*it_T, ncol=it_M)
```

```
colnames(mt_wages) <- paste0('m', seq(1,it_M))
rownames(mt_wages) <- paste0('kxt', seq(1,it_K*it_T))
# HOMETECH changes, random and sorted in ascending order, increasing over time, specific to each k
mt_hometech <- sapply(seq(1,it_K), function(i){sort(runif(it_T))})
colnames(mt_hometech) <- paste0('k', seq(1,dim(mt_hometech)[2]))
rownames(mt_hometech) <- paste0('t', seq(1,dim(mt_hometech)[1]))
# Define a probability assignment function
ffi_logit_prob3_alpha_beta <- function(it_K, it_T, it_M,
  mt_alpha, fl_beta, fl_phi, ar_theta, mt_hometech, mt_wages) {

  # Aggregate probabilities
  mt_prob_o <- matrix(data=NA, nrow=it_K*it_T, ncol=it_M+3)
  colnames(mt_prob_o) <- c('skillgroup', 'time', paste0('m', seq(0,it_M)))
  rownames(mt_prob_o) <- paste0('kxt', seq(1,it_K*it_T))
  # Generate Probabilities
  it_kxt_ctr <- 0
  for (it_k in seq(1, it_K)) {
    for (it_t in seq(1, it_T)) {
      # Counter updating
      it_kxt_ctr <- it_kxt_ctr + 1
      # get current period wages
      ar_wage_at_t <- mt_wages[it_kxt_ctr, ]
      # Value without shocks/errors, M+1
      ar_V_hat_at_t <- c(0, mt_alpha[it_k,] + fl_beta*ar_wage_at_t - fl_phi*it_t - ar_theta[it_k]*mt
      # Probabilities across M+1
      fl_prob_denominator <- sum(exp(ar_V_hat_at_t))
      ar_prob_at_t <- exp(ar_V_hat_at_t)/fl_prob_denominator
      # Fill in
      mt_prob_o[it_kxt_ctr,1] <- it_k
      mt_prob_o[it_kxt_ctr,2] <- it_t
      mt_prob_o[it_kxt_ctr,3:dim(mt_prob_o)[2]] <- ar_prob_at_t
      # row name
      rownames(mt_prob_o)[it_kxt_ctr] <- paste0('rk', it_k, 't', it_t)
    }
  }
  return(mt_prob_o)

}
# Show probabilities
mt_prob_o3_full <- ffi_logit_prob3_alpha_beta(it_K, it_T, it_M,
  mt_alpha, fl_beta, fl_phi, ar_theta, mt_hometech, mt_wages)
# Selected only probability columns
mt_prob_o3 <- mt_prob_o3_full[, 3:dim(mt_prob_o3_full)[2]]
st_caption <- 'Occupation aggregate participation probabilities across time: skill- and time-varying
kable(mt_prob_o3_full, caption=st_caption) %>% kable_styling_fc()
```

Occupation aggregate participation probabilities across time: skill- and time-varying observables, skill and time- and occupation-specific wages, skill and occupation-specific intercepts; common wage coefficient and time coefficient; note reduction in m0, home-activity share over time

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	1	1	0.0887175	0.1995146	0.2020237	0.2757014	0.2340428
rk1t2	1	2	0.0646539	0.1984822	0.2223035	0.2803052	0.2342551
rk1t3	1	3	0.0358582	0.1975615	0.2339696	0.2909961	0.2416147
rk2t1	2	1	0.0942785	0.2435583	0.2481846	0.1610703	0.2529083
rk2t2	2	2	0.0780946	0.2411637	0.2669887	0.1673420	0.2464110
rk2t3	2	3	0.0572989	0.2348487	0.2807791	0.1643585	0.2627147

```
# mt_prob_o
```

6.1.2.3.2 Create Regression Data Inputs

Second, generate relative market shares of various work occupations as prior.

```
mt_prob_rela_m2leisure3 <- mt_prob_o3[, 2:(it_M+1)]/mt_prob_o3[, 1]
# mt_log_prob_rela_m2leisure3 <- log(mt_prob_rela_m2leisure3/mt_prob_o3[, 1])
# kable(log(mt_prob_rela_m2leisure3/mt_prob_o3[, 1]), caption=st_caption) %>% kable_styling_fc()
```

Third, construct the estimation input matrices as before. There are $K \times M$ indicator vectors, a time variable, a HOMETECH variable, and 1 wage vector for data inputs. The indicator vectors are specific interecept for each type of worker (skilled or not) and for each occupation. Note:

1. k and m specific indicators
2. k specific hometech coefficients
3. common time coefficient
4. common coefficient for wage

```
# Regression input matrix
# it_K*it_M+it_K+1+1: 1. it_K*it_M for all indicators; 2. it_k for HOMETECH; 3. 1 for time; 4. 1 for
mt_prob_ols_input3 <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K*it_M+it_K+1+1)
colnames(mt_prob_ols_input3) <- paste0('m', seq(1,dim(mt_prob_ols_input3)[2]))
rownames(mt_prob_ols_input3) <- paste0('rowkxMxT', seq(1,dim(mt_prob_ols_input3)[1]))

# LHS variable meshed store
ar_prob_ols_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
# RHS variables meshed store
mt_indi_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K*it_M)
ar_time_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
ar_wage_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=1)
mt_hometech_mesh_kmt <- matrix(data=NA, nrow=it_K*it_T*it_M, ncol=it_K)

# Loop over columns
it_kxm_ctr <- 0
for (it_r_k in seq(1, it_K)) {
  for (it_r_m in seq(1, it_M)) {
    # Column counter
    it_kxm_ctr <- it_kxm_ctr + 1

    # Update name of indicator column
    colnames(mt_prob_ols_input3)[it_kxm_ctr] <- paste0('i_k', it_r_k, 'm', it_r_m)
    # Start and end row for the indictor function mt_indi_mesh_kmt
    it_indi_str_row <- (it_kxm_ctr-1)*it_T
    it_indi_end_row <- (it_kxm_ctr+0)*it_T

    # Update names of the hometech column
    colnames(mt_prob_ols_input3)[it_K*it_M + it_r_k] <- paste0('hometech_k', it_r_k)
    # Start and end row for the indictor function mt_hometech_mesh_kmt
    it_hometech_str_row <- (it_r_k-1)*it_M*it_T
    it_hometech_end_row <- (it_r_k+0)*it_M*it_T

    # Loop over rows
    it_rowKxT <- 0
    it_rowKxTxM <- 0
    for (it_k in seq(1, it_K)) {
      for (it_m in seq(1, it_M)) {
        for (it_t in seq(1, it_T)) {

          # KxT group counter
```

```
          it_rowKxT <- it_T*(it_k-1) + it_t

          # Row counter
          it_rowKxTxM <- it_rowKxTxM + 1

          # Indicator matrix, heterogeneous by K and M
          if ((it_rowKxTxM > it_indi_str_row) && (it_rowKxTxM <= it_indi_end_row)){
            mt_indi_mesh_kmt[it_rowKxTxM, it_kxm_ctr] <- 1
          } else {
            mt_indi_mesh_kmt[it_rowKxTxM, it_kxm_ctr] <- 0
          }

          # HOMETECH specific matrix, heterogeneous by K, homogeneous by M
          if (it_r_m == 1) {
            if ((it_rowKxTxM > it_hometech_str_row) && (it_rowKxTxM <= it_hometech_end_row)){
              mt_hometech_mesh_kmt[it_rowKxTxM, it_r_k] <- mt_hometech[it_t, it_k]
            } else {
              mt_hometech_mesh_kmt[it_rowKxTxM, it_r_k] <- 0
            }
          }

          # Only need to do once, homogeneous across K, M and T
          if (it_kxm_ctr == 1) {
            rownames(mt_prob_ols_input3)[it_rowKxTxM] <- paste0('ik', it_k, 'm', it_m, 't', it_t)
            # RHS
            ar_time_mesh_kmt[it_rowKxTxM] <- it_t
            ar_wage_mesh_kmt[it_rowKxTxM] <- mt_wages[it_rowKxT, it_m]
            # LHS, log of probability
            ar_prob_ols_mesh_kmt[it_rowKxTxM] <- log(mt_prob_rela_m2leisure3[it_rowKxT, it_m])
          }
        }
      }
    }
  }
}
# Indicators are the earlier columns
mt_prob_ols_input3[, 1:(it_K*it_M)] <- mt_indi_mesh_kmt
# Time variable column
it_col_start <- (it_K*it_M) + 1
it_col_end <- it_col_start + it_K - 1
mt_prob_ols_input3[, it_col_start:it_col_end] <- mt_hometech_mesh_kmt
# Home Tech Column
mt_prob_ols_input3[, it_col_end+1] <- ar_time_mesh_kmt
# Wage is the last column
mt_prob_ols_input3[, it_col_end+2] <- ar_wage_mesh_kmt
# kable out
# kable(mt_prob_ols_input3) %>% kable_styling_fc()
```

Fourth, combine left-hand-side and right-hand-side regression input data structures/

```
# Construct data structure
mt_all_inputs3 <- cbind(ar_prob_ols_mesh_kmt, mt_prob_ols_input3)
colnames(mt_all_inputs3)[1] <- 'log_pm_over_po'
colnames(mt_all_inputs3)[dim(mt_all_inputs3)[2]-1] <- 'time'
colnames(mt_all_inputs3)[dim(mt_all_inputs3)[2]] <- 'wages'
tb_all_inputs3 <- as_tibble(mt_all_inputs3)
# Show data
st_caption <- 'LHS=Log Probability Ratios (column 1); RHS=Indicator, time-trends and data with diffe
```

```
kable(tb_all_inputs3, caption=st_caption) %>% kable_styling_fc_wide()
```

LHS=Log Probability Ratios (column 1); RHS=Indicator, time-trends and data with different assumptions on whether coefficients with be skill specific (hometech), occuption and skill specific (indictor), or homogeneous across skills and occupations (time and wage), OLS inputs (other columns)

log_pm_over_po	i_k1m1	i_k1m2	i_k1m3	i_k1m4	i_k2m1	i_k2m2	i_k2m3	i_k2m4	hometech_k1	hometech_k2	time	wages
0.8104303	1	0	0	0	0	0	0	0	0.2164079	0.0000000	1	1.677571
1.1216510	1	0	0	0	0	0	0	0	0.3181810	0.0000000	2	1.572633
1.7064781	1	0	0	0	0	0	0	0	0.7584595	0.0000000	3	1.102925
0.8229277	0	1	0	0	0	0	0	0	0.2164079	0.0000000	1	1.327921
1.2349948	0	1	0	0	0	0	0	0	0.3181810	0.0000000	2	1.954504
1.8756195	0	1	0	0	0	0	0	0	0.7584595	0.0000000	3	1.889539
1.1338609	0	0	1	0	0	0	0	0	0.2164079	0.0000000	1	1.655706
1.4668305	0	0	1	0	0	0	0	0	0.3181810	0.0000000	2	1.708530
2.0937381	0	0	1	0	0	0	0	0	0.7584595	0.0000000	3	1.544066
0.9700465	0	0	0	1	0	0	0	0	0.2164079	0.0000000	1	1.963024
1.2873623	0	0	0	1	0	0	0	0	0.3181810	0.0000000	2	1.902299
1.9077727	0	0	0	1	0	0	0	0	0.7584595	0.0000000	3	1.690705
0.9491036	0	0	0	0	1	0	0	0	0.0000000	0.1428000	1	1.899825
1.1275553	0	0	0	0	1	0	0	0	0.0000000	0.2316258	2	1.246088
1.4106597	0	0	0	0	1	0	0	0	0.0000000	0.4145463	3	1.042059
0.9679200	0	0	0	0	0	1	0	0	0.0000000	0.1428000	1	1.692803
1.2292855	0	0	0	0	0	1	0	0	0.0000000	0.2316258	2	1.640507
1.5892864	0	0	0	0	0	1	0	0	0.0000000	0.4145463	3	1.994270
0.5355882	0	0	0	0	0	0	1	0	0.0000000	0.1428000	1	1.594142
0.7621188	0	0	0	0	0	0	1	0	0.0000000	0.2316258	2	1.289160
1.0537680	0	0	0	0	0	0	1	0	0.0000000	0.4145463	3	1.147114
0.9867739	0	0	0	0	0	0	0	1	0.0000000	0.1428000	1	1.795467
1.1490801	0	0	0	0	0	0	0	1	0.0000000	0.2316258	2	1.024614
1.5227867	0	0	0	0	0	0	0	1	0.0000000	0.4145463	3	1.477796

6.1.2.3.3 Estimate Wage and Time Coefficients, Skill Specific Fixed Effects and HOMETECH Coefficient

Similar to before, estimate with OLS, and show that predictions based on OLS estimates match with the true parameters. Predicted probabilities are the same as observed probabilities.

```
# Regression
fit_ols_agg_prob3 <- lm(log_pm_over_po ~ . -1, data = tb_all_inputs3)
summary(fit_ols_agg_prob3)
```

```
##
## Call:
## lm(formula = log_pm_over_po ~ . - 1, data = tb_all_inputs3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.978e-16 -1.212e-16  2.245e-17  6.724e-17  5.494e-16
##
## Coefficients:
##              Estimate Std. Error   t value Pr(>|t|)
## i_k1m1      1.438e-01  4.270e-16 3.368e+14   <2e-16 ***
## i_k1m2      2.045e-01  4.796e-16 4.264e+14   <2e-16 ***
## i_k1m3      4.702e-01  4.624e-16 1.017e+15   <2e-16 ***
## i_k1m4      2.641e-01  5.049e-16 5.229e+14   <2e-16 ***
## i_k2m1      3.942e-01  4.277e-16 9.216e+14   <2e-16 ***
## i_k2m2      4.415e-01  5.028e-16 8.780e+14   <2e-16 ***
## i_k2m3      2.278e-02  4.176e-16 5.455e+13   <2e-16 ***
## i_k2m4      4.462e-01  4.348e-16 1.026e+15   <2e-16 ***
## hometech_k1 9.568e-01  7.700e-16 1.243e+15   <2e-16 ***
## hometech_k2 4.533e-01  1.662e-15 2.728e+14   <2e-16 ***
## time        2.283e-01  2.182e-16 1.046e+15   <2e-16 ***
## wages       1.379e-01  2.235e-16 6.168e+14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.412e-16 on 12 degrees of freedom
```

```
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 5.778e+31 on 12 and 12 DF,  p-value: < 2.2e-16
```

```
# Parse coefficients
ls_coefficients_esti <- vector(mode = "list", length = 0)
ls_coefficients_true <- vector(mode = "list", length = 0)
ar_coefficients3 <- fit_ols_agg_prob3$coefficients
it_col_end <- 0
for (it_coef_grp in c(1,2,3,4)) {
  it_col_str <- it_col_end + 1
  if (it_coef_grp == 1) {
    it_grp_coef_cnt <- (it_K*it_M)
    st_coef_name <- 'indi_km'
    ar_esti_true <- mt_alpha
    it_sign <- +1
  } else if (it_coef_grp == 2) {
    it_grp_coef_cnt <- it_K
    st_coef_name <- 'hometech_k'
    ar_esti_true <- ar_theta
    it_sign <- -1
  } else if (it_coef_grp == 3) {
    it_grp_coef_cnt <- 1
    st_coef_name <- 'time'
    ar_esti_true <- fl_phi
    it_sign <- -1
  } else if (it_coef_grp == 4) {
    it_grp_coef_cnt <- 1
    st_coef_name <- 'wage'
    ar_esti_true <- fl_beta
    it_sign <- +1
  }

  # select
  it_col_end <- it_col_end + it_grp_coef_cnt
  ar_esti_curgroup <- ar_coefficients3[it_col_str:it_col_end]
  if (it_coef_grp == 1) {
      ar_esti_curgroup <- t(matrix(data=ar_esti_curgroup, nrow=it_M, ncol=it_K))
  }
  # store
  ls_coefficients_esti[[st_coef_name]] <- it_sign*ar_esti_curgroup
  ls_coefficients_true[[st_coef_name]] <- ar_esti_true
}

# Compare estimates and true
print(ls_coefficients_esti)
```

```
## $indi_km
##           [,1]      [,2]       [,3]      [,4]
## [1,] 0.1437888 0.2044885 0.47023364 0.2640527
## [2,] 0.3941526 0.4415087 0.02277825 0.4462095
##
## $hometech_k
## hometech_k1 hometech_k2
##  -0.9568333  -0.4533342
##
## $time
##       time
## -0.2283074
##
```

```
## $wage
##     wages
## 0.1378588
```

```
print(ls_coefficients_true)
```

```
## $indi_km
##           m1        m2         m3        m4
## k1 0.1437888 0.2044885 0.47023364 0.2640527
## k2 0.3941526 0.4415087 0.02277825 0.4462095
##
## $hometech_k
## [1] -0.9568333 -0.4533342
##
## $time
## [1] -0.2283074
##
## $wage
## [1] 0.1378588
```

```
# Simulate given estimated parameters using earlier function
mt_prob_o3_full_esti <- ffi_logit_prob3_alpha_beta(it_K, it_T, it_M,
  ls_coefficients_esti[['indi_km']],
  ls_coefficients_esti[['wage']],
  ls_coefficients_esti[['time']],
  ls_coefficients_esti[['hometech_k']],
  mt_hometech, mt_wages)
# Results
st_caption <- 'Predicted probabilities based on estimates'
kable(mt_prob_o3_full_esti, caption=st_caption) %>% kable_styling_fc()
```

Predicted probabilities based on estimates

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	1	1	0.0887175	0.1995146	0.2020237	0.2757014	0.2340428
rk1t2	1	2	0.0646539	0.1984822	0.2223035	0.2803052	0.2342551
rk1t3	1	3	0.0358582	0.1975615	0.2339696	0.2909961	0.2416147
rk2t1	2	1	0.0942785	0.2435583	0.2481846	0.1610703	0.2529083
rk2t2	2	2	0.0780946	0.2411637	0.2669887	0.1673420	0.2464110
rk2t3	2	3	0.0572989	0.2348487	0.2807791	0.1643585	0.2627147

```
# Results
st_caption <- 'Compare differences in probability predictions based on estimates and true probabilit
kable(mt_prob_o3_full_esti-mt_prob_o3_full, caption=st_caption) %>% kable_styling_fc()
```

Compare differences in probability predictions based on estimates and true probabilities

	skillgroup	time	m0	m1	m2	m3	m4
rk1t1	0	0	0	0	0	0	0
rk1t2	0	0	0	0	0	0	0
rk1t3	0	0	0	0	0	0	0
rk2t1	0	0	0	0	0	0	0
rk2t2	0	0	0	0	0	0	0
rk2t3	0	0	0	0	0	0	0

6.1.3 Prices from Aggregate Shares in Logistic Choice Model

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

6.1.3.1 Observed Shares and Wages

In Estimate Logistic Choice Model with Aggregate Shares, we described and developed the multinomial logistic model with choice over alternatives.

The scenario here is that we have estimated the logistic choice model using data from some prior years. We know that for another set of years, model parameters, and in particular, the effect of alternative-specific prices are the same. We have market share information, as well as all other observables from other years, however, no price for alternatives. The goal is to use existing model parameters and the aggregate shares to back out the alternative-specific prices that would explain the data.

We know that values for choice-specific alternatives, with p_m as the alternative-specific price/wage, are:

$$V_{im} = \alpha_m + \beta \cdot w_m + \epsilon_{im}$$

Choice probabilities are functions of wages. The probability that individual i chooses alternative m is:

$$P(o = m) = P_m = \frac{\exp\left(\alpha_m + \beta \cdot w_m\right)}{1 + \sum_{\widehat{m}=1}^{M} \exp\left(\alpha_{\widehat{m}} + \beta \cdot w_{\widehat{m}}\right)}$$

We observe $P(o = m)$, we know α_m across alternatives, and we know already β. We do not know w_m across alternatives. Fitting means adjusting $\{w_m\}_{m=1}^{M}$ to fit $\{P(o = m)\}_{m=1}^{M}$ observed.

Moving terms around and cross multiplying, we have:

$$\exp\left(\alpha_m + \beta \cdot w_m\right) = P_m + P_m \sum_{\widehat{m}=1}^{M} \exp\left(\alpha_{\widehat{m}} + \beta \cdot w_{\widehat{m}}\right)$$

$$e^{\alpha_m} \exp\left(\beta \cdot w_m\right) = P_m + P_m \sum_{\widehat{m}=1}^{M} e^{\alpha_{\widehat{m}}} \exp\left(\beta \cdot w_{\widehat{m}}\right)$$

This can be viewed as a linear equation, let $\exp(\alpha_m) = A_m$, which is known, and let $\exp(\beta \cdot w_m) = \theta_m$, which is a function of the known parameter β and unknown price w_m. We have:

$$P_m = A_m \cdot \theta_m - P_m \sum_{\widehat{m}=1}^{M} A_{\widehat{m}} \cdot \theta_{\widehat{m}}$$

Suppose $M = 3$, and we label the categories as m, r, a. Note that implicitly we have an outside option category that we are normalizing against. We have then:

$$\begin{aligned}
P_m &= +A_m\left(1 - P_m\right) \cdot \theta_m - A_r P_m \cdot \theta_r - A_a P_m \cdot \theta_a \\
P_r &= -A_m P_r \cdot \theta_m + A_r\left(1 - P_r\right) \cdot \theta_r - A_a P_r \cdot \theta_a \\
P_a &= -A_m P_a \cdot \theta_m - A_r P_a \cdot \theta_r + A_a\left(1 - P_a\right) \cdot \theta_a
\end{aligned}$$

Above, we have a system of equations, with three unknown parameters. Regressing the left-hand-side aggregate share vectors against the matrix of right-hand values composed of A and P values generates θ values, which then map one to one to the wages.

6.1.3.2 Simulate Market Share

In this section, we now simulate the above model, with $M = 3$ and data over three periods, and estimate α and β via OLS. Note that $m = 0$ is leisure. This is identical what is what the simulate market share section from Estimate Logistic Choice Model with Aggregate Shares.

First, wages across alternatives.

```
# set seed
set.seed(123)
# T periods, and M occupations (+1 leisure)
it_M <- 3
# define alpha and beta parameters
ar_alpha <- runif(it_M) + 0
fl_beta <- runif(1) + 0
# wage matrix, no wage for leisure
mt_wages <- matrix(runif(1 * it_M) + 1, nrow = 1, ncol = it_M)
colnames(mt_wages) <- paste0("m", seq(1, it_M))
rownames(mt_wages) <- paste0("t", seq(1, 1))
# Show wages
st_caption <- "Wages across occupations"
kable(mt_wages, caption = st_caption) %>% kable_styling_fc()
```

Wages across occupations

	m1	m2	m3
t1	1.940467	1.045556	1.528105

Second, shares across alternatives (and outside option).

```
# Define a probability assignment function
ffi_logit_prob_alpha_beta_1t <- function(ar_alpha, fl_beta, mt_wages) {
  # Dimensions
  it_M <- dim(mt_wages)[2]
  # Aggregate probabilities
  mt_prob_o <- matrix(data = NA, nrow = 1, ncol = it_M + 1)
  colnames(mt_prob_o) <- paste0("m", seq(0, it_M))
  rownames(mt_prob_o) <- paste0("t", seq(1, 1))
  # Generate Probabilities
  # Value without shocks/errors, M+1
  ar_V_hat <- c(0, ar_alpha + fl_beta * mt_wages[1, ])
  # Probabilities across M+1
  mt_prob_o[1, ] <- exp(ar_V_hat) / sum(exp(ar_V_hat))
  return(mt_prob_o)
}
# Show probabilities
ar_prob_o <- ffi_logit_prob_alpha_beta_1t(ar_alpha, fl_beta, mt_wages)
st_caption <- "Participation probabilities across categories"
kable(ar_prob_o, caption = st_caption) %>% kable_styling_fc()
```

Participation probabilities across categories

	m0	m1	m2	m3
t1	0.0506663	0.374767	0.2805663	0.2940004

6.1.3.3 Create Inputs for Wages Fit/Estimation

See the linearized structure above, where the LHS is a vector of non-outside-option alternative probabilities. And the RHS is M by M, where each row is multiplying by a different occupation-specific probability, and each column is a different non-wage component of the category specific value (without the error term).

Create the right-hand-side matrix.

```
# A Matrix with share from 1:M columns
mt_rhs_input <- matrix(data = NA, nrow = it_M, ncol = it_M)
colnames(mt_rhs_input) <- paste0("mValNoWage", seq(1, it_M))
```

```
rownames(mt_rhs_input) <- paste0("mProb", seq(1, it_M))

# Loop over rows
for (it_m_r in seq(1, it_M)) {
  # +1 to skip the outside-option category
  P_m <- ar_prob_o[it_m_r + 1]
  # Loop over columns
  for (it_m_c in seq(1, it_M)) {
    # Column value for non-wage component of the category-specific value
    A_m <- exp(ar_alpha[it_m_c])
    # Diagonal or not
    if (it_m_r == it_m_c) {
      fl_rhs_val <- A_m * (1 - P_m)
    } else {
      fl_rhs_val <- -1 * A_m * P_m
    }
    # Fill value
    mt_rhs_input[it_m_r, it_m_c] <- fl_rhs_val
  }
}

# Show rhs matrix
st_caption <- "RHS fit wages matrix"
kable(mt_rhs_input, caption = st_caption) %>% kable_styling_fc()
```

RHS fit wages matrix

	mValNoWage1	mValNoWage2	mValNoWage3
mProb1	0.8335569	-0.8243618	-0.5641281
mProb2	-0.3740494	1.5825131	-0.4223301
mProb3	-0.3919596	-0.6467025	1.0627249

Add in the LHS probability column.

```
# Construct data structure, LHS and RHS, LHS first oclumn
mt_all_inputs <- cbind(ar_prob_o[2:length(ar_prob_o)], mt_rhs_input)
colnames(mt_all_inputs)[1] <- "prob_o"
tb_all_inputs <- as_tibble(mt_all_inputs)
# Show data
st_caption <- "col1=prob in non-outside options; other columns, RHS matrix"
kable(tb_all_inputs, caption=st_caption) %>% kable_styling_fc()
```

col1=prob in non-outside options; other columns, RHS matrix

prob_o	mValNoWage1	mValNoWage2	mValNoWage3
0.3747670	0.8335569	-0.8243618	-0.5641281
0.2805663	-0.3740494	1.5825131	-0.4223301
0.2940004	-0.3919596	-0.6467025	1.0627249

6.1.3.4 Solve/Estimate for Wages that Explain Shares

Given the RHS matrix just generated estimate the wages, and check that the match the wages used to simulate the probabilities.

```
# Regression
fit_ols_agg_prob_to_wages <- lm(prob_o ~ . - 1, data = tb_all_inputs)
summary(fit_ols_agg_prob_to_wages)
```

```
##
```

```
## Call:
## lm(formula = prob_o ~ . - 1, data = tb_all_inputs)
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## mValNoWage1    5.548         NA      NA       NA
## mValNoWage2    2.517         NA      NA       NA
## mValNoWage3    3.855         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

```
# alpha estimates
ar_coefficients <- fit_ols_agg_prob_to_wages$coefficients
ar_wages_esti <- log(ar_coefficients)/fl_beta
# Compare estimates and true
print(paste0("ar_coefficients=", ar_coefficients))
```

```
## [1] "ar_coefficients=5.54816023677087" "ar_coefficients=2.51744522035287"
## [3] "ar_coefficients=3.85489489133316"
```

```
print(paste0("ar_wages_esti=", ar_wages_esti))
```

```
## [1] "ar_wages_esti=1.94046728429384" "ar_wages_esti=1.04555649938993" "ar_wages_esti=1.5281054880
```

```
print(paste0("mt_wages=", mt_wages))
```

```
## [1] "mt_wages=1.94046728429385" "mt_wages=1.04555649938993" "mt_wages=1.528105488047"
```

6.2 Quantile Regression

6.2.1 Quantile Regression Basics

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

6.2.1.1 Estimate Mean and Conditional Quantile Coefficients using mtcars dataset

Here, we conduct tests for using the quantreg package, using the built-in mtcars dataset.

First, estimate the mean (OLS) regression:

```
fit_mean <- lm(mpg ~ disp + hp + factor(am) + factor(vs), data = mtcars)
summary(fit_mean)
```

```
##
## Call:
## lm(formula = mpg ~ disp + hp + factor(am) + factor(vs), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7981 -1.9532  0.0111  1.5665  5.6321
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.832119   2.890418   8.591 3.32e-09 ***
## disp        -0.008304   0.010087  -0.823  0.41757
## hp          -0.037623   0.013846  -2.717  0.01135 *
```

```
## factor(am)1  4.419257   1.493243   2.960  0.00634 **
## factor(vs)1  2.052472   1.627096   1.261  0.21794
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.812 on 27 degrees of freedom
## Multiple R-squared:  0.8104, Adjusted R-squared:  0.7823
## F-statistic: 28.85 on 4 and 27 DF,  p-value: 2.13e-09
```

Now estimate conditional quantile regressions (not that this remains linear) at various quantiles, standard error obtained via bootstrap. Note that there is a gradient in the quantile hp coefficients as well as disp. disp sign reverses, also the coefficient on factor am is different by quantiles:

```
ls_fl_quantiles <- c(0.25, 0.50, 0.75)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
               tau = ls_fl_quantiles,
               data = mtcars)
summary(fit_quantiles, se = "boot")
```

```
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 25.34665  1.50911   16.79575  0.00000
## disp        -0.02441  0.00952   -2.56333  0.01603
## hp          -0.01672  0.01641   -1.01845  0.31718
## factor(am)1  1.39719  1.43085    0.97648  0.33719
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 27.49722  1.78258   15.42554  0.00000
## disp        -0.02253  0.01571   -1.43388  0.16268
## hp          -0.02713  0.02329   -1.16494  0.25387
## factor(am)1  3.37328  2.09860    1.60739  0.11919
##
## Call: rq(formula = mpg ~ disp + hp + factor(am), tau = ls_fl_quantiles,
##     data = mtcars)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept) 28.06384  1.75630   15.97892  0.00000
## disp         0.00445  0.01427    0.31201  0.75734
## hp          -0.06662  0.01781   -3.74146  0.00084
## factor(am)1  7.91402  2.48452    3.18533  0.00353
```

6.2.1.2 Test Conditional Quantile Coefficients if Different

Use the rq.anova function frm the quantile regression packge to conduct WALD test. Remember WALD test says given unrestricted model's estimates, test where null is that the coefficients satisfy some linear

restrictions.

To test, use the returned object from running rq with different numbers of quantiles, and set the option *joint* to true or false. When joint is true: "equality of slopes should be done as joint tests on all slope parameters", when joint is false: "separate tests on each of the slope parameters should be reported". A slope parameter refers to one of the RHS variables.

Note that quantile tests are "parallel line" tests. Meaning that we should except to have different x-intercepts for each quantile, because they represents the levels of the conditional shocks distributions. However, if quantile coefficients for the slopes are all the same, then there are no quantile specific effects, mean effects would be sufficient.

see:

- anova.rq() in quantreg package in R

6.2.1.2.1 Test Statistical Difference between 25th and 50th Conditional Quantiles Given the quantile estimates above, the difference between 0.25 and 0.50 quantiles exists, but are they sufficiently large to be statistically different? What is the p-value? Reviewing the results below, they are not statistically different.

First, joint = TRUE. This is not testing if the coefficien on disp is the same as the coefficient on hp. This is testing jointly if the coefficients for different quantiles of disp, and different quantiles of hp are the same for each RHS variable.

```
ls_fl_quantiles <- c(0.25, 0.50)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
                tau = ls_fl_quantiles,
                data = mtcars)
anova(fit_quantiles, test = "Wald", joint=TRUE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Joint Test of Equality of Slopes: tau in {  0.25 0.5  }
##
##   Df Resid Df F value Pr(>F)
## 1  3       61  0.7986 0.4994
```

Second, joint = False:

```
anova(fit_quantiles, test = "Wald", joint=FALSE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Tests of Equality of Distinct Slopes: tau in {  0.25 0.5  }
##
##              Df Resid Df F value Pr(>F)
## disp          1       63  0.0304 0.8621
## hp            1       63  0.5397 0.4653
## factor(am)1   1       63  1.0957 0.2992
```

6.2.1.2.2 Test Statistical Difference between 25th, 50th, and 75th Conditional Quantiles The 1st quartile and median do not seem to be statistically different, now include the 3rd quartile. As seen earlier, the quartiles jointly show a gradient. Now, we can see that idisp, hp and am are separately have statistically different

First, joint = TRUE:

```
ls_fl_quantiles <- c(0.25, 0.50, 0.75)
fit_quantiles <- rq(mpg ~ disp + hp + factor(am),
                tau = ls_fl_quantiles,
```

```
              data = mtcars)
anova(fit_quantiles, test = "Wald", joint=TRUE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Joint Test of Equality of Slopes: tau in {  0.25 0.5 0.75  }
##
##   Df Resid Df F value   Pr(>F)
## 1  6       90   3.957 0.001475 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Second, joint = False:

```
anova(fit_quantiles, test = "Wald", joint=FALSE)
```

```
## Quantile Regression Analysis of Deviance Table
##
## Model: mpg ~ disp + hp + factor(am)
## Tests of Equality of Distinct Slopes: tau in {  0.25 0.5 0.75  }
##
##             Df Resid Df F value    Pr(>F)
## disp         2       94  9.2284 0.0002191 ***
## hp           2       94  6.5798 0.0021162 **
## factor(am)1  2       94  3.6669 0.0292803 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 7

Optimization

7.1 Bisection

7.1.1 Bisection

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

See the ff_opti_bisect_pmap_multi function from Fan's *REconTools* Package, which provides a resuable function based on the algorithm worked out here.

The bisection specific code does not need to do much.

- list variables in file for grouping, each group is an individual for whom we want to calculate optimal choice for using bisection.
- string variable name of input where functions are evaluated, these are already contained in the dataframe, existing variable names, row specific, rowwise computation over these, each rowwise calculation using different rows.
- scalar and array values that are applied to every rowwise calculation, all rowwise calculations using the same scalars and arrays.
- string output variable name

This is how I implement the bisection algorithm, when we know the bounding minimum and maximum to be below and above zero already.

1. Evaluate $f_a^0 = f(a^0)$ and $f_b^0 = f(b^0)$, min and max points.
2. Evaluate at $f_p^0 = f(p^0)$, where $p_0 = \frac{a^0+b^0}{2}$.
3. if $f_a^i \cdot f_p^i < 0$, then $b_{i+1} = p_i$, else, $a_{i+1} = p_i$ and $f_a^{i+1} = p_i$.
4. iteratre until convergence.

Generate New columns of a and b as we iteratre, do not need to store p, p is temporary. Evaluate the function below which we have already tested, but now, in the dataframe before generating all permutations, *tb_states_choices*, now the *fl_N* element will be changing with each iteration, it will be row specific. *fl_N* are first min and max, then each subsequent ps.

7.1.1.1 Initialize Matrix

Prepare Input Data:

```
# Parameters
fl_rho = 0.20
svr_id_var = 'INDI_ID'

# P fixed parameters, nN is N dimensional, nP is P dimensional
ar_nN_A = seq(-2, 2, length.out = 4)
ar_nN_alpha = seq(0.1, 0.9, length.out = 4)
```

```
# Choice Grid for nutritional feasible choices for each
fl_N_agg = 100
fl_N_min = 0

# Mesh Expand
tb_states_choices <- as_tibble(cbind(ar_nN_A, ar_nN_alpha)) %>%
  rowid_to_column(var=svr_id_var)

# Convert Matrix to Tibble
ar_st_col_names = c(svr_id_var,'fl_A', 'fl_alpha')
tb_states_choices <- tb_states_choices %>% rename_all(~c(ar_st_col_names))
```

Prepare Function:

```
# Define Implicit Function
ffi_nonlin_dplyrdo <- function(fl_A, fl_alpha, fl_N, ar_A, ar_alpha, fl_N_agg, fl_rho){

  ar_p1_s1 = exp((fl_A - ar_A)*fl_rho)
  ar_p1_s2 = (fl_alpha/ar_alpha)
  ar_p1_s3 = (1/(ar_alpha*fl_rho - 1))
  ar_p1 = (ar_p1_s1*ar_p1_s2)^ar_p1_s3
  ar_p2 = fl_N^((fl_alpha*fl_rho-1)/(ar_alpha*fl_rho-1))
  ar_overall = ar_p1*ar_p2
  fl_overall = fl_N_agg - sum(ar_overall)

  return(fl_overall)
}
```

Initialize the matrix with a_0 and b_0, the initial min and max points:

```
# common prefix to make reshaping easier
st_bisec_prefix <- 'bisec_'
svr_a_lst <- paste0(st_bisec_prefix, 'a_0')
svr_b_lst <- paste0(st_bisec_prefix, 'b_0')
svr_fa_lst <- paste0(st_bisec_prefix, 'fa_0')
svr_fb_lst <- paste0(st_bisec_prefix, 'fb_0')

# Add initial a and b
tb_states_choices_bisec <- tb_states_choices %>%
  mutate(!!sym(svr_a_lst) := fl_N_min, !!sym(svr_b_lst) := fl_N_agg)

# Evaluate function f(a_0) and f(b_0)
tb_states_choices_bisec <- tb_states_choices_bisec %>%
  rowwise() %>%
  mutate(!!sym(svr_fa_lst) := ffi_nonlin_dplyrdo(fl_A, fl_alpha, !!sym(svr_a_lst),
                                                 ar_nN_A, ar_nN_alpha,
                                                 fl_N_agg, fl_rho),
         !!sym(svr_fb_lst) := ffi_nonlin_dplyrdo(fl_A, fl_alpha, !!sym(svr_b_lst),
                                                 ar_nN_A, ar_nN_alpha,
                                                 fl_N_agg, fl_rho))
# Summarize
dim(tb_states_choices_bisec)
```

```
## [1] 4 7
```

```
# summary(tb_states_choices_bisec)
```

7.1.1.2 Iterate and Solve for f(p), update f(a) and f(b)

Implement the DPLYR based Concurrent bisection algorithm.

```
# fl_tol = float tolerance criteria
# it_tol = number of interations to allow at most
fl_tol <- 10^-2
it_tol <- 100

# fl_p_dist2zr = distance to zero to initalize
fl_p_dist2zr <- 1000
it_cur <- 0
while (it_cur <= it_tol && fl_p_dist2zr >= fl_tol ) {

  it_cur <- it_cur + 1

  # New Variables
  svr_a_cur <- paste0(st_bisec_prefix, 'a_', it_cur)
  svr_b_cur <- paste0(st_bisec_prefix, 'b_', it_cur)
  svr_fa_cur <- paste0(st_bisec_prefix, 'fa_', it_cur)
  svr_fb_cur <- paste0(st_bisec_prefix, 'fb_', it_cur)

  # Evaluate function f(a_0) and f(b_0)
  # 1. generate p
  # 2. generate f_p
  # 3. generate f_p*f_a
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    rowwise() %>%
    mutate(p = ((!!sym(svr_a_lst) + !!sym(svr_b_lst))/2)) %>%
    mutate(f_p = ffi_nonlin_dplyrdo(fl_A, fl_alpha, p,
                                    ar_nN_A, ar_nN_alpha,
                                    fl_N_agg, fl_rho)) %>%
    mutate(f_p_t_f_a = f_p*!!sym(svr_fa_lst))
  # fl_p_dist2zr = sum(abs(p))
  fl_p_dist2zr <- mean(abs(tb_states_choices_bisec %>% pull(f_p)))

  # Update a and b
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    mutate(!!sym(svr_a_cur) :=
             case_when(f_p_t_f_a < 0 ~ !!sym(svr_a_lst),
                       TRUE ~ p)) %>%
    mutate(!!sym(svr_b_cur) :=
             case_when(f_p_t_f_a < 0 ~ p,
                       TRUE ~ !!sym(svr_b_lst)))
  # Update f(a) and f(b)
  tb_states_choices_bisec <- tb_states_choices_bisec %>%
    mutate(!!sym(svr_fa_cur) :=
             case_when(f_p_t_f_a < 0 ~ !!sym(svr_fa_lst),
                       TRUE ~ f_p)) %>%
    mutate(!!sym(svr_fb_cur) :=
             case_when(f_p_t_f_a < 0 ~ f_p,
                       TRUE ~ !!sym(svr_fb_lst)))
  # Save from last
  svr_a_lst <- svr_a_cur
  svr_b_lst <- svr_b_cur
  svr_fa_lst <- svr_fa_cur
  svr_fb_lst <- svr_fb_cur

  # Summar current round
  print(paste0('it_cur:', it_cur, ', fl_p_dist2zr:', fl_p_dist2zr))
  summary(tb_states_choices_bisec %>%
            select(one_of(svr_a_cur, svr_b_cur, svr_fa_cur, svr_fb_cur)))
```

```
}
```

```
## [1] "it_cur:1, fl_p_dist2zr:1597.93916362849"
## [1] "it_cur:2, fl_p_dist2zr:676.06602535902"
## [1] "it_cur:3, fl_p_dist2zr:286.850590132782"
## [1] "it_cur:4, fl_p_dist2zr:117.225493866655"
## [1] "it_cur:5, fl_p_dist2zr:37.570593471664"
## [1] "it_cur:6, fl_p_dist2zr:4.60826664896022"
## [1] "it_cur:7, fl_p_dist2zr:14.4217689135683"
## [1] "it_cur:8, fl_p_dist2zr:8.38950830086659"
## [1] "it_cur:9, fl_p_dist2zr:3.93347761455868"
## [1] "it_cur:10, fl_p_dist2zr:1.88261338941038"
## [1] "it_cur:11, fl_p_dist2zr:0.744478952222305"
## [1] "it_cur:12, fl_p_dist2zr:0.187061801237917"
## [1] "it_cur:13, fl_p_dist2zr:0.117844913432613"
## [1] "it_cur:14, fl_p_dist2zr:0.0275365951418891"
## [1] "it_cur:15, fl_p_dist2zr:0.0515488156908255"
## [1] "it_cur:16, fl_p_dist2zr:0.0191152349149135"
## [1] "it_cur:17, fl_p_dist2zr:0.00385372194545752"
```

7.1.1.3 Reshape Wide to long to Wide

To view results easily, how iterations improved to help us find the roots, convert table from wide to long. Pivot twice. This allows us to easily graph out how bisection is working out iterationby iteration.

Here, we will first show what the raw table looks like, the wide only table, and then show the long version, and finally the version that is medium wide.

7.1.1.3.1 Table One–Very Wide

Show what the *tb_states_choices_bisec* looks like.

Variables are formatted like: *bisec_xx_yy*, where yy is the iteration indicator, and xx is either a, b, fa, or fb.

```
kable(head(t(tb_states_choices_bisec), 25)) %>%
  kable_styling_fc()
```

```
# str(tb_states_choices_bisec)
```

7.1.1.3.2 Table Two–Very Wide to Very Long

We want to treat the iteration count information that is the suffix of variable names as a variable by itself. Additionally, we want to treat the a,b,fa,fb as a variable. Structuring the data very long like this allows for easy graphing and other types of analysis. Rather than dealing with many many variables, we have only 3 core variables that store bisection iteration information.

Here we use the very nice *pivot_longer* function. Note that to achieve this, we put a common prefix in front of the variables we wanted to convert to long. THis is helpful, because we can easily identify which variables need to be reshaped.

```
# New variables
svr_bisect_iter <- 'biseciter'
svr_abfafb_long_name <- 'varname'
svr_number_col <- 'value'
svr_id_bisect_iter <- paste0(svr_id_var, '_bisect_ier')

# Pivot wide to very long
tb_states_choices_bisec_long <- tb_states_choices_bisec %>%
  pivot_longer(
    cols = starts_with(st_bisec_prefix),
    names_to = c(svr_abfafb_long_name, svr_bisect_iter),
    names_pattern = paste0(st_bisec_prefix, "(.*)_(.*)"),
```

INDI_ID	1.000000e+00	2.0000000	3.0000000	4.0000000
fl_A	-2.000000e+00	-0.6666667	0.6666667	2.0000000
fl_alpha	1.000000e-01	0.3666667	0.6333333	0.9000000
bisec_a_0	0.000000e+00	0.0000000	0.0000000	0.0000000
bisec_b_0	1.000000e+02	100.0000000	100.0000000	100.0000000
bisec_fa_0	1.000000e+02	100.0000000	100.0000000	100.0000000
bisec_fb_0	-1.288028e+04	-1394.7069782	-323.9421599	-51.9716069
p	1.544952e+00	8.5838318	24.8359680	65.0367737
f_p	-7.637200e-03	-0.0052211	-0.0016162	-0.0009405
f_p_t_f_a	-3.800000e-04	-0.0000237	-0.0000025	-0.0000002
bisec_a_1	0.000000e+00	0.0000000	0.0000000	50.0000000
bisec_b_1	5.000000e+01	50.0000000	50.0000000	100.0000000
bisec_fa_1	1.000000e+02	100.0000000	100.0000000	22.5557704
bisec_fb_1	-5.666956e+03	-595.7345364	-106.5105843	-51.9716069
bisec_a_2	0.000000e+00	0.0000000	0.0000000	50.0000000
bisec_b_2	2.500000e+01	25.0000000	25.0000000	75.0000000
bisec_fa_2	1.000000e+02	100.0000000	100.0000000	22.5557704
bisec_fb_2	-2.464562e+03	-224.1460032	-0.6857375	-14.8701831
bisec_a_3	0.000000e+00	0.0000000	12.5000000	62.5000000
bisec_b_3	1.250000e+01	12.5000000	25.0000000	75.0000000
bisec_fa_3	1.000000e+02	100.0000000	50.8640414	3.7940196
bisec_fb_3	-1.041574e+03	-51.1700464	-0.6857375	-14.8701831
bisec_a_4	0.000000e+00	6.2500000	18.7500000	62.5000000
bisec_b_4	6.250000e+00	12.5000000	25.0000000	68.7500000
bisec_fa_4	1.000000e+02	29.4271641	25.2510409	3.7940196

```
    values_to = svr_number_col
  )

# Print
# summary(tb_states_choices_bisec_long)
kable(head(tb_states_choices_bisec_long %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 15)) %>%
  kable_styling_fc()
```

INDI_ID	fl_A	fl_alpha	varname	biseciter	value
1	-2	0.1	a	0	0.000
1	-2	0.1	b	0	100.000
1	-2	0.1	fa	0	100.000
1	-2	0.1	fb	0	-12880.284
1	-2	0.1	a	1	0.000
1	-2	0.1	b	1	50.000
1	-2	0.1	fa	1	100.000
1	-2	0.1	fb	1	-5666.956
1	-2	0.1	a	2	0.000
1	-2	0.1	b	2	25.000
1	-2	0.1	fa	2	100.000
1	-2	0.1	fb	2	-2464.562
1	-2	0.1	a	3	0.000
1	-2	0.1	b	3	12.500
1	-2	0.1	fa	3	100.000

```
kable(tail(tb_states_choices_bisec_long %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 15)) %>%
  kable_styling_fc()
```

INDI_ID	fl_A	fl_alpha	varname	biseciter	value
4	2	0.9	b	14	65.0390625
4	2	0.9	fa	14	0.0047633
4	2	0.9	fb	14	-0.0043628
4	2	0.9	a	15	65.0360107
4	2	0.9	b	15	65.0390625
4	2	0.9	fa	15	0.0002003
4	2	0.9	fb	15	-0.0043628
4	2	0.9	a	16	65.0360107
4	2	0.9	b	16	65.0375366
4	2	0.9	fa	16	0.0002003
4	2	0.9	fb	16	-0.0020812
4	2	0.9	a	17	65.0360107
4	2	0.9	b	17	65.0367737
4	2	0.9	fa	17	0.0002003
4	2	0.9	fb	17	-0.0009405

7.1.1.3.3 Table Two–Very Very Long to Wider Again But the previous results are too long, with the a, b, fa, and fb all in one column as different categories, they are really not different categories, they are in fact different types of variables. So we want to spread those four categories of this variable into four columns, each one representing the a, b, fa, and fb values. The rows would then be uniquly identified by the iteration counter and individual ID.

```
# Pivot wide to very long to a little wide
tb_states_choices_bisec_wider <- tb_states_choices_bisec_long %>%
  pivot_wider(
    names_from = !!sym(svr_abfafb_long_name),
    values_from = svr_number_col
  )

# Print
# summary(tb_states_choices_bisec_wider)
kable(head(tb_states_choices_bisec_wider %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 10)) %>%
  kable_styling_fc_wide()
```

INDI_ID	fl_A	fl_alpha	biseciter	a	b	fa	fb
1	-2	0.1	0	0.000000	100.0000	100.00000	-12880.283918
1	-2	0.1	1	0.000000	50.0000	100.00000	-5666.955763
1	-2	0.1	2	0.000000	25.0000	100.00000	-2464.562178
1	-2	0.1	3	0.000000	12.5000	100.00000	-1041.574253
1	-2	0.1	4	0.000000	6.2500	100.00000	-408.674764
1	-2	0.1	5	0.000000	3.1250	100.00000	-126.904283
1	-2	0.1	6	0.000000	1.5625	100.00000	-1.328965
1	-2	0.1	7	0.781250	1.5625	54.69612	-1.328965
1	-2	0.1	8	1.171875	1.5625	27.46061	-1.328965
1	-2	0.1	9	1.367188	1.5625	13.23495	-1.328965

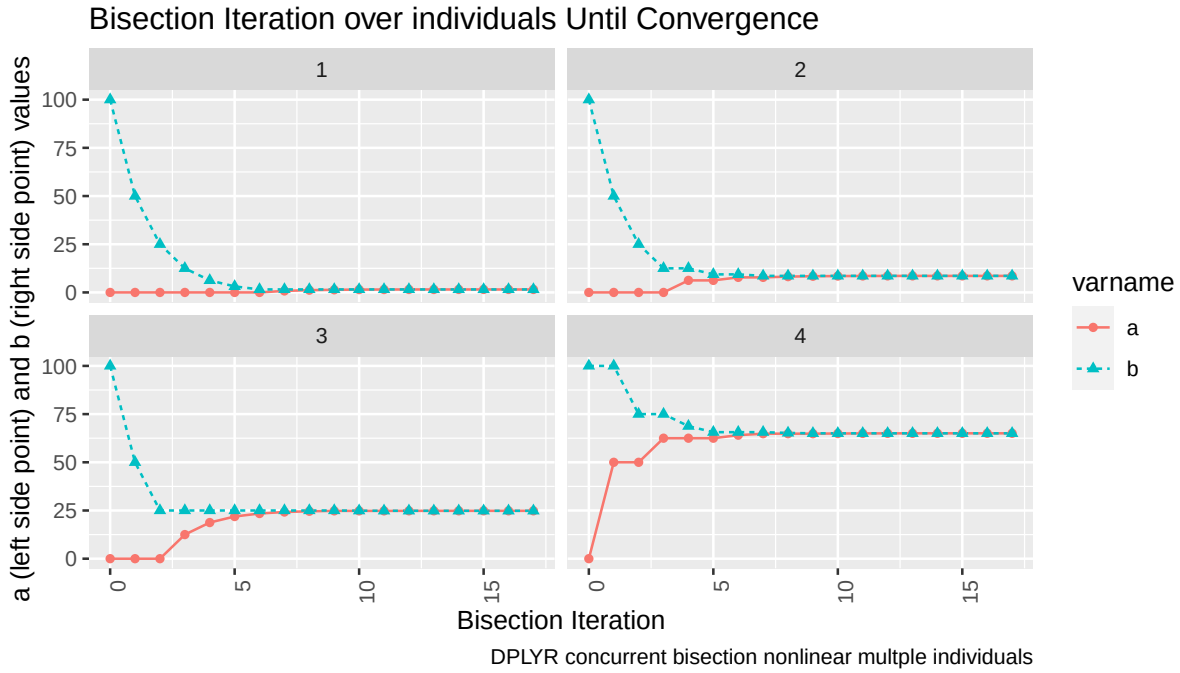
```
kable(head(tb_states_choices_bisec_wider %>%
             select(-one_of('p','f_p','f_p_t_f_a')), 10)) %>%
  kable_styling_fc_wide()
```

7.1.1.4 Graph Bisection Iteration Results

Actually we want to graph based on the long results, not the wider. Wider easier to view in table.

INDI_ID	fl_A	fl_alpha	biseciter	a	b	fa	fb
1	-2	0.1	0	0.000000	100.0000	100.00000	-12880.283918
1	-2	0.1	1	0.000000	50.0000	100.00000	-5666.955763
1	-2	0.1	2	0.000000	25.0000	100.00000	-2464.562178
1	-2	0.1	3	0.000000	12.5000	100.00000	-1041.574253
1	-2	0.1	4	0.000000	6.2500	100.00000	-408.674764
1	-2	0.1	5	0.000000	3.1250	100.00000	-126.904283
1	-2	0.1	6	0.000000	1.5625	100.00000	-1.328965
1	-2	0.1	7	0.781250	1.5625	54.69612	-1.328965
1	-2	0.1	8	1.171875	1.5625	27.46061	-1.328965
1	-2	0.1	9	1.367188	1.5625	13.23495	-1.328965

```
# Graph results
lineplot <- tb_states_choices_bisec_long %>%
    mutate(!!sym(svr_bisect_iter) := as.numeric(!!sym(svr_bisect_iter))) %>%
    filter(!!sym(svr_abfafb_long_name) %in% c('a', 'b')) %>%
    ggplot(aes(x=!!sym(svr_bisect_iter), y=!!sym(svr_number_col),
               colour=!!sym(svr_abfafb_long_name),
               linetype=!!sym(svr_abfafb_long_name),
               shape=!!sym(svr_abfafb_long_name))) +
        facet_wrap( ~ INDI_ID) +
        geom_line() +
        geom_point() +
        labs(title = 'Bisection Iteration over individuals Until Convergence',
             x = 'Bisection Iteration',
             y = 'a (left side point) and b (right side point) values',
             caption = 'DPLYR concurrent bisection nonlinear multple individuals') +
      theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(lineplot)
```



Chapter 8

Mathematics

8.1 Production Function

8.1.1 Nested CES Production Function

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.1.1.1 Denesting the Nested CES Problem

We have the following production function with four inputs x_1, x_2, y_1 and y_2. There are three ρ parameters ρ_x, ρ_y and ρ_o that correspond to inner-nest and outter nest elasticity of substitution between inputs.

The firm's expenditure minimization problem has the following objective:

$$\min_{x_1,x_2,y_1,y_2} \left(x_1 \cdot p_{x_1} + x_2 \cdot p_{x_2} + y_1 \cdot p_{y_1} + y_2 \cdot p_{y_2}\right)$$

The production quantity constraint is, using a constant-returns doubly-nested production function:

$$Y = Z \cdot \left(\beta_{o_1} \left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o} + \beta_{o_2} \left(\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}\right)^{\rho_o}\right)^{\frac{1}{\rho_o}}$$

Note that we are assuming constant-returns to scale in a competitive setting, so firms do not make profits. We solve for expenditure minimization rather than profit maximization.

Rather than solving the problem above directly in an expenditure minimization, we can divide the problem above into three parts, the **X Problem**, the **Y Problem** and the **Z Problem**.

8.1.1.1.1 Marginal Product of Labor A key object to consider is the marginal product of input (labor or capital). Taking derivative of output Y with respect to input x_1, we have:

$$\frac{\partial Y}{\partial x_1} = \left[\frac{1}{\rho_o} Z \left(\beta_{o_1} \left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o} + \beta_{o_2} \left(\left(\beta_{y_1} y_1^{\rho_y} + \beta_{y_2} y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}\right)^{\rho_o}\right)^{\frac{1}{\rho_o}-1}\right] \cdot \left[\rho_o \beta_{o_1} \left(\left(\beta_{x_1} x_1^{\rho_x} + \beta_{x_2} x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}\right)^{\rho_o-}\right.$$

What is the relationship between the marginal product of labor and the wage? Let λ be the lagrange multiplier for the overall problem:

$$p_{x_1} = \lambda \cdot \left(\frac{\partial Y}{\partial x_1}\right)$$

8.1.1.1.2 Three Denested Sub-problems, X, Y and O Problems **The X problem**:

$$\min_{x_1,x_2}\left(x_1\cdot p_{x_1}+x_2\cdot p_{x_2}\right)$$

$$O_x=\left(\beta_{x_1}x_1^{\rho_x}+\beta_{x_2}x_2^{\rho_x}\right)^{\frac{1}{\rho_x}}$$

The Y problem:

$$\min_{y_1,y_2}\left(y_1\cdot p_{y_1}+y_2\cdot p_{y_2}\right)$$

$$O_y=\left(\beta_{y_1}y_1^{\rho_y}+\beta_{y_2}y_2^{\rho_y}\right)^{\frac{1}{\rho_y}}$$

The O problem:

$$\min_{o_1,o_2}\left(O_x\cdot p_{o_1}+O_y\cdot p_{o_2}\right)$$

$$Y=Z\cdot\left(\beta_{o_1}O_x^{\rho_o}+\beta_{o_2}O_y^{\rho_o}\right)^{\frac{1}{\rho_o}}$$

8.1.1.1.3 Marginal Product of Labor for De-nested Problem We can also take the derivative of the output requirement for the X problem with respect to x_1, we have:

$$\frac{\partial O_x}{\partial x_1}=\left[\frac{1}{\rho_x}\left(\beta_{x_1}x_1^{\rho_x}+\beta_{x_2}x_2^{\rho_x}\right)^{\frac{1}{\rho_x}-1}\right]\cdot\left[\rho_x\beta_{x_1}x_1^{\rho_x-1}\right]$$

Which simplifies a little bit to:

$$\frac{\partial O_x}{\partial x_1}=\left[\left(\beta_{x_1}x_1^{\rho_x}+\beta_{x_2}x_2^{\rho_x}\right)^{\frac{1}{\rho_x}-1}\right]\cdot\left[\beta_{x_1}x_1^{\rho_x-1}\right]$$

What is the relationship between the marginal product of labor and the wage for the problem in the subnest? Let λ_x be the lagrange multiplier for the lagrange multiplier specific to the subnest:

$$p_{x_1}=\lambda_x\cdot\left(\frac{\partial O_x}{\partial x_1}\right)$$

This means that we have the following FOC from solving the expenditure minimization problem:

$$p_{x_1}=\lambda_x\cdot\left[\left(\beta_{x_1}x_1^{\rho_x}+\beta_{x_2}x_2^{\rho_x}\right)^{\frac{1}{\rho_x}-1}\right]\cdot\left[\beta_{x_1}x_1^{\rho_x-1}\right]$$

8.2 Basics

8.2.1 Log with Different Bases and Exponents

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.2.1.1 Log of Bases that Are not 10, 2 and e

What is y below, with arbitrary base x? It is $y = \frac{\log(z)}{\log(x)}$, because:

$$\begin{aligned} x^y &= z \\ x^{\frac{\log(z)}{\log(x)}} &= z \\ \log\left(x^{\frac{\log(z)}{\log(x)}}\right) &= \log(z) \\ \frac{\log(z)}{\log(x)}\log(x) &= \log(z) \\ \frac{\log(z)}{\log(x)} &= \frac{\log(z)}{\log(x)} \end{aligned}$$

Given these, we can compute the exponents, y, for non-standard bases, x, given the value for z.

```
# base 1.1
x <- 1.1
y <- 5.5
z <- x^y
# given z and knowing x, and what is y?
y_solved <- log(z) / log(x)
# dispaly
print(paste0("y_solved=", y_solved, ", y=", y))
```

```
## [1] "y_solved=5.5, y=5.5"
```

8.2.1.2 Rescale Bounded Model Parameters to Unconstrained with Exponentiation

We have a parameter to be estimated, the parameter's values can range between positive 1 and negative infinity. We want to use an estimator that us unconstrained. Use exponentiation to rescale the parameter so that is become unconstrained, use different bases so that the speed at which the parameter value approaches its bounds can be controlled.

While y is not bounded, $f(y;x)$ is bounded:

$$\begin{aligned} &f(y;x) = 1 - x^y \\ &\text{where } x > 1 \text{ and } -\infty < y < \infty \\ &\text{then, } 1 > f(y;x) > -\infty \end{aligned}$$

With $x > 1$, as y increases $f(y;x)$ decreases:

$$\frac{df(y;x)}{dy} = -x^y \log(x) < 0 \text{ when } x > 1$$

x controls the speed at which $f(y)$ approaches its bounds. In the simulation below, we try a number of different bases, at higher bases (2, e=2.71, 10), as y value changes $f(y)$ shifts too quickly to the bounds. But a base value of $x = 1.03$ or $x = 1.04$ would work well in an unbounded estimation routine that still generates parameters within bounds, which is below 1 in the case here.

```
# Vector of unbounded values, high and low
ar_y_vals <- sort(rnorm(20, 0, 20))
# Different base values
ar_bases <- c(1.01, 1.02, 1.03, 1.04, 1.1, 2, 2.71, 10)
# Transform back to f(y) scale with different bases
mt_f_of_y_vary_x <- matrix(NA,
  nrow = length(ar_y_vals),
  ncol = 1 + length(ar_bases)
)
ar_st_varnames <- c("yvalidx", "y_vals", paste0("base", ar_bases))
```

```
mt_f_of_y_vary_x[, 1] <- ar_y_vals
for (it_base in seq(1, length(ar_bases))) {
  fl_base <- ar_bases[it_base]
  ar_f_y <- 1 - fl_base^ar_y_vals
  mt_f_of_y_vary_x[, 1 + it_base] <- ar_f_y
}
# To tibble
tb_f_of_y_vary_x <- as_tibble(mt_f_of_y_vary_x) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~ c(ar_st_varnames))
# Print
kable(tb_f_of_y_vary_x) %>% kable_styling_fc_wide()
```

yvalidx	y_vals	base1.01	base1.02	base1.03	base1.04	base1.1	base2	base2.71	base10
1	-33.079054	0.2804631	0.4805850	0.6238537	0.7267544	0.9572664	1.000000e+00	1.000000e+00	1.000000e+00
2	-30.457963	0.2614502	0.4529131	0.5935527	0.6971698	0.9451391	1.000000e+00	1.000000e+00	1.000000e+00
3	-16.676929	0.1529038	0.2812538	0.3891782	0.4800804	0.7959685	9.999905e-01	9.999999e-01	1.000000e+00
4	-12.910841	0.1205575	0.2256014	0.3172517	0.3973221	0.7078636	9.998701e-01	9.999974e-01	1.000000e+00
5	-6.044928	0.0583758	0.1128183	0.1636272	0.2110769	0.4379380	9.848541e-01	9.975860e-01	9.999991e-01
6	-5.758399	0.0556874	0.1077701	0.1565135	0.2021610	0.4223771	9.815265e-01	9.967879e-01	9.999983e-01
7	-5.743401	0.0555464	0.1075051	0.1561395	0.2016916	0.4215508	9.813334e-01	9.967395e-01	9.999982e-01
8	-0.567698	0.0056329	0.0111790	0.0166405	0.0220195	0.0526696	3.253075e-01	4.321885e-01	7.294161e-01
9	2.454721	-0.0247260	-0.0498108	-0.0752558	-0.1010628	-0.2635939	-4.482073e+00	-1.055631e+01	-2.839191e+02
10	5.499020	-0.0562417	-0.1150453	-0.1765006	-0.2406997	-0.6889594	-4.422412e+01	-2.393848e+02	-3.155143e+05
11	5.956853	-0.0610645	-0.1252006	-0.1925304	-0.2631796	-0.7642906	-6.111426e+01	-3.784322e+02	-9.054242e+05
12	6.182477	-0.0634493	-0.1302392	-0.2005102	-0.2744073	-0.8026414	-7.162918e+01	-4.741414e+02	-1.522219e+06
13	6.743568	-0.0694032	-0.1428674	-0.2205869	-0.3027632	-0.9016667	-1.061560e+02	-8.303005e+02	-5.540748e+06
14	8.392309	-0.0870920	-0.1807971	-0.2815453	-0.3897895	-1.2252571	-3.349980e+02	-4.300421e+03	-2.467792e+08
15	8.537745	-0.0886663	-0.1842028	-0.2870664	-0.3977397	-1.2563175	-3.706357e+02	-4.971578e+03	-3.449415e+08
16	11.199607	-0.1178864	-0.2482988	-0.3924252	-0.5515533	-1.9079157	-2.350894e+03	-7.064474e+04	-1.583460e+11
17	12.805015	-0.1358873	-0.2886214	-0.4600941	-0.6523885	-2.3887066	-7.155385e+03	-3.500879e+05	-6.382858e+12
18	13.492957	-0.1436894	-0.3062964	-0.4900887	-0.6975795	-2.6183428	-1.152782e+04	-6.950808e+05	-3.111407e+13
19	23.309229	-0.2610372	-0.5865851	-0.9917087	-1.4947901	-8.2221374	-1.039387e+07	-1.236461e+10	-2.038118e+23
20	33.498931	-0.3956014	-0.9413175	-1.6917412	-2.7204773	-23.3562658	-1.213900e+10	-3.191600e+14	-3.154504e+33

8.2.1.3 Positive Exponents

Define exponents to consider and x-values to consider.

```
# positive value exponents
ar_exponents_posv <- c(0.05, 0.5, 1, 1.5)
# positive and negative values of the base
ar_baseval_pos <- seq(1e-10, 1.5, length.out = 1000)
# base to power
mt_x2a_val <- matrix(data = NA, nrow = length(ar_exponents_posv), ncol = length(ar_baseval_pos))
# Generate values
it_row_ctr <- 0
for (fl_exponents_posv in ar_exponents_posv) {
  it_row_ctr <- it_row_ctr + 1
  mt_x2a_val[it_row_ctr, ] <- ar_baseval_pos^fl_exponents_posv
}
```

Note that the smaller exponents functions are higher when $x < 1$, but lower when $x > 1$.

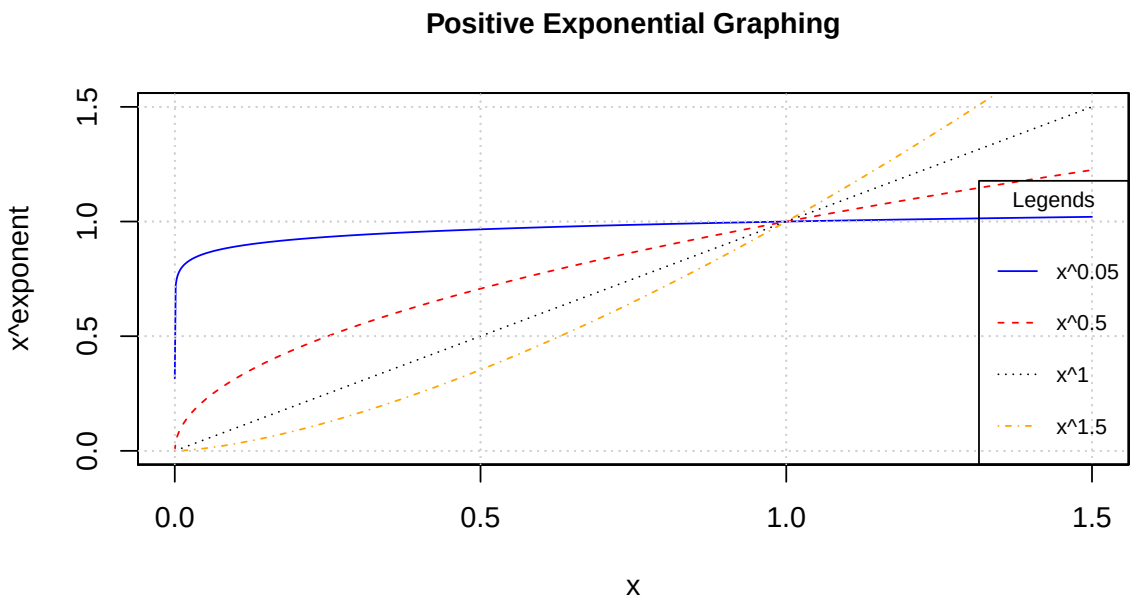
$$\text{if } b > a > 0, \text{ then, } (x^a - x^b) > 0, \text{ for all } 1 > x > 0$$

$$\text{if } b > a > 0, \text{ then, } (x^a - x^b) < 0, \text{ for all } x > 1$$

Note we also have: $\lim_{a \to 0} x^a = 1$ and $\lim_{a \to 1} x^a = x$ bounds. When $a > 1$, function becomes convex.

```
# x and bounds
ar_xlim <- c(min(ar_baseval_pos), max(ar_baseval_pos))
ar_ylim <- c(0, 1.5)
# function line
st_line_1_y_legend <- paste0("x^", ar_exponents_posv[1])
```

```
st_line_2_y_legend <- paste0("x^", ar_exponents_posv[2])
st_line_3_y_legend <- paste0("x^", ar_exponents_posv[3])
st_line_4_y_legend <- paste0("x^", ar_exponents_posv[4])
# Color and line
st_point_1_pch <- 10
st_point_1_cex <- 2
ar_colors <- c("blue", "red", "black", "orange")
ar_ltys <- c("solid", "dashed", "dotted", "dotdash")
# Graph and combine
for (it_graph in c(1, 2, 3, 4)) {
  if (it_graph != 1) {
    par(new = T)
  }
  ar_y_current <- mt_x2a_val[it_graph, ]
  plot(ar_baseval_pos, ar_y_current,
    type = "l",
    col = ar_colors[it_graph], lty = ar_ltys[it_graph],
    pch = 10, cex = 2, xlim = ar_xlim, ylim = ar_ylim, panel.first = grid(),
    ylab = "", xlab = "", yaxt = "n", xaxt = "n", ann = FALSE
  )
  plot_line <- recordPlot()
}
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg <- 1
fl_ces_fig_small <- 0.75
# R Legend
st_title <- paste0("Positive Exponential Graphing")
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_inequality.html"
)
st_x_label <- "x"
st_y_label <- "x^exponent"
title(
  main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
  cex.lab = fl_ces_fig_reg,
  cex.main = fl_ces_fig_reg,
  cex.sub = fl_ces_fig_small
)
axis(1, cex.axis = fl_ces_fig_reg)
axis(2, cex.axis = fl_ces_fig_reg)
grid()
# Legend sizing CEX
legend("bottomright",
  inset = c(0, 0),
  xpd = TRUE,
  c(st_line_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
  col = c(ar_colors[1], ar_colors[2], ar_colors[3], ar_colors[4]),
  cex = fl_ces_fig_small,
  lty = c(ar_ltys[1], ar_ltys[2], ar_ltys[3], ar_ltys[4]),
  title = "Legends",
  y.intersp = 2
)
```



https://fanwangecon.github.io/R4Econ/math/solutions/htmlpdfr/fs_inequality.html

8.2.1.4 Negative Exponents

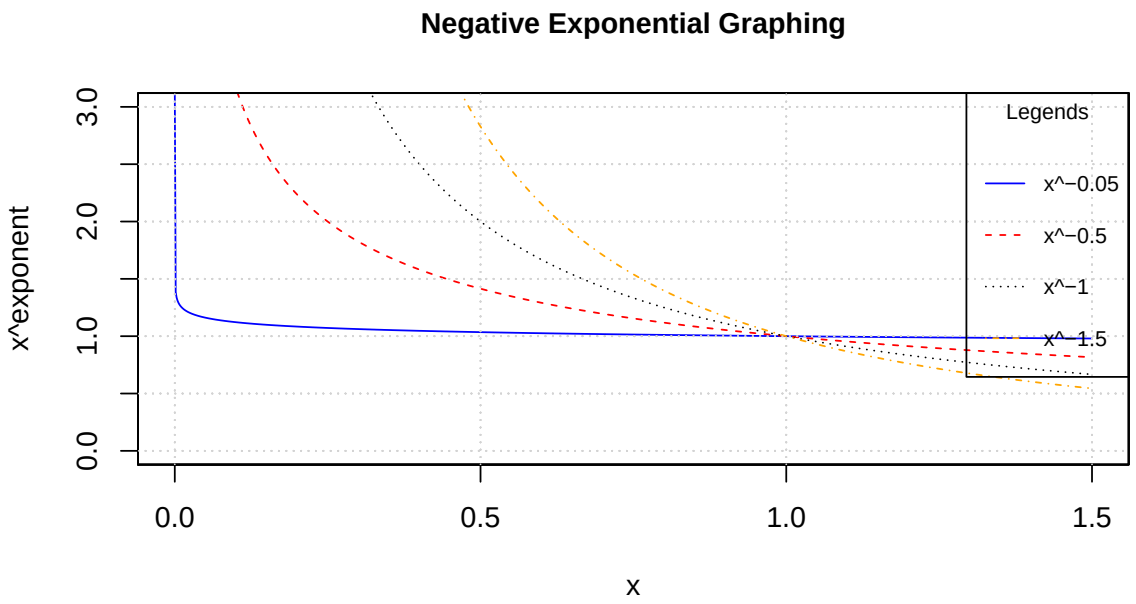
Similar to above, but now with negative exonents.

```
# positive value exponents
ar_exponents_posv <- -c(0.05, 0.5, 1, 1.5)
# positive and negative values of the base
ar_baseval_pos <- seq(1e-10, 1.5, length.out = 1000)
# base to power
mt_x2a_val <- matrix(data = NA, nrow = length(ar_exponents_posv), ncol = length(ar_baseval_pos))
# Generate values
it_row_ctr <- 0
for (fl_exponents_posv in ar_exponents_posv) {
  it_row_ctr <- it_row_ctr + 1
  mt_x2a_val[it_row_ctr, ] <- ar_baseval_pos^fl_exponents_posv
}
```

For positive exponents, when $x < 1$, $x^a < 1$, when $x > 1$, $x^a > 1$. For negative exponents, when $x < 1$, $x^a > 1$, and when $x > 1$, $x^a < 1$. Large positive exponents generate small values when $x < 1$, and large negative exponents generate very large values when $x < 1$.

```
# x and bounds
ar_xlim <- c(min(ar_baseval_pos), max(ar_baseval_pos))
ar_ylim <- c(0, 3)
# function line
st_line_1_y_legend <- paste0("x^", ar_exponents_posv[1])
st_line_2_y_legend <- paste0("x^", ar_exponents_posv[2])
st_line_3_y_legend <- paste0("x^", ar_exponents_posv[3])
st_line_4_y_legend <- paste0("x^", ar_exponents_posv[4])
# Color and line
st_point_1_pch <- 10
st_point_1_cex <- 2
ar_colors <- c("blue", "red", "black", "orange")
ar_ltys <- c("solid", "dashed", "dotted", "dotdash")
# Graph and combine
for (it_graph in c(1, 2, 3, 4)) {
  if (it_graph != 1) {
```

```
    par(new = T)
  }
  ar_y_current <- mt_x2a_val[it_graph, ]
  plot(ar_baseval_pos, ar_y_current,
    type = "l",
    col = ar_colors[it_graph], lty = ar_ltys[it_graph],
    pch = 10, cex = 2, xlim = ar_xlim, ylim = ar_ylim, panel.first = grid(),
    ylab = "", xlab = "", yaxt = "n", xaxt = "n", ann = FALSE
  )
  plot_line <- recordPlot()
}
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg <- 1
fl_ces_fig_small <- 0.75
# R Legend
st_title <- paste0("Negative Exponential Graphing")
st_subtitle <- paste0(
  "https://fanwangecon.github.io/",
  "R4Econ/math/solutions/htmlpdfr/fs_inequality.html"
)
st_x_label <- "x"
st_y_label <- "x^exponent"
title(
  main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
  cex.lab = fl_ces_fig_reg,
  cex.main = fl_ces_fig_reg,
  cex.sub = fl_ces_fig_small
)
axis(1, cex.axis = fl_ces_fig_reg)
axis(2, cex.axis = fl_ces_fig_reg)
grid()
# Legend sizing CEX
legend("topright",
  inset = c(0, 0),
  xpd = TRUE,
  c(st_line_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
  col = c(ar_colors[1], ar_colors[2], ar_colors[3], ar_colors[4]),
  cex = fl_ces_fig_small,
  lty = c(ar_ltys[1], ar_ltys[2], ar_ltys[3], ar_ltys[4]),
  title = "Legends",
  y.intersp = 2
)
```



https://fanwangecon.github.io/R4Econ/math/solutions/htmlpdfr/fs_inequality.html

8.2.1.5 Inequality and Exponents

Suppose we have the inequality $0 < a < b$, if we apply positive exponents to them, the direction of the inequality will stay the same: If $0 < a < b$, then $0 < a^{|\alpha|} < b^{|\alpha|}$ if $\alpha < 0$. Think about the graphs above, think of a and b as points along the x-axis, note that positive exponents are strictly increasing (although some concavely and some convexly) along the x-axis. Comparing x^α at $0 < b < a$ anywhere along the x-axis has still has $b^\alpha < a^\alpha$.

In contrast, if $0 < a < b$, then $a^{-|\alpha|} > b^{-|\alpha|} > 0$ if $\alpha < 0$. Sign flips. Visually from above, the sign-flipping happens because negative exponential is strictly decreasing along $x > 0$.

8.2.2 Rescale a Parameter with Fixed Min and Max

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.2.2.1 Using A Quadratic Function to Fit Three Points Uniquely

Given e < x < f, use f(x) to rescale x, such that f(e)=e, f(f)=f, but f(z)=$\alpha \cdot z$ for one particular z between e and f, where $\alpha > 1$. And in this case, assume that $\alpha \cdot z < f$. We can fit these three points using the Quadratic function uniquely. In another word, there is a unique quadratic function that crosses these three points. Note the quadratic function is either concave or convex through the entire domain.

Suppose that $e = 0$, $f = 10$, $z = 2$, and $\alpha = 1.5$. Using a quadratic to fit:

$$y(x) = a \cdot x^2 + b \cdot x + c$$

We have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c 2 \cdot 1.5 = a \cdot 2^2 + b \cdot 2 + c 10 = a \cdot 10^2 + b \cdot 10 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$3 = a \cdot 4 + b \cdot 210 = a \cdot 100 + b \cdot 10$$

Hence:

$$a = \frac{3 - 2b}{4} 10 = \frac{3 - 2b}{4} \cdot 100 + b \cdot 1010 = 75 - 50b + 10b$$

And finally:

$$a = \frac{3 - 2 * 1.625}{4} = -0.0625b = \frac{65}{40} = 1.625c = 0$$

Generate the a, b and c points above for the quadratic function:

```
# set values
e <- 0
f <- 10
z <- 2
alpha <- 1.5
# apply formulas from above
a <- -0.0625
b <- 1.625
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 11)
# rescale
ar_grid_quad <- a*ar_x^2 + b*ar_x + c
# show values
kable(print(as_tibble(cbind(ar_x, ar_grid_quad))),
      caption = paste0("Quadratic Fit of Three Equations and Three Unknowns\n",
                       "Satisfies: f(0)=0, f(10)=10, f(2)=3")) %>%
  kable_styling_fc()
```

Quadratic Fit of Three Equations and Three Unknowns Satisfies: f(0)=0, f(10)=10, f(2)=3

ar_x	ar_grid_quad
0	0.0000
1	1.5625
2	3.0000
3	4.3125
4	5.5000
5	6.5625
6	7.5000
7	8.3125
8	9.0000
9	9.5625
10	10.0000

We have three equations:

$$0 = a \cdot 0 + b \cdot 0 + c0.75 = a \cdot 0.5^2 + b \cdot 0.5 + c3.5 = a \cdot 3.5^2 + b \cdot 3.5 + c$$

Given these, we have, $c = 0$, and subsequently, 2 equations and 2 unknowns:

$$0.75 = a \cdot 0.25 + b \cdot 0.53.5 = a \cdot 12.25 + b \cdot 3.5$$

Hence:

$$a = \frac{0.75 - 0.5b}{0.25}3.5 = \frac{0.75 - 0.5b}{0.25} \cdot 12.25 + b \cdot 3.53.5 = 36.75 - 24.5b + 3.5b$$

And finally:

$$a = \frac{0.75 - 0.5 * 1.58333}{0.25} = -0.1666b = \frac{36.75 - 3.5}{24.5 - 3.5} = 1.58333c = 0$$

Generate the a, b and c points above for the quadratic function:

```
# set values
e <- 0
f <- 3.5
```

```
z <- 0.5
alpha <- 1.5
# apply formulas from above
a <- -0.16666666
b <- 1.583333333
c <- 0
# grid of values beween a and b, 11 points covering z = 2
ar_x <- seq(e, f, length.out = 100000)
# rescale
ar_grid_quad <- a*ar_x^2 + b*ar_x + c
# show values
# cbind(ar_x, ar_grid_quad)
ar_x[which.min(abs(ar_grid_quad - 0.75))]
```

```
## [1] 0.500015
```

8.2.3 Find Nearest

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.2.3.1 Neart Point Along a Line Through Orgin to Another Point

We first have X_1 an Y_1, given these, we are able to generate $R = \frac{Y_2}{X_2}$, a ratio. We want to iteratively update X_1 and Y_1, where 1 subscript indicates the first iteration, but we only know the ratio. Think of R as a line through the origin with R as the slope.

We generate X_2 and Y_2 by finding the point along the R as slope origin line that is the closest to the X_1 and Y_1. At the resulting point, R will be respected, and it will differ least in distance to the earlier iteration's X_1 and Y_1 points.

1. The slope of the diagonal line is $-\frac{X_2}{Y_2} = -\frac{1}{R}$
2. The diagonal line must cross X_1 and Y_1, solve for this line's y-intercept
3. Solve for the intersection of the diagonal line and the origin line with R as slope

Implementing step (2):

$$
\begin{aligned}
Y_1 &= I - \frac{1}{R} \cdot X_1 \\
I &= Y_1 + \frac{X_1}{R} \\
I &= \frac{Y_1 \cdot R + X_1}{R}
\end{aligned}
$$

Implementing step (3):

$$
\begin{aligned}
Y &= \frac{Y_1 \cdot R + X_1}{R} - \frac{1}{R} \cdot X \\
Y &= R \cdot X \\
R \cdot X &= \frac{Y_1 \cdot R + X_1}{R} - \frac{1}{R} \cdot X \\
\left(R + \frac{1}{R}\right) \cdot X &= \frac{Y_1 \cdot R + X_1}{R} \\
\frac{R^2 + 1}{R} \cdot X &= \frac{Y_1 \cdot R + X_1}{R} \\
X &= \frac{Y_1 \cdot R + X_1}{R} \cdot \frac{R}{R^2 + 1}
\end{aligned}
$$

And we have:

$$X = \frac{Y_1 \cdot R + X_1}{R^2 + 1}$$
$$Y = \frac{Y_1 \cdot R^2 + X_1 \cdot R}{R^2 + 1}$$

Visualize the results:

```
# Set random parameter Values for X1, Y1, and X2/Y2 ratio
set.seed(3)
fl_x1 <- runif(1) * 10
fl_y1 <- runif(1) * 10
fl_r <- runif(1) * 5

# Diaganol
fl_diag_slope <- -1 / fl_r
fl_diag_yintercept <- (fl_y1 * fl_r + fl_x1) / fl_r

# Closest point
fl_x2 <- (fl_y1 * fl_r + fl_x1) / (fl_r^2 + 1)
fl_y2 <- (fl_y1 * fl_r^2 + fl_x1 * fl_r) / (fl_r^2 + 1)

# Print state
print(paste("x1=", fl_x1, "x2=", fl_x2, "R=", fl_r, sep = " "))
```

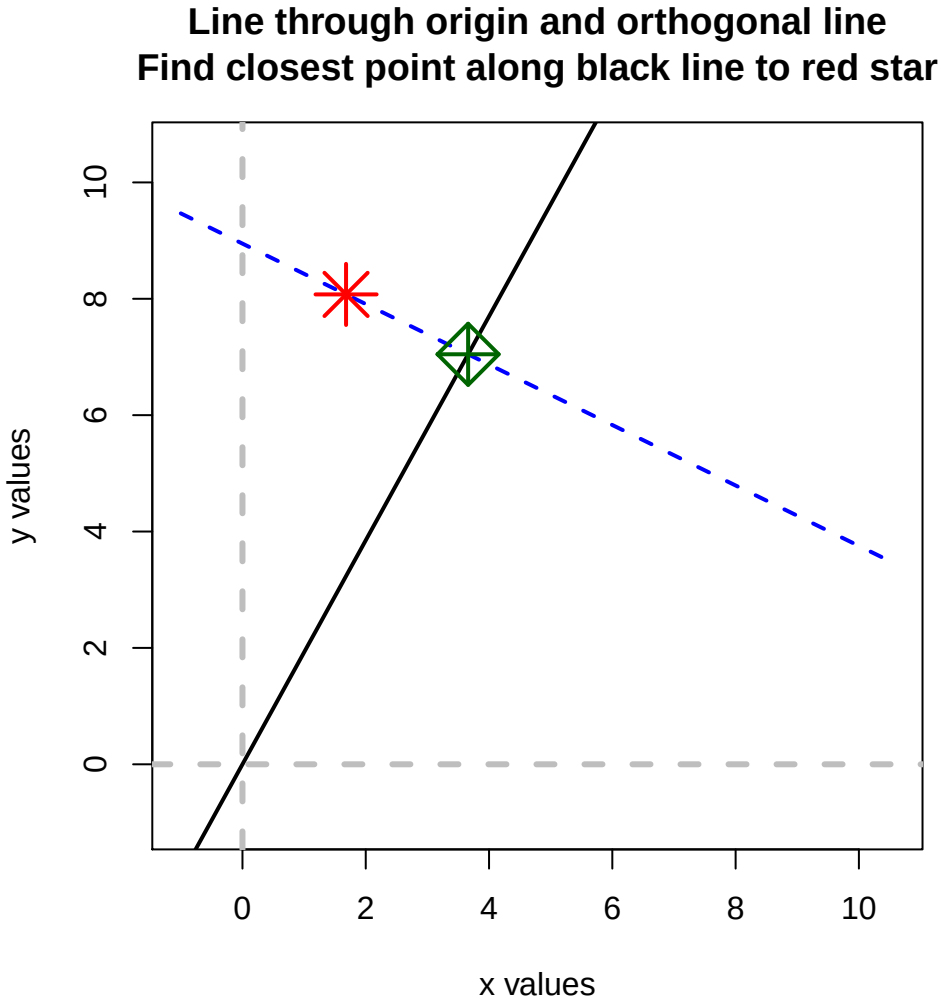
```
## [1] "x1= 1.68041526339948 x2= 3.6609038475849 R= 1.92471175687388"
```

```
print(paste("x2=", fl_x2, "y2=", fl_y2, sep = " "))
```

```
## [1] "x2= 3.6609038475849 y2= 7.04618467623146"
```

```
# X and y lims
ar_xylim <- c(-1, max(fl_x1, fl_y2) * 1.5)

# Visualize
par(mfrow = c(1, 1))
# Line through origin
curve(0 + fl_r * x, ar_xylim[1], ar_xylim[2],
  col = "black", lwd = 2, lty = 1,
  ylim = ar_xylim,
  ylab = "", xlab = ""
)
# Diaganol line
curve(fl_diag_yintercept + fl_diag_slope * x,
  add = TRUE,
  col = "blue", lwd = 2, lty = 2,
  ylim = ar_xylim,
  ylab = "", xlab = ""
)
# Point
points(fl_x1, fl_y1,
  add = TRUE,
  pch = 8, col = "red", cex = 3, lwd = 2,
  ylab = "", xlab = ""
)
points(fl_x2, fl_y2,
  add = TRUE,
  pch = 9, col = "darkgreen", cex = 3, lwd = 2,
  ylab = "", xlab = ""
)
```

```
# Origin lines
abline(
  v = 0, h = 0,
  col = "gray", lwd = 3, lty = 2
)

# Titles
title(
  main = paste0(
    "Line through origin and orthogonal line\n",
    "Find closest point along black line to red star"
  ),
  sub = paste0(
    "Black line goes through origin,",
    " blue line goes through (x1,y1) and (x2, y2)"
  ),
  xlab = "x values", ylab = "y values"
)
```



8.2.4 Linear Scalar f(x)=0 Solutions

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.2.4.1 Ratio

Here are some common ratios.

8.2.4.1.1 Unif Draw Min and Max Ratio We want to draw numbers such that we have some mean b, and that the possible maximum and minimum value drawn are at most a times apart. Given b and a, solve for x.

$$f(x) = \frac{b+x}{b-x} - a = 0$$

$$b \cdot a - x \cdot a = b + xb \cdot a - b = x + x \cdot ab\,(a-1) = x\,(a+1)\,x = \frac{b\,(a-1)}{a+1}$$

Uniformly draw

```
b <- 100
a <- 2
x <- (b*(a-1))/(a+1)
ar_unif_draws <- runif(100, min=b-x, max=b+x)
fl_max_min_ratio <- max(ar_unif_draws)/min(ar_unif_draws)
cat('fl_max_min_ratio =', fl_max_min_ratio, 'is close to a =', a, '\n')
```

```
## fl_max_min_ratio = 1.976291 is close to a = 2
```

8.3 Inequality Models

8.3.1 GINI Discrete Sample

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

This works out how the ff_dist_gini_vector_pos function works from Fan's *REconTools* Package.

8.3.1.1 GINI Formula for Discrete Sample

There is an vector values (all positive). This could be height information for N individuals. It could also be income information for N individuals. Calculate the GINI coefficient treating the given vector as population. This is not an estimation exercise where we want to estimate population GINI based on a sample. The given array is the population. The population is discrete, and only has these N individuals in the length n vector.

Note that when the sample size is small, there is a limit to inequality using the formula defined below given each N. So for small N, can not really compare inequality across arrays with different N, can only compare arrays with the same N.

The GINI formula used here is:

$$GINI = 1 - \frac{2}{N+1} \cdot \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right) \cdot \left(\sum_{i=1}^{N} x_i \right)^{-1}$$

Derive the formula in the steps below.

Step 1 Area Formula

$$\Gamma = \sum_{i=1}^{N} \frac{1}{N} \cdot \left(\sum_{j=1}^{i} \left(\frac{x_j}{\sum_{\hat{j}=1}^{N} x_{\hat{j}}} \right) \right)$$

Step 2 Total Area Given Perfect equality

With perfect equality $x_i = a$ for all i, so need to divide by that.

$$\Gamma^{\text{equal}} = \sum_{i=1}^{N} \frac{1}{N} \cdot \left(\sum_{j=1}^{i} \left(\frac{a}{\sum_{j=1}^{N} a} \right) \right) = \frac{N+1}{N} \cdot \frac{1}{2}$$

As the number of elements of the vecotr increases:

$$\lim_{N \to \infty} \Gamma^{\text{equal}} = \lim_{N \to \infty} \frac{N+1}{N} \cdot \frac{1}{2} = \frac{1}{2}$$

Step 3 Arriving at Finite Vector GINI Formula

Given what we have from above, we obtain the GINI formula, divide by total area below 45 degree line.

$$GINI = 1 - \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right) \cdot \left(N \cdot \sum_{i=1}^{N} x_i \right)^{-1} \cdot \left(\frac{N+1}{N} \cdot \frac{1}{2} \right)^{-1} = 1 - \frac{2}{N+1} \cdot \left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right) \cdot \left(\sum_{i=1}^{N} x_i \right)^{-1}$$

Step 4 Maximum Inequality given N

Suppose $x_i = 0$ for all $i < N$, then:

$$GINI^{x_i=0 \text{ except } i=N} = 1 - \frac{2}{N+1} \cdot X_N \cdot (X_N)^{-1} = 1 - \frac{2}{N+1}$$

$$\lim_{N \to \infty} GINI^{x_i=0 \text{ except } i=N} = 1 - \lim_{N \to \infty} \frac{2}{N+1} = 1$$

Note that for small N, for example if $N = 10$, even when one person holds all income, all others have 0 income, the formula will not produce GINI is zero, but that GINI is equal to $\frac{2}{11} \approx 0.1818$. If $N = 2$, inequality is at most, $\frac{2}{3} \approx 0.667$.

$$MostUnequalGINI\,(N) = 1 - \frac{2}{N+1} = \frac{N-1}{N+1}$$

8.3.1.1.1 Implement GINI Formula for Discrete Sample The **GINI** formula just derived is trivial to compute.

1. scalar: $\frac{2}{N+1}$
2. cumsum: $\sum_{j=1}^{i} x_j$
3. sum of cumsum: $\left(\sum_{i=1}^{N} \sum_{j=1}^{i} x_j \right)$
4. sum: $\sum_{i=1}^{N} X_i$

There are no package dependencies. Define the formula here:

```
# Formula, directly implement the GINI formula Following Step 4 above
ffi_dist_gini_vector_pos_test <- function(ar_pos) {
  # Check length and given warning
  it_n <- length(ar_pos)
  if (it_n <= 100)  warning('Data vector has n=',it_n,', max-inequality/max-gini=',(it_n-1)/(it_n +
  # Sort
  ar_pos <- sort(ar_pos)
  # formula implement
  fl_gini <- 1 - ((2/(it_n+1)) * sum(cumsum(ar_pos))*(sum(ar_pos))^(-1))
  return(fl_gini)
}
```

Generate a number of examples Arrays for testing

```
# Example Arrays of data
ar_equal_n1 = c(1)
ar_ineql_n1 = c(100)

ar_equal_n2 = c(1,1)
ar_ineql_alittle_n2 = c(1,2)
ar_ineql_somewht_n2 = c(1,2^3)
ar_ineql_alotine_n2 = c(1,2^5)
ar_ineql_veryvry_n2 = c(1,2^8)
ar_ineql_mostmst_n2 = c(1,2^13)

ar_equal_n10 = c(2,2,2,2,2,2, 2, 2, 2, 2)
ar_ineql_some_n10 = c(1,2,3,5,8,13,21,34,55,89)
ar_ineql_very_n10 = c(1,2^2,3^2,5^2,8^2,13^2,21^2,34^2,55^2,89^2)
ar_ineql_extr_n10 = c(1,2^2,3^3,5^4,8^5,13^6,21^7,34^8,55^9,89^10)
```

Now test the example arrays above using the function based no our formula:

```
##
## Small N=1 Hard-Code

## ar_equal_n1: 0

## ar_ineql_n1: 0

##
## Small N=2 Hard-Code, converge to 1/3, see formula above

## ar_ineql_alittle_n2: 0.1111111

## ar_ineql_somewht_n2: 0.2592593

## ar_ineql_alotine_n2: 0.3131313

## ar_ineql_veryvry_n2: 0.3307393

##
## Small N=10 Hard-Code, convege to 9/11=0.8181, see formula above

## ar_equal_n10: 0

## ar_ineql_some_n10: 0.5395514

## ar_ineql_very_n10: 0.7059554

## ar_ineql_extr_n10: 0.8181549
```

8.3.2 GINI Formula for Discrete Random Varialbe

For a discrete random variable, we are two arrays, an array of x values, and an array of $f(x)$ probability mass at each x value. Suppose the x values are unique/non-repeating. This is also Implemented in MEconTools with the ff_disc_rand_var_gini function.

Generate two arrays for x and $f(x)$, we will use the binomial distribution:

```
ar_choice_unique_sorted <- seq(0, 100, by=1)
ar_choice_prob <- dbinom(ar_choice_unique_sorted, 100, 0.01)
```

Generate mean and cumulative mean at each point:

```
# 1. to normalize, get mean (binomial so mean is p*N=50)
fl_mean <- sum(ar_choice_unique_sorted*ar_choice_prob);
# 2. get cumulative mean at each point
ar_mean_cumsum <- cumsum(ar_choice_unique_sorted*ar_choice_prob);
```

Normalizing and area calculation, following the same principle as above:

```
# 3. Share of wealth (income etc) accounted for up to this sorted type
ar_height <- ar_mean_cumsum/fl_mean;
# 4. The total area, is the each height times each width summed up
fl_area_drm <- sum(ar_choice_prob*ar_height);
```

Finally GINI coefficient:

```
# 5. area below 45 degree line might not be 1/2, depends on discretness
fl_area_below45 <- sum(ar_choice_prob*(cumsum(ar_choice_prob)/sum(ar_choice_prob)))

# 6. Gini is the distance between
fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45
print(paste0('fl_gini_index=', fl_gini_index))
```

```
## [1] "fl_gini_index=0.468573066002754"
```

8.3.2.1 Discrete Random Variable as Function

Organizing the code above as a function, and testing results out with the binomial distribution as an example.

For the binomial distribution, if the probability of success is very close to zero, that means nearly all mass is at lose all or nearly losing all. There will be non-zero but very small mass at higher levels of wins. Hence this should mean extreme inequality. GINI index should be close to 1. Alternatively, GINI index should be close to 0 when we have near 100 percent chance of success, then all mass is at winning all, perfect equality.

```
# Combining the code from above
ffi_dist_gini_random_var_pos_test <- function(ar_x_sorted, ar_prob_of_x) {
  #' @param ar_x_sorted sorted array of values
  #' @param ar_prob_of_x probability mass for each element along `ar_x_sorted`, sums to 1

  # 1. to normalize, get mean (binomial so mean is p*N=50)
  fl_mean <- sum(ar_x_sorted*ar_prob_of_x);
  # 2. get cumulative mean at each point
  ar_mean_cumsum <- cumsum(ar_x_sorted*ar_prob_of_x);
  # 3. Share of wealth (income etc) accounted for up to this sorted type
  ar_height <- ar_mean_cumsum/fl_mean;
  # 4. The total area, is the each height times each width summed up
  fl_area_drm <- sum(ar_prob_of_x*ar_height);
  # 5. area below 45 degree line might not be 1/2, depends on discretness
  fl_area_below45 <- sum(ar_prob_of_x*(cumsum(ar_prob_of_x)/sum(ar_prob_of_x)))
  # 6. Gini is the distance between
  fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45

  return(fl_gini_index)
}
```

Testing the function with the Binomial Distribution:

```
for (fl_binom_success_prob in seq(0.0001,0.9999,length.out=10)) {
  ar_x_sorted <- seq(0, 100, by=1)
  ar_prob_of_x <- dbinom(ar_x_sorted, 100, fl_binom_success_prob)
  fl_gini_index <- ffi_dist_gini_random_var_pos_test(ar_x_sorted, ar_prob_of_x)
  st_print <- paste0('binom p(success)=', fl_binom_success_prob ,
                     ', the fl_gini_index=', fl_gini_index)
  print(st_print)
}
```

```
## [1] "binom p(success)=1e-04, the fl_gini_index=0.990048846889393"
## [1] "binom p(success)=0.111188888888889, the fl_gini_index=0.147061101509638"
```

```
## [1] "binom p(success)=0.222277777777778, the fl_gini_index=0.0989942681255604"
## [1] "binom p(success)=0.333366666666667, the fl_gini_index=0.0753059593394789"
## [1] "binom p(success)=0.444455555555556, the fl_gini_index=0.0596495299535176"
## [1] "binom p(success)=0.555544444444444, the fl_gini_index=0.0476678493269222"
## [1] "binom p(success)=0.666633333333333, the fl_gini_index=0.0375168214334586"
## [1] "binom p(success)=0.777722222222222, the fl_gini_index=0.0280646430085938"
## [1] "binom p(success)=0.888811111111111, the fl_gini_index=0.0180312757603542"
## [1] "binom p(success)=0.9999, the fl_gini_index=9.90197372312816e-07"
```

8.3.2.2 Compare Discrete Sample and Discrete Random Variable Functions for GINI

ff_dist_gini_random_var provides the GINI implementation for a discrete random variable. The procedure is the same as prior, except now each element of the "x" array has element specific weights associated with it. The function can handle unsorted array with non-unique values.

Test and compare ff_dist_gini_random_var provides the GINI implementation for a discrete random variable and ff_dist_gini_vector_pos.

There is a vector of values from 1 to 100, in ascending order. What is the equal-weighted gini, the gini result when smaller numbers have higher weights, and when larger numbers have higher weights?

First, generate the relevant values.

```
# array
ar_x <- seq(1, 100, length.out = 30)
# prob array
ar_prob_x_unif <- rep.int(1, length(ar_x))/sum(rep.int(1, length(ar_x)))
# prob higher at lower values
ar_prob_x_lowval_highwgt <- rev(cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# prob higher at lower values
ar_prob_x_highval_highwgt <- (cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# show
kable(cbind(ar_x, ar_prob_x_unif, ar_prob_x_lowval_highwgt, ar_prob_x_highval_highwgt)) %>%
  kable_styling_fc()
```

Second, generate GINI values. What should happen?

1. The ff_dist_gini_random_var and ff_dist_gini_vector_pos results should be the same when the uniform distribution is used.
2. GINI should be higher, more inequality, if there is higher weights on the lower values.
3. GINI should be lower, more equality, if there is higher weight on the higher values.

```
ff_dist_gini_vector_pos(ar_x)
```

```
## [1] 0.3267327
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_unif)
```

```
## [1] 0.3267327
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_lowval_highwgt)
```

```
## [1] 0.4010343
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_highval_highwgt)
```

```
## [1] 0.1926849
```

8.3.3 Atkinson Inequality Index

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

ar_x	ar_prob_x_unif	ar_prob_x_lowval_highwgt	ar_prob_x_highval_highwgt
1.000000	0.0333333	0.0645161	0.0021505
4.413793	0.0333333	0.0623656	0.0043011
7.827586	0.0333333	0.0602151	0.0064516
11.241379	0.0333333	0.0580645	0.0086022
14.655172	0.0333333	0.0559140	0.0107527
18.068966	0.0333333	0.0537634	0.0129032
21.482759	0.0333333	0.0516129	0.0150538
24.896552	0.0333333	0.0494624	0.0172043
28.310345	0.0333333	0.0473118	0.0193548
31.724138	0.0333333	0.0451613	0.0215054
35.137931	0.0333333	0.0430108	0.0236559
38.551724	0.0333333	0.0408602	0.0258065
41.965517	0.0333333	0.0387097	0.0279570
45.379310	0.0333333	0.0365591	0.0301075
48.793103	0.0333333	0.0344086	0.0322581
52.206897	0.0333333	0.0322581	0.0344086
55.620690	0.0333333	0.0301075	0.0365591
59.034483	0.0333333	0.0279570	0.0387097
62.448276	0.0333333	0.0258065	0.0408602
65.862069	0.0333333	0.0236559	0.0430108
69.275862	0.0333333	0.0215054	0.0451613
72.689655	0.0333333	0.0193548	0.0473118
76.103448	0.0333333	0.0172043	0.0494624
79.517241	0.0333333	0.0150538	0.0516129
82.931034	0.0333333	0.0129032	0.0537634
86.344828	0.0333333	0.0107527	0.0559140
89.758621	0.0333333	0.0086022	0.0580645
93.172414	0.0333333	0.0064516	0.0602151
96.586207	0.0333333	0.0043011	0.0623656
100.000000	0.0333333	0.0021505	0.0645161

8.3.3.1 Atkinson Inequality Measures

Atkinson (JET, 1970) studies five standard inequality measures. Atkinson finds that given the same income data across countries, different inequality measure lead to different rankings of which country is more unequal. Atkinson develops an measure of inequality that changes depending on an inequality aversion parameter.

$$\text{Atkinson Inequality} = A\left(\{Y_i\}_{i=1}^{N}, \lambda\right) = 1 - \left(\sum_{i=1}^{N} \frac{1}{N}\left(\frac{Y_i}{\sum_{j=1}^{N}\left(\frac{Y_j}{N}\right)}\right)^{\lambda}\right)^{\frac{1}{\lambda}} \in [0,1]$$

$A\left(\{Y_i\}_{i=1}^{N}, \lambda\right)$ equals to zero is perfect equality. 1 is Perfect inequality. If $\lambda = 1$, the inequality measure is always equal to 0 because the planner does not care abouot inequality anymore.

8.3.3.2 Atkinson Inequality Function

Programming up the equation above, we have, given a sample of data:

```
# Formula
ffi_atkinson_ineq <- function(ar_data, fl_rho) {
  ar_data_demean <- ar_data/mean(ar_data)
  it_len <- length(ar_data_demean)
  fl_atkinson <- 1 - sum(ar_data_demean^{fl_rho}*(1/it_len))^(1/fl_rho)
  return(fl_atkinson)
}
```

When each element of the data array has weight, we have:

```
# Formula
ffi_atkinson_random_var_ineq <- function(ar_data, ar_prob_data, fl_rho) {
  #' @param ar_data array sorted array values
  #' @param ar_prob_data array probability mass for each element along `ar_data`, sums to 1
  #' @param fl_rho float inequality aversion parameter fl_rho = 1 for planner
  #' without inequality aversion. fl_rho = -infinity for fully inequality averse.

  fl_mean <- sum(ar_data*ar_prob_data);
  fl_atkinson <- 1 - (sum(ar_prob_data*(ar_data^{fl_rho}))^(1/fl_rho))/fl_mean
  return(fl_atkinson)
}
```

8.3.3.3 Atkinson Inequality Examples

Given a vector of observables, compute the Atkinson inequality measure given different inequality aversion.

8.3.3.3.1 Data Samples and Weighted Data

The ρ preference vector.

```
# Preference Vector
ar_rho <- 1 - (10^(c(seq(-2.0,2.0, length.out=30))))
ar_rho <- unique(ar_rho)
mt_rho <- matrix(ar_rho, nrow=length(ar_rho), ncol=1)
```

Sampled version for N sampled points, random, uniform and one-rich.

```
# Random normal Data Vector (not equal outcomes)
set.seed(123)
it_sample_N <- 30
fl_rnorm_mean <- 100
fl_rnorm_sd <- 6
ar_data_rnorm <- rnorm(it_sample_N, mean=fl_rnorm_mean, sd=fl_rnorm_sd)
# Uniform Data Vector (Equal)
ar_data_unif <- rep(1, length(ar_data_rnorm))

# One Rich (last person has income equal to the sum of all others*100)
ar_data_onerich <- rep(0.1, length(ar_data_rnorm))
ar_data_onerich[length(ar_data_onerich)] = sum(head(ar_data_onerich,-1))*10
```

Given the same distributions, random, uniform and one-rich, generate discrete random variable versions below. We approximate continuous normal with discrete binomial:

```
# Use binomial to approximate normal
fl_p_binom <- 1 - fl_rnorm_sd^2/fl_rnorm_mean
fl_n_binom <- round(fl_rnorm_mean^2/(fl_rnorm_mean - fl_rnorm_sd^2))
fl_binom_mean <- fl_n_binom*fl_p_binom
fl_binom_sd <- sqrt(fl_n_binom*fl_p_binom*(1-fl_p_binom))
# drv = discrete random variable
ar_drv_rbinom_xval <- seq(1, fl_n_binom)
ar_drv_rbinom_prob <- dbinom(ar_drv_rbinom_xval, size=fl_n_binom, prob=fl_p_binom)
# ignore weight at x=0
ar_drv_rbinom_prob <- ar_drv_rbinom_prob/sum(ar_drv_rbinom_prob)
```

Additionally, for the one-rich vector created earlier, now consider several probability mass over them, change the weight assigned to the richest person.

```
# This should be the same as the unweighted version
ar_drv_onerich_prob_unif <- rep(1/it_sample_N, it_sample_N)
# This puts almost no weight on the last rich person
```

```
# richlswgt = rich less weight
ar_drv_onerich_prob_richlswgt <- ar_drv_onerich_prob_unif
ar_drv_onerich_prob_richlswgt[it_sample_N] <- (1/it_sample_N)*0.1
ar_drv_onerich_prob_richlswgt <- ar_drv_onerich_prob_richlswgt/sum(ar_drv_onerich_prob_richlswgt)
# This puts more weight on the rich person
# richmrwgt = rich more weight
ar_drv_onerich_prob_richmrwgt <- ar_drv_onerich_prob_unif
ar_drv_onerich_prob_richmrwgt[it_sample_N] <- (1/it_sample_N)*10
ar_drv_onerich_prob_richmrwgt <- ar_drv_onerich_prob_richmrwgt/sum(ar_drv_onerich_prob_richmrwgt)
```

8.3.3.3.2 Testing Atkinson Index at Single rho Value Atkinson index with $\rho = -1$, which is a planner with some aversion towards inequality, this is equivalent to CRRA=2.

Testing with normal sample draw vs normal approximated with binomial discrete random variable:

```
# ATK = 0.05372126
ffi_atkinson_ineq(ar_data_rnorm, -1)
```

```
## [1] 0.00335581
```

```
# ATK = 0.03443246
ffi_atkinson_random_var_ineq(ar_drv_rbinom_xval, ar_drv_rbinom_prob, -1)
```

```
## [1] 0.003642523
```

```
# ATK = 0
ffi_atkinson_ineq(ar_data_unif, -1)
```

```
## [1] 0
```

Testing with

```
# ATK = 0.89, sample
ffi_atkinson_ineq(ar_data_onerich, -1)
```

```
## [1] 0.9027248
```

```
# ATK = 0.89, drv, uniform weight
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_unif, -1)
```

```
## [1] 0.9027248
```

```
# ATK = 0.49, drv, less weight on rich
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_richlswgt, -1)
```

```
## [1] 0.4965518
```

```
# ATK = 0.97, drv, more weight on rich
ffi_atkinson_random_var_ineq(ar_data_onerich, ar_drv_onerich_prob_richmrwgt, -1)
```

```
## [1] 0.9821147
```

Create vector of inequality aversion parameters and graph legends.

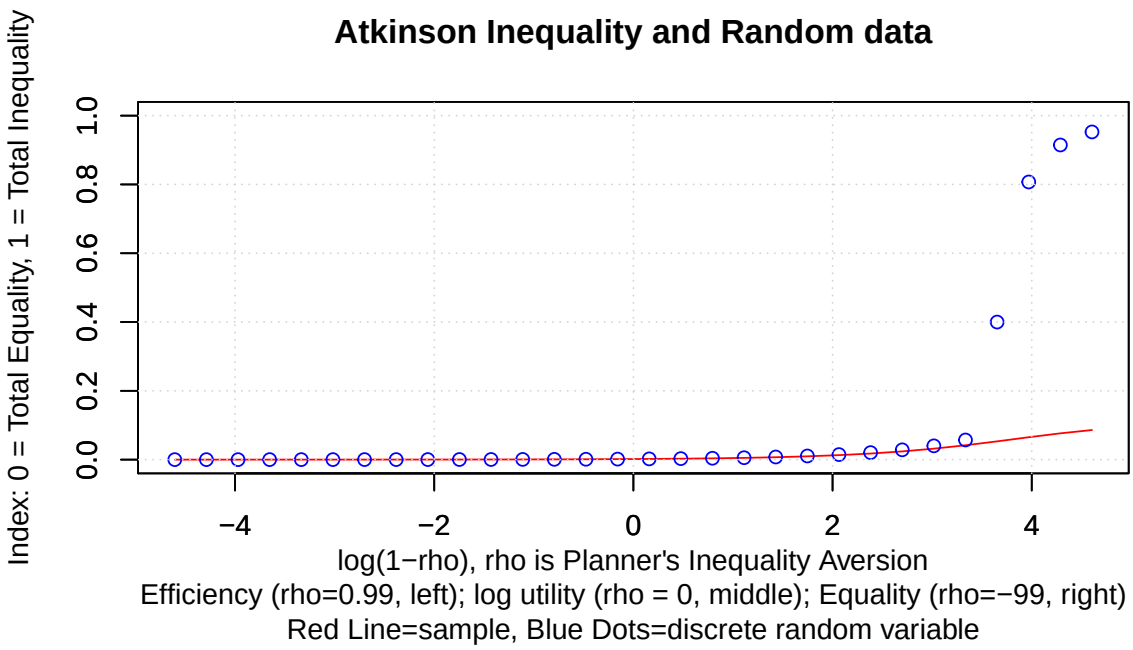
```
ar_log_1_minus_rho <- log(1-ar_rho)
st_x_label <- 'log(1-rho), rho is Planner\'s Inequality Aversion\nEfficiency (rho=0.99, left); log u
st_y_label <- 'Index: 0 = Total Equality, 1 = Total Inequality'
```

8.3.3.3.3 Atkinson Inequality and Normally Distributed Data How does Atkinson Inequality measure change with respect to a vector of normal random data as inequality aversion shifts? Note that in the example below, the bionomial approximated version is very similar to the normally drawn sample version. until rho beocmes very negative.

At very negative rho values, the binomial approximation has very tiny, but positive mass for all small values starting from 0, the normally drawn sample has no mass at those points. The Atkinson inequality

planner increasingly only cares about the individual with the lowest value from the binomial approximated version, and given the low value of those individuals compared to others, despite having no mass, the inequality index is almost 1.

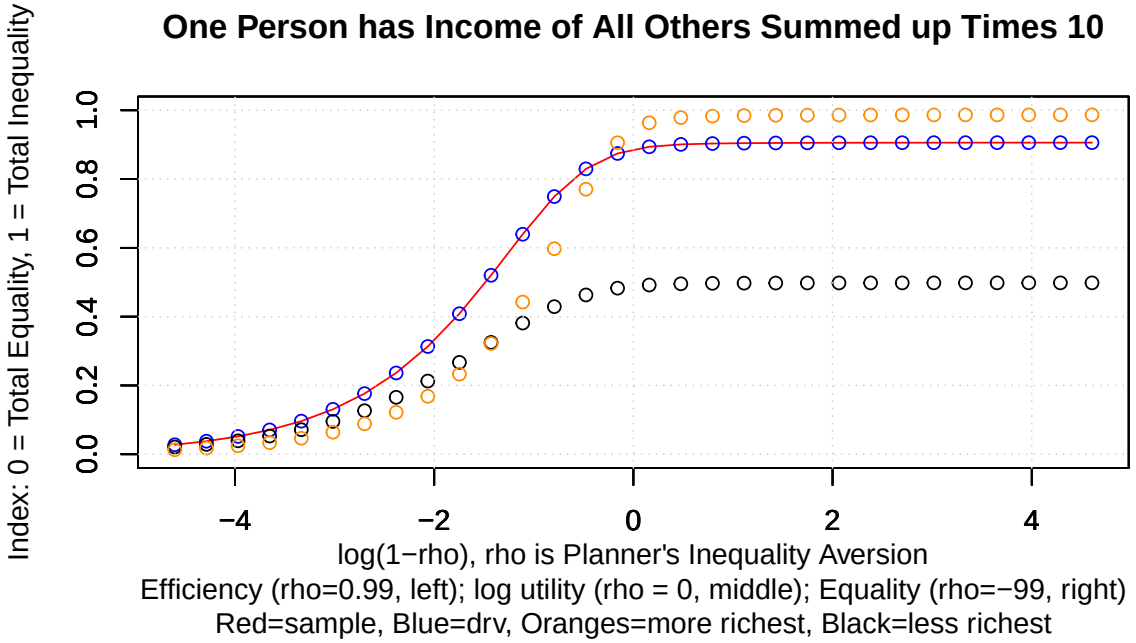
```
ar_ylim = c(0,1)
# First line
par(new=FALSE)
ar_atkinson_sample <- apply(mt_rho, 1, function(row){
  ffi_atkinson_ineq(ar_data_rnorm, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_sample,
     ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label,
     type="l", col = 'red')
# Second line
par(new=T)
ar_atkinson_drv <- apply(mt_rho, 1, function(row){
  ffi_atkinson_random_var_ineq(ar_drv_rbinom_xval, ar_drv_rbinom_prob, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'blue')
# Title
title(main = 'Atkinson Inequality and Random data',
      sub = 'Red Line=sample, Blue Dots=discrete random variable')
grid()
```



8.3.3.3.4 Atkinson Inequality with an Extremely Wealthy Individual Now with the one person has the wealth of all others in the vector times 10.

```
# First line
par(new=FALSE)
ar_atkinson <- apply(mt_rho, 1, function(row){ffi_atkinson_ineq(
  ar_data_onerich, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson,
     ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label,
     type="l", col = 'red')
# Second line
```

```
par(new=T)
ar_atkinson_drv <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_unif, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'blue')
# Third line
par(new=T)
ar_atkinson_drv_richlswgt <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_richlswgt, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv_richlswgt,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'black')
# Fourth line
par(new=T)
ar_atkinson_drv_richmrwgt <- apply(mt_rho, 1, function(row){ffi_atkinson_random_var_ineq(
  ar_data_onerich, ar_drv_onerich_prob_richmrwgt, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson_drv_richmrwgt,
     ylim = ar_ylim, xlab = '', ylab = '',
     type="p", col = 'darkorange')
# Title
title(main = 'One Person has Income of All Others Summed up Times 10',
      sub  = 'Red=sample, Blue=drv, Oranges=more richest, Black=less richest')
grid()
```



8.3.3.3.5 Atkinson Inequality with an Uniform Distribution

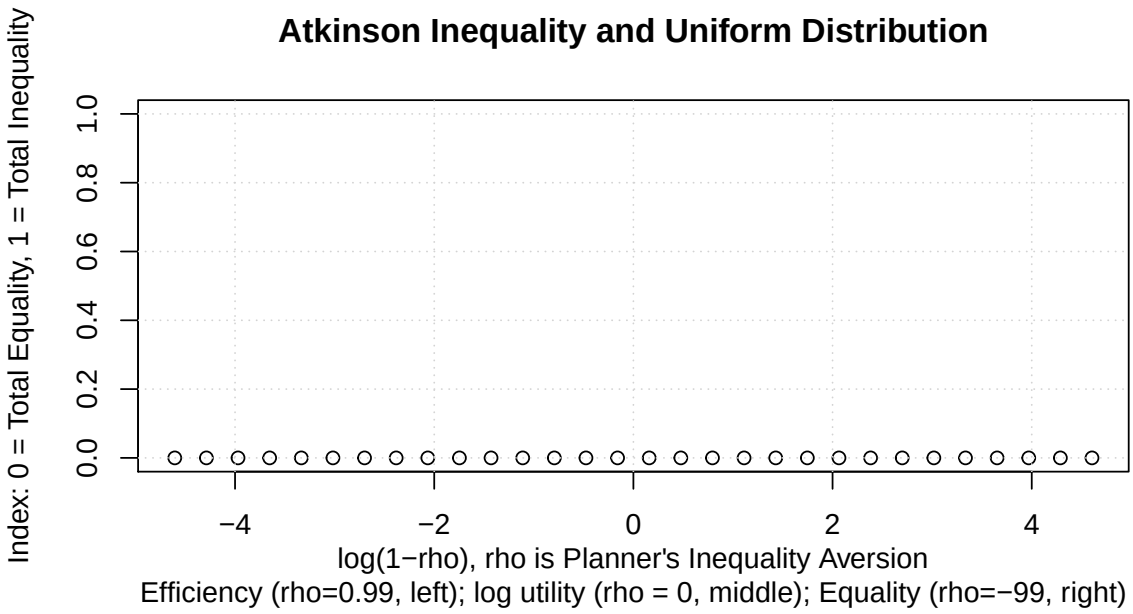
The Uniform Results, since allocations are uniform, zero for all.

```
par(new=FALSE)
ffi_atkinson_ineq(ar_data_unif, -1)
```

```
## [1] 0
```

```
ar_atkinson <- apply(mt_rho, 1, function(row){ffi_atkinson_ineq(ar_data_unif, row[1])})
plot(ar_log_1_minus_rho, ar_atkinson, ylim = ar_ylim, xlab = st_x_label, ylab = st_y_label)
```

```
title(main = 'Atkinson Inequality and Uniform Distribution')
grid()
```



8.3.3.4 Analyzing Equation Mechanics

How does the Aktinson Family utility function work? THe Atkinson Family Utility has the following functional form.

$$V^{\text{social}} = \left(\alpha \cdot A^{\lambda} + \beta \cdot B^{\lambda}\right)^{\frac{1}{\lambda}}$$

Several key issues here:

1. V^{social} is the utility of some social planner
2. A and B are allocations for Alex and Ben.
3. α and β are biases that a social planner has for Alex and Ben: $\alpha + \beta = 1$, $\alpha > 0$, and $\beta > 0$
4. $-\infty < \lambda \leq 1$ is a measure of inequality aversion
   - $\lambda = 1$ is when the planner cares about weighted total allocations (efficient, Utilitarian)
   - $\lambda = -\infty$ is when the planner cares about only the minimum between A and B allocations (equality, Rawlsian)

What if only care about Alex? Clearly, if the planner only cares about Ben, $\beta = 1$, then:

$$V^{\text{social}} = \left(B^{\lambda}\right)^{\frac{1}{\lambda}} = B$$

Clearly, regardless of the value of λ, as B increases V increases. What Happens to V when A or B increases? What is the derivative of V with respect to A or B?

$$\frac{\partial V}{\partial A} = \frac{1}{\lambda}\left(\alpha A^{\lambda} + \beta B^{\lambda}\right)^{\frac{1}{\lambda}-1} \cdot \lambda \alpha A^{\lambda-1}$$

$$\frac{\partial V}{\partial A} = \left(\alpha A^{\lambda} + \beta B^{\lambda}\right)^{\frac{1-\lambda}{\lambda}} \cdot \alpha A^{\lambda-1} > 0$$

Note that $\frac{\partial V}{\partial A} > 0$. When $\lambda < 0$, $Z^{\lambda} > 0$. For example $10^{-2} = \frac{1}{100}$. And For example $0.1^{\frac{3}{-2}} = \frac{1}{0.1^{1.5}}$. Still Positive.

While the overall V increases with increasing A, but if we did not have the outter power term, the situation is different. In particular, when $\lambda < 0$:

$$\text{if } \lambda < 0 \text{ then } \frac{d\left(\alpha A^{\lambda} + \beta B^{\lambda}\right)}{dA} = \alpha\lambda A^{\lambda-1} < 0$$

Without the outter $\frac{1}{\lambda}$ power, negative λ would lead to decreasing weighted sum. But:

$$\text{if } \lambda < 0 \text{ then } \frac{dG^{\frac{1}{\lambda}}}{dG} = \frac{1}{\lambda} \cdot G^{\frac{1-\lambda}{\lambda}} < 0$$

so when G is increasing and $\lambda < 0$, V would decrease. But when $G(A, B)$ is decreasing, as is the case with increasing A when $\lambda < 0$, V will actually increase. This confirms that $\frac{\partial V}{\partial A} > 0$ for $\lambda < 0$. The result is symmetric for $\lambda > 0$.

8.3.3.5 Indifference Curve Graph

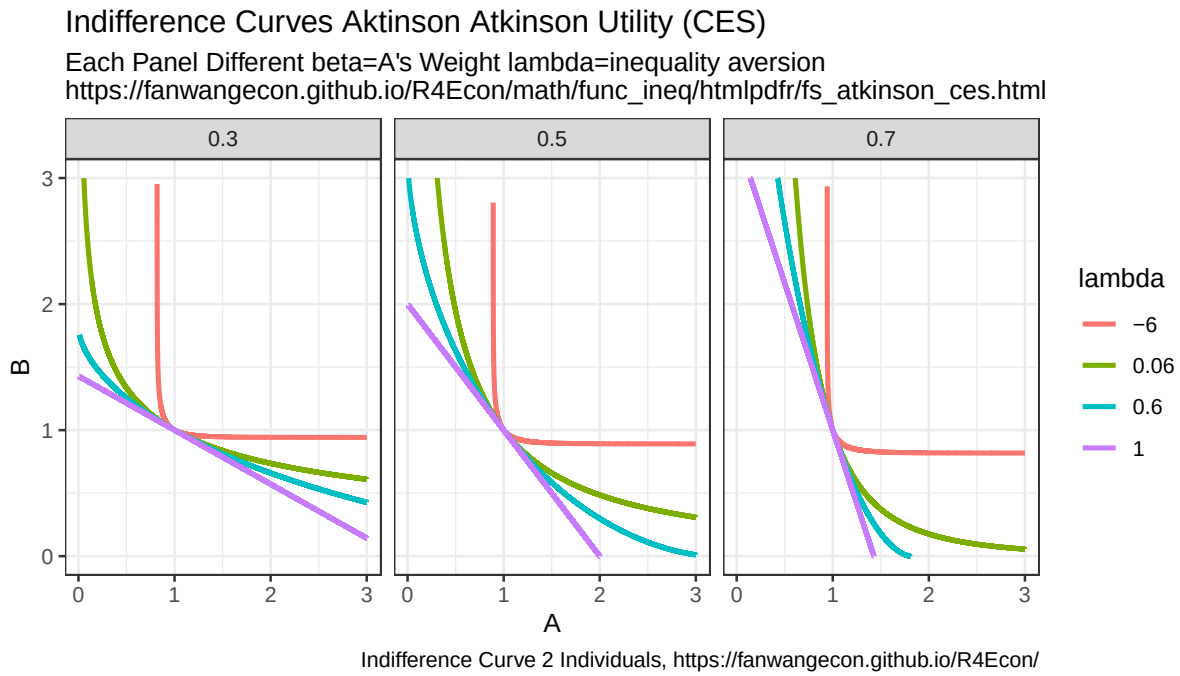
Given V^*, we can show the combinations of A and B points that provide the same utility. We want to be able to potentially draw multiple indifference curves at the same time. Note that indifference curves are defined by α, λ only. Each indifference curve is a set of A and B coordinates. So to generate multiple indifference curves means to generate many sets of A, B associated with different planner preferences, and then these could be graphed out.

```
# A as x-axis, need bounds on A
fl_A_min = 0.01
fl_A_max = 3
it_A_grid = 50000

# Define parameters
# ar_lambda <- 1 - (10^(c(seq(-2,2, length.out=3))))
ar_lambda <- c(1, 0.6, 0.06, -6)
ar_beta <- seq(0.25, 0.75, length.out = 3)
ar_beta <- c(0.3, 0.5, 0.7)
ar_v_star <- seq(1, 2, length.out = 1)
tb_pref <- as_tibble(cbind(ar_lambda)) %>%
  expand_grid(ar_beta) %>% expand_grid(ar_v_star) %>%
  rename_all(~c('lambda', 'beta', 'vstar')) %>%
  rowid_to_column(var = "indiff_id")

# Generate indifference points with apply and anonymous function
# tb_pref, whatever is selected from it, must be all numeric
# if there are strings, would cause conversion error.
ls_df_indiff <- apply(tb_pref, 1, function(x){
  indiff_id <- x[1]
  lambda <- x[2]
  beta <- x[3]
  vstar <- x[4]
  ar_fl_A_indiff <- seq(fl_A_min, fl_A_max, length.out=it_A_grid)
  ar_fl_B_indiff <- (((vstar^lambda) -
                        (beta*ar_fl_A_indiff^(lambda)))/(1-beta))^(1/lambda)
  mt_A_B_indiff <- cbind(indiff_id, lambda, beta, vstar,
                         ar_fl_A_indiff, ar_fl_B_indiff)
  colnames(mt_A_B_indiff) <- c('indiff_id', 'lambda', 'beta', 'vstar',
                               'indiff_A', 'indiff_B')
  tb_A_B_indiff <- as_tibble(mt_A_B_indiff) %>%
    rowid_to_column(var = "A_grid_id") %>%
    filter(indiff_B >= 0 & indiff_B <= max(ar_fl_A_indiff))
  return(tb_A_B_indiff)
```

```
})
df_indiff <- do.call(rbind, ls_df_indiff) %>% drop_na()
```

Note that many more A grid points are needed to fully plot out the leontief line.

```
# Labeling
st_title <- paste0('Indifference Curves Aktinson Atkinson Utility (CES)')
st_subtitle <- paste0('Each Panel Different beta=A\'s Weight lambda=inequality aversion\n',
                      'https://fanwangecon.github.io/',
                      'R4Econ/math/func_ineq/htmlpdfr/fs_atkinson_ces.html')
st_caption <- paste0('Indifference Curve 2 Individuals, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'A'
st_y_label <- 'B'

# Graphing
plt_indiff <-
  df_indiff %>% mutate(lambda = as_factor(lambda),
                       beta = as_factor(beta),
                       vstar = as_factor(vstar)) %>%
  ggplot(aes(x=indiff_A, y=indiff_B,
             colour=lambda)) +
  facet_wrap( ~ beta) +
  geom_line(size=1) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_indiff)
```



8.3.4 Location, Population, and Pollution

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

8.3.4.1 Simulate Population Distribution over Location and Demographics

Use the binomial distribution to generate heterogenous demographic break-down by location. There are N demographic cells, and the binomial distribution provides the probability mass in each of the N cell. Different bernoulli "win" chance for each location. There is also probability distribution over population in each location.

First, construct empty population share dataframe:

```
# 7 different age groups and 12 different locationso
it_N_pop_groups <- 7
it_M_location <- 12
# Matrix of demographics by location
mt_pop_data_frac <- matrix(data=NA, nrow=it_M_location, ncol=it_N_pop_groups)
colnames(mt_pop_data_frac) <- paste0('popgrp', seq(1,it_N_pop_groups))
rownames(mt_pop_data_frac) <- paste0('location', seq(1,it_M_location))
# Display
mt_pop_data_frac %>% kable() %>% kable_styling_fc()
```

	popgrp1	popgrp2	popgrp3	popgrp4	popgrp5	popgrp6	popgrp7
location1	NA	NA	NA	NA	NA	NA	NA
location2	NA	NA	NA	NA	NA	NA	NA
location3	NA	NA	NA	NA	NA	NA	NA
location4	NA	NA	NA	NA	NA	NA	NA
location5	NA	NA	NA	NA	NA	NA	NA
location6	NA	NA	NA	NA	NA	NA	NA
location7	NA	NA	NA	NA	NA	NA	NA
location8	NA	NA	NA	NA	NA	NA	NA
location9	NA	NA	NA	NA	NA	NA	NA
location10	NA	NA	NA	NA	NA	NA	NA
location11	NA	NA	NA	NA	NA	NA	NA
location12	NA	NA	NA	NA	NA	NA	NA

Second, generate conditional population distribution for each location, and then multiply by the share of population in each locality:

```
# Share of population per location
set.seed(123)
ar_p_loc <- dbinom(0:(3*it_M_location-1), 3*it_M_location-1, 0.5)
it_start <- length(ar_p_loc)/2-it_M_location/2
ar_p_loc <- ar_p_loc[it_start:(it_start+it_M_location+1)]
ar_p_loc <- ar_p_loc/sum(ar_p_loc)

# Different bernoulli "win" probability for each location
set.seed(234)
# ar_fl_unif_prob <- sort(runif(it_M_location)*(0.25)+0.4)
ar_fl_unif_prob <- sort(runif(it_M_location))

# Generate population proportion by locality
for (it_loc in 1:it_M_location ) {
  ar_p_pop_condi_loc <- dbinom(0:(it_N_pop_groups-1), it_N_pop_groups-1, ar_fl_unif_prob[it_loc])
  mt_pop_data_frac[it_loc,] <- ar_p_pop_condi_loc*ar_p_loc[it_loc]
}

# Sum of cells, should equal to 1
print(paste0('pop frac sum = ', sum(mt_pop_data_frac)))
```

```
## [1] "pop frac sum = 0.962953679726938"
```

```
# Display
round(mt_pop_data_frac*100, 2) %>%
  kable(caption='Share of population in each location and demographic cell') %>%
  kable_styling_fc()
```

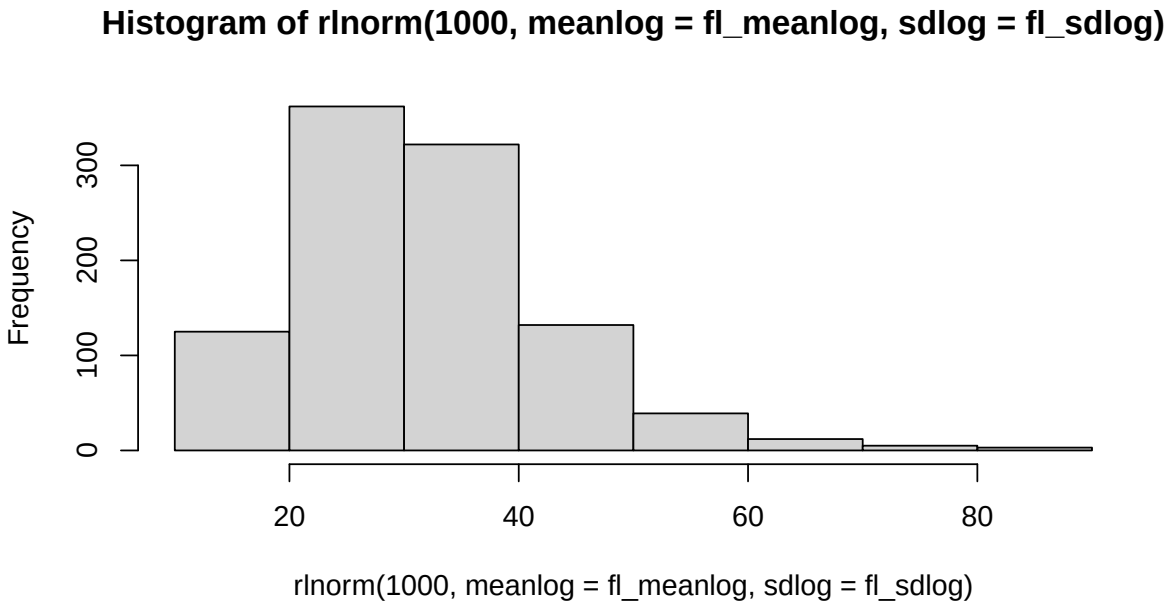
Share of population in each location and demographic cell

	popgrp1	popgrp2	popgrp3	popgrp4	popgrp5	popgrp6	popgrp7
location1	1.09	0.13	0.01	0.00	0.00	0.00	0.00
location2	1.63	0.70	0.13	0.01	0.00	0.00	0.00
location3	0.59	1.40	1.39	0.74	0.22	0.03	0.00
location4	0.06	0.43	1.29	2.09	1.90	0.92	0.19
location5	0.07	0.55	1.73	2.89	2.71	1.36	0.28
location6	0.02	0.26	1.19	2.89	3.93	2.85	0.86
location7	0.01	0.10	0.66	2.23	4.26	4.33	1.83
location8	0.00	0.06	0.47	1.83	4.03	4.72	2.31
location9	0.00	0.03	0.27	1.26	3.28	4.55	2.63
location10	0.00	0.02	0.20	0.96	2.57	3.68	2.19
location11	0.00	0.00	0.00	0.04	0.40	2.05	4.38
location12	0.00	0.00	0.00	0.02	0.24	1.28	2.82

8.3.4.2 Simulate Enviromental Exposure

Use log-normal distribution to describe average daily PM10 exposures distribution by locality:

```
fl_meanlog <- 3.4
fl_sdlog <- 0.35
hist(rlnorm(1000, meanlog = fl_meanlog, sdlog = fl_sdlog))
```



First, draw pollution measure for each locality:

```
# draw
set.seed(123)
ar_pollution_loc <- rlnorm(it_M_location, meanlog = fl_meanlog, sdlog = fl_sdlog)
# pollution dataframe
```

```
# 5 by 3 matrix

# Column Names
ar_st_varnames <- c('location','avgdailypm10')

# Combine to tibble, add name col1, col2, etc.
tb_loc_pollution <- as_tibble(ar_pollution_loc) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames)) %>%
  mutate(location = paste0('location', location))

# Display
kable(tb_loc_pollution) %>% kable_styling_fc()
```

location	avgdailypm10
location1	24.62676
location2	27.64481
location3	51.70466
location4	30.71275
location5	31.35114
location6	54.61304
location7	35.20967
location8	19.24456
location9	23.56121
location10	25.63653
location11	45.99021
location12	33.98553

Second, reshape population data:

```
# Reshape population data, so each observation is location/demo
df_pop_data_frac_long <- as_tibble(mt_pop_data_frac, rownames='location') %>%
  pivot_longer(cols = starts_with('popgrp'),
               names_to = c('popgrp'),
               names_pattern = paste0("popgrp(.*)"),
               values_to = "pop_frac")
```

Third, join with pollution data:

```
# Reshape population data, so each observation is location/demo
df_pop_pollution_long <- df_pop_data_frac_long %>%
  left_join(tb_loc_pollution, by='location')

# display
df_pop_pollution_long[1:round(it_N_pop_groups*2.5),] %>% kable() %>% kable_styling_fc()
```

8.3.4.3 Compute Demographic Group Specific Exposure Distributions

What is the p10, median, p90 and mean pollution exposure for each demographic group?

1. group by population group
2. sort by pollution exposure within group
3. generate population group specific conditional population weights
4. generate population CDF for each population group (sorted by pollution)

```
# Follow four steps above
df_pop_pollution_by_popgrp_cdf <- df_pop_pollution_long %>%
  arrange(popgrp, avgdailypm10) %>%
  group_by(popgrp) %>%
```

location	popgrp	pop_frac	avgdailypm10
location1	1	0.0109366	24.62676
location1	2	0.0013417	24.62676
location1	3	0.0000686	24.62676
location1	4	0.0000019	24.62676
location1	5	0.0000000	24.62676
location1	6	0.0000000	24.62676
location1	7	0.0000000	24.62676
location2	1	0.0163003	27.64481
location2	2	0.0070132	27.64481
location2	3	0.0012573	27.64481
location2	4	0.0001202	27.64481
location2	5	0.0000065	27.64481
location2	6	0.0000002	27.64481
location2	7	0.0000000	27.64481
location3	1	0.0058760	51.70466
location3	2	0.0140000	51.70466
location3	3	0.0138984	51.70466
location3	4	0.0073587	51.70466

```
  mutate(cdf_pop_condi_popgrp_sortpm10 = cumsum(pop_frac/sum(pop_frac)),
         pmf_pop_condi_popgrp_sortpm10 = (pop_frac/sum(pop_frac)))
# display
df_pop_pollution_by_popgrp_cdf[1:round(it_N_pop_groups*5.5),] %>%
  kable() %>% kable_styling_fc_wide()
```

8.3.4.4 Compute the Gini Index by Population Subgroup

The Gini index from fs_gini_disc.

```
ffi_dist_gini_random_var_pos_test <- function(ar_x_sorted, ar_prob_of_x) {
  fl_mean <- sum(ar_x_sorted*ar_prob_of_x);
  ar_mean_cumsum <- cumsum(ar_x_sorted*ar_prob_of_x);
  ar_height <- ar_mean_cumsum/fl_mean;
  fl_area_drm <- sum(ar_prob_of_x*ar_height);
  fl_area_below45 <- sum(ar_prob_of_x*(cumsum(ar_prob_of_x)/sum(ar_prob_of_x)))
  fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45
  return(fl_gini_index)
}
```

Compute Gini index for sub-group:

```
# Compute GINI by group
df_pop_pollu_gini <- df_pop_pollution_by_popgrp_cdf %>%
  group_by(popgrp) %>%
  do(popgrp_gini = ffi_dist_gini_random_var_pos_test(
    .$avgdailypm10, .$pmf_pop_condi_popgrp_sortpm10)) %>%
  unnest(c(popgrp_gini)) %>%
  left_join(df_pop_pollution_by_popgrp_cdf %>%
              group_by(popgrp) %>% slice(1L) %>%
              select(popgrp)
              , by="popgrp")
# Display
df_pop_pollu_gini %>% kable() %>% kable_styling_fc()
```

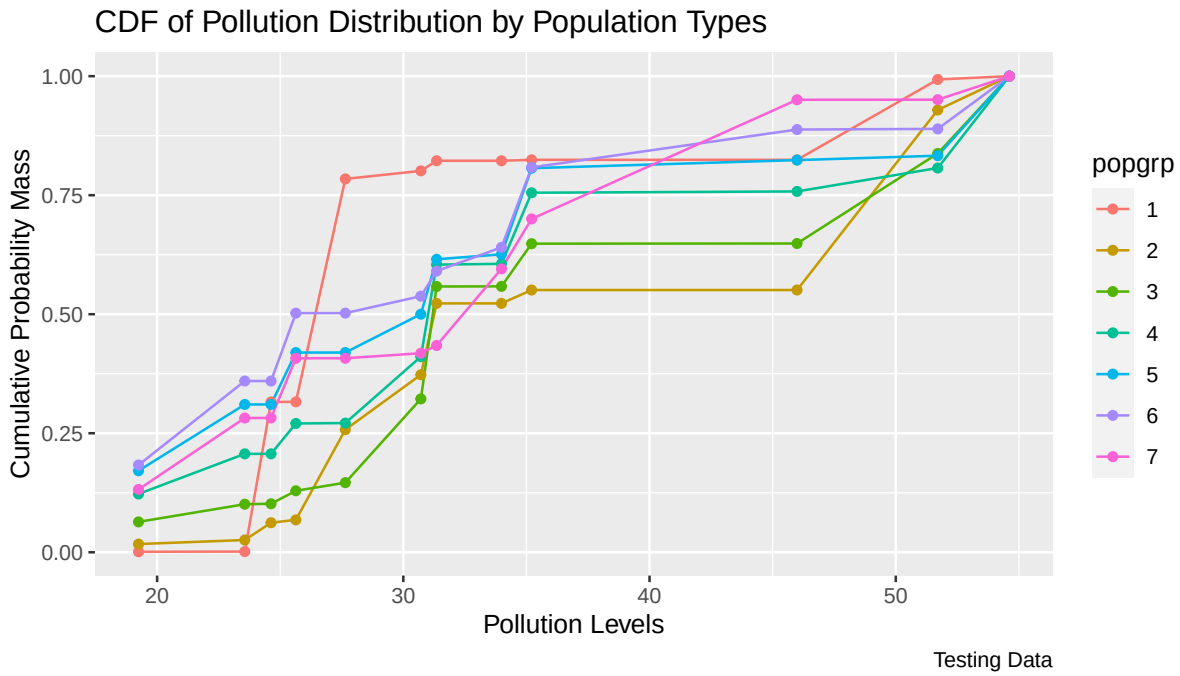
8.3.4.5 Visualize the Distributions

Visualizing distributions. Visualize the CDF:

location	popgrp	pop_frac	avgdailypm10	cdf_pop_condi_popgrp_sortpm10	pmf_pop_condi_popgrp_sortpm10
location8	1	0.0000364	19.24456	0.0010453	0.0010453
location9	1	0.0000151	23.56121	0.0014804	0.0004351
location1	1	0.0109366	24.62676	0.3156484	0.3141680
location10	1	0.0000104	25.63653	0.3159471	0.0002988
location2	1	0.0163003	27.64481	0.7841942	0.4682471
location4	1	0.0005879	30.71275	0.8010816	0.0168874
location5	1	0.0007392	31.35114	0.8223166	0.0212350
location12	1	0.0000000	33.98553	0.8223168	0.0000002
location7	1	0.0000681	35.20967	0.8242718	0.0019550
location11	1	0.0000000	45.99021	0.8242721	0.0000003
location3	1	0.0058760	51.70466	0.9930669	0.1687948
location6	1	0.0002413	54.61304	1.0000000	0.0069331
location8	2	0.0006400	19.24456	0.0172871	0.0172871
location9	2	0.0003150	23.56121	0.0257947	0.0085076
location1	2	0.0013417	24.62676	0.0620374	0.0362427
location10	2	0.0002235	25.63653	0.0680736	0.0060362
location2	2	0.0070132	27.64481	0.2575157	0.1894421
location4	2	0.0042712	30.71275	0.3728918	0.1153760
location5	2	0.0055479	31.35114	0.5227547	0.1498629
location12	2	0.0000004	33.98553	0.5227662	0.0000116
location7	2	0.0010378	35.20967	0.5508009	0.0280347
location11	2	0.0000008	45.99021	0.5508213	0.0000203
location3	2	0.0140000	51.70466	0.9289930	0.3781718
location6	2	0.0026287	54.61304	1.0000000	0.0710070
location8	3	0.0046896	19.24456	0.0638166	0.0638166
location9	3	0.0027290	23.56121	0.1009539	0.0371373
location1	3	0.0000686	24.62676	0.1018872	0.0009333
location10	3	0.0020006	25.63653	0.1291118	0.0272246
location2	3	0.0012573	27.64481	0.1462207	0.0171089
location4	3	0.0129304	30.71275	0.3221799	0.1759592
location5	3	0.0173492	31.35114	0.5582709	0.2360910
location12	3	0.0000141	33.98553	0.5584625	0.0001916
location7	3	0.0065945	35.20967	0.6482016	0.0897391
location11	3	0.0000242	45.99021	0.6485305	0.0003290
location3	3	0.0138984	51.70466	0.8376617	0.1891312
location6	3	0.0119295	54.61304	1.0000000	0.1623383
location8	4	0.0183277	19.24456	0.1224562	0.1224562
location9	4	0.0126118	23.56121	0.2067219	0.0842656

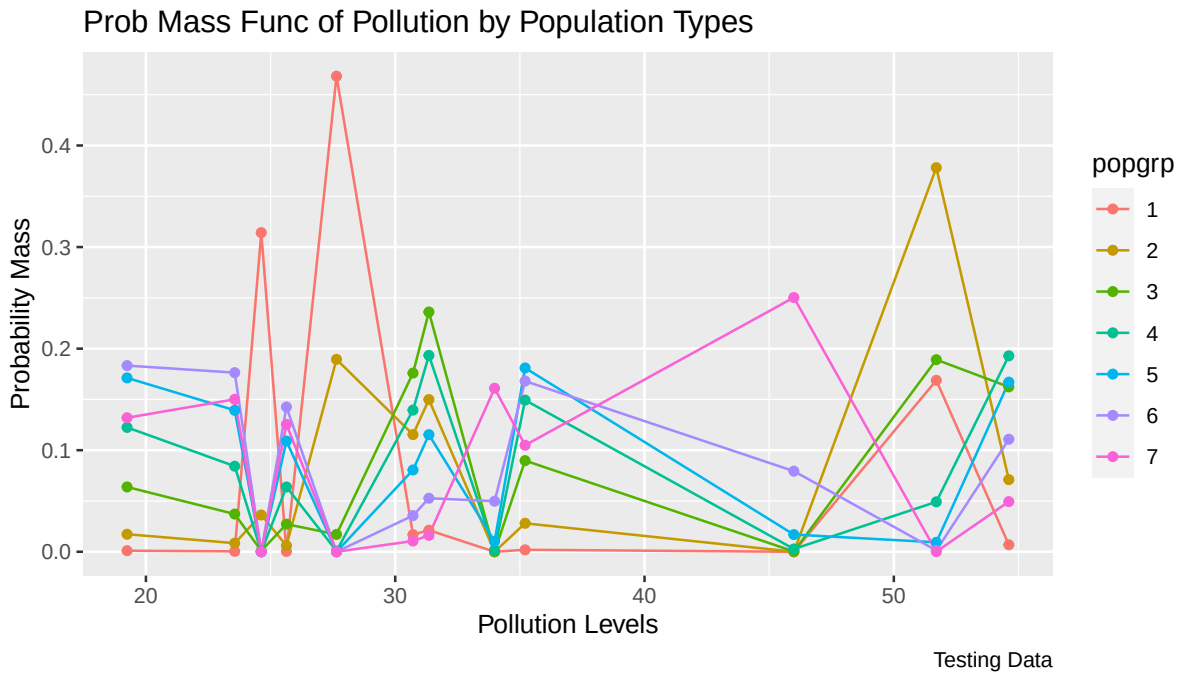
popgrp	popgrp_gini
1	0.1206861
2	0.1075096
3	0.1380319
4	0.1543002
5	0.1680265
6	0.1754309
7	0.1453136

```
# Visaulize Distributions, CDF for different population groups
lineplot <- df_pop_pollution_by_popgrp_cdf %>%
    select(popgrp, avgdailypm10, cdf_pop_condi_popgrp_sortpm10 ) %>%
    ggplot(aes(x=avgdailypm10, y=cdf_pop_condi_popgrp_sortpm10,
               colour=popgrp)) +
        geom_line() +
        geom_point() +
        labs(title = 'CDF of Pollution Distribution by Population Types',
             x = 'Pollution Levels',
             y = 'Cumulative Probability Mass',
             caption = 'Testing Data')
print(lineplot)
```



Visualize the Probability Mass (real data should look much less chaotic than this):

```
# Visaulize Distributions, CDF for different population groups
lineplot_pmf <- df_pop_pollution_by_popgrp_cdf %>%
    select(popgrp, avgdailypm10, pmf_pop_condi_popgrp_sortpm10 ) %>%
    ggplot(aes(x=avgdailypm10, y=pmf_pop_condi_popgrp_sortpm10,
               colour=popgrp)) +
        geom_line() +
        geom_point() +
        labs(title = 'Prob Mass Func of Pollution by Population Types',
             x = 'Pollution Levels',
             y = 'Probability Mass',
             caption = 'Testing Data')
print(lineplot_pmf)
```



8.3.4.6 Various Quantiles

Measure quantiles of pollution exposures for different population groups:

1. Consider CDF larger than current quantile of interest.
2. Slice group-specific CDF that is higher and closest to quantile of interest.
3. Merge results for different quantiles together.

```
# Generate pollution quantiles by population groups
df_pop_pollution_distribution <- df_pop_pollution_by_popgrp_cdf %>%
  mutate(pm10_mean = weighted.mean(avgdailypm10, pop_frac)) %>%
  filter(cdf_pop_condi_popgrp_sortpm10 >= 0.10) %>%
  slice(1) %>%
  mutate(pm10_p10 = avgdailypm10) %>%
  select(popgrp, pm10_mean, pm10_p10) %>%
  left_join(df_pop_pollu_gini, by='popgrp') %>%
  left_join(df_pop_pollution_by_popgrp_cdf %>%
              filter(cdf_pop_condi_popgrp_sortpm10 >= 0.20) %>%
              slice(1) %>%
              mutate(pm10_p20 = avgdailypm10) %>%
              select(popgrp, pm10_p20),
            by='popgrp') %>%
  left_join(df_pop_pollution_by_popgrp_cdf %>%
              filter(cdf_pop_condi_popgrp_sortpm10 >= 0.50) %>%
              slice(1) %>%
              mutate(pm10_p50 = avgdailypm10) %>%
              select(popgrp, pm10_p50),
            by='popgrp') %>%
  left_join(df_pop_pollution_by_popgrp_cdf %>%
              filter(cdf_pop_condi_popgrp_sortpm10 >= 0.80) %>%
              slice(1) %>%
              mutate(pm10_p80 = avgdailypm10) %>%
              select(popgrp, pm10_p80),
            by='popgrp') %>%
  left_join(df_pop_pollution_by_popgrp_cdf %>%
              filter(cdf_pop_condi_popgrp_sortpm10 >= 0.90) %>%
              slice(1) %>%
```

```
              mutate(pm10_p90 = avgdailypm10) %>%
              select(popgrp, pm10_p90),
            by='popgrp') %>%
  select(popgrp, pm10_mean, popgrp_gini, everything())
# display
df_pop_pollution_distribution %>%
  kable(caption = 'PM10 Exposure Distribution by Population Groups') %>%
  kable_styling_fc()
```

PM10 Exposure Distribution by Population Groups

popgrp	pm10_mean	popgrp_gini	pm10_p10	pm10_p20	pm10_p50	pm10_p80	pm10_p90
1	31.07894	0.1206861	24.62676	24.62676	27.64481	30.71275	51.70466
2	39.47897	0.1075096	27.64481	27.64481	31.35114	51.70466	51.70466
3	37.92901	0.1380319	23.56121	30.71275	31.35114	51.70466	54.61304
4	34.86470	0.1543002	19.24456	23.56121	31.35114	51.70466	54.61304
5	32.56731	0.1680265	19.24456	23.56121	30.71275	35.20967	54.61304
6	31.46626	0.1754309	19.24456	23.56121	25.63653	35.20967	54.61304
7	33.50541	0.1453136	19.24456	23.56121	33.98553	45.99021	45.99021

Chapter 9

Statistics

9.1 Distributions

9.1.1 Integrate Over Normal Guassian Process Shock

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Some Common parameters

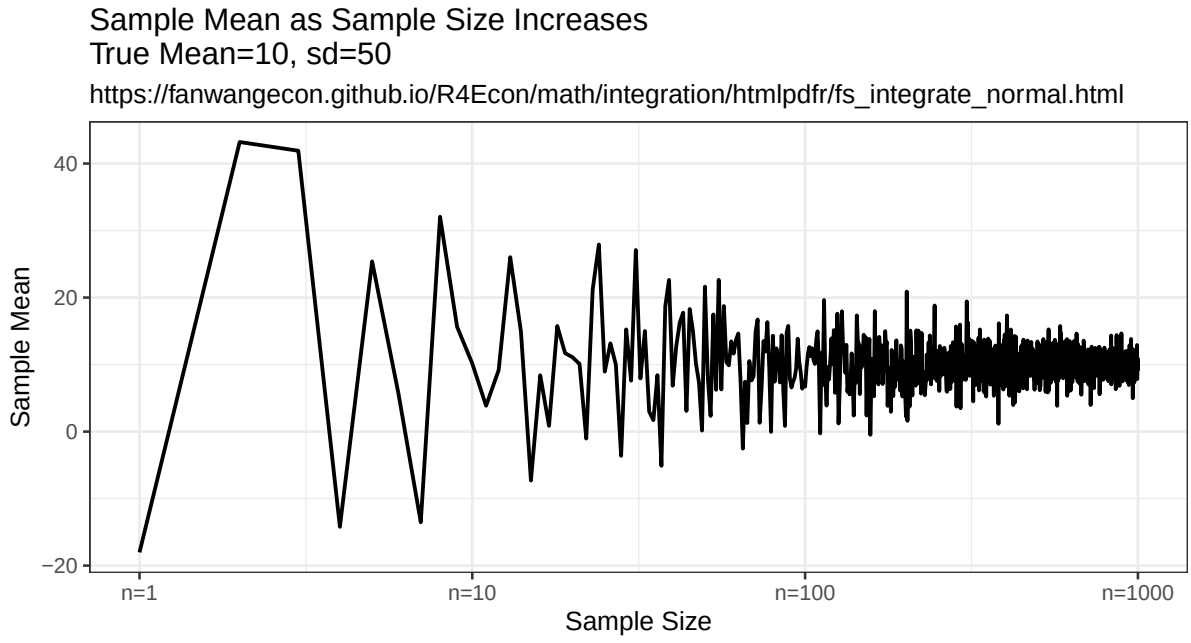
```
fl_eps_mean = 10
fl_eps_sd = 50
fl_cdf_min = 0.000001
fl_cdf_max = 0.999999
ar_it_draws <- seq(1, 1000)
```

9.1.1.1 Randomly Sample and Integrate (Monte Carlo Integration)

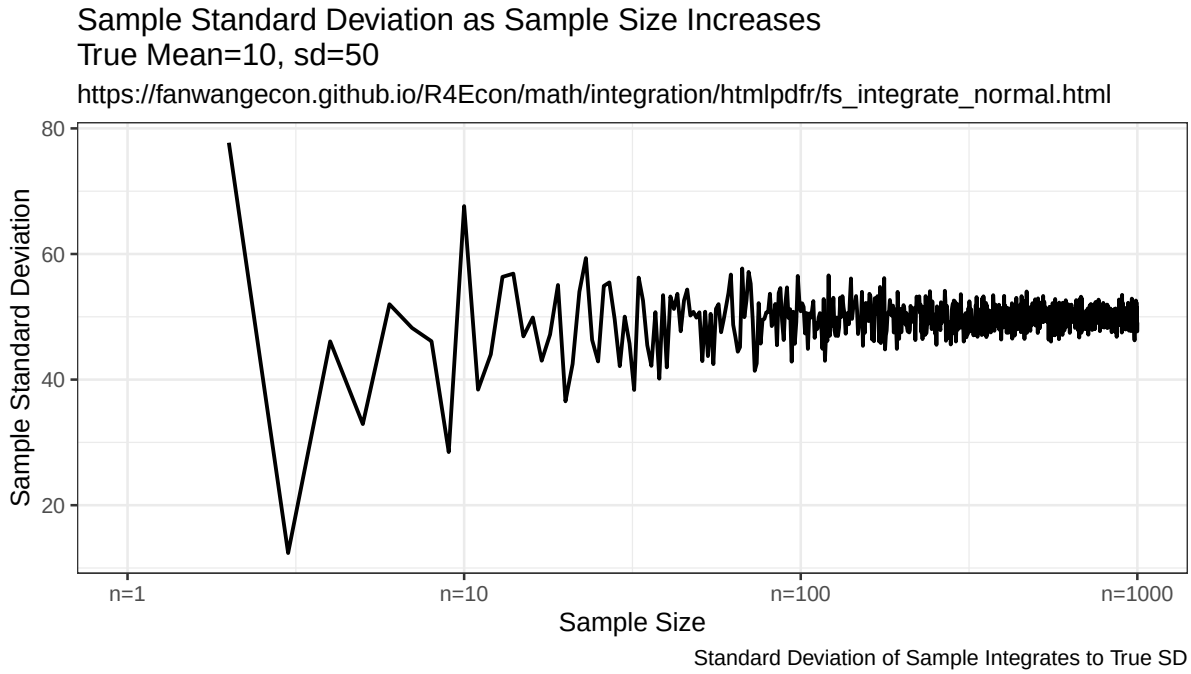
Compare randomly drawn normal shock mean and known mean. How does simulated mean change with draws. Actual integral equals to 10, as sample size increases, the sample mean approaches the integration results, but this is expensive, even with ten thousand draws, not very exact.

```
# Simulate Draws
set.seed(123)
ar_fl_means <-
  sapply(ar_it_draws, function(x)
    return(mean(rnorm(x[1], mean=fl_eps_mean, sd=fl_eps_sd))))
ar_fl_sd <-
  sapply(ar_it_draws, function(x)
    return(sd(rnorm(x[1], mean=fl_eps_mean, sd=fl_eps_sd))))

mt_sample_means <- cbind(ar_it_draws, ar_fl_means, ar_fl_sd)
colnames(mt_sample_means) <- c('draw_count', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)

# Shared Subtitle
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/math/integration/htmlpdfr/fs_integrate_normal.html')
```

```
# Shared Labels
slb_title_shr = paste0('as Sample Size Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Sample Size')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Sample Mean ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sample Mean',
       caption = 'Mean of Sample Integrates to True Mean') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```




```
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Sample Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sample Standard Deviation',
       caption = 'Standard Deviation of Sample Integrates to True SD') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_sd)
```



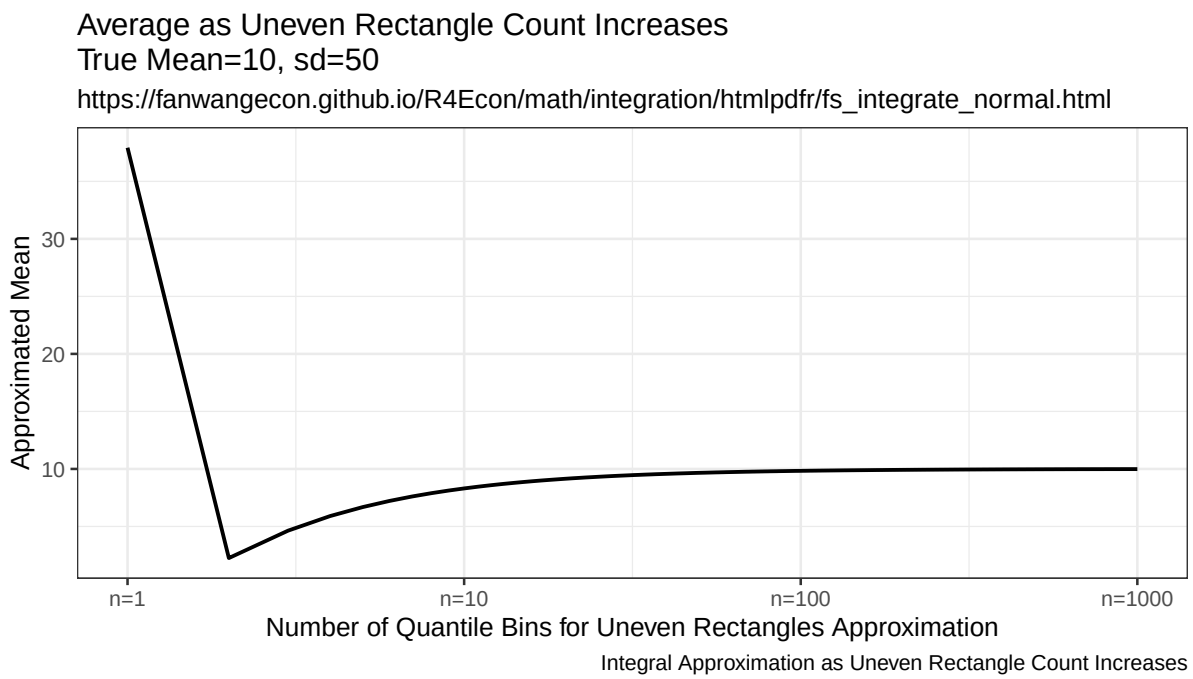
9.1.1.2 Integration By Symmetric Uneven Rectangle

Draw on even grid from close to 0 to close to 1. Get the corresponding x points to these quantile levels. Distance between x points are not equi-distance but increasing and symmetric away from the mean. Under this approach, each rectangle aims to approximate the same area.

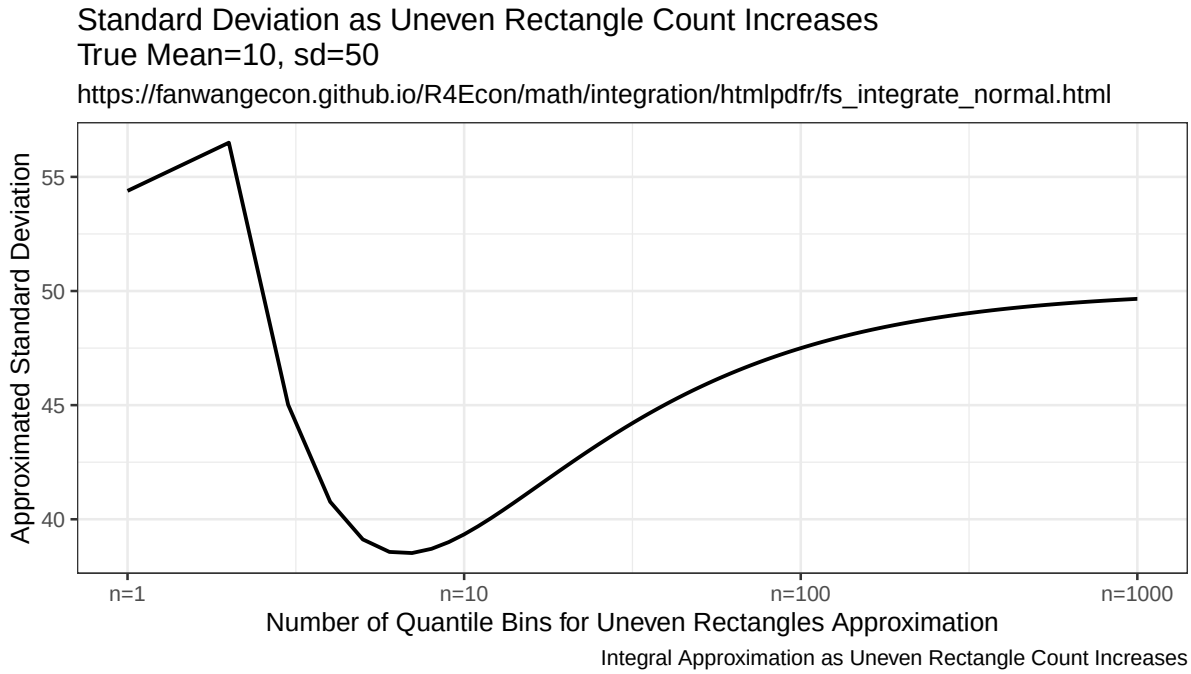
Resulting integration is rectangle based, but rectangle width differ. The rectangles have wider width as they move away from the mean, and thinner width close to the mean. This is much more stable than the random draw method, but note that it converges somewhat slowly to true values as well.

```
mt_fl_means <-
  sapply(ar_it_draws, function(x) {

    fl_prob_break = (fl_cdf_max - fl_cdf_min)/(x[1])
    ar_eps_bounds <- qnorm(seq(fl_cdf_min, fl_cdf_max,
                               by=(fl_cdf_max - fl_cdf_min)/(x[1])),
                           mean = fl_eps_mean, sd = fl_eps_sd)
    ar_eps_val <- (tail(ar_eps_bounds, -1) + head(ar_eps_bounds, -1))/2
    ar_eps_prb <- rep(fl_prob_break/(fl_cdf_max - fl_cdf_min), x[1])
    ar_eps_fev <- dnorm(ar_eps_val,
                        mean = fl_eps_mean, sd = fl_eps_sd)

    fl_cdf_total_approx <- sum(ar_eps_fev*diff(ar_eps_bounds))
    fl_mean_approx <- sum(ar_eps_val*(ar_eps_fev*diff(ar_eps_bounds)))
    fl_sd_approx <- sqrt(sum((ar_eps_val-fl_mean_approx)^2*(ar_eps_fev*diff(ar_eps_bounds))))

    return(list(cdf=fl_cdf_total_approx, mean=fl_mean_approx, sd=fl_sd_approx))
  })

mt_sample_means <- cbind(ar_it_draws, as_tibble(t(mt_fl_means)) %>% unnest())
colnames(mt_sample_means) <- c('draw_count', 'cdf', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)
```

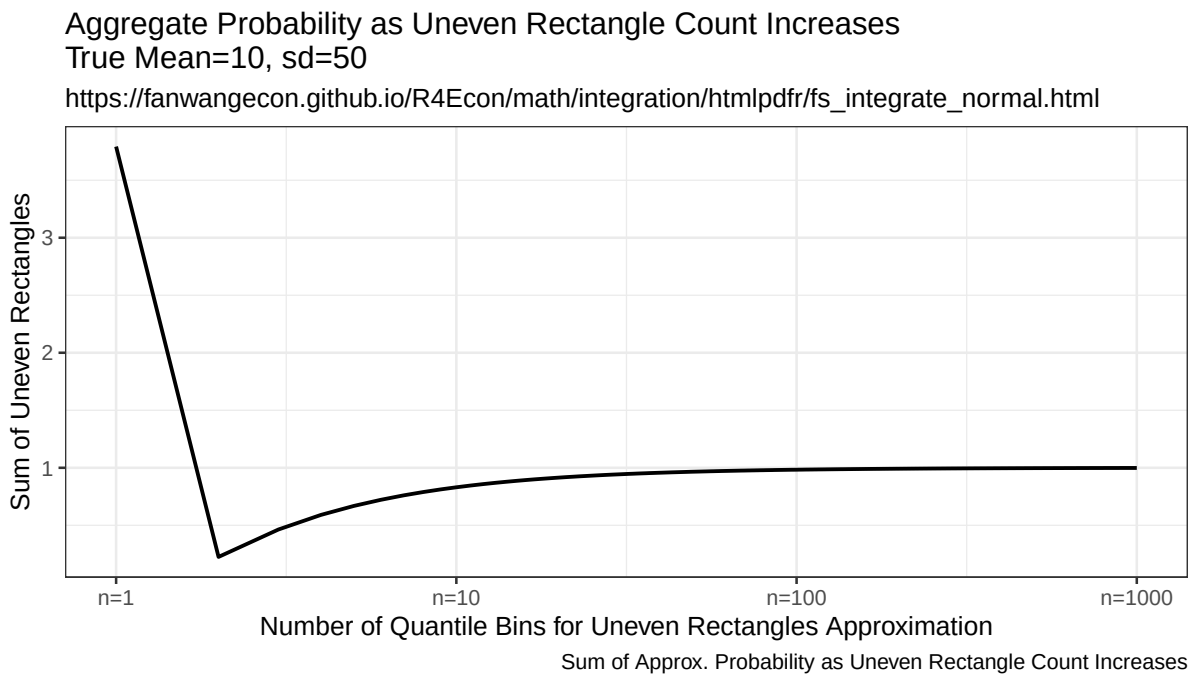
```
# Shared Labels
slb_title_shr = paste0('as Uneven Rectangle Count Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Number of Quantile Bins for Uneven Rectangles Approximation')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Average ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Approximated Mean',
       caption = 'Integral Approximation as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```




```
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Approximated Standard Deviation',
       caption = 'Integral Approximation as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_sd)
```



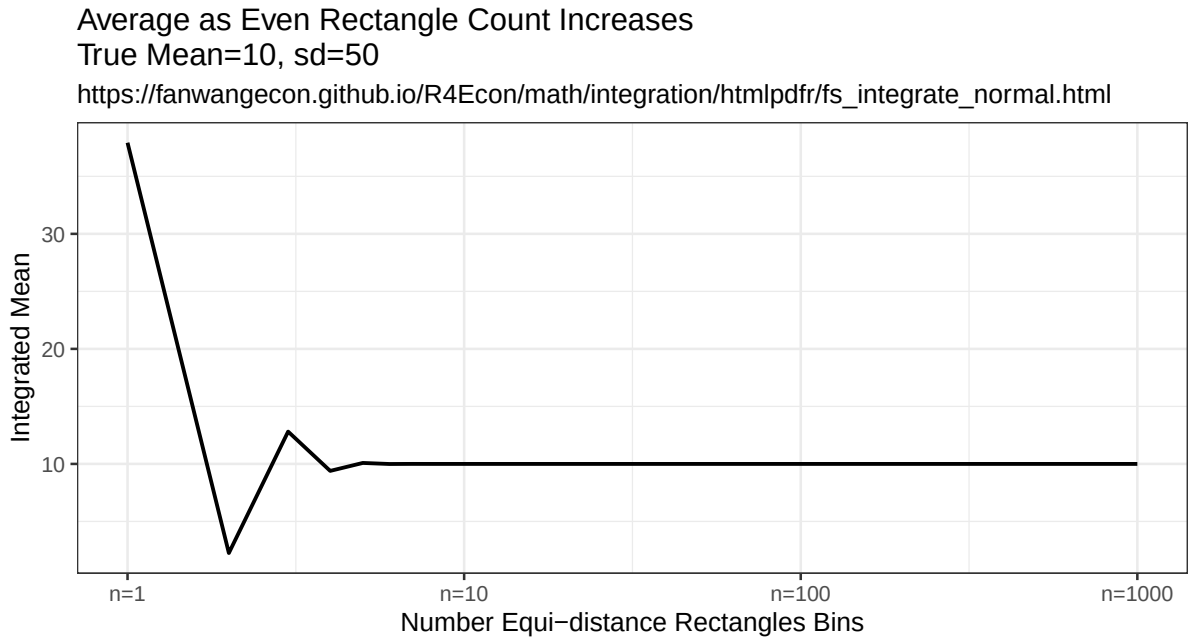

```
plt_cdf <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=cdf)) +
  geom_line(size=0.75) +
  labs(title = paste0('Aggregate Probability ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sum of Uneven Rectangles',
       caption = 'Sum of Approx. Probability as Uneven Rectangle Count Increases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_cdf)
```



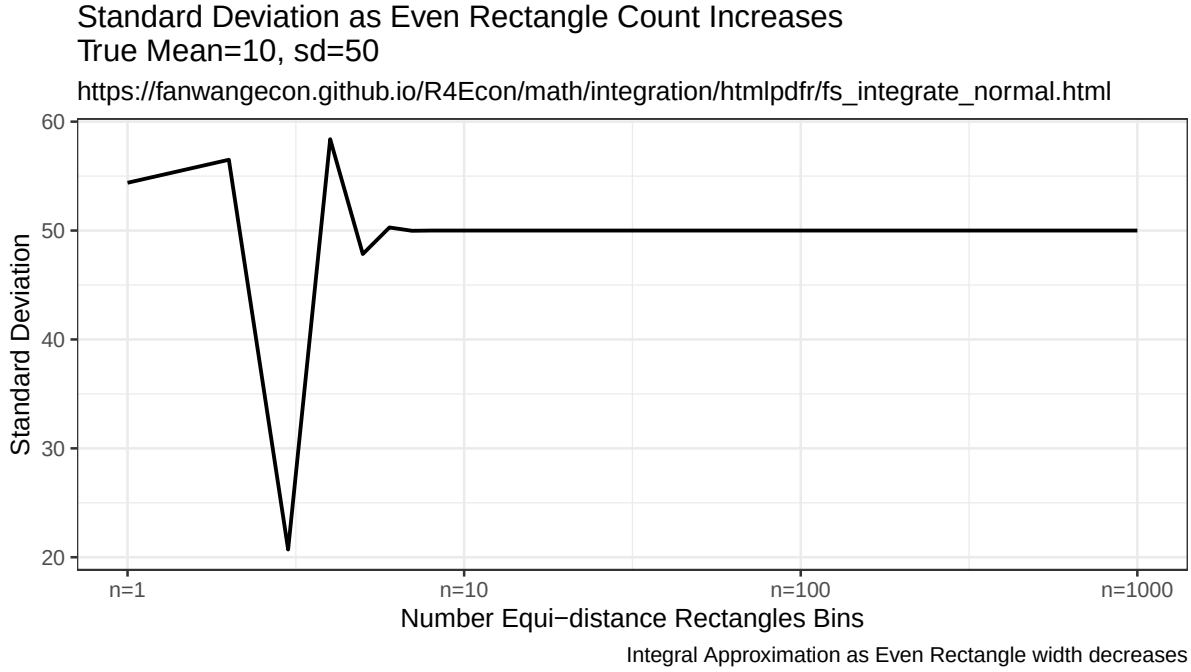
9.1.1.3 Integration By Constant Width Rectangle (Trapezoidal rule)

This is implementing even width recentagle, even along x-axix. Rectangle width are the same, height is $f(x)$. This is even width, but uneven area. Note that this method approximates the true answer much better and more quickly than the prior methods.

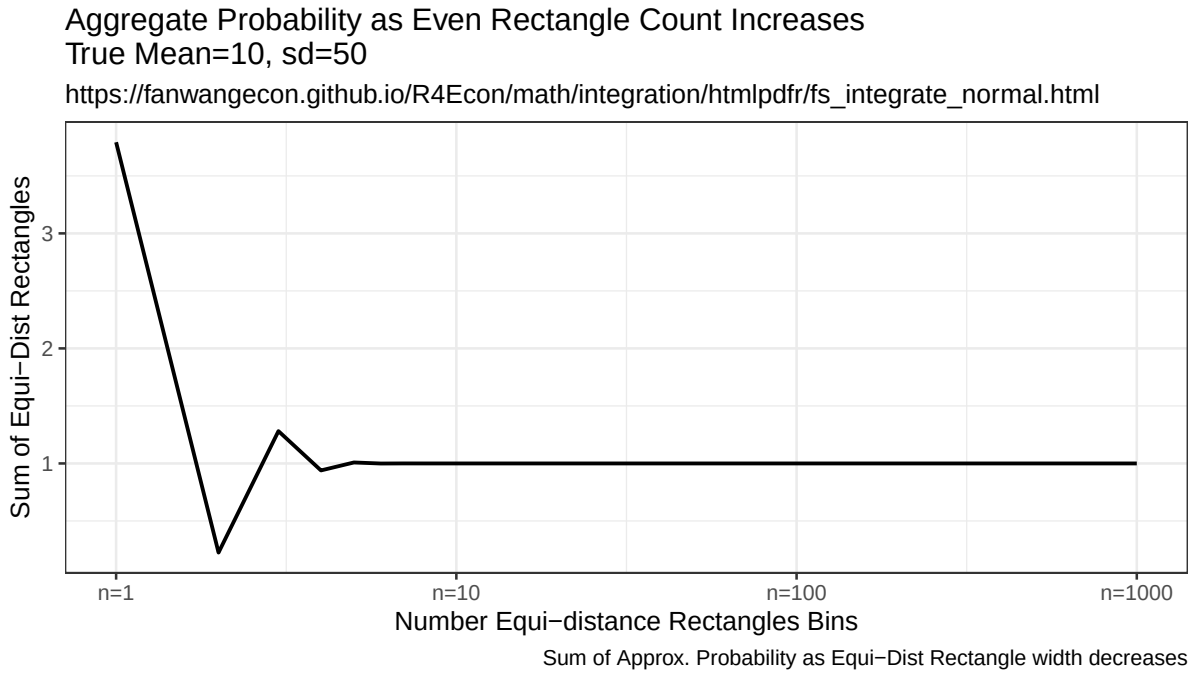
```
mt_fl_means <-
  sapply(ar_it_draws, function(x) {

    fl_eps_min <- qnorm(fl_cdf_min, mean = fl_eps_mean, sd = fl_eps_sd)
    fl_eps_max <- qnorm(fl_cdf_max, mean = fl_eps_mean, sd = fl_eps_sd)
    fl_gap <- (fl_eps_max-fl_eps_min)/(x[1])
    ar_eps_bounds <- seq(fl_eps_min, fl_eps_max, by=fl_gap)
    ar_eps_val <- (tail(ar_eps_bounds, -1) + head(ar_eps_bounds, -1))/2
    ar_eps_prb <- dnorm(ar_eps_val, mean = fl_eps_mean, sd = fl_eps_sd)*fl_gap

    fl_cdf_total_approx <- sum(ar_eps_prb)
    fl_mean_approx <- sum(ar_eps_val*ar_eps_prb)
    fl_sd_approx <- sqrt(sum((ar_eps_val-fl_mean_approx)^2*ar_eps_prb))

    return(list(cdf=fl_cdf_total_approx, mean=fl_mean_approx, sd=fl_sd_approx))
  })

mt_sample_means <- cbind(ar_it_draws, as_tibble(t(mt_fl_means)) %>% unnest())
colnames(mt_sample_means) <- c('draw_count', 'cdf', 'mean', 'sd')
tb_sample_means <- as_tibble(mt_sample_means)

# Graph
# x-labels
x.labels <- c('n=1', 'n=10', 'n=100', 'n=1000')
x.breaks <- c(1, 10, 100, 1000)

# Shared Labels
slb_title_shr = paste0('as Even Rectangle Count Increases\n',
                       'True Mean=', fl_eps_mean,', sd=',fl_eps_sd)
slb_xtitle = paste0('Number Equi-distance Rectangles Bins')

# Graph Results--Draw
plt_mean <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=mean)) +
  geom_line(size=0.75) +
  labs(title = paste0('Average ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Integrated Mean',
       caption = 'Integral Approximation as Even Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_mean)
```




```
plt_sd <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=sd)) +
  geom_line(size=0.75) +
  labs(title = paste0('Standard Deviation ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Standard Deviation',
       caption = 'Integral Approximation as Even Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_sd)
```



```
plt_cdf <- tb_sample_means %>%
  ggplot(aes(x=draw_count, y=cdf)) +
  geom_line(size=0.75) +
  labs(title = paste0('Aggregate Probability ', slb_title_shr),
       subtitle = st_subtitle,
       x = slb_xtitle,
       y = 'Sum of Equi-Dist Rectangles',
       caption = 'Sum of Approx. Probability as Equi-Dist Rectangle width decreases') +
  scale_x_continuous(trans='log10', labels = x.labels, breaks = x.breaks) +
  theme_bw()
print(plt_cdf)
```



9.2 Discrete Random Variable

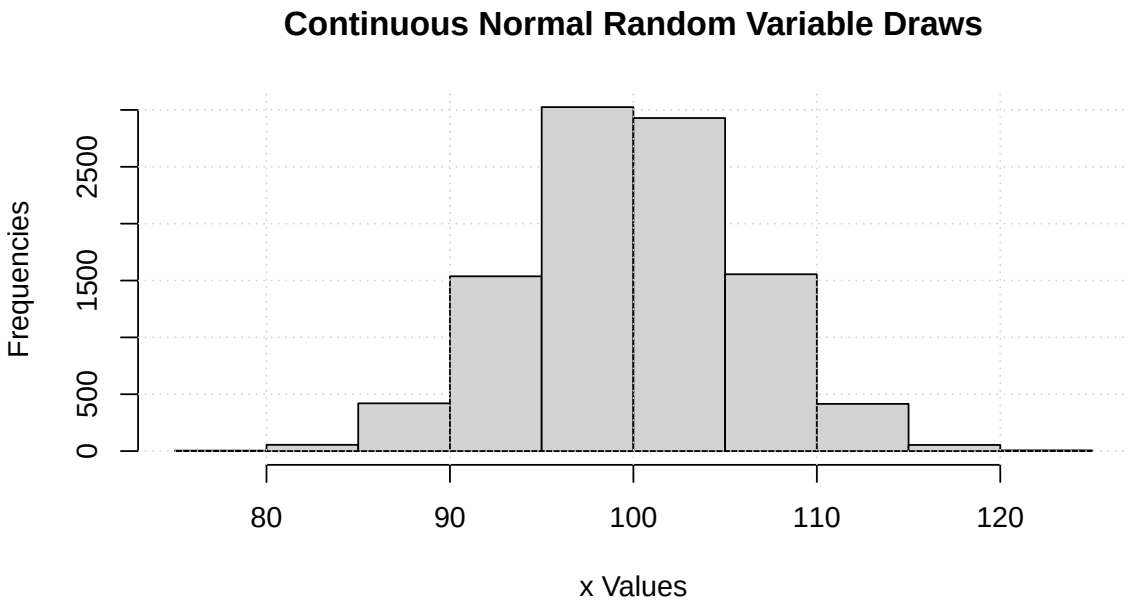
9.2.1 Discrete Approximation of Continuous Random Variables

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

9.2.1.1 Use Binomial Discrete Random Variable to Approximate Continuous Normal

First, draw from a Continuous Random Variable. Sample N draws from a normal random variable.

```
# Random normal Data Vector (not equal outcomes)
set.seed(123)
it_sample_N <- 10000
fl_rnorm_mean <- 100
fl_rnorm_sd <- 6
ar_data_rnorm <- rnorm(it_sample_N, mean = fl_rnorm_mean, sd = fl_rnorm_sd)
# Visualize
par(new = FALSE)
hist(ar_data_rnorm, xlab = "x Values", ylab = "Frequencies", main = "")
title(main = "Continuous Normal Random Variable Draws")
grid()
```



We use the binomial to approximate the normal distribution. Let μ and σ be the mean and standard deviations of the normal random variable, and n and p be the number of "trials" and the "probability-of-success" for the binomial distribution. We know that these relationships are approximately true, :

$$\begin{aligned}\mu &= n \cdot p \\ n &= \frac{\mu}{p} \\ \sigma^2 &= n \cdot p \cdot (1-p) = \mu \cdot (1-p)\end{aligned}$$

Given these, we have can translate between the normal random variable's parameters and the binomial discrete random variable's parameters:

$$\begin{aligned}p &= 1 - \frac{\sigma^2}{\mu} \\ n &= \frac{\mu}{1 - \frac{\sigma^2}{\mu}} = \frac{\mu}{\frac{\mu - \sigma^2}{\mu}} = \frac{\mu^2}{\mu - \sigma^2}\end{aligned}$$

There are two important aspects to note here:

1. Since p must be positive, this means $\frac{\sigma^2}{\mu} < 1$ and $\sigma^2 < \mu$, which is the condition for the above transformation to work.
2. The binomial discrete random variable will have non-zero mass for very small probability events at the left-tail. These very low outcome events are highly unlikely to be observed or drawn from sampling the continuous random variable. The presence of these left-tail values might impact the computation of certain statistics, for example the Atkinson Index for highly inequality averse planners.

Create a function for converting between normal and binomial parameters:

```
ffi_binom_approx_nomr <- function(fl_rnorm_mean, fl_rnorm_sd) {
  #' @param fl_rnorm_mean float normal mean
  #' @param fl_rnorm_sd float normal standard deviation
  if (fl_rnorm_mean <= fl_rnorm_sd^2) {
    stop("Normal mean must be larger than the variance for conversion")
  } else {
```

```
    # Use binomial to approximate normal
    fl_p_binom <- 1 - fl_rnorm_sd^2 / fl_rnorm_mean
    fl_n_binom <- round(fl_rnorm_mean^2 / (fl_rnorm_mean - fl_rnorm_sd^2))
    fl_binom_mean <- fl_n_binom * fl_p_binom
    fl_binom_sd <- sqrt(fl_n_binom * fl_p_binom * (1 - fl_p_binom))
    # return
    return(list(
      fl_p_binom = fl_p_binom, fl_n_binom = fl_n_binom,
      fl_binom_mean = fl_binom_mean, fl_binom_sd = fl_binom_sd
    ))
  }
}
```

Call the function to generate binomial parameters:

```
# with these parameters, does not work
ls_binom_params <- ffi_binom_approx_nomr(fl_rnorm_mean = 10, fl_rnorm_sd = 3)
# Call function with parameters, defined earlier, that work
ls_binom_params <- ffi_binom_approx_nomr(fl_rnorm_mean, fl_rnorm_sd)
fl_binom_mean <- ls_binom_params$fl_binom_mean
fl_binom_sd <- ls_binom_params$fl_binom_sd
fl_n_binom <- ls_binom_params$fl_n_binom
fl_p_binom <- ls_binom_params$fl_p_binom
# Mean and sd
print(paste0("BINOMI mean=", ls_binom_params$fl_binom_mean))
```

```
## [1] "BINOMI mean=99.84"
```

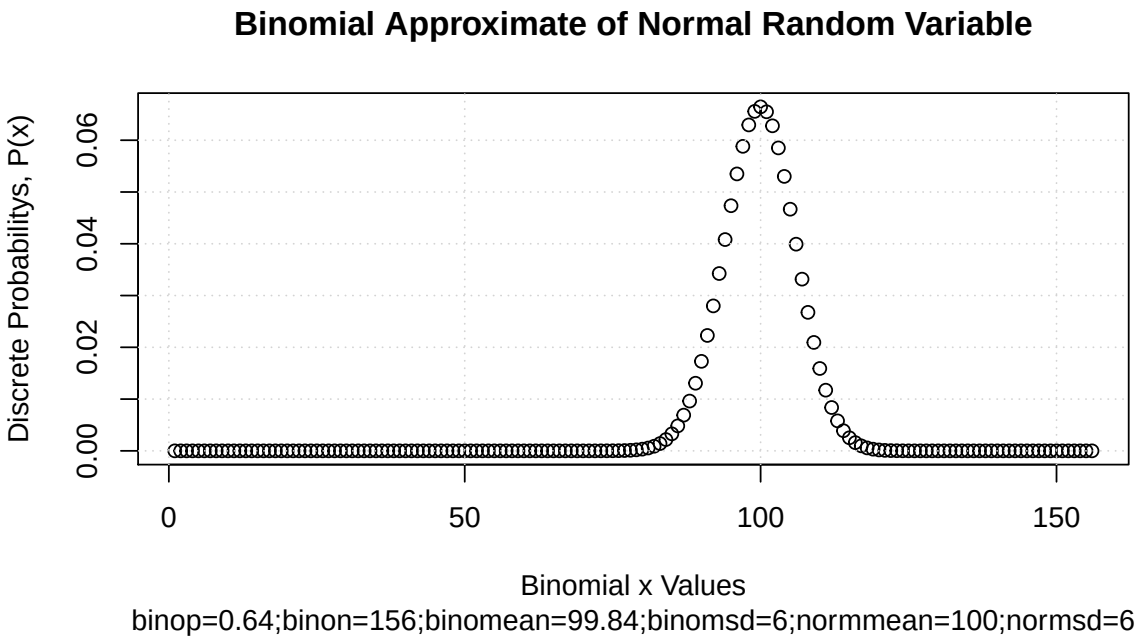
```
print(paste0("BINOMI mean=", ls_binom_params$fl_binom_sd))
```

```
## [1] "BINOMI mean=5.99519807846246"
```

```
# drv = discrete random variable
ar_drv_rbinom_xval <- seq(1, fl_n_binom)
ar_drv_rbinom_prob <- dbinom(ar_drv_rbinom_xval,
  size = fl_n_binom, prob = fl_p_binom
)
# ignore weight at x=0
ar_drv_rbinom_prob <- ar_drv_rbinom_prob / sum(ar_drv_rbinom_prob)
```

Visualize the binomial discrete random variable:

```
# graph
par(new = FALSE)
ar_ylim <- c(0, 1)
plot(ar_drv_rbinom_xval, ar_drv_rbinom_prob,
  xlab = "Binomial x Values", ylab = "Discrete Probabilitys, P(x)"
)
title(
  main = paste0("Binomial Approximate of Normal Random Variable"),
  sub = paste0(
    "binop=", round(fl_p_binom, 2),
    ";binon=", round(fl_n_binom, 2),
    ";binomean=", round(fl_binom_mean, 2),
    ";binomsd=", round(fl_binom_sd, 2),
    ";normmean=", round(fl_rnorm_mean, 2), ";normsd=", round(fl_rnorm_sd, 2)
  )
)
grid()
```



9.2.2 Obtaining Joint Distribution from Marginal with Rectilinear Restrictions

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Suppose we want to know the joint probability mass function $P(E, A)$ where E and A are both discrete. E could be education groups, and A could be age groups. But we only know $P(E)$ and $P(A)$, under what conditions can we obtain the joint distribution from the marginals?

9.2.2.1 Unrestricted Joint 2 by 2 Distribution

Suppose there are two unique states for E and A. For example, suppose we know the unemployment probability for better or worse educated, and also for low and high age groups. We want to know the joint probability of been both better educated and lower age, better educated and higher age, worse educated and lower age, and worse educated and higher age. Then:

$$
\begin{aligned}
P(E_1) &= P(A_1, E_1) + P(A_2, E_1) \\
P(E_2) &= P(A_1, E_2) + P(A_2, E_2) \\
P(A_1) &= P(A_1, E_1) + P(A_1, E_2) \\
P(A_2) &= P(A_2, E_1) + P(A_2, E_2)
\end{aligned}
$$

We know $P(E_1)$, $P(E_2)$, $P(A_1)$ and $P(A_2)$, but not $P(A_i, E_i)$. Let X, W, Y, Z be unknowns and A, B, C, D are known. It might seem like that with four equations and four unknowns, we can find X, W, Y, Z. But because what we know are probabilities: the marginals along each dimension sums up to 1. Without restrictions, there are no unique solutions to this problem. but many possible solutions. For example, Suppose $P(E_1) = 0.5$ and $P(A_1) = 0.5$, rewrite the above problem as:

$$
\begin{aligned}
0.5 &= X + W \\
0.5 &= Y + Z \\
0.5 &= W + Y \\
0.5 &= X + Z
\end{aligned}
$$

Solutions include:

- $X = 0.2$, $W = 0.3$, $Y = 0.2$, $Z = 0.3$
- $X = 0.4$, $W = 0.1$, $Y = 0.4$, $Z = 0.1$
- and infinitely many others ...

There are no unique solutions, because, when we write the linear system above in matrix form as shown below, the A coefficient matrix is not full rank, but has a rank of 3.

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$$

$$A \cdot \mathbb{X} = b$$

We can see the rank of a matrix with the qr function (QR decomposition):

```
# Construct The coefficent Matrix
mt_a = t(matrix(data=c(1, 1, 0, 0,
                       0, 0, 1, 1,
                       1, 0, 1, 0,
                       0, 1, 0, 1), nrow=4, ncol=4))
# rank Check with the qr function:
print(qr(mt_a))
```

```
## $qr
##            [,1]       [,2]       [,3]          [,4]
## [1,] -1.4142136 -0.7071068 -0.7071068  0.000000e+00
## [2,]  0.0000000 -1.2247449  0.4082483 -8.164966e-01
## [3,]  0.7071068 -0.5773503 -1.1547005 -1.154701e+00
## [4,]  0.0000000  0.8164966 -0.4184316 -2.775558e-16
##
## $rank
## [1] 3
##
## $qraux
## [1] 1.707107e+00 1.000000e+00 1.908248e+00 2.775558e-16
##
## $pivot
## [1] 1 2 3 4
##
## attr(,"class")
## [1] "qr"
```

9.2.2.2 Rectilinear Restriction on Joint 2 by 2 Distribution

So in the section above, it is demonstrated that it is not possible to uniquely identify the joint probability mass function from marginal probability mass functions.

However, sometimes, we need to find some reasonable joint distribution, when we only observe marginal distributions. This joint distribution might be an input transition matrix in a model we simulate. If we just use one of the infinitely possible joint mass that match up with the marginals, then the model would have infinitely many simulation results depending on our arbitrary choice of joint mass.

Ideally, one should try to obtain data to estimate the underlying joint distribution, when this is not possible, we can impose additional non-parametric restrictions on the structures of the joint probability mass that would lead to unique joint mass from marginals.

Specifically, I will assume the incremental changes across rows and across columns of the joint mass matrix are row or column specific, is this sufficient? (In Some Cases it Will Not be):

$$\Delta_{12}^{E} = P(A_1, E_2) - P(A_1, E_1) = P(A_2, E_2) - P(A_2, E_1)$$
$$\Delta_{12}^{A} = P(A_2, E_1) - P(A_1, E_1) = P(A_2, E_2) - P(A_1, E_2)$$

The assumption is non-parametric. This is effectively an rectilinear assumption on the joint Cumulative Probability Mass Function.

Given this assumption, now we have:

$$P(E_2) - P(E_1) = P(A_1, E_2) + P(A_2, E_2) - P(A_1, E_1) - P(A_2, E_1)$$
$$P(A_2) - P(A_1) = P(A_2, E_1) + P(A_2, E_2) - P(A_1, E_1) - P(A_1, E_2)$$

Which become:

$$P(E_2) - P(E_1) = 2 \cdot \Delta_{12}^{E}$$
$$P(A_2) - P(A_1) = 2 \cdot \Delta_{12}^{A}$$

Suppose $P(E_1) = 0.5$ and $P(A_1) = 0.5$:

- $\phi = 0$
- $\rho = 0$
- hence: $P(A_1, E_1) = P(A_1, E_2) = P(A_2, E_1) = P(A_2, E_2) = 0.25$

Suppose $P(E_1) = 0.4$ and $P(A_1) = 0.7$:

- $\Delta_{12}^{E} = 0.1$
- $\Delta_{12}^{A} = -0.20$

Hence:

$$0.4 = P(A_1, E_1) + P(A_2, E_1) = P(A_1, E_1) + P(A_1, E_1) + \Delta_{12}^{A} = 2 \cdot P(A_1, E_1) - 0.20$$

And:

$$P(A_1, E_1) = \frac{0.60}{2} = 0.30$$
$$P(A_2, E_1) = 0.4 - 0.30 = 0.10$$
$$P(A_1, E_2) = 0.7 - 0.30 = 0.40$$
$$P(A_2, E_2) = 0.3 - 0.10 = 0.20$$

These joint mass sum up to 1, satisfy the marginal mass requirements from the data, and are unique given the rectilinear assumption.

9.2.2.3 Rectilinear Restriction Diamond on Joint 2 by 2 Distribution

The rectilinear assumptions, however, do not necessarily lead to positive values for each element of the joint mass function. In this section, I discuss under what conditions the rectilinear restriction leads to positive mass at all points of the joint probability mass function.

We can write these:

$$P(A_1, E_1) = \frac{1}{4}\left(1 - \Delta_{12}^{E} \cdot 2 - \Delta_{12}^{A} \cdot 2\right)$$
$$P(A_2, E_1) = P(A_1, E_1) + \Delta_{12}^{A}$$
$$P(A_1, E_2) = P(A_1, E_1) + \Delta_{12}^{E}$$
$$P(A_2, E_2) = P(A_1, E_1) + \Delta_{12}^{A} + \Delta_{12}^{E}$$

Plugging the Values for $P(A_1, E_1)$ in, we have:

$$P(A_1, E_1) = \frac{1}{4}\left(1 - \Delta_{12}^{E} \cdot 2 - \Delta_{12}^{A} \cdot 2\right)$$
$$P(A_2, E_1) = \frac{1}{4}\left(1 - \Delta_{12}^{E} \cdot 2 + \Delta_{12}^{A} \cdot 2\right)$$
$$P(A_1, E_2) = \frac{1}{4}\left(1 + \Delta_{12}^{E} \cdot 2 - \Delta_{12}^{A} \cdot 2\right)$$
$$P(A_2, E_2) = \frac{1}{4}\left(1 + \Delta_{12}^{E} \cdot 2 + \Delta_{12}^{A} \cdot 2\right)$$

When are these terms positive:

$$P(A_1, E_1) \geq 0 \text{ iff } \frac{1}{2} \geq \Delta_{12}^E + \Delta_{12}^A$$
$$P(A_2, E_1) \geq 0 \text{ iff } \frac{1}{2} \geq \Delta_{12}^E - \Delta_{12}^A$$
$$P(A_1, E_2) \geq 0 \text{ iff } \frac{1}{2} \geq -\Delta_{12}^E + \Delta_{12}^A$$
$$P(A_2, E_2) \geq 0 \text{ iff } \frac{1}{2} \geq -\Delta_{12}^E - \Delta_{12}^A$$

Rewriting the positivity conditions, we have:

$$\Delta_{12}^E \leq \frac{1}{2} - \Delta_{12}^A$$
$$\Delta_{12}^E \leq \frac{1}{2} + \Delta_{12}^A$$
$$\Delta_{12}^E \geq -\frac{1}{2} + \Delta_{12}^A$$
$$\Delta_{12}^E \geq -\frac{1}{2} - \Delta_{12}^A$$

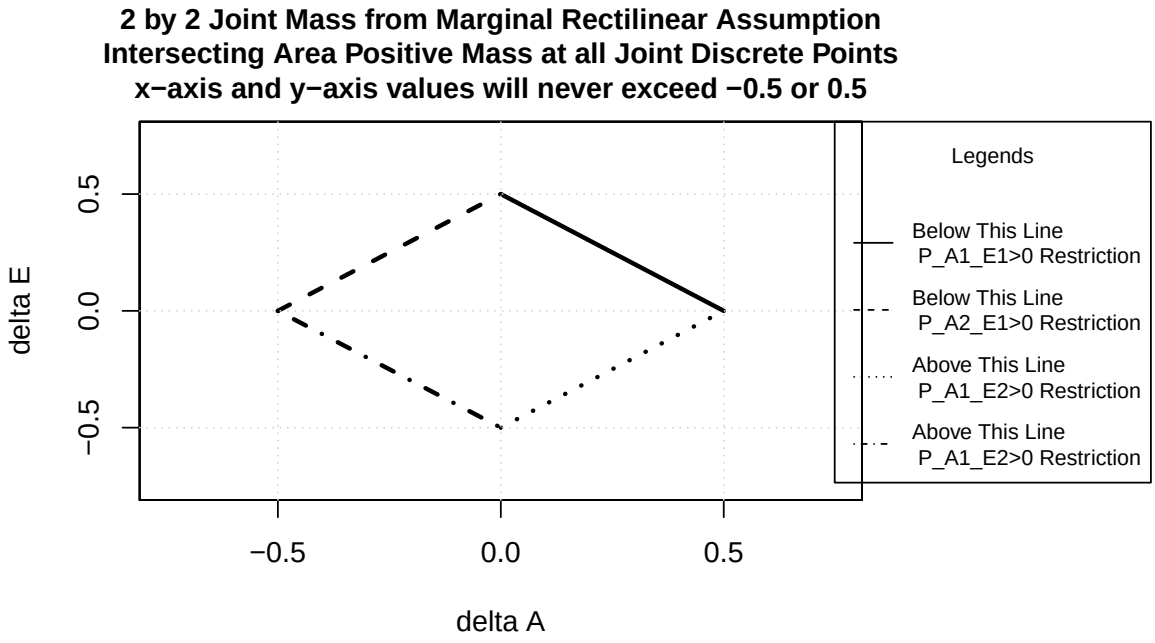
The four conditions above create a diamond, Following the rectilinear restriction, if the δ^E and δ^A values fall within the diamond region, then the mass for at all joint probability points will be positive. Basically the restriction is that when the jumps in mass are more extreme, rectilinear restriction will not return positive mass at all points. The requirement is that:

$$| \Delta_{12}^A | + | \Delta_{12}^E | \leq \frac{1}{2}$$

Graphically:

```
# Labeling
st_title <- paste0('2 by 2 Joint Mass from Marginal Rectilinear Assumption\n',
                   'Intersecting Area Positive Mass at all Joint Discrete Points\n',
                   'x-axis and y-axis values will never exceed -0.5 or 0.5')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/math/discrandvar/htmlpdfr/fs_drm_mass.html')
st_x_label <- 'delta A'
st_y_label <- 'delta E'

# Line 1
x1 <- seq(0, 0.5, length.out=50)
y1 <- 0.5-x1
st_legend1 <- 'Below This Line\n P_A1_E1>0 Restriction'
# Line 2
x2 <- seq(-0.5, 0, length.out=50)
y2 <- 0.5+x2
st_legend2 <- 'Below This Line\n P_A2_E1>0 Restriction'
# Line 3
x3 <- seq(0, 0.5, length.out=50)
y3 <- -0.5+x3
st_legend3 <- 'Above This Line\n P_A1_E2>0 Restriction'
# Line 4
x4 <- seq(-0.5, 0, length.out=50)
y4 <- -0.5-x4
st_legend4 <- 'Above This Line\n P_A1_E2>0 Restriction'

# line lty
st_line_1_lty <- 'solid'
```

```
st_line_2_lty <- 'dashed'
st_line_3_lty <- 'dotted'
st_line_4_lty <- 'dotdash'

# Share xlim and ylim
ar_xlim = c(-0.75, 0.75)
ar_ylim = c(-0.75, 0.75)

# Graph
par(new=FALSE, mar=c(5, 4, 4, 10))
plot(x1, y1, type="l", col = 'black', lwd = 2.5, lty = st_line_1_lty,
      xlim = ar_xlim, ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
par(new=T)
plot(x2, y2, type="l", col = 'black', lwd = 2.5, lty = st_line_2_lty,
      xlim = ar_xlim, ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
par(new=T)
plot(x3, y3, type="l", col = 'black', lwd = 2.5, lty = st_line_3_lty,
      xlim = ar_xlim, ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
par(new=T)
plot(x4, y4, type="l", col = 'black', lwd = 2.5, lty = st_line_4_lty,
      xlim = ar_xlim, ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)

# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 1
fl_ces_fig_small = 0.75

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topright",
       inset=c(-0.4,0),
       xpd=TRUE,
       c(st_legend1, st_legend2, st_legend3, st_legend4),
       cex = fl_ces_fig_small,
       lty = c(st_line_1_lty, st_line_2_lty, st_line_3_lty, st_line_4_lty),
       title = 'Legends',
       y.intersp=2)
```



https://fanwangecon.github.io/R4Econ/math/discrandvar/htmlpdfr/fs_drm_mass.html

Programmatically, With different random values, we have:

```
it_warning = 0
it_neg = 0
for (it_rand_seed in 1:1000) {
  # Generate two marginal MASS
  set.seed(it_rand_seed)
  ar_E_marginal <- runif(2)
  # ar_E_marginal <- c(0.01, 0.99)
  # ar_E_marginal <- c(0.49, 0.51)
  ar_E_marginal <- ar_E_marginal/sum(ar_E_marginal)
  ar_A_marginal <- runif(2)
  # ar_A_marginal <- c(0.01, 0.99)
  # ar_A_marginal <- c(0.01, 0.99)
  ar_A_marginal <- ar_A_marginal/sum(ar_A_marginal)
  # print(ar_E_marginal)
  # print(ar_A_marginal)
  # Differences in Marginal Points
  ar_delta_E = diff(ar_E_marginal)/2
  ar_delta_A = diff(ar_A_marginal)/2
  # print(paste0('deltaE + deltaA:', diff(ar_E_marginal) + diff(ar_A_marginal)))
  # some cell negativity condition:
  if (sum(abs(diff(ar_E_marginal))) + sum(abs(diff(ar_A_marginal))) > 1){
    it_warning = it_warning + 1
    # warning('Outside of Diamond, Rectilinear Restriction Leads to Negative Values in Some Cells\n'
  }
  # What is P(A1,E1), implemetning the formula above
  fl_P_A1_E1 = (1 - c(1,1) %*% rbind(ar_delta_E, ar_delta_A) %*% t(t(c(2))))/(4)
  # Getting the Entire P_A_E matrix
  mt_P_A_E = matrix(data=NA, nrow=2, ncol=2)
  for (it_row in 1:length(ar_E_marginal)){
    for (it_col in 1:length(ar_A_marginal)){
      fl_p_value = fl_P_A1_E1
      if (it_row >= 2){
        fl_p_value = fl_p_value + sum(ar_delta_E[1:(it_row-1)])
      }
```

```
      if (it_col >= 2){
        fl_p_value = fl_p_value + sum(ar_delta_A[1:(it_col-1)])
      }
      mt_P_A_E[it_row, it_col] = fl_p_value
    }
  }
  # print(mt_P_A_E)
  sum(mt_P_A_E)
  rowSums(mt_P_A_E)
  colSums(mt_P_A_E)
  if (length(mt_P_A_E[mt_P_A_E<0])>0){
      it_neg = it_neg + 1
  }
}
print(paste0('it_warning:',it_warning))
```

```
## [1] "it_warning:283"
```

```
print(paste0('it_neg:',it_neg))
```

```
## [1] "it_neg:283"
```

9.2.2.4 Restricted Joint 3 by 3 Distribution

The idea can be applied to when there are three discrete random outcomes along each dimension. Find an unique 3 by 3 probability joint mass distribution from marginal distributions. Similar to the 2 by 2 case, only when marginal mass changes are within a change diamond will this method lead to positive mass at all points of the joint distribution.

Given this assumption:

$$\begin{aligned}
\Delta^E_{12} &= P(A_1, E_2) - P(A_1, E_1) = P(A_2, E_2) - P(A_2, E_1) = P(A_3, E_2) - P(A_3, E_1) \\
\Delta^E_{23} &= P(A_1, E_3) - P(A_1, E_2) = P(A_2, E_3) - P(A_2, E_2) = P(A_3, E_3) - P(A_3, E_2) \\
\Delta^A_{12} &= P(A_2, E_1) - P(A_1, E_1) = P(A_2, E_2) - P(A_1, E_2) = P(A_2, E_3) - P(A_1, E_3) \\
\Delta^A_{23} &= P(A_3, E_1) - P(A_2, E_1) = P(A_3, E_2) - P(A_2, E_2) = P(A_3, E_3) - P(A_2, E_3)
\end{aligned}$$

Following the two by two example, the restriction above just means we can use the differences between the marginal distribution's discrete points to back out.

$$\begin{aligned}
\Delta^E_{12} &= \frac{P(A_2) - P(A_1)}{3} \\
\Delta^E_{23} &= \frac{P(A_3) - P(A_2)}{3} \\
\Delta^A_{12} &= \frac{P(E_2) - P(E_1)}{3} \\
\Delta^A_{23} &= \frac{P(E_3) - P(E_2)}{3}
\end{aligned}$$

Given these Δ values, we can solve for (A_1, E_1):

$$\begin{aligned}
1 &= 3 \cdot 3 \cdot P(A_1, E_1) + \Delta^E_{12} \cdot 3 \cdot (3-1) + \Delta^E_{23} \cdot 3 + \Delta^A_{12} \cdot 3 \cdot (3-1) + \Delta^A_{12} \cdot 3 \\
P(A_1, E_1) &= \frac{1}{9}\left(1 - \Delta^E_{12} \cdot 3 \cdot (3-1) - \Delta^E_{23} \cdot 3 - \Delta^A_{12} \cdot 3 \cdot (3-1) - \Delta^A_{12} \cdot 3\right)
\end{aligned}$$

In Matrix form:

$$P(A_1, E_1) = \frac{1}{3 \cdot 3}\left(1 - \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta^E_{12} & \Delta^E_{23} \\ \Delta^A_{12} & \Delta^A_{23} \end{bmatrix} \cdot \begin{bmatrix} 3 \cdot (3-1) \\ 3 \end{bmatrix}\right)$$

Following the 2 by 2 case, the condition needed for positive mass at all points is:

$$| \Delta_{12}^{A} | + | \Delta_{23}^{A} | + | \Delta_{12}^{E} + | \Delta_{23}^{E} | \leq \frac{1}{3}$$

Implementing the formulas, we have:

```
# Generate two marginal MASS
it_warning = 0
it_neg = 0
it_concur = 0
for (it_rand_seed in 1:1000) {
  set.seed(it_rand_seed)
  # set.seed(333)
  ar_E_marginal <- runif(3)
  ar_E_marginal <- ar_E_marginal/sum(ar_E_marginal)
  ar_A_marginal <- runif(3)
  ar_A_marginal <- ar_A_marginal/sum(ar_A_marginal)
  # print(ar_E_marginal)
  # print(ar_A_marginal)
  # Differences in Marginal Points
  ar_delta_E_m = diff(ar_E_marginal)
  ar_delta_A_m = diff(ar_A_marginal)
  ar_delta_E = diff(ar_E_marginal)/3
  ar_delta_A = diff(ar_A_marginal)/3
  # some cell negativity condition: this condition is incorrect
  bl_count_warn = FALSE
  for (it_row in 1:length(ar_delta_E)){
    for (it_col in 1:length(ar_delta_A)){
      if ((abs(sum(ar_delta_E_m[1:it_row])) + abs(sum(ar_delta_A_m[1:it_col]))) > 2/4) {
        bl_count_warn = TRUE
      }
      if ((abs(ar_delta_E_m[it_row]) + abs(ar_delta_A_m[it_col])) > 2/4) {
        bl_count_warn = TRUE
      }
    }
  }
  if (bl_count_warn) {
    # if (max(abs(diff(ar_E_marginal))) + max(abs(diff(ar_A_marginal))) > 2/3){
    it_warning = it_warning + 1
    # }
    # warning('Outside of Diamond, Rectilinear Restriction Leads to Negative Values in Some Cells\n'
  }
  # What is P(A1,E1), implemetning the formula above
  fl_P_A1_E1 = (1 - c(1,1) %*% rbind(ar_delta_E, ar_delta_A) %*% t(t(c(3*2, 3))))/(3*3)
  # Getting the Entire P_A_E matrix
  mt_P_A_E = matrix(data=NA, nrow=3, ncol=3)
  for (it_row in 1:length(ar_E_marginal)){
    for (it_col in 1:length(ar_A_marginal)){
      fl_p_value = fl_P_A1_E1
      if (it_row >= 2){
        fl_p_value = fl_p_value + sum(ar_delta_E[1:(it_row-1)])
      }
      if (it_col >= 2){
        fl_p_value = fl_p_value + sum(ar_delta_A[1:(it_col-1)])
      }
      mt_P_A_E[it_row, it_col] = fl_p_value
    }
  }
```

```
  # print(mt_P_A_E)
  sum(mt_P_A_E)
  rowSums(mt_P_A_E)
  colSums(mt_P_A_E)
  if (length(mt_P_A_E[mt_P_A_E<0])>0){
      it_neg = it_neg + 1
      if (bl_count_warn) {
        it_concur = it_concur + 1
      }
  }
}
print(paste0('it_warning:',it_warning))
```

```
## [1] "it_warning:797"
```

```
print(paste0('it_neg:',it_neg))
```

```
## [1] "it_neg:592"
```

```
print(paste0('it_concur:',it_concur))
```

```
## [1] "it_concur:592"
```

9.2.2.5 Restricted Joint 5 by 5 Distribution

For a Five by Five Problem, we have:

In Matrix form:

$$P(A_1, E_1) = \frac{1}{5 \cdot 5} \left(1 - \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta_{12}^{E} & \Delta_{23}^{E} & \Delta_{34}^{E} & \Delta_{45}^{E} \\ \Delta_{45}^{A} & \Delta_{45}^{A} & \Delta_{45}^{A} & \Delta_{45}^{A} \end{bmatrix} \cdot \begin{bmatrix} 5 \cdot 4 \\ 5 \cdot 3 \\ 5 \cdot 2 \\ 5 \cdot 1 \end{bmatrix} \right)$$

```
# pi_j=[0.22;0.175;0.16;0.165;0.22]; % Probability of unemployment in 2020 by age groups from Cajner
# pi_w=[0.360;0.22;0.17;0.14;0.09]; % Probability of unemployment in 2020 by wage quintiles from Caj
# Generate two marginal MASS
set.seed(111)
# set.seed(333)
ar_E_marginal <- c(0.22, 0.175, 0.16, 0.165, 0.22)
ar_E_marginal <- ar_E_marginal/sum(ar_E_marginal)
ar_A_marginal <- c(0.360, 0.22, 0.17, 0.14, 0.09)
ar_A_marginal <- ar_A_marginal/sum(ar_A_marginal)
print(ar_E_marginal)
```

```
## [1] 0.2340426 0.1861702 0.1702128 0.1755319 0.2340426
```

```
print(ar_A_marginal)
```

```
## [1] 0.36734694 0.22448980 0.17346939 0.14285714 0.09183673
```

```
# Differences in Marginal Points
ar_delta_E = diff(ar_E_marginal)/5
ar_delta_A = diff(ar_A_marginal)/5
# What is P(A1,E1), implemetning the formula above
fl_P_A1_E1 = (1 - c(1,1) %*% rbind(ar_delta_E, ar_delta_A) %*% t(t(c(20,15,10,5))))/(5*5)
# Getting the Entire P_A_E matrix
mt_P_A_E = matrix(data=NA, nrow=5, ncol=5)
for (it_row in 1:length(ar_E_marginal)){
  for (it_col in 1:length(ar_A_marginal)){
    fl_p_value = fl_P_A1_E1
```

```
    if (it_row >= 2){
      fl_p_value = fl_p_value + sum(ar_delta_E[1:(it_row-1)])
    }
    if (it_col >= 2){
      fl_p_value = fl_p_value + sum(ar_delta_A[1:(it_col-1)])
    }
    mt_P_A_E[it_row, it_col] = fl_p_value
  }
}
print(mt_P_A_E)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.08027790 0.05170647 0.04150239 0.03537994 0.02517586
## [2,] 0.07070343 0.04213200 0.03192792 0.02580547 0.01560139
## [3,] 0.06751194 0.03894051 0.02873643 0.02261398 0.01240990
## [4,] 0.06857577 0.04000434 0.02980026 0.02367781 0.01347373
## [5,] 0.08027790 0.05170647 0.04150239 0.03537994 0.02517586
```

```
sum(mt_P_A_E)
```

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

```
rowSums(mt_P_A_E)
```

```
## [1] 0.2340426 0.1861702 0.1702128 0.1755319 0.2340426
```

```
colSums(mt_P_A_E)
```

```
## [1] 0.36734694 0.22448980 0.17346939 0.14285714 0.09183673
```

9.2.3 Obtaining Joint Distribution from Conditional with Rectilinear Restrictions

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

9.2.3.1 2 by 2 Joint from Marginal Probability Mass Functions

We know the $P(U|E)$, the probability of unemployment by educational types. I also know $P(U|A)$, the probability of unemployment by age groups. Additionally, I also know the $P(E, A)$, the mass of individuals at discrete (E, A) combinations. What is the $P(U|E, A)$, the probability of unemployment by age and education groups?

9.2.3.1.1 Generate Data Structure Generate the data structure. First, a function that generates random joint probabilities of mass at different points. This data-structure will work for 2 by 2, 3 by 3, or 4 by 4 problems.

```
ffi_gen_rand_joint_mass <- function(it_seed = 123,
                                    it_Q = 2, it_M = 2,
                                    bl_verbose = FALSE) {

  # set random seed
  set.seed(it_seed)

  # Generate prob list
  ls_2d <- vector(mode = "list", length = it_Q*it_M)
  dim(ls_2d) <- c(it_Q, it_M)

  # Random joint mass
  ar_rand <- runif(it_Q*it_M)
  ar_rand <- ar_rand/sum(ar_rand)
```

```
  # Fill with values
  it_ctr <- 0
  for (it_Q_ctr in seq(1,it_Q)) {
    for (it_M_ctr in seq(1,it_M)) {
      # linear index
      ls_2d[[it_M_ctr, it_Q_ctr]] <- ar_rand[(it_Q_ctr-1)*it_M+it_M_ctr]
    }
  }

  # Replace row names, note rownames does not work
  dimnames(ls_2d)[[1]] <- paste0('E',seq(1,it_M))
  dimnames(ls_2d)[[2]] <- paste0('A',seq(1,it_Q))

  # rename
  ls_prob_joint_E_A <- ls_2d
  mt_prob_joint_E_A <- matrix(unlist(ls_prob_joint_E_A), ncol=it_M, byrow=F)

  if (bl_verbose) {
    print('ls_prob_joint_E_A')
    print(ls_prob_joint_E_A)
    print(mt_prob_joint_E_A)
  }

  # return
  return(list(mt_prob_joint_E_A=mt_prob_joint_E_A,
              ls_prob_joint_E_A=ls_prob_joint_E_A))
}
```

Second, given joint mass at different points, generate conditional unemployment probabilities along each discrete variable's values that works for 2 by 2 case. We have two unique levels for E and for A separately, so these are four points where there is mass. Generate random mass at these points. Then generate given these four additional mass points: $F = P(A_1|E_1)$, $B = P(A_1|E_2)$, $C = P(E_1|A_1)$, $D = P(E_1|A_2)$. Use code from fs_lst_basics.

```
ffi_gen_condi_unemploy_prob_2by2 <- function(mt_prob_joint_E_A,
                                        fl_alpha = 0.25, fl_beta = 0.05,
                                        fl_gamma = 0.31, fl_delta = 0.50,
                                        bl_verbose = FALSE) {

  # From joint probability, generate conditional probabilities
  fl_F <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[1,])
  fl_B <- mt_prob_joint_E_A[2,1]/sum(mt_prob_joint_E_A[2,])
  fl_C <- mt_prob_joint_E_A[1,1]/sum(mt_prob_joint_E_A[,1])
  fl_D <- mt_prob_joint_E_A[1,2]/sum(mt_prob_joint_E_A[,2])
  if (bl_verbose) {
    print(paste0('fl_F=', fl_F, ',fl_B=',fl_B,',fl_C=',fl_C,',fl_D=',fl_D))
  }

  # Also generate random W X Y Z
  # ar_b <- runif(4)

  # fl_Delta_A_true <- 0.05
  # fl_Delta_E_true <- 0.11

  # fl_alpha_true <- 0.25
  # fl_beta_true <- fl_alpha_true + fl_Delta_A_true
  # fl_gamma_true <- fl_alpha_true + fl_Delta_E_true
  # fl_delta_true <- fl_alpha_true + fl_Delta_E_true + fl_Delta_A_true
```

```
  # fl_beta_true <- 0.31
  # fl_gamma_true <- 0.50
  # fl_delta_true <- 0.16

  fl_W <- fl_alpha*fl_F + fl_beta*(1-fl_F)
  fl_X <- fl_gamma*fl_B + fl_delta*(1-fl_B)
  fl_Y <- fl_alpha*fl_C + fl_gamma*(1-fl_C)
  fl_Z <- fl_beta*fl_D + fl_delta*(1-fl_D)
  fl_V <- mt_prob_joint_E_A[1,1]*fl_alpha_true +
    mt_prob_joint_E_A[1,2]*fl_beta_true +
    mt_prob_joint_E_A[2,1]*fl_gamma_true +
    mt_prob_joint_E_A[2,2]*fl_delta_true
  ar_b <- c(fl_W, fl_X, fl_Y, fl_Z)

  if (bl_verbose) {
    print(paste0('ar_b=',ar_b))
    print(paste0('fl_V=',fl_V))
  }

  # return
  return(list(F=fl_F, B=fl_B, C=fl_C, D=fl_D,
    W=fl_W, X=fl_X, Y=fl_Y, Z=fl_Z, V=fl_V))
}
```

9.2.3.1.2 Unrestricted Joint 2 by 2 Distribution Suppose there are two unique states for E and A. For example, suppose we know the unemployment probability for better or worse educated, and also for low and high age groups. We also know the proportion of people who are in each one of the four cells (regardless of unemployment or not). We also know the aggregate proportion of people in the population that is unemployed.

We want to know the joint probability of unemployment for the four types: both better educated and lower age, better educated and higher age, worse educated and lower age, and worse educated and higher age. Then:

$$\begin{aligned}
P(U|E_1) &= P(U|A_1,E_1)\cdot P(A_1|E_1) + P(U|A_2,E_1)\cdot P(A_2|E_1)\\
P(U|E_2) &= P(U|A_1,E_2)\cdot P(A_1|E_2) + P(U|A_2,E_2)\cdot P(A_2|E_2)\\
P(U|A_1) &= P(U|A_1,E_1)\cdot P(E_1|A_1) + P(U|A_1,E_2)\cdot P(E_2|A_1)\\
P(U|A_2) &= P(U|A_2,E_1)\cdot P(E_1|A_2) + P(U|A_2,E_2)\cdot P(E_2|A_2)
\end{aligned}$$

And, we also have the extra equation:

$$P(U) = P(U|A_1,E_1)\cdot P(E_1,A_1)+P(U|A_2,E_1)\cdot P(E_2,A_1)+P(U|A_1,E_2)\cdot P(E_1,A_2)+P(U|A_2,E_2)\cdot P(E_2,A_2)$$

We know $P(A|E)$, and we know $P(U|E)$ as well as $P(U|A)$. Let Roman letter represent what we know, and greek letters represent what we do not know, then we have four equations and four unknowns in a linear system. We use the letter F because A and E are taken.

$$\begin{aligned}
W &= \alpha F + \beta(1-F)\\
X &= \gamma B + \delta(1-B)\\
Y &= \alpha C + \gamma(1-C)\\
Z &= \beta D + \delta(1-D)
\end{aligned}$$

And we have also:

$$V = \alpha P(E_1, A_1) + \beta P(E_2, A_1) + \gamma P(E_1, A_2) + \delta P(E_2, A_2)$$

There are no unique solutions for the linear system. When we write the linear system above in matrix form as shown below, the Ω coefficient matrix is not full rank.

$$\begin{bmatrix} F & (1-F) & 0 & 0 \\ 0 & 0 & B & (1-B) \\ C & 0 & (1-C) & 0 \\ 0 & D & 0 & (1-D) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix}$$

$$\Omega \cdot \mathbb{X} = b$$

Get data from the data generation functions created prior:

```
# JOint mass of population at all cells
ls_ffi_rand_joint_mass <- ffi_gen_rand_joint_mass(it_seed = 123, it_Q = 2, it_M = 2, bl_verbose = TR
```

```
## [1] "ls_prob_joint_E_A"
##    A1        A2
## E1 0.1214495 0.1727188
## E2 0.3329164 0.3729152
##           [,1]      [,2]
## [1,] 0.1214495 0.1727188
## [2,] 0.3329164 0.3729152
```

```
mt_prob_joint_E_A <- ls_ffi_rand_joint_mass$mt_prob_joint_E_A
# retlinear restrictions for conditional unemployment probabilities
fl_Delta_A_true <- 0.05
fl_Delta_E_true <- 0.11
fl_alpha_true <- 0.25
fl_beta_true <- fl_alpha_true + fl_Delta_A_true
fl_gamma_true <- fl_alpha_true + fl_Delta_E_true
fl_delta_true <- fl_alpha_true + fl_Delta_E_true + fl_Delta_A_true
ls_FBCDWXYZV <- ffi_gen_condi_unemploy_prob_2by2(mt_prob_joint_E_A,
  fl_alpha = fl_alpha_true, fl_beta = fl_beta_true,
  fl_gamma = fl_gamma_true, fl_delta = fl_delta_true, bl_verbose=TRUE)
```

```
## [1] "fl_F=0.412857205138471,fl_B=0.471665472604598,fl_C=0.267294503388642,fl_D=0.316546995323062"
## [1] "ar_b=0.279357139743076" "ar_b=0.38641672636977"  "ar_b=0.330597604627249" "ar_b=0.3751798305
## [1] "fl_V=0.354923185445584"
```

```
fl_F <- ls_FBCDWXYZV$F
fl_B <- ls_FBCDWXYZV$B
fl_C <- ls_FBCDWXYZV$C
fl_D <- ls_FBCDWXYZV$D
fl_W <- ls_FBCDWXYZV$W
fl_X <- ls_FBCDWXYZV$X
fl_Y <- ls_FBCDWXYZV$Y
fl_Z <- ls_FBCDWXYZV$Z
fl_V <- ls_FBCDWXYZV$V
```

We can see the rank of a matrix with the qr function (QR decomposition), regardless of the random seed chosen above, the Ω matrix is not full ranked.

```
# does not matter if joint sum is used or the alternative
bl_use_joint_sum <- FALSE
if (bl_use_joint_sum) {
  # Construct The coefficent Matrix
  mt_OMEGA = t(
    matrix(data=c(fl_F, 1-fl_F, 0, 0,
```

```
                  0, 0, fl_B, 1-fl_B,
                  fl_C, 0, 1-fl_C, 0,
                  mt_prob_joint_E_A[1,1],  mt_prob_joint_E_A[1,2],
                  mt_prob_joint_E_A[2,1],  mt_prob_joint_E_A[2,2]), nrow=4, ncol=4))
} else {
  # Construct The coefficent Matrix
  mt_OMEGA = t(matrix(data=c(fl_F, 1-fl_F, 0, 0,
                             0, 0, fl_B, 1-fl_B,
                             fl_C, 0, 1-fl_C, 0,
                             0, fl_D, 0, 1-fl_D), nrow=4, ncol=4))

}
# rank Check with the qr function:
print(qr(mt_OMEGA))
```

```
## $qr
##            [,1]       [,2]       [,3]          [,4]
## [1,] -0.4918307 -0.4928650 -0.3982024  0.000000e+00
## [2,]  0.0000000 -0.4494694  0.4366482 -4.813342e-01
## [3,]  0.5434685 -0.7099340 -0.6403896 -7.173303e-01
## [4,]  0.0000000  0.7042682 -0.0385102 -2.081668e-17
##
## $rank
## [1] 3
##
## $qraux
## [1] 1.839430e+00 1.000000e+00 1.999258e+00 2.081668e-17
##
## $pivot
## [1] 1 2 3 4
##
## attr(,"class")
## [1] "qr"
```

```
# rank Check with the qr function:
```

We cannot solve the linear equations using *solve* because this is NOT full rank.

9.2.3.1.3 Rectilinear Restriction on Joint 2 by 2 Distribution So in the section above, it is demonstrated that it is not possible to uniquely identify the joint probability mass function from marginal probability mass functions.

However, sometimes, we need to find some reasonable joint distribution, when we only observe marginal distributions. This joint distribution might be an input transition matrix in a model we simulate. If we just use one of the infinitely possible joint mass that match up with the marginals, then the model would have infinitely many simulation results depending on our arbitrary choice of joint mass.

Ideally, one should try to obtain data to estimate the underlying joint distribution, when this is not possible, we can impose additional non-parametric restrictions on the structures of the joint probability mass that would lead to unique joint mass from marginals.

Specifically, we will assume the incremental changes across rows and across columns of the joint mass matrix are row or column specific, is this sufficient? (In Some Cases it Will Not be):

$$\Delta_{12}^{E} = P(A_1, E_2) - P(A_1, E_1) = \gamma - \alpha = P(A_2, E_2) - P(A_2, E_1) = \delta - \beta$$
$$\Delta_{12}^{A} = P(A_2, E_1) - P(A_1, E_1) = \beta - \alpha = P(A_2, E_2) - P(A_1, E_2) = \delta - \gamma$$

The assumption is non-parametric. This is effectively an rectilinear assumption on the joint Cumulative Probability Mass Function. The assumption means that if we know Δ_{12}^{E} and Δ_{12}^{A} and say α, we have:

$$\beta = \alpha + \Delta_{12}^{A}\gamma = \alpha + \Delta_{12}^{E}\delta = \gamma\left(\alpha, \Delta_{12}^{E}\right) + \Delta_{12}^{A}$$

Alernatively, we can also say that if we know α, β and γ, then we know δ:

$$\Delta_{12}^{A} = \beta - \alpha\delta = \gamma + \Delta_{12}^{A} = \gamma + \beta - \alpha$$

9.2.3.1.4 2 by 2 Problem with Rectlinear Restrictions We have these three equations, where α, Δ^A and Δ^E are not known:

$$\begin{aligned} W &= \alpha F + \alpha(1-F) + \Delta^A(1-F) \\ X &= \alpha B + \Delta^E B + \alpha(1-B) + \Delta^A(1-B) + \Delta^E(1-B) \\ Y &= \alpha C + \alpha(1-C) + \Delta^E(1-C) \end{aligned}$$

Rewriting a little bit, we have the following linear system:

$$\begin{aligned} W &= \alpha + \Delta^A(1-F) \\ X &= \alpha + \Delta^A(1-B) + \Delta^E \\ Y &= \alpha + \Delta^E(1-C) \\ V &= \alpha + \Delta^A\left(P(E_2, A_1) + P(E_2, A_2)\right) + \Delta^E\left(P(E_1, A_2) + P(E_2, A_2)\right) \end{aligned}$$

Using the First three equations, we have:

$$\begin{bmatrix} 1 & (1-F) & 0 \\ 1 & (1-B) & 1 \\ 1 & 0 & (1-C) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \Delta^A \\ \Delta^E \end{bmatrix} = \begin{bmatrix} W \\ X \\ Y \end{bmatrix}$$

$$\widehat{\Omega} \cdot \widehat{\mathbb{X}} = \hat{b}$$

9.2.3.1.5 Solution Program for 2 by 2 Problem with Rectlinear Restrictions Using the same F, B and C values obtained from prior, we have now a full ranked 3 by 3 matrix:

```
ffi_solve_2by2_rectilinear <- function(fl_F, fl_B, fl_C,
                                       fl_W, fl_X, fl_Y,
                                       bl_verbose=FALSE) {

  # Construct The coefficent Matrix
  mt_OMEGA_hat = t(matrix(
    data=c(1, 1-fl_F, 0,
           1, 1-fl_B, 1,
           1, 0, 1-fl_C), nrow=3, ncol=3))

  # rank Check with the qr function:
  if (bl_verbose) {
    print(qr(mt_OMEGA_hat))
  }

  # bhat
  ar_b_hat <- c(fl_W, fl_X, fl_Y)

  # solve
  ar_solution <- solve(mt_OMEGA_hat, ar_b_hat)

  # Get alpha and Delta from solution
  fl_alpha <- ar_solution[1]
  fl_Delta_A <- ar_solution[2]
  fl_Delta_E <- ar_solution[3]
  if (bl_verbose) {
```

```
    print(paste0('fl_Delta_A=',fl_Delta_A))
    print(paste0('fl_Delta_E=',fl_Delta_E))
  }

  # Get beta gamma , delta
  fl_beta <- fl_alpha + fl_Delta_A
  fl_gamma <- fl_alpha + fl_Delta_E
  fl_delta <- fl_gamma + fl_Delta_A
  if (bl_verbose) {
    print(paste0('fl_alpha=',fl_alpha))
    print(paste0('fl_beta=',fl_beta))
    print(paste0('fl_gamma=',fl_gamma))
    print(paste0('fl_delta=',fl_delta))
    if (abs(ar_b[1] - (fl_alpha*fl_F + fl_beta*(1-fl_F))) < 1e-10) {
      print('W matched')
    }
    if (abs(ar_b[2] - (fl_gamma*fl_B + fl_delta*(1-fl_B))) < 1e-10) {
      print('X matched')
    }
    if (abs(ar_b[3] - (fl_alpha*fl_C + fl_gamma*(1-fl_C))) < 1e-10) {
      print('Y matched')
    }
    if (abs(ar_b[4] - (fl_beta*fl_D + fl_delta*(1-fl_D))) < 1e-10) {
      print('Z matched')
    }
    if (abs(fl_V - (mt_prob_joint_E_A[1,1]*fl_alpha +
                    mt_prob_joint_E_A[1,2]*fl_beta +
                    mt_prob_joint_E_A[2,1]*fl_gamma +
                    mt_prob_joint_E_A[2,2]*fl_delta)) < 1e-10) {
      print('V matched')
    }
  }

  return(list(Delta_A=fl_Delta_A, Delta_E=fl_Delta_E,
              alpha=fl_alpha, beta=fl_beta,
              gamma=fl_gamma, delta=fl_delta))
}
```

Now solve for $\widehat{\mathbb{X}}$ using *solve* given $\widehat{\Omega}$ and $\hat{b}$. Solving different assumptions on underlying joint probabilities.

9.2.3.1.6 Testing Program for 2 by 2 Problem with Rectlinear Restrictions Solving first when rectilinear assumption is valid:

```
for (it_i in c(1,2,3)) {

  # JOint mass of population at all cells
  ls_ffi_rand_joint_mass <- ffi_gen_rand_joint_mass(
    it_seed = 123, it_Q = 2, it_M = 2, bl_verbose = FALSE)
  mt_prob_joint_E_A <- ls_ffi_rand_joint_mass$mt_prob_joint_E_A

  # retlinear restrictions for conditional unemployment probabilities
  fl_alpha_true <- 0.25
  if (it_i == 1) {
    fl_Delta_A_true <- 0.05
    fl_Delta_E_true <- 0.11
  } else if (it_i == 2) {
    fl_Delta_A_true <- 0.21
    fl_Delta_E_true <- 0.03
```

```
  } else if (it_i == 3) {
    fl_Delta_A_true <- -0.05
    fl_Delta_E_true <- -0.11
  }

  fl_beta_true <- fl_alpha_true + fl_Delta_A_true
  fl_gamma_true <- fl_alpha_true + fl_Delta_E_true
  fl_delta_true <- fl_alpha_true + fl_Delta_E_true + fl_Delta_A_true
  ls_FBCDWXYZV <- ffi_gen_condi_unemploy_prob_2by2(
    mt_prob_joint_E_A,
    fl_alpha = fl_alpha_true, fl_beta = fl_beta_true,
    fl_gamma = fl_gamma_true, fl_delta = fl_delta_true, bl_verbose=FALSE)

  # call solution function
  fl_F <- ls_FBCDWXYZV$F
  fl_B <- ls_FBCDWXYZV$B
  fl_C <- ls_FBCDWXYZV$C
  fl_D <- ls_FBCDWXYZV$D
  fl_W <- ls_FBCDWXYZV$W
  fl_X <- ls_FBCDWXYZV$X
  fl_Y <- ls_FBCDWXYZV$Y
  fl_Z <- ls_FBCDWXYZV$Z
  fl_V <- ls_FBCDWXYZV$V
  ls_solution <- ffi_solve_2by2_rectilinear(
    fl_F, fl_B, fl_C,
    fl_W, fl_X, fl_Y)

  fl_alpha <- ls_solution$alpha
  fl_beta <- ls_solution$beta
  fl_gamma <- ls_solution$gamma
  fl_delta <- ls_solution$delta

  # check
  print(paste0('it_i=', it_i))
  print(paste0('fl_alpha=', fl_alpha, ', fl_alpha_true=', fl_alpha_true))
  print(paste0('fl_beta=', fl_beta, ', fl_beta_true=', fl_beta_true))
  print(paste0('fl_gamma=', fl_gamma, ', fl_gamma_true=', fl_gamma_true))
  print(paste0('fl_delta=', fl_delta, ', fl_delta_true=', fl_delta_true))
}
```

```
## [1] "it_i=1"
## [1] "fl_alpha=0.25, fl_alpha_true=0.25"
## [1] "fl_beta=0.3, fl_beta_true=0.3"
## [1] "fl_gamma=0.36, fl_gamma_true=0.36"
## [1] "fl_delta=0.41, fl_delta_true=0.41"
## [1] "it_i=2"
## [1] "fl_alpha=0.25, fl_alpha_true=0.25"
## [1] "fl_beta=0.46, fl_beta_true=0.46"
## [1] "fl_gamma=0.28, fl_gamma_true=0.28"
## [1] "fl_delta=0.49, fl_delta_true=0.49"
## [1] "it_i=3"
## [1] "fl_alpha=0.25, fl_alpha_true=0.25"
## [1] "fl_beta=0.2, fl_beta_true=0.2"
## [1] "fl_gamma=0.14, fl_gamma_true=0.14"
## [1] "fl_delta=0.0900000000000001, fl_delta_true=0.09"
```

Solving when rectilinear assumption is NOT valid, results can be approximately correct. However, note that we do not have positive mass in all cases. See case three below, the alpha backed out is negative.

```
for (it_i in c(1,2,3)) {

  # JOint mass of population at all cells
  ls_ffi_rand_joint_mass <- ffi_gen_rand_joint_mass(
    it_seed = 123, it_Q = 2, it_M = 2, bl_verbose = FALSE)
  mt_prob_joint_E_A <- ls_ffi_rand_joint_mass$mt_prob_joint_E_A

  # retlinear restrictions for conditional unemployment probabilities
  if (it_i == 1) {
    fl_alpha_true <- 0.25
    fl_Delta_A_true <- 0.05
    fl_Delta_E_true <- 0.11
    fl_Delta_AE_true <- 0.10
  } else if (it_i == 2) {
    fl_alpha_true <- 0.55
    fl_Delta_A_true <- 0.21
    fl_Delta_E_true <- 0.03
    fl_Delta_AE_true <- 0.15
  } else if (it_i == 3) {
    fl_alpha_true <- 0.15
    fl_Delta_A_true <- -0.05
    fl_Delta_E_true <- -0.11
    fl_Delta_AE_true <- 0.30
  }

  fl_beta_true <- fl_alpha_true + fl_Delta_A_true
  fl_gamma_true <- fl_alpha_true + fl_Delta_E_true
  fl_delta_true <- fl_alpha_true + fl_Delta_AE_true
  ls_FBCDWXYZV <- ffi_gen_condi_unemploy_prob_2by2(
    mt_prob_joint_E_A,
    fl_alpha = fl_alpha_true, fl_beta = fl_beta_true,
    fl_gamma = fl_gamma_true, fl_delta = fl_delta_true, bl_verbose=FALSE)

  # call solution function
  fl_F <- ls_FBCDWXYZV$F
  fl_B <- ls_FBCDWXYZV$B
  fl_C <- ls_FBCDWXYZV$C
  fl_D <- ls_FBCDWXYZV$D
  fl_W <- ls_FBCDWXYZV$W
  fl_X <- ls_FBCDWXYZV$X
  fl_Y <- ls_FBCDWXYZV$Y
  fl_Z <- ls_FBCDWXYZV$Z
  fl_V <- ls_FBCDWXYZV$V
  ls_solution <- ffi_solve_2by2_rectilinear(
    fl_F, fl_B, fl_C,
    fl_W, fl_X, fl_Y)

  fl_alpha <- ls_solution$alpha
  fl_beta <- ls_solution$beta
  fl_gamma <- ls_solution$gamma
  fl_delta <- ls_solution$delta

  # check
  print(paste0('it_i=', it_i))
  print(paste0('fl_alpha=', fl_alpha, ', fl_alpha_true=', fl_alpha_true))
  print(paste0('fl_beta=', fl_beta, ', fl_beta_true=', fl_beta_true))
  print(paste0('fl_gamma=', fl_gamma, ', fl_gamma_true=', fl_gamma_true))
```

```
  print(paste0('fl_delta=', fl_delta, ', fl_delta_true=', fl_delta_true))
}
```

```
## [1] "it_i=1"
## [1] "fl_alpha=0.275066384524696, fl_alpha_true=0.25"
## [1] "fl_beta=0.282374240902959, fl_beta_true=0.3"
## [1] "fl_gamma=0.350855661880163, fl_gamma_true=0.36"
## [1] "fl_delta=0.358163518258426, fl_delta_true=0.35"
## [1] "it_i=2"
## [1] "fl_alpha=0.587599576787045, fl_alpha_true=0.55"
## [1] "fl_beta=0.733561361354439, fl_beta_true=0.76"
## [1] "fl_gamma=0.566283492820244, fl_gamma_true=0.58"
## [1] "fl_delta=0.712245277387639, fl_delta_true=0.7"
## [1] "it_i=3"
## [1] "fl_alpha=-0.0421756146893388, fl_alpha_true=0.15"
## [1] "fl_beta=0.235130819743977, fl_beta_true=0.1"
## [1] "fl_gamma=0.110106592252085, fl_gamma_true=0.04"
## [1] "fl_delta=0.387413026685401, fl_delta_true=0.45"
```

9.2.3.2 3 by 3 Joint from Marginal Probability Mass Functions with Rectilinear Assumptions

9.2.3.2.1 The 3 by 3 Problem We append the 2 by 2 problem to a 3 by 3 problem.

$$
\begin{aligned}
P(U|E_1) &= P(U|A_1, E_1) \cdot P(A_1|E_1) + P(U|A_2, E_1) \cdot P(A_2|E_1) + P(U|A_3, E_1) \cdot P(A_3|E_1) \\
P(U|E_2) &= P(U|A_1, E_2) \cdot P(A_1|E_2) + P(U|A_2, E_2) \cdot P(A_2|E_2) + P(U|A_3, E_2) \cdot P(A_3|E_2) \\
P(U|E_3) &= P(U|A_1, E_3) \cdot P(A_1|E_3) + P(U|A_2, E_3) \cdot P(A_2|E_3) + P(U|A_3, E_3) \cdot P(A_3|E_3) \\
P(U|A_1) &= P(U|A_1, E_1) \cdot P(E_1|A_1) + P(U|A_1, E_2) \cdot P(E_2|A_1) + P(U|A_1, E_3) \cdot P(E_3|A_1) \\
P(U|A_2) &= P(U|A_2, E_1) \cdot P(E_1|A_2) + P(U|A_2, E_2) \cdot P(E_2|A_2) + P(U|A_2, E_3) \cdot P(E_3|A_2) \\
P(U|A_3) &= P(U|A_3, E_1) \cdot P(E_1|A_3) + P(U|A_3, E_2) \cdot P(E_2|A_3) + P(U|A_3, E_3) \cdot P(E_3|A_3)
\end{aligned}
$$

Similar to before, let Roman letter represent what we know, and greek letters represent what we do not know. There are nine potential unknown conditional unemployment probabilities α_{ij}.

$$
\begin{aligned}
V_1 &= \alpha_{11} B_{11} + \alpha_{21} B_{21} + \alpha_{31} B_{31} \\
V_2 &= \alpha_{12} B_{12} + \alpha_{22} B_{22} + \alpha_{32} B_{32} \\
V_3 &= \alpha_{13} B_{13} + \alpha_{23} B_{23} + \alpha_{33} B_{33} \\
W_1 &= \alpha_{11} C_{11} + \alpha_{12} C_{21} + \alpha_{13} C_{31} \\
W_2 &= \alpha_{21} C_{12} + \alpha_{22} C_{22} + \alpha_{23} C_{32} \\
W_3 &= \alpha_{31} C_{13} + \alpha_{32} C_{23} + \alpha_{33} C_{33}
\end{aligned}
$$

9.2.3.2.2 3 by 3 Problem with Rectlinear Restrictions With rectilinear assumptions, we have now rather than nine parameters, five parameters.

$$
\begin{aligned}
V_1 &= \alpha B_{11} + (\alpha + \delta^{21}) B_{21} + (\alpha + \delta^{21} + \delta^{31}) B_{31} \\
V_2 &= (\alpha + \delta^{12}) B_{12} + (\alpha + \delta^{12} + \delta^{21}) B_{22} + (\alpha + \delta^{12} + \delta^{21} + \delta^{31}) B_{32} \\
V_3 &= (\alpha + \delta^{12} + \delta^{13}) B_{13} + (\alpha + \delta^{12} + \delta^{13} + \delta^{21}) B_{23} + (\alpha + \delta^{12} + \delta^{13} + \delta^{21} + \delta^{31}) B_{33} \\
W_1 &= \alpha C_{11} + (\alpha + \delta^{12}) C_{21} + (\alpha + \delta^{12} + \delta^{13}) C_{31} \\
W_2 &= (\alpha + \delta^{21}) C_{12} + (\alpha + \delta^{12} + \delta^{21}) C_{22} + (\alpha + \delta^{12} + \delta^{13} + \delta^{21}) C_{32} \\
W_3 &= (\alpha + \delta^{21} + \delta^{31}) C_{13} + (\alpha + \delta^{12} + \delta^{21} + \delta^{31}) C_{23} + (\alpha + \delta^{12} + \delta^{13} + \delta^{21} + \delta^{31}) C_{33}
\end{aligned}
$$

Write these in matrix form:

$$\begin{bmatrix} (B_{11}+B_{21}+B_{31}) & (B_{21}+B_{31}) & (B_{31}) & 0 & 0 \\ (B_{12}+B_{22}+B_{32}) & (B_{22}+B_{32}) & (B_{32}) & (B_{12}+B_{22}+B_{32}) & 0 \\ (B_{13}+B_{23}+B_{33}) & (B_{23}+B_{33}) & (B_{33}) & (B_{13}+B_{23}+B_{33}) & (B_{13}+B_{23}+B_{33}) \\ (C_{11}+C_{21}+C_{31}) & 0 & 0 & (C_{21}+C_{31}) & (C_{31}) \\ (C_{12}+C_{22}+C_{32}) & (C_{12}+C_{22}+C_{32}) & 0 & (C_{22}+C_{32}) & (C_{32}) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \delta^{21} \\ \delta^{31} \\ \delta^{12} \\ \delta^{13} \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ W_1 \\ W_2 \end{bmatrix}$$

$$\widehat{\Omega} \cdot \widehat{\mathbb{X}} = \hat{b}$$

Simplify slightly for where the probabilities sum to 1, which arrives at a matrix form that is similar to the matrix form arrived at for the 2 by 2 problem.

$$\begin{bmatrix} 1 & (B_{21}+B_{31}) & (B_{31}) & 0 & 0 \\ 1 & (B_{22}+B_{32}) & (B_{32}) & 1 & 0 \\ 1 & (B_{23}+B_{33}) & (B_{33}) & 1 & 1 \\ 1 & 0 & 0 & (C_{21}+C_{31}) & (C_{31}) \\ 1 & 1 & 0 & (C_{22}+C_{32}) & (C_{32}) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \delta^{21} \\ \delta^{31} \\ \delta^{12} \\ \delta^{13} \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ W_1 \\ W_2 \end{bmatrix}$$

$$\widehat{\Omega} \cdot \widehat{\mathbb{X}} = \hat{b}$$

```
ffi_gen_condi_unemploy_prob_3by3 <- function(mt_prob_joint_E_A, mt_alpha, bl_verbose = FALSE) {

  # From joint probability, generate conditional probabilities
  fl_B21 <- mt_prob_joint_E_A[1,2]/sum(mt_prob_joint_E_A[1,])
  fl_B31 <- mt_prob_joint_E_A[1,3]/sum(mt_prob_joint_E_A[1,])

  fl_B22 <- mt_prob_joint_E_A[2,2]/sum(mt_prob_joint_E_A[2,])
  fl_B32 <- mt_prob_joint_E_A[2,3]/sum(mt_prob_joint_E_A[2,])

  fl_B23 <- mt_prob_joint_E_A[3,2]/sum(mt_prob_joint_E_A[3,])
  fl_B33 <- mt_prob_joint_E_A[3,3]/sum(mt_prob_joint_E_A[3,])

  fl_C21 <- mt_prob_joint_E_A[2,1]/sum(mt_prob_joint_E_A[,1])
  fl_C31 <- mt_prob_joint_E_A[3,1]/sum(mt_prob_joint_E_A[,1])

  fl_C22 <- mt_prob_joint_E_A[2,2]/sum(mt_prob_joint_E_A[,2])
  fl_C32 <- mt_prob_joint_E_A[3,2]/sum(mt_prob_joint_E_A[,2])

  if (bl_verbose) {
    print(paste0('fl_B21=', fl_B21, ',fl_B31=',fl_B31))
    print(paste0('fl_B22=', fl_B22, ',fl_B22=',fl_B22))
    print(paste0('fl_B23=', fl_B23, ',fl_B33=',fl_B33))
    print(paste0('fl_C21=', fl_C21, ',fl_C31=',fl_C31))
    print(paste0('fl_C22=', fl_C22, ',fl_C32=',fl_C32))
  }

  # Generate random W X Y
  fl_V1 <- mt_alpha[1,1]*(1 - fl_B21 - fl_B31) + mt_alpha[2,1]*fl_B21 + mt_alpha[3,1]*fl_B31
  fl_V2 <- mt_alpha[1,2]*(1 - fl_B22 - fl_B32) + mt_alpha[2,2]*fl_B22 + mt_alpha[3,2]*fl_B32
  fl_V3 <- mt_alpha[1,3]*(1 - fl_B23 - fl_B33) + mt_alpha[2,3]*fl_B23 + mt_alpha[3,3]*fl_B33
  fl_W1 <- mt_alpha[1,1]*(1 - fl_C21 - fl_C31) + mt_alpha[1,2]*fl_C21 + mt_alpha[1,3]*fl_C31
  fl_W2 <- mt_alpha[2,1]*(1 - fl_C22 - fl_C32) + mt_alpha[2,2]*fl_C22 + mt_alpha[2,3]*fl_C32
  ar_b <- c(fl_V1, fl_V2, fl_V3, fl_W1, fl_W2)

  if (bl_verbose) {
    print(paste0('ar_b=',ar_b))
  }
```

```
  # return
  return(ar_b)
}
```

9.2.3.2.3 Data Program for 3 by 3 Problem with Rectlinear Restrictions

```
ffi_solve_3by3_rectilinear <- function(mt_prob_joint_E_A, ar_b, bl_verbose=FALSE) {

  # From joint probability, generate conditional probabilities
  fl_B21 <- mt_prob_joint_E_A[1,2]/sum(mt_prob_joint_E_A[1,])
  fl_B31 <- mt_prob_joint_E_A[1,3]/sum(mt_prob_joint_E_A[1,])

  fl_B22 <- mt_prob_joint_E_A[2,2]/sum(mt_prob_joint_E_A[2,])
  fl_B32 <- mt_prob_joint_E_A[2,3]/sum(mt_prob_joint_E_A[2,])

  fl_B23 <- mt_prob_joint_E_A[3,2]/sum(mt_prob_joint_E_A[3,])
  fl_B33 <- mt_prob_joint_E_A[3,3]/sum(mt_prob_joint_E_A[3,])

  fl_C21 <- mt_prob_joint_E_A[2,1]/sum(mt_prob_joint_E_A[,1])
  fl_C31 <- mt_prob_joint_E_A[3,1]/sum(mt_prob_joint_E_A[,1])

  fl_C22 <- mt_prob_joint_E_A[2,2]/sum(mt_prob_joint_E_A[,2])
  fl_C32 <- mt_prob_joint_E_A[3,2]/sum(mt_prob_joint_E_A[,2])

  # Construct The coefficent Matrix
  mt_OMEGA = t(matrix(
    data=c(1, fl_B21+fl_B31, fl_B31, 0, 0,
           1, fl_B22+fl_B32, fl_B32, 1, 0,
           1, fl_B23+fl_B33, fl_B33, 1, 1,
           1, 0, 0, fl_C21+fl_C31, fl_C31,
           1, 1, 0, fl_C22+fl_C32, fl_C32), nrow=5, ncol=5))

  # rank Check with the qr function:
  if (bl_verbose) {
    print(qr(mt_OMEGA_hat))
  }

  # solve
  ar_solution <- solve(mt_OMEGA, ar_b)

  # Get alpha and Delta from solution
  fl_alpha <- ar_solution[1]
  fl_Delta_21 <- ar_solution[2]
  fl_Delta_31 <- ar_solution[3]
  fl_Delta_12 <- ar_solution[4]
  fl_Delta_13 <- ar_solution[5]
  if (bl_verbose) {
    print(paste0('fl_Delta_21=',fl_Delta_21))
    print(paste0('fl_Delta_31=',fl_Delta_31))
    print(paste0('fl_Delta_12=',fl_Delta_12))
    print(paste0('fl_Delta_13=',fl_Delta_13))
  }

  # Get beta gamma , delta
  mt_alpha = matrix(0, 3, 3)
  for (it_row in c(1,2,3)) {
```

```
    for (it_col in c(1,2,3)) {

      fl_current_val <- fl_alpha

      if (it_row == 2) {
        fl_current_val <- fl_current_val + fl_Delta_21
      }
      if (it_row == 3) {
        fl_current_val <- fl_current_val + fl_Delta_21 + fl_Delta_31
      }
      if (it_col == 2) {
        fl_current_val <- fl_current_val + fl_Delta_12
      }
      if (it_col == 3) {
        fl_current_val <- fl_current_val + fl_Delta_12 + fl_Delta_13
      }

      mt_alpha[it_row, it_col] <- fl_current_val
    }
  }


  return(mt_alpha)
}
```

9.2.3.2.4 Solution Program for 3 by 3 Problem with Rectlinear Restrictions

9.2.3.2.5 Testing Program for 3 by 3 Problem with Rectlinear Restrictions Solving first when rectilinear assumption is valid:

```
for (it_i in c(1,2,3)) {

  # JOint mass of population at all cells
  ls_ffi_rand_joint_mass <- ffi_gen_rand_joint_mass(
    it_seed = 123, it_Q = 3, it_M = 3, bl_verbose = FALSE)
  mt_prob_joint_E_A <- ls_ffi_rand_joint_mass$mt_prob_joint_E_A

  # retlinear restrictions for conditional unemployment probabilities
  if (it_i == 1) {
    fl_alpha <- 0.1
    fl_Delta_21 <- 0.05
    fl_Delta_31 <- 0.07
    fl_Delta_12 <- 0.09
    fl_Delta_13 <- 0.03
  } else if (it_i == 2) {
    fl_alpha <- 0.20
    fl_Delta_21 <- -0.05
    fl_Delta_31 <- +0.07
    fl_Delta_12 <- 0.09
    fl_Delta_13 <- -0.03
  } else if (it_i == 3) {
    fl_alpha <- 0.15
    fl_Delta_21 <- 0.11
    fl_Delta_31 <- -0.01
    fl_Delta_12 <- 0
    fl_Delta_13 <- +0.1
  }
```

```
  mt_alpha_true = matrix(0, 3, 3)
  for (it_row in c(1,2,3)) {
    for (it_col in c(1,2,3)) {

      fl_current_val <- fl_alpha

      if (it_row == 2) {
        fl_current_val <- fl_current_val + fl_Delta_21
      }
      if (it_row == 3) {
        fl_current_val <- fl_current_val + fl_Delta_21 + fl_Delta_31
      }
      if (it_col == 2) {
        fl_current_val <- fl_current_val + fl_Delta_12
      }
      if (it_col == 3) {
        fl_current_val <- fl_current_val + fl_Delta_12 + fl_Delta_13
      }

      mt_alpha_true[it_row, it_col] <- fl_current_val
    }
  }

  ar_b <- ffi_gen_condi_unemploy_prob_3by3(mt_prob_joint_E_A, mt_alpha_true, bl_verbose=FALSE)

  # call solution function
  mt_alpha_solu <- ffi_solve_3by3_rectilinear(mt_prob_joint_E_A, ar_b)

  # check
  print(paste0('it_i=', it_i))
  print('mt_alpha_true')
  print(mt_alpha_true)
  print('mt_alpha_solu')
  print(mt_alpha_solu)

}
```

```
## [1] "it_i=1"
## [1] "mt_alpha_true"
##      [,1] [,2] [,3]
## [1,] 0.10 0.19 0.22
## [2,] 0.15 0.24 0.27
## [3,] 0.22 0.31 0.34
## [1] "mt_alpha_solu"
##      [,1] [,2] [,3]
## [1,] 0.10 0.19 0.22
## [2,] 0.15 0.24 0.27
## [3,] 0.22 0.31 0.34
## [1] "it_i=2"
## [1] "mt_alpha_true"
##      [,1] [,2] [,3]
## [1,] 0.20 0.29 0.26
## [2,] 0.15 0.24 0.21
## [3,] 0.22 0.31 0.28
## [1] "mt_alpha_solu"
##      [,1] [,2] [,3]
## [1,] 0.20 0.29 0.26
## [2,] 0.15 0.24 0.21
```

```
## [3,] 0.22 0.31 0.28
## [1] "it_i=3"
## [1] "mt_alpha_true"
##      [,1] [,2] [,3]
## [1,] 0.15 0.15 0.25
## [2,] 0.26 0.26 0.36
## [3,] 0.25 0.25 0.35
## [1] "mt_alpha_solu"
##      [,1] [,2] [,3]
## [1,] 0.15 0.15 0.25
## [2,] 0.26 0.26 0.36
## [3,] 0.25 0.25 0.35
```

Solving first when rectilinear assumption is NOT VALID, same as above, except add randomness at each point. Note that the solution is approximately the same as the actual in the examples here.

```
# set seed
set.seed(123)
# loop over alternatives
for (it_i in c(1,2,3)) {

  # JOint mass of population at all cells
  ls_ffi_rand_joint_mass <- ffi_gen_rand_joint_mass(
    it_seed = 123, it_Q = 3, it_M = 3, bl_verbose = FALSE)
  mt_prob_joint_E_A <- ls_ffi_rand_joint_mass$mt_prob_joint_E_A

  # retlinear restrictions for conditional unemployment probabilities
  if (it_i == 1) {
    fl_alpha <- 0.1
    fl_Delta_21 <- 0.05
    fl_Delta_31 <- 0.07
    fl_Delta_12 <- 0.09
    fl_Delta_13 <- 0.03
  } else if (it_i == 2) {
    fl_alpha <- 0.20
    fl_Delta_21 <- -0.05
    fl_Delta_31 <- +0.07
    fl_Delta_12 <- 0.09
    fl_Delta_13 <- -0.03
  } else if (it_i == 3) {
    fl_alpha <- 0.15
    fl_Delta_21 <- 0.11
    fl_Delta_31 <- -0.01
    fl_Delta_12 <- 0
    fl_Delta_13 <- +0.1
  }

  mt_alpha_true = matrix(0, 3, 3)
  for (it_row in c(1,2,3)) {
    for (it_col in c(1,2,3)) {

      fl_current_val <- fl_alpha

      if (it_row == 2) {
        fl_current_val <- fl_current_val + fl_Delta_21 + runif(1)*0.2
      }
      if (it_row == 3) {
        fl_current_val <- fl_current_val + fl_Delta_21 + fl_Delta_31 + runif(1)*0.2
      }
```

```
      if (it_col == 2) {
        fl_current_val <- fl_current_val + fl_Delta_12 + runif(1)*0.2
      }
      if (it_col == 3) {
        fl_current_val <- fl_current_val + fl_Delta_12 + fl_Delta_13 + runif(1)*0.2
      }

      mt_alpha_true[it_row, it_col] <- fl_current_val
    }
  }

  ar_b <- ffi_gen_condi_unemploy_prob_3by3(mt_prob_joint_E_A, mt_alpha_true, bl_verbose=FALSE)

  # call solution function
  mt_alpha_solu <- ffi_solve_3by3_rectilinear(mt_prob_joint_E_A, ar_b)

  # check
  print(paste0('it_i=', it_i))
  print('mt_alpha_true')
  print(mt_alpha_true)
  print('mt_alpha_solu')
  print(mt_alpha_solu)

}
```

```
## [1] "it_i=1"
## [1] "mt_alpha_true"
##           [,1]      [,2]      [,3]
## [1,] 0.1000000 0.2813229 0.4113667
## [2,] 0.2406668 0.4900408 0.4705499
## [3,] 0.2692175 0.3839961 0.7088086
## [1] "mt_alpha_solu"
##            [,1]      [,2]      [,3]
## [1,] 0.06922205 0.2610615 0.4720626
## [2,] 0.26540144 0.4572409 0.6682420
## [3,] 0.24462011 0.4364596 0.6474607
## [1] "it_i=2"
## [1] "mt_alpha_true"
##           [,1]      [,2]      [,3]
## [1,] 0.2000000 0.3813229 0.4513667
## [2,] 0.2406668 0.4900408 0.4105499
## [3,] 0.2692175 0.3839961 0.6488086
## [1] "mt_alpha_solu"
##           [,1]      [,2]      [,3]
## [1,] 0.1692221 0.3610615 0.5120626
## [2,] 0.2654014 0.4572409 0.6082420
## [3,] 0.2446201 0.4364596 0.5874607
## [1] "it_i=3"
## [1] "mt_alpha_true"
##           [,1]      [,2]      [,3]
## [1,] 0.1500000 0.2413229 0.4413667
## [2,] 0.3506668 0.5100408 0.5605499
## [3,] 0.2992175 0.3239961 0.7188086
## [1] "mt_alpha_solu"
##           [,1]      [,2]      [,3]
## [1,] 0.1192221 0.2210615 0.5020626
## [2,] 0.3754014 0.4772409 0.7582420
## [3,] 0.2746201 0.3764596 0.6574607
```

Chapter 10

Tables and Graphs

10.1 R Base Plots

10.1.1 Plot Curve, Line and Points

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Work with the R plot function.

10.1.1.1 One Point, One Line and Two Curves

- r curve on top of plot
- r plot specify pch lty both scatter and line
- r legend outside

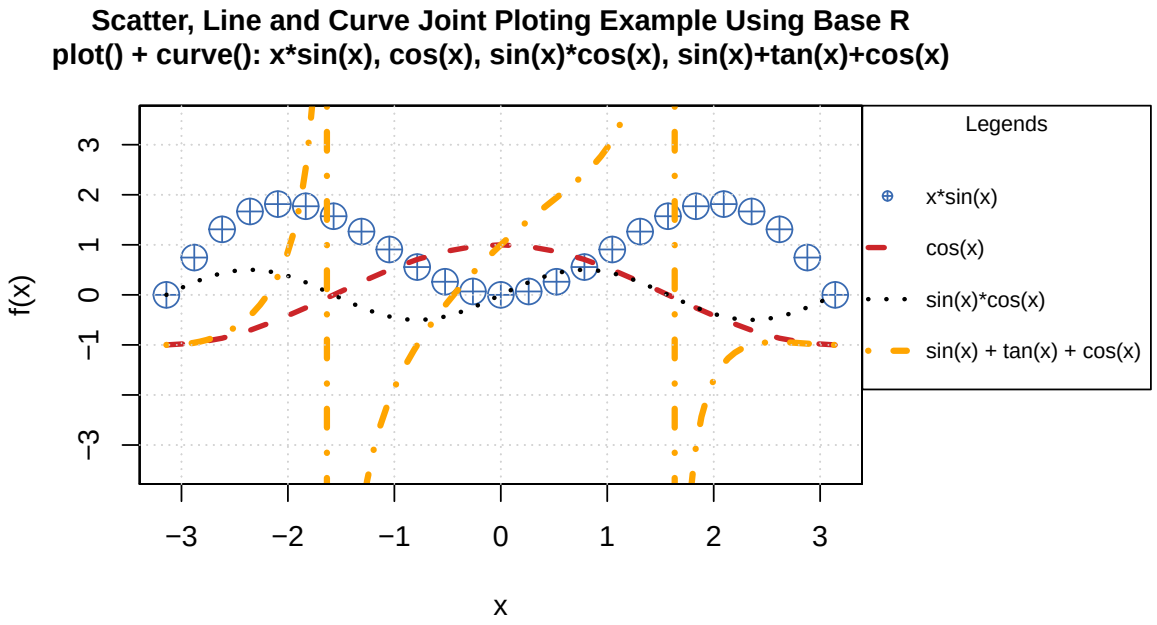
Jointly plot:

- 1 scatter plot
- 1 line plot
- 2 function curve plots

```
##########################################################
# First, Some common Labels:
##########################################################
# Labeling
st_title <- paste0('Scatter, Line and Curve Joint Ploting Example Using Base R\n',
                   'plot() + curve(): x*sin(x), cos(x), sin(x)*cos(x), sin(x)+tan(x)+cos(x)')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/inout/htmlpdfr/fs_base_curve.html')
st_x_label <- 'x'
st_y_label <- 'f(x)'

##########################################################
# Second, Generate the Graphs Functions and data points:
##########################################################
# x only used for Point 1 and Line 1
x <- seq(-1*pi, 1*pi, length.out=25)
# Line (Point) 1: Generate X and Y
y1 <- x*sin(x)
st_point_1_y_legend <- 'x*sin(x)'
# Line 2: Line Plot
y2 <- cos(x)
st_line_2_y_legend <- 'cos(x)'
# Line 3: Function
```

```
fc_sin_cos_diff <- function(x) sin(x)*cos(x)
st_line_3_y_legend <- 'sin(x)*cos(x)'
# Line 4: Function
fc_sin_cos_tan <- function(x) sin(x) + cos(x) + tan(x)
st_line_4_y_legend <- 'sin(x) + tan(x) + cos(x)'

#########################################################
# Third, set:
# - point shape and size: *pch* and *cex*
# - line type and width: *lty* and *lwd*
#########################################################
# http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r
# http://www.sthda.com/english/wiki/line-types-in-r-lty
# for colors, see: https://fanwangecon.github.io/M4Econ/graph/tools/fs_color.html
st_point_1_blue <- rgb(57/255,106/255,177/255)
st_line_2_red <- rgb(204/255, 37/255, 41/255,)
st_line_3_black <- 'black'
st_line_4_purple <- 'orange'

# point type
st_point_1_pch <- 10
# point size
st_point_1_cex <- 2

# line type
st_line_2_lty <- 'dashed'
st_line_3_lty <- 'dotted'
st_line_4_lty <- 'dotdash'
# line width
st_line_2_lwd <- 3
st_line_3_lwd <- 2.5
st_line_4_lwd <- 3.5

#########################################################
# Fourth: Share xlim and ylim
#########################################################
ar_xlim = c(min(x), max(x))
ar_ylim = c(-3.5, 3.5)

#########################################################
# Fifth: the legend will be long, will place it to the right of figure,
#########################################################
par(new=FALSE, mar=c(5, 4, 4, 10))

#########################################################
# Sixth, the four objects and do not print yet:
#########################################################
# pdf(NULL)
# Graph Scatter 1
plot(x, y1, type="p",
     col = st_point_1_blue,
     pch = st_point_1_pch, cex = st_point_1_cex,
     xlim = ar_xlim, ylim = ar_ylim,
     panel.first = grid(),
     ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_scatter_1 <- recordPlot()
```

```
# Graph Line 2
par(new=T)
plot(x, y2, type="l",
     col = st_line_2_red,
     lwd = st_line_2_lwd, lty = st_line_2_lty,
     xlim = ar_xlim, ylim = ar_ylim,
     ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_12 <- recordPlot()

# Graph Curve 3
par(new=T)
curve(fc_sin_cos_diff,
      col = st_line_3_black,
      lwd = st_line_3_lwd, lty = st_line_3_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_123 <- recordPlot()

# Graph Curve 4
par(new=T)
curve(fc_sin_cos_tan,
      col = st_line_4_purple,
      lwd = st_line_4_lwd, lty = st_line_4_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
pl_1234 <- recordPlot()
# invisible(dev.off())

##########################################################
# Seventh, Set Title and Legend and Plot Jointly
##########################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 1
fl_ces_fig_small = 0.75

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topright",
       inset=c(-0.4,0),
       xpd=TRUE,
       c(st_point_1_y_legend, st_line_2_y_legend, st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_point_1_blue, st_line_2_red, st_line_3_black, st_line_4_purple),
       pch = c(st_point_1_pch, NA, NA, NA),
       cex = fl_ces_fig_small,
       lty = c(NA, st_line_2_lty, st_line_3_lty, st_line_4_lty),
       lwd = c(NA, st_line_2_lwd, st_line_3_lwd,st_line_4_lwd),
       title = 'Legends',
       y.intersp=2)
```

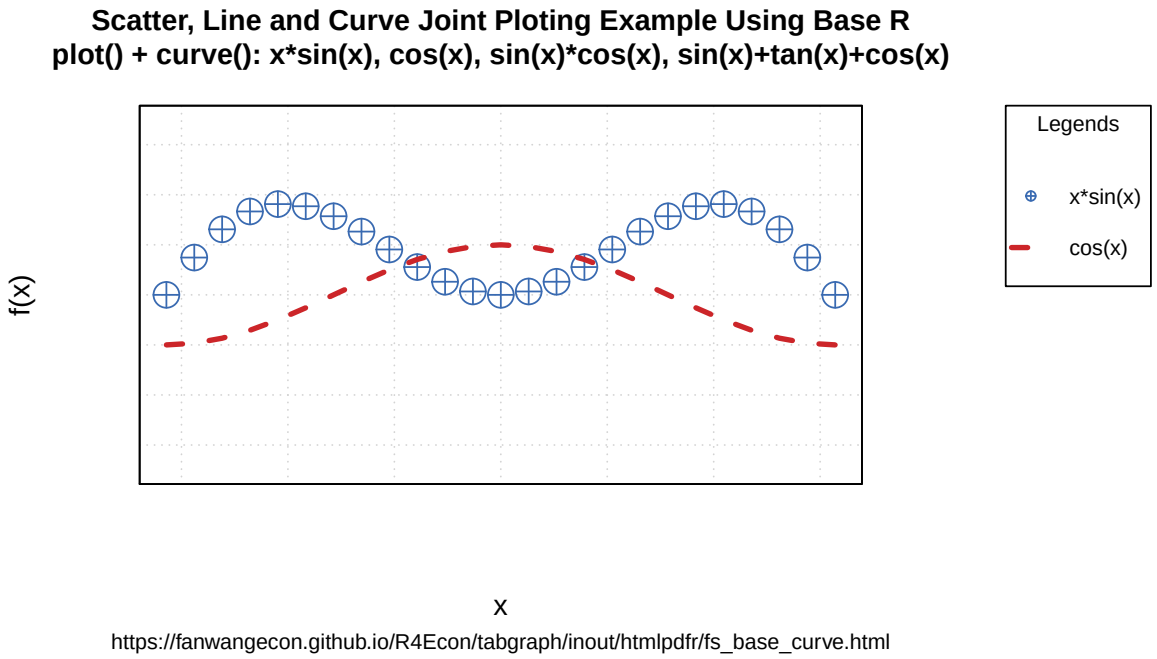


https://fanwangecon.github.io/R4Econ/tabgraph/inout/htmlpdfr/fs_base_curve.html

```
# record final plot
pl_1234_final <- recordPlot()
```

We used recordplot() earlier. So now we can print just the first two constructed plots.

```
############################################################
# Eighth, Plot just the first two saved lines
############################################################
# mar: margin, bottom, left, top, right
pl_12
# R Legend
par(new=T)
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab = fl_ces_fig_reg,
      cex.main = fl_ces_fig_reg,
      cex.sub = fl_ces_fig_small)

# Legend sizing CEX
par(new=T)
legend("topright",
       inset=c(-0.4,0),
       xpd=TRUE,
       c(st_point_1_y_legend, st_line_2_y_legend),
       col = c(st_point_1_blue, st_line_2_red),
       pch = c(st_point_1_pch, NA),
       cex = fl_ces_fig_small,
       lty = c(NA, st_line_2_lty),
       lwd = c(NA, st_line_2_lwd),
       title = 'Legends',
       y.intersp=2)
```



10.2 ggplot Line Related Plots

10.2.1 ggplot Line Plot

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

10.2.1.1 Three Categories, One is Subplot

The outcome is CEV, generated for results with different productivity types (subplot), generated for PE vs GE (linetype), and at different parameter specifications (lower and higher gamma).

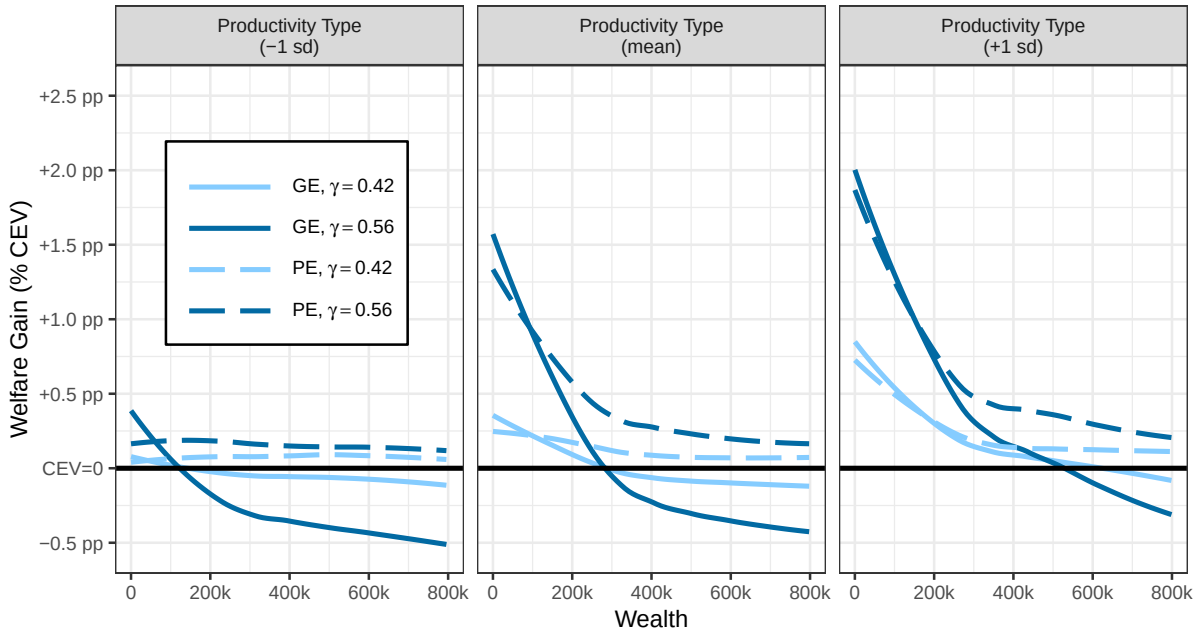
The graphs rely on this csv file cev_data.csv.

```
# Libraries
# library(tidyverse)

# Load in CSV
bl_save_img <- FALSE
spt_csv_root <- c('C:/Users/fan/R4Econ/tabgraph/ggline/_file/')
spt_img_root <- c('G:/repos/R4Econ/tabgraph/ggline/_file/')
spn_cev_data <- paste0(spt_csv_root, 'cev_data.csv')
spn_cev_graph <- paste0(spt_img_root, 'cev_graph.png')
spn_cev_graph_eps <- paste0(spt_img_root, 'cev_graph.eps')
df_cev_graph <- as_tibble(read.csv(spn_cev_data)) %>% select(-X)

# Dataset subsetting ------

# Line Patterns and Colors ------
# ar_st_age_group_leg_labels <- c("\nGE\n\u03B3=0.42\n", "\nGE\n\u03B3=0.56\n",
#                                 "\nPE\n\u03B3=0.42\n", "\nPE\n\u03B3=0.42\n")
ar_st_age_group_leg_labels <- c(bquote("GE,"~gamma == .(0.42)),
                                bquote("GE,"~gamma == .(0.56)),
                                bquote("PE,"~gamma == .(0.42)),
                                bquote("PE,"~gamma == .(0.56)))
ar_st_colours <- c("#85ccff", "#026aa3", "#85ccff", "#026aa3")
```

```
ar_st_linetypes <- c("solid", "solid", "longdash", "longdash")

# Labels and Other Strings -------
st_title <- ''
st_x <- 'Wealth'
st_y <- 'Welfare Gain (% CEV)'
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/ggline/htmlpdfr/fs_ggline_mgrp_ncts.html')

# ar_st_age_group_leg_labels <- c("C\u2013Optimal", "V\u2013Optimal")

prod_type_recode <- c("Productivity Type\n(-1 sd)" = "8993",
                      "Productivity Type\n(mean)" = "10189",
                      "Productivity Type\n(+1 sd)" = "12244")

x.labels <- c('0', '200k', '400k', '600k', '800k')
x.breaks <- c(0,
              5,
              10,
              15,
              20)
x.min <- 0
x.max <- 20

# y.labels <- c('-0.01',
#               '\u2191\u2191\nWelfare\nGain\n\nCEV=0\n\nWelfare\nLoss\n\u2193\u2193',
#               '+0.01', '+0.02', '+0.03', '+0.04','+0.05')
y.labels <- c('-0.5 pp',
              'CEV=0',
              '+0.5 pp', '+1.0 pp', '+1.5 pp', '+2.0 pp','+2.5 pp')
y.breaks <- c(-0.01, 0, 0.01, 0.02, 0.03, 0.04, 0.05)
y.min <- -0.011
y.max <- 0.051

# data change -------
df_cev_graph <- df_cev_graph %>%
  filter(across(counter_policy, ~ grepl('70|42', .))) %>%
  mutate(prod_type_lvl = as.factor(prod_type_lvl)) %>%
  mutate(prod_type_lvl = fct_recode(prod_type_lvl, !!!prod_type_recode))

# graph ------
pl_cev <- df_cev_graph %>%
  group_by(prod_type_st, cash_tt) %>%
  ggplot(aes(x=cash_tt, y=cev_lvl,
             colour=counter_policy, linetype=counter_policy, shape=counter_policy)) +
  facet_wrap( ~ prod_type_lvl, nrow=1) +
  geom_smooth(method="auto", se=FALSE, fullrange=FALSE, level=0.95)

# labels
pl_cev <- pl_cev +
  labs(x = st_x,
       y = st_y,
       subtitle = st_subtitle)

# set shapes and colors
pl_cev <- pl_cev +
  scale_colour_manual(values=ar_st_colours, labels=ar_st_age_group_leg_labels) +
```

```
  scale_shape_discrete(labels=ar_st_age_group_leg_labels) +
  scale_linetype_manual(values=ar_st_linetypes, labels=ar_st_age_group_leg_labels) +
  scale_x_continuous(labels = x.labels, breaks = x.breaks,
                     limits = c(x.min, x.max)) +
  scale_y_continuous(labels = y.labels, breaks = y.breaks,
                     limits = c(y.min, y.max))

# Horizontal line
pl_cev <- pl_cev +
  geom_hline(yintercept=0, linetype='solid', colour="black", size =1)
  # geom_hline(yintercept=0, linetype='dotted', colour="black", size=2)

# theme
pl_cev <- pl_cev +
  theme_bw() +
  theme(text = element_text(size = 10),
       legend.title = element_blank(),
       legend.position = c(0.16, 0.65),
       legend.background = element_rect(fill = "white", colour = "black", linetype='solid'),
       legend.key.width = unit(1.5, "cm"))


# Save Image Outputs -----
if (bl_save_img) {
  png(spn_cev_graph,
      width = 160,
      height = 105, units='mm',
      res = 150, pointsize=7)
  ggsave(
    spn_cev_graph_eps,
    plot = last_plot(),
    device = 'eps',
    path = NULL,
    scale = 1,
    width = 160,
    height =105,
    units = c("mm"),
    dpi = 150,
    limitsize = TRUE
  )

}
print(pl_cev)
```

https://fanwangecon.github.io/R4Econ/tabgraph/ggline/htmlpdfr/fs_ggline_mgrp_ncts.html




```
if (bl_save_img) {
  dev.off()
}
```

10.3 ggplot Scatter Related Plots

10.3.1 ggplot Scatter Plot

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

10.3.1.1 Three Continuous Variables and Two Categorical Variables

We will generate a graph that is very similar to the graph shown for fs_tib_factors, with the addition that scatter color and shape will be for two separate variables, and with the addition that scatter size will be for an additional continuous variable.

We have three continuous variables:

1. y-axis: time for 1/4 Miles (QSEC)
2. x-axis: horsepower
3. scatter-size: miles per gallon (mpg)

We have two categorical ariables:

1. color: vs engine shape (vshaped or straight)
2. shape: am shift type (auto or manual)

First, Load in the mtcars dataset and convert to categorical variables to factor with labels.

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
  mutate(vs = as_factor(vs), am = as_factor(am))
# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c(vshaped_engine = "0", straight_engine = "1")
tb_mtcars <- tb_mtcars %>%
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))
```

Second, generate the core graph, a scatterplot with a nonlinear trendline. Note that in the example below color and shpae only apply to the jitter scatter, but not the trendline graph.

```
# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars, aes(x=hp, y=qsec)) +
  geom_jitter(aes(size=mpg, colour=vs, shape=am), width = 0.15) +
  geom_smooth(span = 0.50, se=FALSE) +
  theme_bw()
```

Third, control Color and Shape Information. There will be two colors and two shapes. See all shape listing.

```
# Color controls
ar_st_colors <- c("#33cc33", "#F8766D")
ar_st_colors_label <- c("v-shaped", "straight")
fl_legend_color_symbol_size <- 5
st_leg_color_lab <- "Engine-Shape"
# Shape controls
ar_it_shapes <- c(9, 15)
ar_st_shapes_label <- c("auto", "manuel")
fl_legend_shape_symbol_size <- 5
st_leg_shape_lab <- "Transmission"
```

Fourth, control the size of the scatter, which will be the MPG variable.

```
# Control scatter point size
fl_min_size <- 3
fl_max_size <- 6
ar_size_range <- c(fl_min_size, fl_max_size)
st_leg_size_lab <- "MPG"
```

Fifth, control graph strings.

```
# Labeling
st_title <- paste0('Distribution of HP and QSEC from mtcars')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/ggscatter/htmlpdfr/fs_ggscatter_3cts_mdisc.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'HP = Horse Power'
st_y_label <- 'QSEC = time for 1/4 Miles'
```

Sixth, combine graphical components.

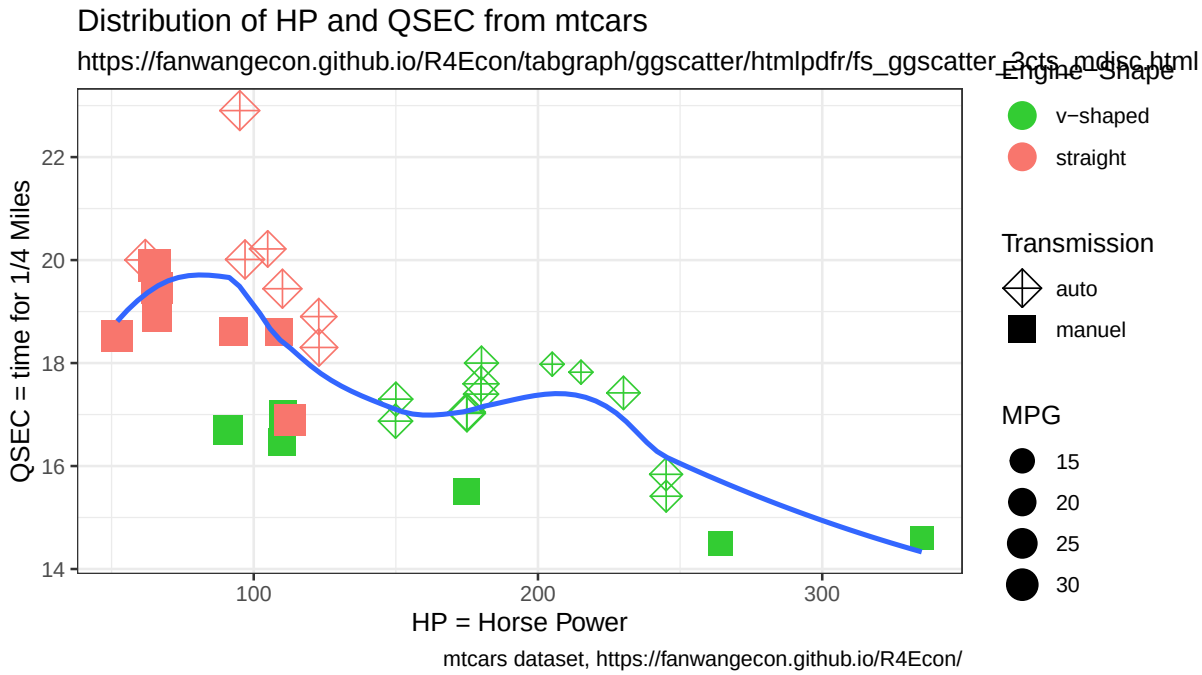
```
# Add titles and labels
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption)

# Color, shape and size controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
  scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
  scale_size_continuous(range = ar_size_range)
```

Eigth, replace default legends titles for color, shape and size.

```
# replace the default labels for each legend segment
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(colour = st_leg_color_lab,
       shape = st_leg_shape_lab,
       size = st_leg_size_lab)
```

Ninth, additional controls for the graph.

```
# Control the order of legend display
# Show color, show shape, then show size.
plt_mtcars_scatter <- plt_mtcars_scatter + guides(
  colour = guide_legend(order = 1, override.aes = list(size = fl_legend_color_symbol_size)),
  shape = guide_legend(order = 2, override.aes = list(size = fl_legend_shape_symbol_size)),
  size = guide_legend(order = 3))

# show
print(plt_mtcars_scatter)
```



10.3.2 ggplot Multiple Scatter-Lines with Facet Wrap

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

10.3.2.1 Multiple Scatter-Lines with Facet Wrap

Two subplots, for auto and manual transitions. The x-axis is horse-power, the y-axis shows QSEC. Different colors represent v-shaped and straight-engines.

1. y-axis: time for 1/4 Miles (QSEC)
2. x-axis: horsepower (hp)
3. facet-wrap: auto or manual (am)
4. colored line and point shapes: vshaped or straight engine (vs)

First, Load in the mtcars dataset and convert to categorical variables to factor with labels.

```
# First make sure these are factors
tb_mtcars <- as_tibble(mtcars) %>%
  mutate(vs = as_factor(vs), am = as_factor(am))
# Second Label the Factors
am_levels <- c(auto_shift = "0", manual_shift = "1")
vs_levels <- c("vshaped engine" = "0", "straight engine" = "1")
tb_mtcars <- tb_mtcars %>%
```

```
  mutate(vs = fct_recode(vs, !!!vs_levels),
         am = fct_recode(am, !!!am_levels))
```

Second, generate the core graph, a line plot and facet wrapping over the *am* variable. Note that *vs* variable has different color as well as line type and shape

```
# Graphing
plt_mtcars_scatter <-
  ggplot(tb_mtcars, aes(x=hp, y=qsec,
                        colour=am, shape=am, linetype=am)) +
  geom_smooth(se = FALSE, lwd = 1.5) + # Lwd = line width
  geom_point(size = 5, stroke = 2) + # stroke = point shape width
  facet_wrap(~ vs,
             scales = "free_x",
             nrow = 1, ncol = 2,
             labeller = label_wrap_gen(multi_line=FALSE))
```

Third, control Color, Shape and Line-type Information. There will be two colors, two shapes and two linetypes. See all shape listing and linetype listing., See all shape listing.

```
# Color controls
ar_st_colors <- c("#33cc33", "#F8766D")
ar_st_colors_label <- c("auto", "manual")
fl_legend_color_symbol_size <- 5
st_leg_color_lab <- "Transmission"
# Shape controls
ar_it_shapes <- c(1, 5)
ar_st_shapes_label <- c("auto", "manual")
fl_legend_shape_symbol_size <- 5
st_leg_shape_lab <- "Transmission"
# Line-Type controls
ar_st_linetypes <- c('solid', 'dashed')
ar_st_linetypes_label <- c("auto", "manual")
fl_legend_linetype_symbol_size <- 5
st_leg_linetype_lab <- "Transmission"
```

Fourth, manaully specify an x-axis.

```
# x labeling and axis control
ar_st_x_labels <- c('50 hp', '150 hp', '250 hp', '350 hp')
ar_fl_x_breaks <- c(50, 150, 250, 350)
ar_fl_x_limits <- c(40, 360)
# y labeling and axis control
ar_st_y_labels <- c('15 QSEC', '18', '21', '24 QSEC')
ar_fl_y_breaks <- c(15, 18, 21, 24)
ar_fl_y_limits <- c(13.5, 25.5)
```

Fifth, control graph strings.

```
# Labeling
st_title <- paste0('How QSEC varies by Horse-power, by Engine and Transmission Types')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'HP = Horse Power'
st_y_label <- 'QSEC = time for 1/4 Miles'
```

Sixth, combine graphical components.

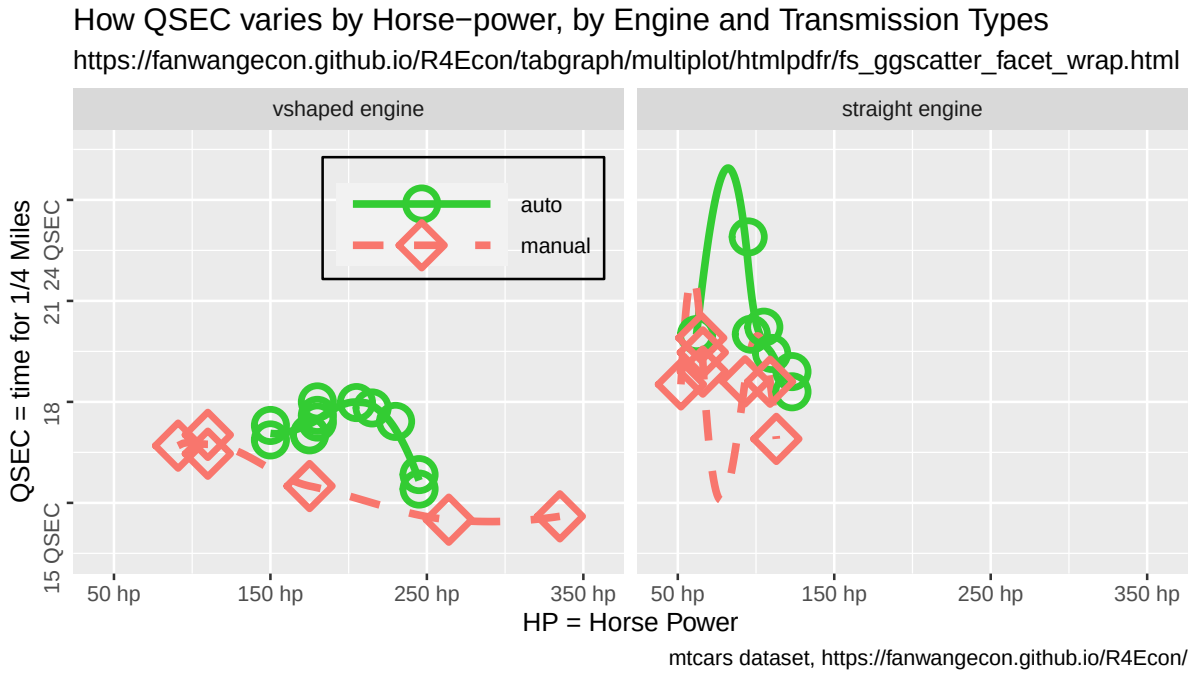
```
# Add titles and labels
plt_mtcars_scatter <- plt_mtcars_scatter +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption)

# x and y-axis ticks controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_x_continuous(labels = ar_st_x_labels,
                     breaks = ar_fl_x_breaks,
                     limits = ar_fl_x_limits) +
  scale_y_continuous(labels = ar_st_y_labels,
                     breaks = ar_fl_y_breaks,
                     limits = ar_fl_y_limits)

# Color, shape and linetype controls
plt_mtcars_scatter <- plt_mtcars_scatter +
  scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
  scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
  scale_linetype_manual(values=ar_st_linetypes, labels=ar_st_linetypes_label)
```

Seventh, replace default legends, and set figure font overall etc.

```
# has legend theme
theme_custom <- theme(
  text = element_text(size = 11),
  axis.text.y = element_text(angle = 90),
  legend.title = element_blank(),
  legend.position = c(0.35, 0.80),
  legend.key.width = unit(5, "line"),
  legend.background =
    element_rect(fill = "transparent", colour = "black", linetype='solid'))
# no legend theme (no y)
theme_custom_blank <- theme(
  text = element_text(size = 12),
  legend.title = element_blank(),
  legend.position = "none",
  axis.title.y=element_blank(),
  axis.text.y=element_blank(),
  axis.ticks.y=element_blank())
```

Eighth, graph out.

```
# replace the default labels for each legend segment
plt_mtcars_scatter <- plt_mtcars_scatter + theme_custom
# show
print(plt_mtcars_scatter)
```



10.3.2.2 Divide Facet Wrapped Plot into Subplots

Given the facet-wrapped plot just generated, now save alternative plot versions, where each subplot is saved by itself. Will simply use the code from above, but call inside lapply over different am categories.

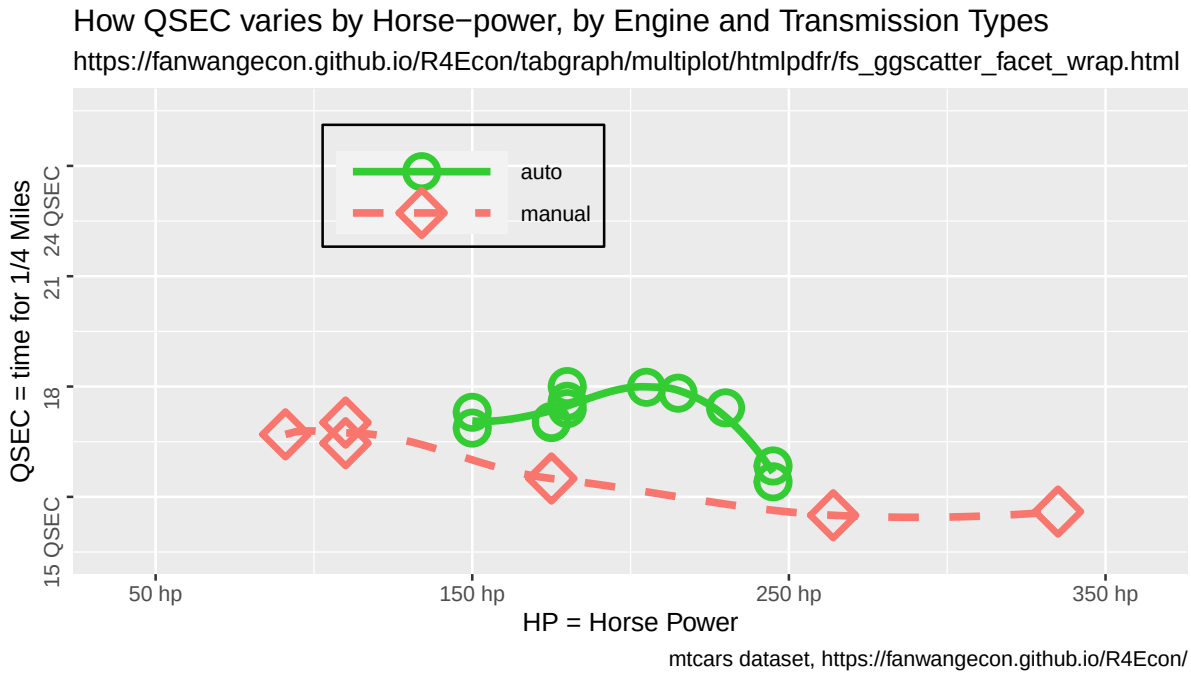
Below generate a matrix with multiple data columns, then use apply over each row of the matrix and select different columns of values for each subplot generation.

First generate with legend versions, then without legend versions. These are versions that would be used to more freely compose graph together.

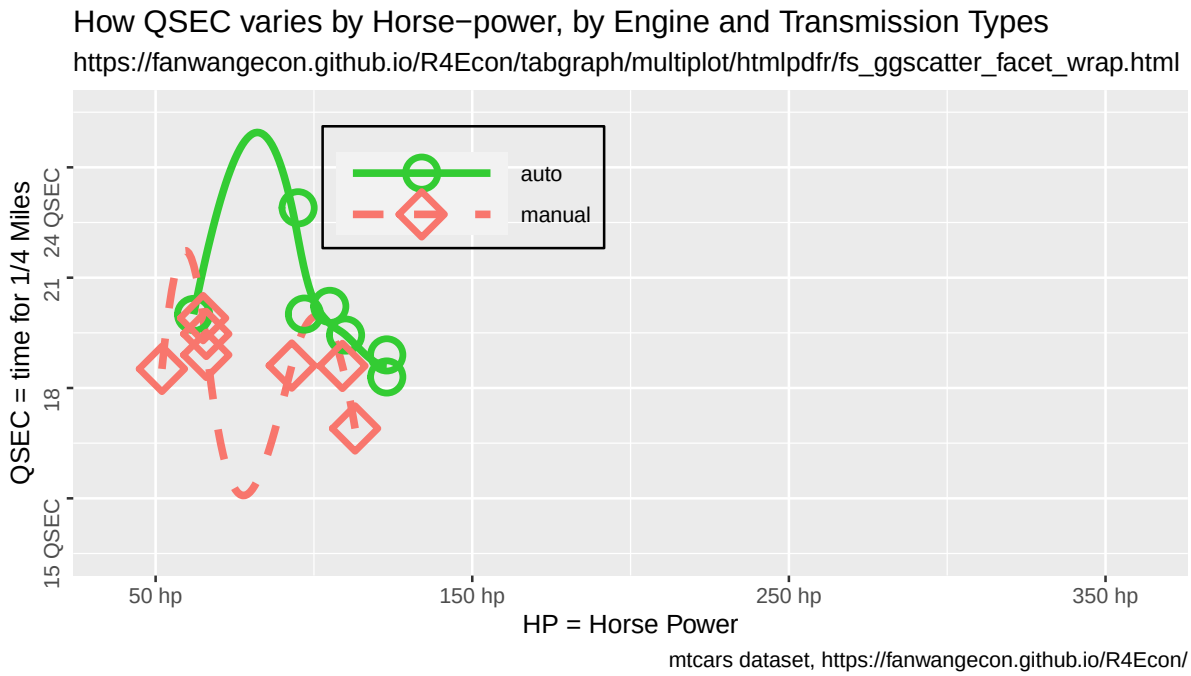
```
for (it_subplot_as_own_vsr in c(1,2)) {

  if (it_subplot_as_own_vsr == 1) {
    theme_custom_use <- theme_custom
    # st_file_suffix <- '_haslegend'
    # it_width <- 100
  } else if (it_subplot_as_own_vsr == 2) {
    theme_custom_use <- theme_custom_blank
    # st_file_suffix <- '_nolegend'
    # it_width <- 88
  }

  # unique vs as matrix
  # ar_uniques <-sort(unique(tb_mtcars$vs))
  # mt_unique_vs <- matrix(data=ar_uniques, nrow=length(ar_uniques), ncol=1)
  mt_unique_vs <- tb_mtcars %>% group_by(vs) %>%
    summarize(mpg=mean(mpg)) %>% ungroup()
  # apply over
  ls_plots <- apply(mt_unique_vs, 1, function(ar_vs_cate_row) {
    # 1. Graph main
    plt_mtcars_scatter <-
      ggplot(tb_mtcars %>% filter(vs == ar_vs_cate_row[1]),
             aes(x=hp, y=qsec,
                 colour=am, shape=am, linetype=am)) +
      geom_smooth(se = FALSE, lwd = 1.5) + # Lwd = line width
      geom_point(size = 5, stroke = 2)
```

```
    # 2. Add titles and labels
    plt_mtcars_scatter <- plt_mtcars_scatter +
      labs(title = st_title, subtitle = st_subtitle,
           x = st_x_label, y = st_y_label, caption = st_caption)

    # 3. x and y ticks
    plt_mtcars_scatter <- plt_mtcars_scatter +
      scale_x_continuous(labels = ar_st_x_labels, breaks = ar_fl_x_breaks, limits = ar_fl_x_limits) +
      scale_y_continuous(labels = ar_st_y_labels, breaks = ar_fl_y_breaks, limits = ar_fl_y_limits)

    # 4. Color, shape and linetype controls
    plt_mtcars_scatter <- plt_mtcars_scatter +
      scale_colour_manual(values=ar_st_colors, labels=ar_st_colors_label) +
      scale_shape_manual(values=ar_it_shapes, labels=ar_st_shapes_label) +
      scale_linetype_manual(values=ar_st_linetypes, labels=ar_st_linetypes_label)

    # 5. replace the default labels for each legend segment
    plt_mtcars_scatter <- plt_mtcars_scatter + theme_custom_use
  })
  # show
  print(ls_plots)
}
```

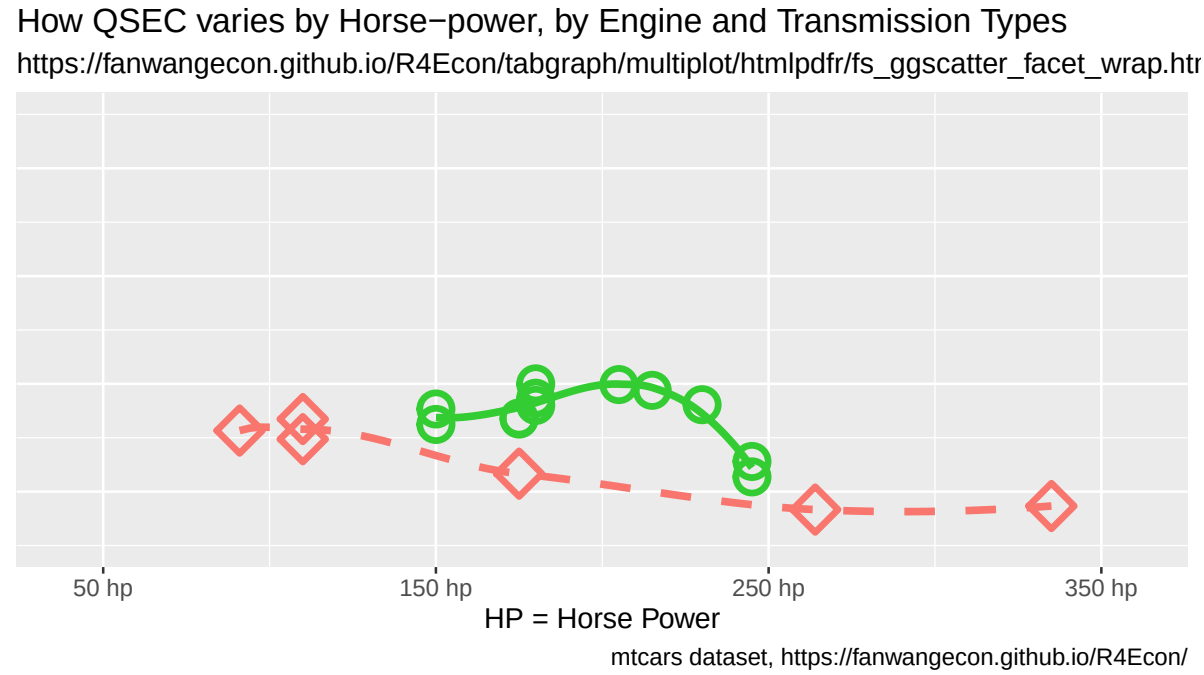
```
## [[1]]
```



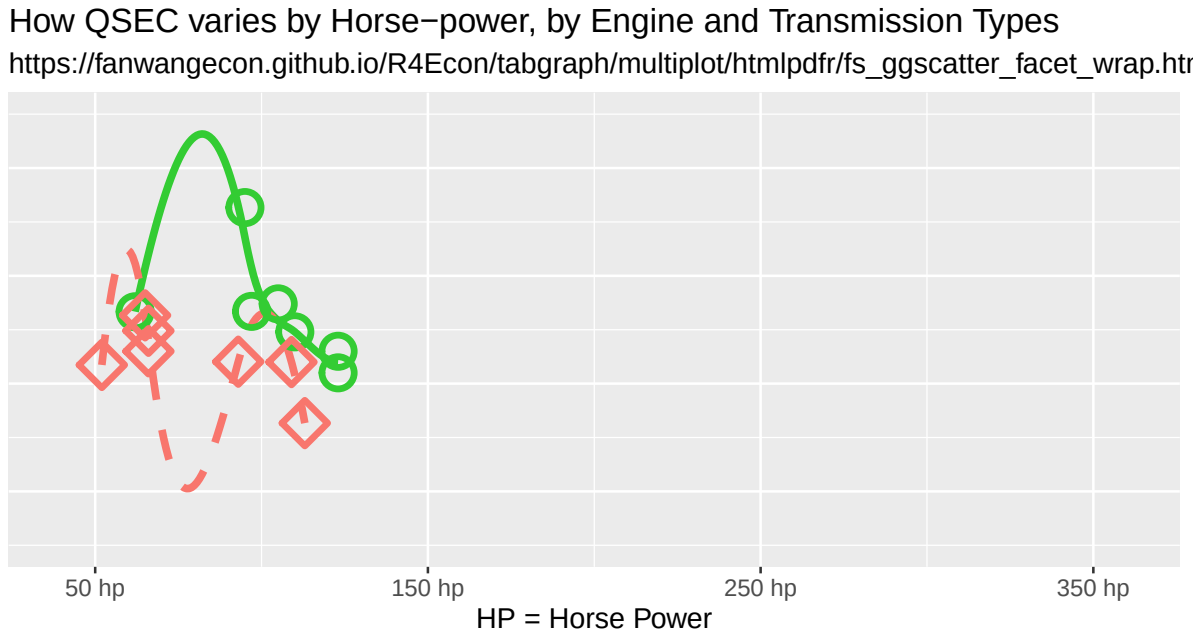

```
##
## [[2]]
```


How QSEC varies by Horse−power, by Engine and Transmission Types
https://fanwangecon.github.io/R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.html
auto
manual
QSEC = time for 1/4 Miles
15 QSEC
18
21
24 QSEC
50 hp
150 hp
250 hp
350 hp
HP = Horse Power
mtcars dataset, https://fanwangecon.github.io/R4Econ/

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


How QSEC varies by Horse−power, by Engine and Transmission Types
https://fanwangecon.github.io/R4Econ/tabgraph/multiplot/htmlpdfr/fs_ggscatter_facet_wrap.htr
50 hp
150 hp
250 hp
350 hp
HP = Horse Power
mtcars dataset, https://fanwangecon.github.io/R4Econ/

```
##
## [[2]]
```



10.4 Write and Read Plots

10.4.1 Import and Export Images

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

Work with the R plot function.

10.4.1.1 Export Images Different Formats with Plot()

10.4.1.1.1 Generate and Record A Plot Generate a graph and recordPlot() it. The generated graph does not have legends Yet. Crucially, there are no titles, legends, axis, labels in the figures. As we stack the figures together, do not add those. Only add at the end jointly for all figure elements together to control at one spot things.

```
#########################################################
# First, Strings
#########################################################
# Labeling
st_title <- paste0('Scatter, Line and Curve Joint Ploting Example Using Base R\n',
                   'plot() + curve():sin(x)*cos(x), sin(x)+tan(x)+cos(x)')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/tabgraph/inout/htmlpdfr/fs_base_curve.html')
st_x_label <- 'x'
st_y_label <- 'f(x)'

#########################################################
# Second, functions
#########################################################
fc_sin_cos_diff <- function(x) sin(x)*cos(x)
st_line_3_y_legend <- 'sin(x)*cos(x)'
fc_sin_cos_tan <- function(x) sin(x) + cos(x) + tan(x)
st_line_4_y_legend <- 'sin(x) + tan(x) + cos(x)'

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

```
# Third, patterns
#########################################################
st_line_3_black <- 'black'
st_line_4_purple <- 'orange'
# line type
st_line_3_lty <- 'dotted'
st_line_4_lty <- 'dotdash'
# line width
st_line_3_lwd <- 2.5
st_line_4_lwd <- 3.5

#########################################################
# Fourth: Share xlim and ylim
#########################################################
ar_xlim = c(-3, 3)
ar_ylim = c(-3.5, 3.5)

#########################################################
# Fifth: Even margins
#########################################################
par(new=FALSE)

#########################################################
# Sixth, the four objects and do not print yet:
#########################################################
# Graph Curve 3
par(new=T)
curve(fc_sin_cos_diff,
      col = st_line_3_black,
      lwd = st_line_3_lwd, lty = st_line_3_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
# Graph Curve 4
par(new=T)
curve(fc_sin_cos_tan,
      col = st_line_4_purple,
      lwd = st_line_4_lwd, lty = st_line_4_lty,
      from = ar_xlim[1], to = ar_xlim[2], ylim = ar_ylim,
      ylab = '', xlab = '', yaxt='n', xaxt='n', ann=FALSE)
```

```
pl_curves_save <- recordPlot()
```

10.4.1.1.2 Generate Large Font and Small Font Versions of PLot

Generate larger font version:

```
# Replay
pl_curves_save

#########################################################
# Seventh, Set Title and Legend and Plot Jointly
#########################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 0.75
fl_ces_fig_leg = 0.75
fl_ces_fig_small = 0.65

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
```

```
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topleft",
       bg="transparent",
       bty = "n",
       c(st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_line_3_black, st_line_4_purple),
       pch = c(NA, NA),
       cex = fl_ces_fig_leg,
       lty = c(st_line_3_lty, st_line_4_lty),
       lwd = c(st_line_3_lwd,st_line_4_lwd),
       y.intersp=2)
```

```
# record final plot
pl_curves_large <- recordPlot()
dev.off()
```

Generate smaller font version:

```
# Replay
pl_curves_save

############################################################
# Seventh, Set Title and Legend and Plot Jointly
############################################################
# CEX sizing Contorl Titling and Legend Sizes
fl_ces_fig_reg = 0.45
fl_ces_fig_leg = 0.45
fl_ces_fig_small = 0.25

# R Legend
title(main = st_title, sub = st_subtitle, xlab = st_x_label, ylab = st_y_label,
      cex.lab=fl_ces_fig_reg,
      cex.main=fl_ces_fig_reg,
      cex.sub=fl_ces_fig_small)
axis(1, cex.axis=fl_ces_fig_reg)
axis(2, cex.axis=fl_ces_fig_reg)
grid()

# Legend sizing CEX
legend("topleft",
       bg="transparent",
       bty = "n",
       c(st_line_3_y_legend, st_line_4_y_legend),
       col = c(st_line_3_black, st_line_4_purple),
       pch = c(NA, NA),
       cex = fl_ces_fig_leg,
       lty = c(st_line_3_lty, st_line_4_lty),
       lwd = c(st_line_3_lwd,st_line_4_lwd),
       y.intersp=2)
```

```
# record final plot
pl_curves_small <- recordPlot()
dev.off()
```

10.4.1.1.3 Save Plot with Varying Resolutions and Heights

Export recorded plot.

A4 paper is 8.3 x 11.7, with 1 inch margins, the remaining area is 6.3 x 9.7. For figures that should take half of the page, the height should be 4.8 inch. One third of a page should be 3.2 inch. 6.3 inch is 160mm and 3 inch is 76 mm. In the example below, use

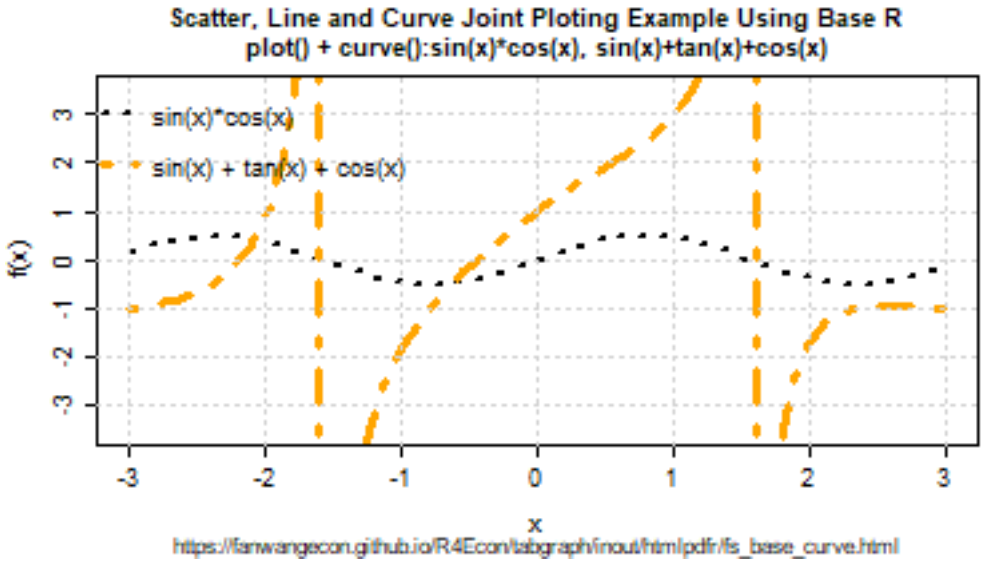
```
# Store both in within folder directory and root image directory:
# C:\Users\fan\R4Econ\tabgraph\inout\_img
# C:\Users\fan\R4Econ\_img
# need to store in both because bookdown and indi pdf path differ.
# Wrap in try because will not work underbookdown, but images already created

ls_spt_root <- c('..//..//', '')
spt_prefix <- '_img/fs_img_io_2curve'

for (spt_root in ls_spt_root) {
  # Changing pointsize will not change font sizes inside, just rescale
  # PNG 72
  try(png(paste0(spt_root, spt_prefix, "_w135h76_res72.png"),
      width = 135 , height = 76, units='mm', res = 72, pointsize=7))
  print(pl_curves_large)
  dev.off()
  # PNG 300
  try(png(paste0(spt_root, spt_prefix, "_w135h76_res300.png"),
      width = 135, height = 76, units='mm', res = 300, pointsize=7))
  print(pl_curves_large)
  dev.off()
  # PNG 300, SMALL, POINT SIZE LOWER
  try(png(paste0(spt_root, spt_prefix, "_w80h48_res300.png"),
      width = 80, height = 48, units='mm', res = 300, pointsize=7))
  print(pl_curves_small)
  dev.off()
  # PNG 300
  try(png(paste0(spt_root, spt_prefix, "_w160h100_res300.png"),
      width = 160, height = 100, units='mm', res = 300))
  print(pl_curves_large)
  dev.off()

  # EPS
  setEPS()
  try(postscript(paste0(spt_root, spt_prefix, "_fs_2curve.eps")))
  print(pl_curves_large)
  dev.off()
}
```

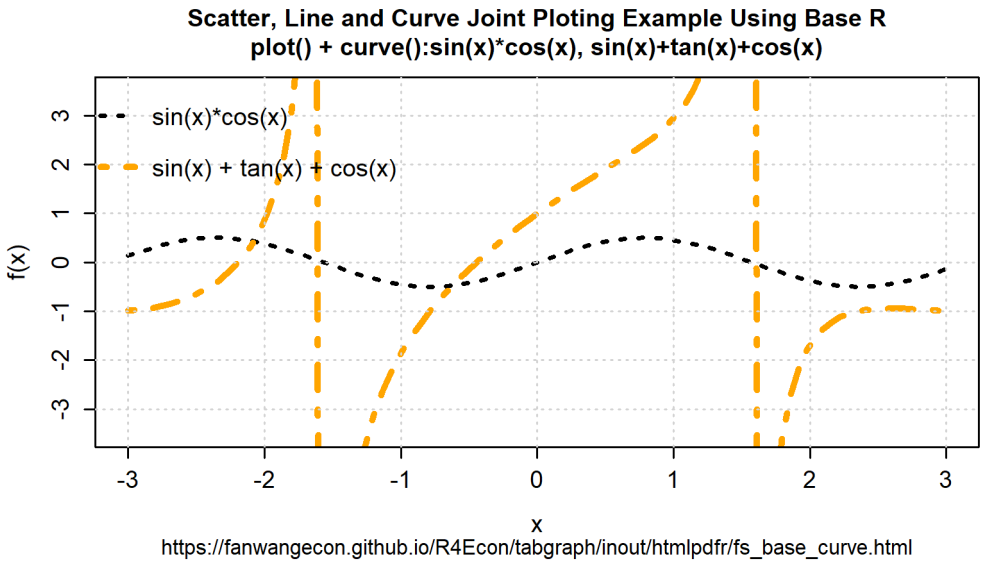
```
## Error in png(paste0(spt_root, spt_prefix, "_w135h76_res72.png"), width = 135,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w135h76_res300.png"), width = 135,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w80h48_res300.png"), width = 80,  :
##   unable to start png() device
## Error in png(paste0(spt_root, spt_prefix, "_w160h100_res300.png"), width = 160,  :
##   unable to start png() device
## Error in postscript(paste0(spt_root, spt_prefix, "_fs_2curve.eps")) :
##   cannot open file '..//..//_img/fs_img_io_2curve_fs_2curve.eps'
```

10.4.1.1.4 Low and High Resolution Figure

The standard resolution often produces very low quality images. Resolution should be increased. See figure comparison.

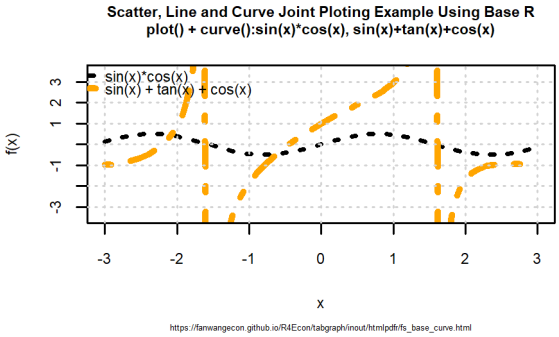
RES=72 (DEFAULT R) TOP, RES=300 Bottom, (Width=160, Height=81, PNG)

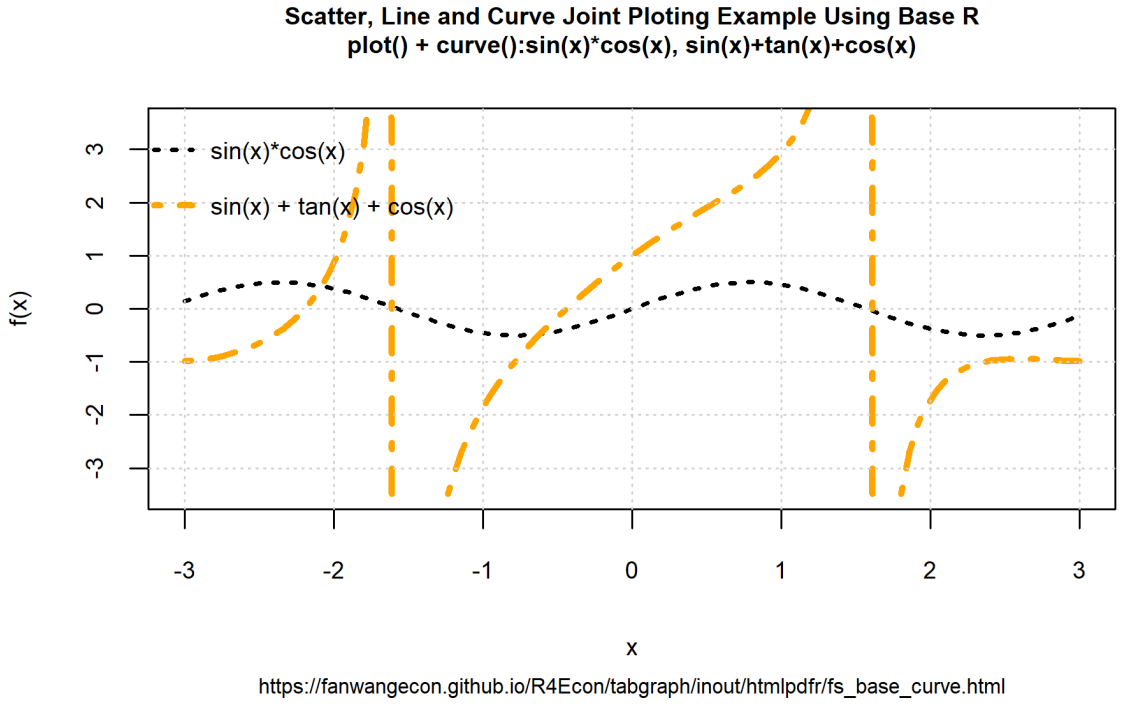




10.4.1.1.5 Smaller and Larger Figures Smaller and larger figures with different font size comparison. Note that earlier, we generated the figure without legends, labels, etc first, recorded the figure. Then we associated the same underlying figure with differently sized titles, legends, axis, labels.

Top Small (small font saved), Bottom Large, PNG





Chapter 11

Get Data

11.1 Environmental Data

11.1.1 ECMWF ERA5 Data

> Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

This files uses R with the reticulate package to download ECMWF ERA5 data. See this file for instructions and tutorials for downloading the data.

11.1.1.1 Program to Download, Unzip, Convert to combined CSV, derived-utci-historical data

The data downloaded from CDS climate could become very large in size. We want to process parts of the data one part at a time, summarize and aggregate over each part, and generate a file output file with aggregate statistics over the entire time period of interest.

This code below accompalishes the following tasks:

1. download data from derived-utci-historical as ZIP
2. unzip
3. convert *nc* files to *csv* files
4. individual csv files are half year groups

Parameter Control for the code below:

1. *spt_root*: root folder where everything will be at
2. *spth_conda_env*: the conda virtual environment python path, eccodes and cdsapi packages are installed in the conda virtual environment. In the example below, the first env is: wk_ecmwf
3. *st_nc_prefix*: the downloaded individual nc files have dates and prefix before and after the date string in the nc file names. This is the string before that.
4. *st_nc_suffix*: see (3), this is the suffix
5. *ar_years*: array of years to download and aggregate over
6. *ar_months_g1*: months to download in first half year
7. *ar_months_g2*: months to download in second half year

Note: *area* below corresponds to *North, West, South, East*.

```
################################################
# ------------ Parameters
################################################

# Where to store everything
spt_root <- "C:/Users/fan/Downloads/_data/"
spth_conda_env <- "C:/ProgramData/Anaconda3/envs/wk_ecmwf/python.exe"
```

```
# nc name prefix
st_nc_prefix <- "ECMWF_utci_"
st_nc_suffix <- "_v1.0_con.nc"

# Years list
# ar_years <- 2001:2019
ar_years <- c(2005, 2015)
# ar_months_g1 <- c('01','02','03','04','05','06')
ar_months_g1 <- c('01', '03')
# ar_months_g2 <- c('07','08','09','10','11','12')
ar_months_g2 <- c('07', '09')

# Area
# # China
# fl_area_north <- 53.31
# fl_area_west <- 73
# fl_area_south <- 4.15
# fl_area_east <- 135
fl_area_north <- 53
fl_area_west <- 73
fl_area_south <- 52
fl_area_east <- 74

# folder to download any nc zips to
nczippath <- spt_root
# we are changing the python api file with different requests stirngs and storing it here
pyapipath <- spt_root
# output directory for AGGREGATE CSV with all DATES from this search
csvpath <- spt_root

################################################
# ------------ Packages
################################################

library("ncdf4")
library("chron")
library("lattice")
library("RColorBrewer")
library("stringr")
library("tibble")
library("dplyr")
Sys.setenv(RETICULATE_PYTHON = spth_conda_env)
library("reticulate")

################################################
# ------------ Define Loops
################################################
for (it_yr in ar_years) {
  for (it_mth_group in c(1,2)) {
    if(it_mth_group == 1) {
      ar_months = ar_months_g1
    }
    if(it_mth_group == 2) {
      ar_months = ar_months_g2
    }

    ################################################
    # ------------ Define Python API Call
```

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

    # name of zip file
    nczipname <- "derived_utci_2010_2.zip"
    unzipfolder <- "derived_utci_2010_2"

    st_file <- paste0("import cdsapi
import urllib.request
# download folder
spt_root = '", nczippath, "'
spn_dl_test_grib = spt_root + '", nczipname, "'
# request
c = cdsapi.Client()
res = c.retrieve(
    'derived-utci-historical',
    {
        'format': 'zip',
        'variable': 'Universal thermal climate index',
        'product_type': 'Consolidated dataset',
        'year': '",it_yr, "',
        'month': [
            ", paste("'", ar_months, "'", sep = "", collapse = ", "), "
        ],
        'day': [
            '01','03'
        ],
        'area'  : [", fl_area_north ,", ", fl_area_west ,", ", fl_area_south ,", ", fl_area_east ,"]
        'grid'  : [0.25, 0.25],
    },
    spn_dl_test_grib)
# show results
print('print results')
print(res)
print(type(res))")

    # st_file = "print(1+1)"

    # Store Python Api File
    fl_test_tex <- paste0(pyapipath, "api.py")
    fileConn <- file(fl_test_tex)
    writeLines(st_file, fileConn)
    close(fileConn)

    ################################################
    # ------------ Run Python File
    ################################################
    # Set Path
    setwd(pyapipath)
    # Run py file, api.py name just defined
    use_python(spth_conda_env)
    source_python('api.py')

    ################################################
    # ------------ uNZIP
    ################################################
    spn_zip <- paste0(nczippath, nczipname)
    spn_unzip_folder <- paste0(nczippath, unzipfolder)
    unzip(spn_zip, exdir=spn_unzip_folder)
```

```
################################################
# ------------ Find All files
################################################
# Get all files with nc suffix in folder
ncpath <- paste0(nczippath, unzipfolder)
ls_sfls <- list.files(path=ncpath, recursive=TRUE, pattern=".nc", full.names=T)

################################################
# ------------ Combine individual NC files to JOINT Dataframe
################################################
# List to gather dataframes
ls_df <- vector(mode = "list", length = length(ls_sfls))
# Loop over files and convert nc to csv
it_df_ctr <- 0
for (spt_file in ls_sfls) {
  it_df_ctr <- it_df_ctr + 1

  # Get file name without Path
  snm_file_date <- sub(paste0('\\',st_nc_suffix,'$'), '', basename(spt_file))
  snm_file_date <- sub(st_nc_prefix, '', basename(snm_file_date))

  # Dates Start and End: list.files is auto sorted in ascending order
  if (it_df_ctr == 1) {
    snm_start_date <- snm_file_date
  }
  else {
    # this will give the final date
    snm_end_date <- snm_file_date
  }

  # Given this structure: ECMWF_utci_20100702_v1.0_con, sub out prefix and suffix
  print(spt_file)
  ncin <- nc_open(spt_file)

  nchist <- ncatt_get(ncin, 0, "history")

  # not using this missing value flag at the moment
  missingval <- str_match(nchist$value, "setmisstoc,\\s*(.*?)\\s* ")[,2]
  missingval <- as.numeric(missingval)

  lon <- ncvar_get(ncin, "lon")
  lat <- ncvar_get(ncin, "lat")
  tim <- ncvar_get(ncin, "time")
  tunits <- ncatt_get(ncin, "time", "units")

  nlon <- dim(lon)
  nlat <- dim(lat)
  ntim <- dim(tim)

  # convert time -- split the time units string into fields
  # tustr <- strsplit(tunits$value, " ")
  # tdstr <- strsplit(unlist(tustr)[3], "-")
  # tmonth <- as.integer(unlist(tdstr)[2])
  # tday <- as.integer(unlist(tdstr)[3])
  # tyear <- as.integer(unlist(tdstr)[1])
  # mytim <- chron(tim, origin = c(tmonth, tday, tyear))
```

```
      tmp_array <- ncvar_get(ncin, "utci")
      tmp_array <- tmp_array - 273.15

      lonlat <- as.matrix(expand.grid(lon = lon, lat = lat, hours = tim))
      temperature <- as.vector(tmp_array)
      tmp_df <- data.frame(cbind(lonlat, temperature))

      # extract a rectangle
      eps <- 1e-8
      minlat <- 22.25 - eps
      maxlat <- 23.50 + eps
      minlon <- 113.00 - eps
      maxlon <- 114.50 + eps
      # subset data
      subset_df <- tmp_df[tmp_df$lat >= minlat & tmp_df$lat <= maxlat &
                            tmp_df$lon >= minlon & tmp_df$lon <= maxlon, ]

      # add Date
      subset_df_date <- as_tibble(subset_df) %>% mutate(date = snm_file_date)

      # Add to list
      ls_df[[it_df_ctr]] <- subset_df_date

      # Close NC
      nc_close(ncin)
    }

    # List of DF to one DF
    df_all_nc <- do.call(rbind, ls_df)

    # Save File
    fname <- paste0(paste0(st_nc_prefix,
                           snm_start_date, "_to_", snm_end_date,
                           ".csv"))
    csvfile <- paste0(csvpath, fname)
    write.table(na.omit(df_all_nc), csvfile, row.names = FALSE, sep = ",")

    # Delete folders
    unlink(spn_zip, recursive=TRUE, force=TRUE)
    unlink(spn_unzip_folder, recursive=TRUE, force=TRUE)

  # end loop months groups
  }
# end loop year
}
```

Chapter 12

Code and Development

12.1 Files In and Out

12.1.1 File Path

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

12.1.1.1 Compose a Path

File Path might contain information related to the file, decompose the file path, keep the final N folder names, to be possibled stored as a variable in the datafile stored inside.

```
# Compose together a path
print(paste0('.Platform$file.sep=', .Platform$file.sep))
```

```
## [1] ".Platform$file.sep=/"
```

```
spn_file_path = file.path("C:", "Users", "fan", "R4Econ", "amto", "tibble",
                          "fs_tib_basics.Rmd",
                          fsep = .Platform$file.sep)
# print
print(spn_file_path)
```

```
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd"
```

12.1.1.2 Substring and File Name

From path, get file name without suffix.

- r string split
- r list last element
- r get file name from path
- r get file path no name

```
st_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
st_file_wth_suffix_s <- tail(strsplit(st_example, "/")[[1]],n=1)
st_file_wno_suffix_s <- tools::file_path_sans_ext(basename(st_example))
st_fullpath_nosufx_s <- sub('\\.Rmd$', '', st_example)
st_fullpath_noname_s <- dirname(st_example)

print(paste0('st_file_wth_suffix_s:', st_file_wth_suffix_s))
```

```
## [1] "st_file_wth_suffix_s:fs_tib_basics.Rmd"
```

```
print(paste0('st_file_wno_suffix_s:', st_file_wno_suffix_s))
```

```
## [1] "st_file_wno_suffix_s:fs_tib_basics"
```

```
print(paste0('st_fullpath_nosufx_s:', st_fullpath_nosufx_s))
```

```
## [1] "st_fullpath_nosufx_s:C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics"
```

```
print(paste0('st_fullpath_noname_s:', st_fullpath_noname_s))
```

```
## [1] "st_fullpath_noname_s:C:/Users/fan/R4Econ/amto/tibble"
```

12.1.1.3 Get Subset of Path Folder Names

File Path might contain information related to the file, decompose the file path, keep the final N folder names, to be possibled stored as a variable in the datafile stored inside.

```
# Compose together a path
spn_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
# Replace default system slash, assume spn was generated by system.
ls_srt_folders_file <- strsplit(st_example, .Platform$file.sep)[[1]]
# Keep the last N layers
it_folders_names_to_keep = 2
snm_file_name <- tail(strsplit(st_example, "/")[[1]],n=1)
ls_srt_folders_keep <- head(tail(ls_srt_folders_file, it_folders_names_to_keep+1),
                            it_folders_names_to_keep)
# Show folder names
print(paste0('snm_file_name:', snm_file_name))
```

```
## [1] "snm_file_name:fs_tib_basics.Rmd"
```

```
print(paste0('last ', it_folders_names_to_keep, ' folders:'))
```

```
## [1] "last 2 folders:"
```

```
print(ls_srt_folders_keep)
```

```
## [1] "amto"   "tibble"
```

Shorter lines, to make copying easier.

```
# inputs
it_folders_names_to_keep = 2
spn_example <- 'C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd'
# copy these
ls_srt_folders_name_keep <- tail(strsplit(st_example, "/")[[1]], n=it_folders_names_to_keep+1)
snm_file_name <- tail(ls_srt_folders_name_keep, 1)
ls_srt_folders_keep <- head(ls_srt_folders_name_keep, it_folders_names_to_keep)
# print
print(paste0('snm_file_name:', snm_file_name))
```

```
## [1] "snm_file_name:fs_tib_basics.Rmd"
```

```
print(paste0('last ', it_folders_names_to_keep, ' folders:'))
```

```
## [1] "last 2 folders:"
```

```
print(ls_srt_folders_keep)
```

```
## [1] "amto"   "tibble"
```

12.1.2 Text to File

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

12.1.2.1 Latex Table to File

Tabular outputs, text outputs, etc are saved as variables, which could be printed in console. They can also be saved to file for future re-used. For example, latex outputs need to be saved to file.

```
# Load Data
dt <- mtcars[1:4, 1:6]
# Generate latex string variable
st_out_tex <- kable(dt, "latex")
print(st_out_tex)
# File out
# fileConn <- file("./../../_file/tex/tex_sample_a_tab.tex")
fileConn <- file("_file/tex/tex_sample_a_tab.tex")
writeLines(st_out_tex, fileConn)
close(fileConn)
```

12.1.2.2 Create a Text File from Strings

```
st_file <- "\\documentclass[12pt,english]{article}
\\usepackage[bottom]{footmisc}
\\usepackage[urlcolor=blue]{hyperref}
\\begin{document}
\\title{A Latex Testing File}
\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{https://fan
\\maketitle
Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc orci.
\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}
Village closure information is taken from a village head survey.\\footnote{Generally students went t
output:
  pdf_document:
    pandoc_args: '..//..//_output_kniti_pdf.yaml'
    includes:
      in_header: '..//..//preamble.tex'
  html_document:
    pandoc_args: '..//..//_output_kniti_html.yaml'
    includes:
      in_header: '..//..//hdga.html'
\\end{document}"

print(st_file)
```

```
## [1] "\\documentclass[12pt,english]{article}\n\\usepackage[bottom]{footmisc}\n\\usepackage[urlcolo
```

```
fl_test_tex <- "_file/tex/test_fan.tex"
fileConn <- file(fl_test_tex)
writeLines(st_file, fileConn)
close(fileConn)
```

12.1.2.3 Open A File and Read Lines

Open and Replace Text in File:

```
fileConn <- file(fl_test_tex, "r")
st_file_read <- readLines(fileConn)
print(st_file_read)
```

```
##  [1] "\\documentclass[12pt,english]{article}"
##  [2] "\\usepackage[bottom]{footmisc}"
##  [3] "\\usepackage[urlcolor=blue]{hyperref}"
##  [4] "\\begin{document}"
```

```
##  [5] "\\title{A Latex Testing File}"
##  [6] "\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{ht
##  [7] "\\maketitle"
##  [8] "Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc
##  [9] "\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}"
## [10] "Village closure information is taken from a village head survey.\\footnote{Generally studen
## [11] "output:"
## [12] "  pdf_document:"
## [13] "    pandoc_args: '..//..//_output_kniti_pdf.yaml'"
## [14] "    includes:"
## [15] "      in_header: '..//..//preamble.tex'"
## [16] "  html_document:"
## [17] "    pandoc_args: '..//..//_output_kniti_html.yaml'"
## [18] "    includes:"
## [19] "      in_header: '..//..//hdga.html'"
## [20] "\\end{document}"
```

```
close(fileConn)
```

12.1.2.4 Open a File and Replace Some Lines

Append additional strings into the file after *html_document* with proper spacings:

```
# Read in Lines from existing file
fileConn <- file(fl_test_tex, "r")
st_file_read <- readLines(fileConn)
close(fileConn)

# Search and Replace String
st_search <- "html_document:"
st_replace <- paste0("html_document:\r\n",
                     "    toc: true\r\n",
                     "    number_sections: true\r\n",
                     "    toc_float:\r\n",
                     "      collapsed: false\r\n",
                     "      smooth_scroll: false\r\n",
                     "      toc_depth: 3")

# Search and Replace
st_file_updated <- gsub(x = st_file_read,
                        pattern = st_search,
                        replacement = st_replace)

# Print
print(st_file_updated)
```

```
##  [1] "\\documentclass[12pt,english]{article}"
##  [2] "\\usepackage[bottom]{footmisc}"
##  [3] "\\usepackage[urlcolor=blue]{hyperref}"
##  [4] "\\begin{document}"
##  [5] "\\title{A Latex Testing File}"
##  [6] "\\author{\\href{http://fanwangecon.github.io/}{Fan Wang} \\thanks{See information \\href{ht
##  [7] "\\maketitle"
##  [8] "Ipsum information dolor sit amet, consectetur adipiscing elit. Integer Latex placerat nunc
##  [9] "\\paragraph{\\href{https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3140132}{Data}}"
## [10] "Village closure information is taken from a village head survey.\\footnote{Generally studen
## [11] "output:"
## [12] "  pdf_document:"
## [13] "    pandoc_args: '..//..//_output_kniti_pdf.yaml'"
## [14] "    includes:"
```

```
## [15] "      in_header: '..//..//preamble.tex'"
## [16] "  html_document:\r\n    toc: true\r\n    number_sections: true\r\n    toc_float:\r\n      c
## [17] "    pandoc_args: '..//..//_output_kniti_html.yaml'"
## [18] "    includes:"
## [19] "      in_header: '..//..//hdga.html'"
## [20] "\\end{document}"
```

```
# Save Updated File
fl_srcrep_tex <- "_file/tex/test_fan_search_replace.tex"
fileConn_sr <- file(fl_srcrep_tex)
writeLines(st_file_updated, fileConn_sr)
close(fileConn_sr)
```

12.1.3 Rmd to HTML

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

12.1.3.1 Search and Find all Files in Repository

Search inside directories, for all files in a repository that have a particular suffix and that don't contain skip pattern list string items.

```
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/amto', 'C:/Users/fan/R4Econ/summarize')
spn_skip <- c('summarize', 'panel', 'support')

# Search and get all Path
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)

# Skip path if contains words in skip list
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
    st_fullpath_nosufx <- tail(strsplit(spt_file, "/")[[1]],n=1)
    print(paste0(spt_file, '---', st_fullpath_nosufx))
}
```

```
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_basics.Rmd---fs_ary_basics.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_generate.Rmd---fs_ary_generate.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_mesh.Rmd---fs_ary_mesh.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/fs_ary_string.Rmd---fs_ary_string.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/array/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/list/fs_lst_basics.Rmd---fs_lst_basics.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/list/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/fs_mat_generate.Rmd---fs_mat_generate.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/fs_mat_linear_algebra.Rmd---fs_mat_linear_algebra.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/matrix/main.Rmd---main.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd---fs_tib_basics.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_factors.Rmd---fs_tib_factors.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_na.Rmd---fs_tib_na.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_random_draws.Rmd---fs_tib_random_draws.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/fs_tib_string.Rmd---fs_tib_string.Rmd"
## [1] "C:/Users/fan/R4Econ/amto/tibble/main.Rmd---main.Rmd"
```

12.1.3.2 Search and Find all Git Modified or New Rmd

Search inside directories, for all files in a git repo folder that are new or have been modified. Ignore possible subset of file based on string search.

```
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/amto', 'C:/Users/fan/R4Econ/development')
spn_skip <- c('summarize', 'panel', 'support')
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
  spg_check_git_status <- paste0('git status -s ', spt_file)
  st_git_status <- toString(system(spg_check_git_status, intern=TRUE))
  bl_modified <- grepl(' M ', st_git_status, fixed=TRUE)
  bl_anewfile <- grepl('?? ', st_git_status, fixed=TRUE)
  bl_nochange <- (st_git_status == "")

  if (bl_modified == 1) {
    print(paste0('MODIFIED: ', spt_file))
  } else if (bl_anewfile == 1) {
    print(paste0('A NEW FL: ', spt_file))
  } else {
    print(paste0('NO CHNGE: ', spt_file))
  }
}
```

```
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_basics.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_generate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_mesh.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/fs_ary_string.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/array/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/list/fs_lst_basics.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/list/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/fs_mat_generate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/fs_mat_linear_algebra.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/matrix/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_basics.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_factors.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_na.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_random_draws.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/fs_tib_string.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/amto/tibble/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/fs_text_save_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/_file/rmd/main_mod.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_path.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_rmd_pdf_html.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/fs_text_save.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/inout/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/python/fs_python_reticulate.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/python/main.Rmd"
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/system/fs_system_shell.Rmd"
```

```
## [1] "NO CHNGE: C:/Users/fan/R4Econ/development/system/main.Rmd"
```

12.1.3.3 Resave an Existing File with Different Name Different Folder

Given an existing Rmd File, Resave it with a different name (add to name suffix), and then save in a different folder:

- old file: */R4Econ/development/fs_rmd_pdf_html.Rmd*
- new file: *R4Econ/development/inout/_file/rmd/fs_rmd_pdf_html_mod.Rmd*

```
# Serch Folder and skip list
spt_roots <- c('C:/Users/fan/R4Econ/development/inout/')
spn_skip <- c('_main', '_file')
ls_sfls <- list.files(path=spt_roots, recursive=T, pattern=".Rmd", full.names=T)
if(!missing(spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(spn_skip, collapse = "|"), ls_sfls)]
}

# Loop and print
for (spt_file in ls_sfls) {
  spt_new <- paste0('_file/rmd/')
  spn_new <- paste0(spt_new, sub('\\.Rmd$', '', basename(spt_file)), '_mod.Rmd')
  print(spt_new)
  print(spn_new)

  fileConn_rd <- file(spt_file, "r")
  st_file_read <- readLines(fileConn_rd)

  fileConn_sr <- file(spn_new)
  writeLines(st_file_read, fileConn_sr)

  close(fileConn_rd)
  close(fileConn_sr)
}
```

12.1.3.3.1 Replacment Function Change Markdown Hierarchy and Add to YAML

Given an existing Rmd File, Resave it with a different name, and replace (add in) additional yaml contents.

```
spn_file = '_file/rmd/fs_rmd_pdf_html_mod.Rmd'
fileConn_sr <- file(spn_file)
st_file <- readLines(fileConn_sr)
# print(st_file)

st_search <- "html_document:
    toc: true
    number_sections: true
    toc_float:
      collapsed: false
      smooth_scroll: false
      toc_depth: 3"
st_replace <- paste0("html_document:
    toc: true
    number_sections: true
    toc_float:
      collapsed: false
      smooth_scroll: false
      toc_depth: 3\n",
                     "    toc: true\n",
                     "    number_sections: true\n",
                     "    toc_float:\n",
```

```
                      "      collapsed: false\n",
                      "      smooth_scroll: false\n",
                      "      toc_depth: 3")
st_file_updated <- gsub(x = st_file,
                        pattern = st_search,
                        replacement = st_replace)

st_search <- "../../"
st_replace <- paste0("../../../../")
st_file_updated <- gsub(x = st_file_updated,
                        pattern = st_search,
                        replacement = st_replace)

st_file_updated <- gsub(x = st_file_updated, pattern = '# ', replacement = '# ')
st_file_updated <- gsub(x = st_file_updated, pattern = '## ', replacement = '## ')
st_file_updated <- gsub(x = st_file_updated, pattern = '### ', replacement = '# ')

spn_file = '_file/rmd/fs_rmd_pdf_html_mod.Rmd'
fileConn_sr <- file(spn_file)
st_file <- writeLines(st_file_updated, fileConn_sr)
```

12.1.3.4 Search and Render Rmd File and Save HTML, PDF or R

1. Search files satisfying conditions in a folder
2. knit files to HTML (and re-run the contents of the file)
3. Save output to a different folder

```
# Specify Parameters
ar_spt_root = c('C:/Users/fan/R4Econ/amto/array/', 'C:/Users/fan/R4Econ/math/integration')
bl_recursive = TRUE
st_rmd_suffix_pattern = "*.Rmd"
ar_spn_skip <- c('basics', 'integrate', 'main', 'mesh')
ls_bool_convert <- list(bl_pdf=TRUE, bl_html=TRUE, bl_R=TRUE)
spt_out_directory <- 'C:/Users/fan/Downloads/_data'
bl_verbose <- TRUE

# Get Path
ls_sfls  <- list.files(path=ar_spt_root,
                       recursive=bl_recursive,
                       pattern=st_rmd_suffix_pattern,
                       full.names=T)

# Exclude Some Files given ar_spn_skip
if(!missing(ar_spn_skip)) {
  ls_sfls <- ls_sfls[!grepl(paste(ar_spn_skip, collapse = "|"), ls_sfls)]
}

# Loop over files
for (spn_file in ls_sfls) {

  # Parse File Name
  spt_file <- dirname(spn_file)
  sna_file <- tools::file_path_sans_ext(basename(spn_file))

  # Output FIles
  spn_file_pdf <- paste0(spt_file, sna_file, '.pdf')
  spn_file_html <- paste0(spt_file, sna_file, '.html')
  spn_file_R <- paste0(spt_file, sna_file, '.R')
```

```
  # render to PDF
  if (ls_bool_convert$bl_pdf) {
    if (bl_verbose) message(paste0('spn_file_pdf:',spn_file_pdf, ', PDF started'))
    rmarkdown::render(spn_file, output_format='pdf_document',
                      output_dir = spt_out_directory, output_file = sna_file)
    if (bl_verbose) message(paste0('spn_file_pdf:',spn_file_pdf, ', PDF finished'))
    spn_pdf_generated <- paste0(spt_out_directory, '/', spn_file_pdf)
  }

  # render to HTML
  if (ls_bool_convert$bl_html) {
    if (bl_verbose) message(paste0('spth_html:',spn_file_html, ', HTML started.'))
    rmarkdown::render(spn_file, output_format='html_document',
                      output_dir = spt_out_directory, output_file = sna_file)
    if (bl_verbose) message(paste0('spth_html:',spn_file_html, ', HTML finished.'))
    spn_html_generated <- paste0(spt_out_directory, '/', spn_file_html)
  }

  # purl to R
  if (ls_bool_convert$bl_R) {
    if (bl_verbose) message(paste0('purl_to:', paste0(spn_file_R, ".R")))
    knitr::purl(spn_file, output=paste0(spt_out_directory, '/', sna_file, '.R'), documentation = 1)
    spn_R_generated <- paste0(spt_out_directory, '/', sna_file, '.R')
  }

  # return(list(ls_spt_pdf_generated=ls_spt_pdf_generated,
  #             ls_spt_html_generated=ls_spt_html_generated,
  #             ls_spt_R_generated=ls_spt_R_generated))


}
```

12.2 Python with R

12.2.1 Reticulate Basics

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

12.2.1.1 Basic Python Tests with RMD

Could specify: *python, engine.path = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe"*, this is already set inside Rprofile: *knitr::opts_chunk$set(engine.path = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.ex*

```
1+1
```

```
## 2
```

12.2.1.2 Install and Python Path

Install reticulate from github directly to get latest version: *devtools::install_github("rstudio/reticulate")*

Check python version on computer:

```
Sys.which('python')
```

```
##                                                    python
## "G:\\ProgramData\\Anaconda3\\envs\\wk_pyfan\\python.exe"
```

After installing reticulate, load in the library: library(reticulate). With "py_config()" to see python config. First time, might generate "No non-system installation of Python could be found." and ask if want to install Miniconda. Answer NO.

Correct outputs upon checking py_config():

```
python:         C:/ProgramData/Anaconda3/python.exe
libpython:      C:/ProgramData/Anaconda3/python37.dll
pythonhome:     C:/ProgramData/Anaconda3
version:        3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)]
Architecture:   64bit
numpy:          C:/ProgramData/Anaconda3/Lib/site-packages/numpy
numpy_version:  1.19.1

python versions found:
 C:/ProgramData/Anaconda3/python.exe
 C:/ProgramData/Anaconda3/envs/wk_cgefi/python.exe
 C:/ProgramData/Anaconda3/envs/wk_jinja/python.exe
 C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe
```

Set which python to use:

```
# Sys.setenv(RETICULATE_PYTHON = "C:/programdata/Anaconda3/python.exe")
# Sys.setenv(RETICULATE_PYTHON = "C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe")
library(reticulate)
# What is the python config
py_config()
```

```
## python:         G:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe
## libpython:      G:/ProgramData/Anaconda3/envs/wk_pyfan/python38.dll
## pythonhome:     G:/ProgramData/Anaconda3/envs/wk_pyfan
## version:        3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
## Architecture:   64bit
## numpy:          G:/ProgramData/Anaconda3/envs/wk_pyfan/Lib/site-packages/numpy
## numpy_version:  1.19.4
##
## NOTE: Python version was forced by use_python function
```

```
# set python
# use_python("C:/programdata/Anaconda3/python.exe")
# use_python("C:/ProgramData/Anaconda3/envs/wk_pyfan/python.exe")
use_condaenv('wk_pyfan')
# Sys.which('python')
py_run_string('print(1+1)')
```

12.2.1.3 Error

12.2.1.3.1 py__call__impl error The error appeared when calling any python operations, including "1+1", resolved after installing reticulate from github: *devtools::install_github("rstudio/reticulate")*

```
Error in py_call_impl(callable, dots$args, dots$keywords) :
  TypeError: use() got an unexpected keyword argument 'warn'
```

12.3 Command Line

12.3.1 Shell and System Commands

Go back to fan's REconTools Package, R Code Examples Repository (bookdown site), or Intro Stats with R Repository (bookdown site).

12.3.1.1 Basic Shell Commands

Run basic shell commands in windows:

```
# detect current path
print(toString(shell(paste0("echo %cd%"), intern=TRUE)))
```

```
## [1] "G:\\repos\\R4Econ"
```

```
# Show directory
print(toString(shell(paste0("dir"), intern=TRUE)))
```

```
## [1] " Volume in drive G is New Volume,  Volume Serial Number is 009D-37E9, ,  Directory of G:\\re
```

12.3.1.2 Run Python Inside a Conda Environment

Use shell rather than system to activate a conda environment, check python version:

```
# activate conda env
print(toString(shell(paste0("activate base & python --version"), intern=TRUE)))
```

```
## [1] ", G:\\repos\\R4Econ>conda.bat activate base , Python 3.8.5"
```

Activate conda env and run a line:

```
spg_runpython <- paste0("activate base &",
                        "python --version &",
                        "python -c ",
                        "\"st_var='this is string var';",
                        "print(f'{st_var}');",
                        "\"")
print(toString(shell(spg_runpython, intern=TRUE)))
```

```
## [1] ", G:\\repos\\R4Econ>conda.bat activate base , Python 3.8.5, this is string var"
```

Appendix A

Index and Code Links

A.1 Array, Matrix, Dataframe links

A.1.1 Section 1.1 List links

1. Multi-dimensional Named Lists: **rmd** | **r** | **pdf** | **html**
   - Initiate Empty List. Named one and two dimensional lists. List of Dataframes.
   - Collapse named and unamed list to string and print input code.
   - **r**: *deparse(substitute()) + vector(mode = "list", length = it_N) + names(list) <- paste0('e',seq()) + dimnames(ls2d)[[1]] <- paste0('r',seq()) + dimnames(ls2d)[[2]] <- paste0('c',seq())*
   - **tidyr**: *unnest()*

A.1.2 Section 1.2 Array links

1. Arrays Operations in R: **rmd** | **r** | **pdf** | **html**
   - Basic array operations in R, rep, head, tail, na, etc.
   - E notation.
   - Get N cuts from M points.
   - **r**: *rep() + head() + tail() + na_if() + Re()*
2. Generate Special Arrays: **rmd** | **r** | **pdf** | **html**
   - Generate equi-distance, special log spaced array.
   - Generate probability mass function with non-unique and non-sorted value and probability arrays.
   - **r**: *seq() + sort() + runif() + ceiling()*
   - **stats**: *aggregate()*
3. String Operations: **rmd** | **r** | **pdf** | **html**
   - Split, concatenate, subset, replace, substring strings.
   - Convert number to string without decimal and negative sign.
   - **r**: *paste0() + sub() + gsub() + grepl() + sprintf()*
4. Meshgrid Matrices, Arrays and Scalars: **rmd** | **r** | **pdf** | **html**
   - Meshgrid Matrices, Arrays and Scalars to form all combination dataframe.
   - **tidyr**: *expand_grid() + expand.grid()*

A.1.3 Section 1.3 Matrix links

1. Matrix Basics: **rmd** | **r** | **pdf** | **html**
   - Generate and combine NA, fixed and random matrixes. Name columns and rows.
   - **R**: *rep() + rbind() + matrix(NA) + matrix(NA_real_) + matrix(NA_integer_) + colnames() + rownames()*
2. Linear Algebra Operations: **rmd** | **r** | **pdf** | **html**

A.1.4 Section 1.4 Variables in Dataframes links

1. Tibble Basics: **rmd** | **r** | **pdf** | **html**
   - generate tibbles, rename tibble variables, tibble row and column names
   - rename numeric sequential columns with string prefix and suffix
   - **dplyr**: *as_tibble(mt) + rename_all(~c(ar_names)) + rename_at(vars(starts_with("xx")), funs(str_replace(., "yy", "yyyy")) + rename_at(vars(num_range('',ar_it)), funs(paste0(st,.))) + rowid_to_column() + colnames + rownames*
2. Label and Combine Factor Variables: **rmd** | **r** | **pdf** | **html**
   - Convert numeric variables to factor variables, generate joint factors, and label factors.
   - Graph MPG and 1/4 Miles Time (qsec) from the mtcars dataset over joint shift-type (am) and engine-type (vs) categories.
   - **forcats**: *as_factor() + fct_recode() + fct_cross()*
3. Randomly Draw Subsets of Rows from Matrix: **rmd** | **r** | **pdf** | **html**
   - Given matrix, randomly sample rows, or select if random value is below threshold.
   - **r**: *rnorm() + sample() + df[sample(dim(df)[1], it_M, replace=FALSE),]*
   - **dplyr**: *case_when() + mutate(var = case_when(rnorm(n(),mean=0,sd=1) < 0 ~ 1, TRUE ~ 0)) %>% filter(var == 1)*
4. Generate Variables Conditional on Other Variables: **rmd** | **r** | **pdf** | **html**
   - Use case_when to generate elseif conditional variables: NA, approximate difference, etc.
   - **dplyr**: *case_when() + na_if() + mutate(var = na_if(case_when(rnorm(n())< 0 ~ -99, TRUE ~ mpg), -99))*
   - **r**: *e-notation + all.equal() + isTRUE(all.equal(a,b,tol)) + is.na() + NA_real_ + NA_character_ + NA_integer_*
5. R Tibble Dataframe String Manipulations: **rmd** | **r** | **pdf** | **html**
   - There are multiple CEV files, each containing the same file structure but simulated
   - with different parameters, gather a subset of columns from different files, and provide
   - with correct attributes based on CSV file names.
   - **r**: *cbind(ls_st, ls_st) + as_tibble(mt_st)*

A.2 Summarize Data links

A.2.1 Section 2.1 Counting Observation links

1. Counting Basics: **rmd** | **r** | **pdf** | **html**
   - uncount to generate panel skeleton from years in survey
   - **dplyr**: *uncount(yr_n) + group_by() + mutate(yr = row_number() + start_yr)*

A.2.2 Section 2.2 Sorting, Indexing, Slicing links

1. Sorted Index, Interval Index and Expand Value from One Row: **rmd** | **r** | **pdf** | **html**
   - Sort and generate index for rows
   - Generate negative and positive index based on deviations
   - Populate Values from one row to other rows
   - **dplyr**: *arrange() + row_number() + mutate(lowest = min(Sepal.Length)) + case_when(row_number()==x ~ Septal.Length) + mutate(Sepal.New = Sepal.Length[Sepal.Index == 1])*
2. Group and sort, and Slice and Summarize: **rmd** | **r** | **pdf** | **html**
   - Group a dataframe by a variable, sort within group by another variable, keep only highest rows.
   - **dplyr**: *arrange() + group_by() + slice_head(n=1)*

A.2.3 Section 2.3 Group Statistics links

1. Cummean Test, Cumulative Mean within Group: **rmd** | **r** | **pdf** | **html**
   - There is a dataframe with a grouping variable and some statistics sorted by another within group
   - variable, calculate the cumulative mean of that variable.
   - **dplyr**: *cummean() + group_by(id, isna = is.na(val)) + mutate(val_cummean = ifelse(isna, NA, cummean(val)))*

2. Count Unique Groups and Mean within Groups: **rmd** | **r** | **pdf** | **html**
   - Unique groups defined by multiple values and count obs within group.
   - Mean, sd, observation count for non-NA within unique groups.
   - **dplyr**: *group_by() + summarise(n()) + summarise_if(is.numeric, funs(mean = mean(., na.rm = TRUE), n = sum(is.na(.)==0)))*
3. By Groups, One Variable All Statistics: **rmd** | **r** | **pdf** | **html**
   - Pick stats, overall, and by multiple groups, stats as matrix or wide row with name=(ctsvar + catevar + catelabel).
   - **tidyr**: *group_by() + summarize_at(, funs()) + rename(!!var := !!sym(var)) + mutate(!!var := paste0(var,'str',!!!syms(vars))) + gather() + unite() + spread(varcates, value)*
4. By within Individual Groups Variables, Averages: **rmd** | **r** | **pdf** | **html**
   - By Multiple within Individual Groups Variables.
   - Averages for all numeric variables within all groups of all group variables. Long to Wide to very Wide.
   - **tidyr**: *gather() + group_by() + summarise_if(is.numeric, funs(mean(., na.rm = TRUE))) + mutate(all_m_cate = paste0(variable, '_c', value)) + unite() + spread()*

A.2.4 Section 2.4 Distributional Statistics links

1. Tibble Basics: **rmd** | **r** | **pdf** | **html**
   - input multiple variables with comma separated text strings
   - quantitative/continuous and categorical/discrete variables
   - histogram and summary statistics
   - **tibble**: *ar_one <- c(107.72,101.28) + ar_two <- c(101.72,101.28) + mt_data <- cbind(ar_one, ar_two) + as_tibble(mt_data)*

A.2.5 Section 2.5 Summarize Multiple Variables links

1. Apply the Same Function over Columns of Matrix: **rmd** | **r** | **pdf** | **html**
   - Replace NA values in selected columns by alternative values.
   - Cumulative sum over multiple variables.
   - Rename various various with common prefix and suffix appended.
   - **r**: *cumsum() + gsub() + mutate_at(vars(contains('V')), .funs = list(cumu = ~cumsum(.))) + rename_at(vars(contains("V")), list(~gsub("M", "", .)))*
   - **dplyr**: *rename_at() + mutate_at() + rename_at(vars(starts_with("V")), funs(str_replace(., "V", "var"))) + mutate_at(vars(one_of(c('var1', 'var2'))), list(~replace_na(., 99)))*

A.3 Functions links

A.3.1 Section 3.1 Dataframe Mutate links

1. Nonlinear Function of Scalars and Arrays over Rows: **rmd** | **r** | **pdf** | **html**
   - Five methods to evaluate scalar nonlinear function over matrix.
   - Evaluate non-linear function with scalar from rows and arrays as constants.
   - **r**: $.fl_A + fl_A$ *='(., 'fl_A') + .[[svr_fl_A]]*
   - **dplyr**: *rowwise() + mutate(out = funct(inputs))*
2. Evaluate Functions over Rows of Meshes Matrices: **rmd** | **r** | **pdf** | **html**
   - Mesh states and choices together and rowwise evaluate many matrixes.
   - Cumulative sum over multiple variables.
   - Rename various various with common prefix and suffix appended.
   - **r**: *ffi <- function(fl_A, ar_B)*
   - **tidyr**: *expand_grid() + rowwise() + df %>% rowwise() %>% mutate(var = ffi(fl_A, ar_B))*
   - **ggplot2**: *geom_line() + facet_wrap() + geom_hline() + facet_wrap(. ~ var_id, scales = 'free') + geom_hline(yintercept=0, linetype="dashed", color="red", size=1) +*

A.3.2 Section 3.2 Dataframe Do Anything links

1. Dataframe Row to Array (Mx1 by N) to (MxQ by N+1): **rmd** | **r** | **pdf** | **html**

- Generate row value specific arrays of varying Length, and stack expanded dataframe.
- Given row-specific information, generate row-specific arrays that expand matrix.
- **dplyr**: *do() + unnest() + left_join() + df %>% group_by(ID) %>% do(inc = rnorm(.Q, mean = .mean, sd=.$sd)) %>% unnest(c(inc))*

2. Dataframe Subset to Scalar (MxP by N) to (Mx1 by 1): **rmd** | **r** | **pdf** | **html**
   - MxQ rows to Mx1 Rows. Group dataframe by categories, compute category specific output scalar or arrays based on within category variable information.
   - **dplyr**: *group_by(ID) + do(inc = rnorm(.N, mean = .mn, sd=.$sd)) + unnest(c(inc)) + left_join(df, by="ID")*
3. Dataframe Subset to Dataframe (MxP by N) to (MxQ by N+Z-1): **rmd** | **r** | **pdf** | **html**
   - Group by mini dataframes as inputs for function. Stack output dataframes with group id.
   - **dplyr**: *group_by() + do() + unnest()*

A.3.3 Section 3.3 Apply and pmap links

1. Apply and Sapply function over arrays and rows: **rmd** | **r** | **pdf** | **html**
   - Evaluate function f(x_i,y_i,c), where c is a constant and x and y vary over each row of a matrix, with index i indicating rows.
   - Get same results using apply and sapply with defined and anonymous functions.
   - **r**: *do.call() + apply(mt, 1, func) + sapply(ls_ar, func, ar1, ar2)*
2. Mutate rowwise, mutate pmap, and rowwise do unnest: **rmd** | **r** | **pdf** | **html**
   - Evaluate function f(x_i,y_i,c), where c is a constant and x and y vary over each row of a matrix, with index i indicating rows.
   - Get same results using various types of mutate rowwise, mutate pmap and rowwise do unnest.
   - **dplyr**: *rowwise() + do() + unnest()*
   - **purrr**: *pmap(func)*
   - **tidyr**: *unlist()*

A.4 Multi-dimensional Data Structures links

A.4.1 Section 4.1 Generate, Gather, Bind and Join links

1. R dplyr Group by Index and Generate Panel Data Structure: **rmd** | **r** | **pdf** | **html**
   - Build skeleton panel frame with N observations and T periods with gender and height.
   - Generate group Index based on a list of grouping variables.

   - **r**: *runif() + rnorm() + rbinom(n(), 1, 0.5) + cumsum()*
   - **dplyr**: *group_by() + row_number() + ungroup() + one_of() + mutate(var = (row_number()==1)*1)**
   - **tidyr**: *uncount()*
2. R DPLYR Join Multiple Dataframes Together: **rmd** | **r** | **pdf** | **html**
   - Join dataframes together with one or multiple keys. Stack dataframes together.
   - **dplyr**: *filter() + rename(!!sym(vsta) := !!sym(vstb)) + mutate(var = rnom(n())) + left_join(df, by=(c('id'='id', 'vt'='vt'))) + left_join(df, by=setNames(c('id', 'vt'), c('id', 'vt'))) + bind_rows()*
3. R Gather Data Columns from Multiple CSV Files: **rmd** | **r** | **pdf** | **html**
   - There are multiple CEV files, each containing the same file structure but simulated
   - with different parameters, gather a subset of columns from different files, and provide
   - with correct attributes based on CSV file names.
   - Separate numeric and string components of a string variable value apart.
   - **r**: *file() + writeLines() + readLines() + close() + gsub() + read.csv() + do.call(bind_rows, ls_df) + apply()*
   - **tidyr**: *separate()*
   - **regex**: *(?<=[A-Za-z])(?=[-0-9])*

A.4.2 Section 4.2 Wide and Long links

1. TIDYR Pivot Wider and Pivot Longer Examples: **rmd** | **r** | **pdf** | **html**

- Long roster to wide roster and cumulative sum attendance by date.
- **dplyr**: *mutate(var = case_when(rnorm(n()) < 0 ~ 1, TRUE ~ 0)) + rename_at(vars(num_range('', ar_it)), list(~paste0(st_prefix, . ,''))) + mutate_at(vars(contains(str)), list(~replace_na(., 0))) + mutate_at(vars(contains(str)), list(~cumsum(.)))*

2. R Wide Data to Long Data Example (TIDYR Pivot Longer): **rmd** | **r** | **pdf** | **html**
   - A matrix of ev given states, rows are states and cols are shocks. Convert to Long table with shock and state values and ev.
   - **dplyr**: *left_join() + pivot_longer(cols = starts_with('zi'), names_to = c('zi'), names_pattern = paste0("zi(.)"), values_to = "ev")*

A.4.3 Section 4.3 Join and Compare links

1. Find Closest Values Along Grids: **rmd** | **r** | **pdf** | **html**
   - There is an array (matrix) of values, find the index of the values closest to another value.
   - **r**: *do.call(bind_rows, ls_df)*
   - **dplyr**: *left_join(tb, by=(c('vr_a'='vr_a', 'vr_b'='vr_b')))*

A.5 Linear Regression links

A.5.1 Section 5.1 OLS and IV links

1. IV/OLS Regression: **rmd** | **r** | **pdf** | **html**
   - R Instrumental Variables and Ordinary Least Square Regression store all Coefficients and Diagnostics as Dataframe Row.
   - **aer**: *library(aer) + ivreg(as.formula, diagnostics = TRUE)*
2. M Outcomes and N RHS Alternatives: **rmd** | **r** | **pdf** | **html**
   - There are M outcome variables and N alternative explanatory variables. Regress all M outcome variables on N endogenous/independent right hand side variables one by one, with controls and/or IVs, collect coefficients.
   - **dplyr**: *bind_rows(lapply(listx, function(x)(bind_rows(lapply(listy, regf.iv))) + starts_with() + ends_with() + reduce(full_join)*

A.5.2 Section 5.2 Decomposition links

1. Regression Decomposition: **rmd** | **r** | **pdf** | **html**
   - Post multiple regressions, fraction of outcome variables' variances explained by multiple subsets of right hand side variables.
   - **dplyr**: *gather() + group_by(var) + mutate_at(vars, funs(mean = mean(.))) + rowSums(mat*mat) + mutate_if(is.numeric, funs(frac = (./value_var)))*

A.6 Nonlinear and Other Regressions links

A.6.1 Section 6.1 Logit Regression links

1. Logit Regression: **rmd** | **r** | **pdf** | **html**
   - Logit regression testing and prediction.
   - **stats**: *glm(as.formula(), data, family='binomial') + predict(rs, newdata, type = "response")*
2. Estimate Logistic Choice Model with Aggregate Shares: **rmd** | **r** | **pdf** | **html**
   - Aggregate share logistic OLS with K worker types, T time periods and M occupations.
   - Estimate logistic choice model with aggregate shares, allowing for occupation-specific wages and occupation-specific intercepts.
   - Estimate allowing for K and M specific intercepts, K and M specific coefficients, and homogeneous coefficients.
   - Create input matrix data structures for logistic aggregate share estimation.
   - **stats**: *lm(y ~ . -1)*
3. Fit Prices Given Quantities Logistic Choice with Aggregate Data: **rmd** | **r** | **pdf** | **html**
   - A multinomial logistic choice problem generates choice probabilities across alternatives, find the prices that explain aggregate shares.

- **stats**: *lm(y ~ . -1)*

A.6.2 Section 6.2 Quantile Regression links

1. Quantile Regressions with Quantreg: **rmd** | **r** | **pdf** | **html**
   - Quantile regression with continuous outcomes. Estimates and tests quantile coefficients.
   - **quantreg**: *rq(mpg ~ disp + hp + factor(am), tau = c(0.25, 0.50, 0.75), data = mtcars) + anova(rq(), test = "Wald", joint=TRUE) + anova(rq(), test = "Wald", joint=FALSE)*

A.7 Optimization links

A.7.1 Section 7.1 Bisection links

1. Concurrent Bisection over Dataframe Rows: **rmd** | **r** | **pdf** | **html**
   - Post multiple regressions, fraction of outcome variables' variances explained by multiple subsets of right hand side variables.
   - **tidyr**: *pivot_longer(cols = starts_with('abc'), names_to = c('a', 'b'), names_pattern = paste0('prefix', "(.)_(.)"), values_to = val) + pivot_wider(names_from = !!sym(name), values_from = val) + mutate(!!sym(abc) := case_when(efg < 0 ~ !!sym(opq), TRUE ~ iso))*
   - **gglot2**: *geom_line() + facet_wrap() + geom_hline()*

A.8 Mathematics links

A.8.1 Section 8.1 Production Function links

1. Nested Constant Elasticity of Substitution Production Function: **rmd** | **r** | **pdf** | **html**
   - A nested-CES production function with nest-specific elasticities.
   - Re-state the nested-CES problem as several sub-problems.
   - Marginal products and its relationship to prices in expenditure minimization.

A.8.2 Section 8.2 Basics links

1. Rescaling Bounded Parameter to be Unbounded and Positive and Negative Exponents with Different Bases: **rmd** | **r** | **pdf** | **html**
   - Log of alternative bases, bases that are not e, 10 or 2.
   - A parameter is constrained between 1 and negative infinity, use exponentials of different bases to scale the bounded parameter to an unbounded parameter.
   - Positive exponentials are strictly increasing. Negative exponentials are strictly decreasing.
   - A positive number below 1 to a negative exponents is above 1, and a positive number above 1 to a negative exponents is below 1.
   - **graphics**: *plot(x, y) + title() + legend()*
2. Quadratic and other Rescaling of Parameters with Fixed Min and Max: **rmd** | **r** | **pdf** | **html**
   - Given a < x < b, use f(x) to rescale x, such that f(a)=a, f(b)=b, but f(z)=0.5*z for some z between a and b. Solve using the quadratic function with three equations and three unknowns uniquely.
3. Find the Closest Point Along a Line to Another Point: **rmd** | **r** | **pdf** | **html**
   - A line crosses through the origin, what is the closest point along this line to another point.
   - Graph several functions jointly with points and axis.
   - **graphics**: *par(mfrow = c(1, 1)) + curve(fc) + points(x, y) + abline(v=0, h=0)*
4. linear solve x with f(x) = 0: **rmd** | **r** | **pdf** | **html**
   - Evaluate and solve statistically relevant problems with one equation and one unknown that permit analytical solutions.

A.8.3 Section 8.3 Inequality Models links

1. GINI for Discrete Samples or Discrete Random Variable: **rmd** | **r** | **pdf** | **html**
   - Given sample of data points that are discrete, compute the approximate GINI coefficient.

- Given a discrete random variable, compute the GINI coefficient.
- **r**: *sort() + cumsum() + sum()*

2. CES and Atkinson Inequality Index: **rmd** | **r** | **pdf** | **html**
   - Analyze how changing individual outcomes shift utility given inequality preference parameters.
   - Discrete a continuous normal random variable with a binomial discrete random variable.
   - Draw Cobb-Douglas, Utilitarian and Leontief indifference curve.
   - **r**: *apply(mt, 1, funct(x){}) + do.call(rbind, ls_mt)*
   - **tidyr**: *expand_grid()*
   - **ggplot2**: *geom_line() + facet_wrap()*
   - **econ**: *Atkinson (JET, 1970)*
3. Inequality in Environmental Exposure Across Population Groups: **rmd** | **r** | **pdf** | **html**
   - Simulate population distribution by location and demographic groups.
   - Simulate pollution exposures by location.
   - Compute inequality in environmental exposure across population groups, given location-specific environmental data and location-specific population information.
   - **r**: *matrix()*
   - **stats**: *runif() + sum()*
   - **dplyr**: *arrange() + group_by() + left_join() + filter() + slice()*

A.9 Statistics links

A.9.1 Section 9.1 Distributions links

1. Integrate Normal Shocks: **rmd** | **r** | **pdf** | **html**
   - Random Sampling (Monte Carlo) integrate shocks.
   - Trapezoidal rule (symmetric rectangles) integrate normal shock.

A.9.2 Section 9.2 Discrete Random Variable links

1. Binomial Approximation of Normal: **rmd** | **r** | **pdf** | **html**
   - Approximate a continuous normal random variable with a discrete binomial random variable.
   - **r**: *hist() + plot()*
   - **stats**: *dbinom() + rnorm()*
2. Obtaining Joint Distribution from Marginal with Rectilinear Restrictions: **rmd** | **r** | **pdf** | **html**
   - Solve for joint distributional mass given marginal distributional mass given rectilinear assumptions.
   - **r**: *qr()*
3. Obtaining Joint Distribution from Conditional with Rectilinear Restrictions: **rmd** | **r** | **pdf** | **html**
   - Solve for joint distributional mass given conditional distributional mass given rectilinear assumptions.
   - **r**: *qr() + solve() + matrix()*

A.10 Tables and Graphs links

A.10.1 Section 10.1 R Base Plots links

1. R Base Plot Line with Curves and Scatter: **rmd** | **r** | **pdf** | **html**
   - Plot scatter points, line plot and functional curve graphs together.
   - Set margins for legend to be outside of graph area, change line, point, label and legend sizes.
   - Generate additional lines for plots successively, record successively, and plot all steps, or initial steps results.
   - **r**: *plot() + curve() + legend() + title() + axis() + par() + recordPlot()*

A.10.2 Section 10.2 ggplot Line Related Plots links

1. ggplot Line Plot Multiple Categorical Variables With Continuous Variable: **rmd** | **r** | **pdf** | **html**
   - One category is subplot, one category is line-color, one category is line-type.

- **ggplot**: *ggplot() + facet_wrap() + geom_smooth() + geom_hline() + scale_colour_manual() + scale_shape_discrete() + scale_linetype_manual() + scale_x_continuous() + scale_y_continuous() + theme_bw() + theme()*

A.10.3 Section 10.3 ggplot Scatter Related Plots links

1. ggplot Scatter Plot Three Continuous Variables and Multiple Categorical Variables: **rmd** | **r** | **pdf** | **html**
   - Two continuous variables for the x-axis and the y-axis, another continuous variable for size of scatter, other categorical variables for scatter shape and size.
   - **ggplot**: *ggplot() + geom_jitter() + geom_smooth() + scale_colour_manual() + scale_shape_discrete() + scale_linetype_manual() + scale_x_continuous() + scale_y_continuous() + theme_bw() + theme()*
2. ggplot Multiple Scatter-Lines and Facet Wrap Over Categories: **rmd** | **r** | **pdf** | **html**
   - ggplot multiple lines with scatter as points and connecting lines.
   - Facet wrap to generate subfigures for sub-categories.
   - Generate separate plots from data saved separately.
   - **r**: *apply*
   - **ggplot**: *facet_wrap() + geom_smooth() + geom_point() + facet_wrap() + scale_colour_manual() + scale_shape_manual() + scale_linetype_manual()*

A.10.4 Section 10.4 Write and Read Plots links

1. Base R Save Images At Different Sizes: **rmd** | **r** | **pdf** | **html**
   - Base R store image core, add legends/titles/labels/axis of different sizes to save figures of different sizes.
   - **r**: *png() + setEPS() + postscript() + dev.off()*

A.11 Get Data links

A.11.1 Section 11.1 Environmental Data links

1. CDS ECMWF Global Enviornmental Data Download: **rmd** | **r** | **pdf** | **html**
   - Using Python API get get ECMWF ERA5 data.
   - Dynamically modify a python API file, run python inside a Conda virtual environment with R-reticulate.
   - **r**: *file() + writeLines() + unzip() + list.files() + unlink()*
   - **r-reticulate**: *use_python() + Sys.setenv(RETICULATE_PYTHON = spth_conda_env)*

A.12 Code and Development links

A.12.1 Section 12.1 Files In and Out links

1. Decompose File Paths to Get Folder and Files Names: **rmd** | **r** | **pdf** | **html**
   - Decompose file path and get file path folder names and file name.
   - **r**: *.Platform$file.sep + tail() + strsplit() + basename() + dirname() + substring()*
2. Save Text to File, Read Text from File, Replace Text in File: **rmd** | **r** | **pdf** | **html**
   - Save data to file, read text from file, replace text in file.
   - **r**: *kable() + file() + writeLines() + readLines() + close() + gsub()*
3. Convert R Markdown File to R, PDF and HTML: **rmd** | **r** | **pdf** | **html**
   - Find all files in a folder with a particula suffix, with exclusion.
   - Convert R Markdow File to R, PDF and HTML.
   - Modify markdown pounds hierarchy.
   - **r**: *file() + writeLines() + readLines() + close() + gsub()*

A.12.2 Section 12.2 Python with R links

1. Python in R with Reticulate: **rmd** | **r** | **pdf** | **html**

- Use Python in R with Reticulate
- **reticulate**: *py_config() + use_condaenv() + py_run_string() + Sys.which('python')*

A.12.3 Section 12.3 Command Line links

1. System and Shell Commands in R: **rmd** | **r** | **pdf** | **html**
   - Run system executable and shell commands.
   - Activate conda environment with shell script.
   - **r**: *system() + shell()*

Bibliography

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Wang, F. (2020). *REconTools: R Tools for Panel Data and Optimization*. R package version 0.0.0.9000.

Wickham, H. (2019). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.3.0.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.18.