ECON 390 Homework 1

Creating Variables

1. Create a vector of submission IDs of you and your group members called "sub ids"

```
sub_ids = c("SID2")
sub_ids
```

```
## [1] "SID2"
```

2. Create a vector of the integers from 1 to 5 called "my vec".

```
my_vec = c(1:5)
my_vec
```

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

3. Update "my vec" by adding 10 to each element of "my vec"

```
my_vec = my_vec + 10
my_vec
```

```
## [1] 11 12 13 14 15
```

4. Create a vector of integers from 1994 to 2021 called "years alive."

```
years_alive = c(1994:2021)
years_alive
```

```
##  [1] 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
## [16] 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021
```

5. Create a vector of logicals testing if an element in the "years alive" vector is a leap year

```
names(years_alive) = c(1994:2021)
ifleap = list(names(years_alive), years_alive%%4 == 0)
ifleap
```

```
## [[1]]
##  [1] "1994" "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003"
## [11] "2004" "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013"
## [21] "2014" "2015" "2016" "2017" "2018" "2019" "2020" "2021"
##
## [[2]]
##  1994  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004  2005  2006
## FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
##  2007  2008  2009  2010  2011  2012  2013  2014  2015  2016  2017  2018  2019
## FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  2020  2021
##  TRUE FALSE
```

6. Draw 1000 simulations from a N(0, 1) distribution and call it "norm draws".

```
norm_draws = rnorm(1000, mean = 0, sd = 1)
```

7. Create a vector of that is the log of the draws from item 6 which is called "log draws".

```
log_draws = log(norm_draws)
```

```
## Warning in log(norm_draws): NaNs produced
```

8. Create a logical vector called "draws NaN" that tests which elements of "log draws" are NaNs.

```
draws_NaN = is.na(log_draws)
```

9. Create a sequence of integers from 1 to N, where N is the number of NaN elements in "log draws".

```
sequenceOfNaNs =  seq(which(is.na(log_draws)))
sequenceOfNaNs
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
## [163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## [199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
## [217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
## [235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
## [253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## [271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
## [289] 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
## [307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
```

```
## [325] 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
## [343] 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
## [361] 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
## [379] 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
## [397] 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
## [415] 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
## [433] 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
## [451] 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
## [469] 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
## [487] 487 488 489 490 491 492 493 494 495 496 497
```

10. Use modular arithmetic operations to create a vector called "class length" whose first element is the number of whole hours in one of our class meetings and the second is the number of leftover minutes. E.g. an hour and a half would be "c(1,30)".

```
class_length = c(75%/%60, 75%%60)
class_length
```

```
## [1]  1 15
```

Creating Data Sets

1. Create a variable called "mtcars list" that is each variable in mtcars as a different in the list.

```
mtcars_list = as.list(ls(mtcars))
mtcars_list
```

```
## [[1]]
## [1] "am"
##
## [[2]]
## [1] "carb"
##
## [[3]]
## [1] "cyl"
##
## [[4]]
## [1] "disp"
##
## [[5]]
## [1] "drat"
##
## [[6]]
## [1] "gear"
##
## [[7]]
## [1] "hp"
##
## [[8]]
## [1] "mpg"
##
```

```
## [[9]]
## [1] "qsec"
##
## [[10]]
## [1] "vs"
##
## [[11]]
## [1] "wt"
```

2. Create a 2×13 matrix of the letters of the alphabet where they are read from left to right. Call this matrix "letter mat".

```
letter_mat = matrix(letters, nrow = 2, ncol = 13, byrow = 1)
letter_mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"
## [2,] "n"  "o"  "p"  "q"  "r"  "s"  "t"  "u"  "v"  "w"   "x"   "y"   "z"
```

3. Create a matrix that is 5×5 of new N(0, 1) draws called "norm draws mat".

```
norm_draws_mat = matrix(rnorm(n = 25, mean = 0, sd = 1), nrow = 5, ncol = 5)
norm_draws_mat
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] -0.62695680 -1.6430087 -3.1130609  0.3762678 -0.7124337
## [2,]  0.07313033  3.1963888 -0.3752030  0.5815334 -0.8501695
## [3,]  0.58434304 -1.0524661  0.7568669  0.4887443  0.6090658
## [4,] -0.27341715  0.4139112 -0.4819447 -0.7741562  1.5972052
## [5,] -0.50276675 -0.9001823  1.4148422  0.9907354  1.0623354
```

4. Using the "iris" data set, store the Species variable as a factor variable named "iris species" where "versicolor" is the first level, "virginica" is the second, and "setosa" is the third.

```
iris_species = factor(iris$Species, levels = c("versicolor", "virginica", "setosa"))
iris_species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
```

```
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: versicolor virginica setosa
```

5. Create a sequence of numbers from 1.2 to 5.3 by 0.05 called "first seq".

```
first_seq = seq(from = 1.2, to = 5.3, by = 0.05)
first_seq
```

```
##  [1] 1.20 1.25 1.30 1.35 1.40 1.45 1.50 1.55 1.60 1.65 1.70 1.75 1.80 1.85 1.90
## [16] 1.95 2.00 2.05 2.10 2.15 2.20 2.25 2.30 2.35 2.40 2.45 2.50 2.55 2.60 2.65
## [31] 2.70 2.75 2.80 2.85 2.90 2.95 3.00 3.05 3.10 3.15 3.20 3.25 3.30 3.35 3.40
## [46] 3.45 3.50 3.55 3.60 3.65 3.70 3.75 3.80 3.85 3.90 3.95 4.00 4.05 4.10 4.15
## [61] 4.20 4.25 4.30 4.35 4.40 4.45 4.50 4.55 4.60 4.65 4.70 4.75 4.80 4.85 4.90
## [76] 4.95 5.00 5.05 5.10 5.15 5.20 5.25 5.30
```

6. Create a sequence of numbers from 1.2 to 5.3 that is 100 elements long called "second seq".

```
second_seq = seq(from = 1.2, to = 5.3, length.out = 100)
second_seq
```

```
##   [1] 1.200000 1.241414 1.282828 1.324242 1.365657 1.407071 1.448485 1.489899
##   [9] 1.531313 1.572727 1.614141 1.655556 1.696970 1.738384 1.779798 1.821212
##  [17] 1.862626 1.904040 1.945455 1.986869 2.028283 2.069697 2.111111 2.152525
##  [25] 2.193939 2.235354 2.276768 2.318182 2.359596 2.401010 2.442424 2.483838
##  [33] 2.525253 2.566667 2.608081 2.649495 2.690909 2.732323 2.773737 2.815152
##  [41] 2.856566 2.897980 2.939394 2.980808 3.022222 3.063636 3.105051 3.146465
##  [49] 3.187879 3.229293 3.270707 3.312121 3.353535 3.394949 3.436364 3.477778
##  [57] 3.519192 3.560606 3.602020 3.643434 3.684848 3.726263 3.767677 3.809091
##  [65] 3.850505 3.891919 3.933333 3.974747 4.016162 4.057576 4.098990 4.140404
##  [73] 4.181818 4.223232 4.264646 4.306061 4.347475 4.388889 4.430303 4.471717
##  [81] 4.513131 4.554545 4.595960 4.637374 4.678788 4.720202 4.761616 4.803030
##  [89] 4.844444 4.885859 4.927273 4.968687 5.010101 5.051515 5.092929 5.134343
##  [97] 5.175758 5.217172 5.258586 5.300000
```

7. Load the AER library and attach the GSOEP9402 data set (data("GSOEP9402")). What are the dimensions of the data set? What are the names of the variables? Rename the variables to the same name, but all in uppercase rather than lowercase.

```
library(AER)
```

```
## Loading required package: car
```

```
## Loading required package: carData

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: survival
```

```
data("GSOEP9402")
dim(GSOEP9402)
```

```
## [1] 675  12
```

```
toupper(names(GSOEP9402))
```

```
##  [1] "SCHOOL"      "BIRTHYEAR"   "GENDER"      "KIDS"        "PARITY"
##  [6] "INCOME"      "SIZE"        "STATE"       "MARITAL"     "MEDUCATION"
## [11] "MEMPLOYMENT" "YEAR"
```

8. Create the following vectors: • "person id" which is a numeric vector from 1 to 50 that increases by 1. • "time" which is a numeric vector from 2001 to 2020 that increases by 1.

```
person_id = seq(from = 1, to = 50, by = 1)
person_id
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```
time = seq(from = 2001, to = 2020, by = 1)
time
```

```
##  [1] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015
## [16] 2016 2017 2018 2019 2020
```

9. Finally, create a data.frame called "panel data" that have the following variables: • "id" where each element of "person id" is repeated 20 times. • "time" where the time vector is repeated 50 times. • "draw" that is the vector "norm draws". • Note: this data set is pointless; this is just to get you familiar with creating data.frames

```
time = rep(time, times=50)
id = rep(person_id, times  = 20)
panel_data = data.frame(id, time, norm_draws)
```

Indexing

1. Store a vector of the id's of the elements in"draws NaN" that are NaN called "NaN ids".

```
nan_Ids = which(draws_NaN == T)
```

2. Store a vector of the id's of the elements in "norm draws" that are positive called "pos ids".

```
pos_ids = which(norm_draws > 0)
```

3. Store the elements with even indices from "norm draws" called "even id draws".

```
even_id_draws = which(norm_draws %% 2 == 0)
```

4. Extract the value from the 20th observation of the "mpg" variable in the "mtcars" data set using three different ways we discussed in class.

```
mtcars$mpg[20]
```

```
## [1] 33.9
```

```
mtcars[[1]][20]
```

```
## [1] 33.9
```

```
mtcars[["mpg"]][20]
```

```
## [1] 33.9
```

5. Extract the value from the 17th observation of the "cyl" and "hp" variables in the "mtcars" data set.

```
c(mtcars$cyl[17], mtcars$hp[17])
```

```
## [1]   8 230
```

6. (Harder) Extract all observations from "mtcars" that have 8 cylinders (cyl) using logical expressions/indexing.

```
subset(mtcars, mtcars$cyl == 8)
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    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
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## 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
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

7. Load the "Titanic" data set. This data set documents the survivors and victims of The Titanic. Use the "str()" function to determine the dimensions and structure of the data set. Print the outcomes for the following subgroups of passengers:

(a) The outcomes of the female, adult, crew.
(b) The outcomes for the male, first-class, adults.
(c) The outcomes of the female, first-class, children.
(d) The outcomes of the male, second-class, adults.

```
str(Titanic)
```

```
##  'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
##  - attr(*, "dimnames")=List of 4
##   ..$ Class   : chr [1:4] "1st" "2nd" "3rd" "Crew"
##   ..$ Sex     : chr [1:2] "Male" "Female"
##   ..$ Age     : chr [1:2] "Child" "Adult"
##   ..$ Survived: chr [1:2] "No" "Yes"
```

```
Titanic = data.frame(Titanic)
partA = Titanic[Titanic$Sex== "Female" & Titanic$Age== "Adult" & Titanic$Class =="Crew",]
partB = Titanic[Titanic$Sex== "Male" & Titanic$Class =="1st" & Titanic$Age== "Adult",]
partC = Titanic[Titanic$Sex== "Female" & Titanic$Class =="1st" & Titanic$Age== "Child",]
partD = Titanic[Titanic$Sex== "Male" & Titanic$Class =="2nd" & Titanic$Age== "Adult",]
```

8. Using "mtcars list", extract the "wt" as an atomic vector using two different ways of indexing a list discussed in class.

```
mtcars[[mtcars_list[[11]]]]
```

```
##  [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440 4.070
## [13] 3.730 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520 3.435 3.840
## [25] 3.845 1.935 2.140 1.513 3.170 2.770 3.570 2.780
```

9. Using "mtcars list", extract the "wt" variable as a list.

```
mtcars[mtcars_list[[11]]]
```

```
##                          wt
## Mazda RX4             2.620
## Mazda RX4 Wag         2.875
## Datsun 710            2.320
## Hornet 4 Drive        3.215
## Hornet Sportabout     3.440
## Valiant               3.460
## Duster 360            3.570
## Merc 240D             3.190
## Merc 230              3.150
## Merc 280              3.440
## Merc 280C             3.440
## Merc 450SE            4.070
## Merc 450SL            3.730
## Merc 450SLC           3.780
## Cadillac Fleetwood    5.250
## Lincoln Continental   5.424
## Chrysler Imperial     5.345
## Fiat 128              2.200
## Honda Civic           1.615
## Toyota Corolla        1.835
## Toyota Corona         2.465
## Dodge Challenger      3.520
## AMC Javelin           3.435
## Camaro Z28            3.840
## Pontiac Firebird      3.845
## Fiat X1-9             1.935
## Porsche 914-2         2.140
## Lotus Europa          1.513
## Ford Pantera L        3.170
## Ferrari Dino          2.770
## Maserati Bora         3.570
## Volvo 142E            2.780
```