Lab 2

Jonathan Eng

11:59PM February 17, 2020

More Basic R Skills

- Calculate the average of 1000 realizations of Bernoullis with p = 0.9 in one line using `rbinom`.

```
mean(rbinom(1000, size = 1, prob = 0.9))
```

```
## [1] 0.901
```

- In class we considered a variable `x_3` which measured "criminality". We imagined L = 4 levels "none", "infraction", "misdimeanor" and "felony". Create a variable `x3` here with 100 random elements (equally probable). Create it as a nominal (i.e. unordered) factor.

```
x3 = factor(sample(c("none", "infraction", "misdimeanor", "felony"), size = 100, replace = TRUE))
x3
```

```
##   [1] none        felony      infraction  misdimeanor misdimeanor none
##   [7] none        misdimeanor none        felony      misdimeanor none
##  [13] misdimeanor none        misdimeanor infraction  felony      infraction
##  [19] none        none        none        infraction  none        infraction
##  [25] none        misdimeanor misdimeanor misdimeanor misdimeanor felony
##  [31] infraction  felony      felony      misdimeanor felony      felony
##  [37] none        infraction  misdimeanor misdimeanor none        infraction
##  [43] misdimeanor infraction  felony      misdimeanor none        felony
##  [49] misdimeanor misdimeanor infraction  none        felony      misdimeanor
##  [55] none        none        infraction  misdimeanor none        misdimeanor
##  [61] felony      felony      none        none        misdimeanor none
##  [67] felony      infraction  infraction  none        misdimeanor felony
##  [73] infraction  none        misdimeanor felony      infraction  misdimeanor
##  [79] felony      none        misdimeanor infraction  infraction  none
##  [85] misdimeanor none        misdimeanor none        misdimeanor none
##  [91] felony      infraction  misdimeanor misdimeanor felony      misdimeanor
##  [97] misdimeanor misdimeanor none        misdimeanor
## Levels: felony infraction misdimeanor none
```

- Convert this variable into three binary variables without any information loss and put them into a data matrix.

```r
X = matrix(NA, nrow = length(x3), ncol = 3)
X[ ,1] = as.numeric(x3 == "infraction")
X[ ,2] = as.numeric(x3 == "misdimeanor")
X[ ,3] = as.numeric(x3 == "felony")
colnames(X) = c("is_infraction", "is_misdimeanor", "is_felony")
X
```

```
##       is_infraction is_misdimeanor is_felony
##  [1,]             0              0         0
##  [2,]             0              0         1
##  [3,]             1              0         0
##  [4,]             0              1         0
##  [5,]             0              1         0
##  [6,]             0              0         0
##  [7,]             0              0         0
##  [8,]             0              1         0
##  [9,]             0              0         0
## [10,]             0              0         1
## [11,]             0              1         0
## [12,]             0              0         0
## [13,]             0              1         0
## [14,]             0              0         0
## [15,]             0              1         0
## [16,]             1              0         0
## [17,]             0              0         1
## [18,]             1              0         0
## [19,]             0              0         0
## [20,]             0              0         0
## [21,]             0              0         0
## [22,]             1              0         0
## [23,]             0              0         0
## [24,]             1              0         0
## [25,]             0              0         0
## [26,]             0              1         0
## [27,]             0              1         0
## [28,]             0              1         0
## [29,]             0              1         0
## [30,]             0              0         1
## [31,]             1              0         0
## [32,]             0              0         1
## [33,]             0              0         1
## [34,]             0              1         0
## [35,]             0              0         1
## [36,]             0              0         1
## [37,]             0              0         0
## [38,]             1              0         0
## [39,]             0              1         0
## [40,]             0              1         0
## [41,]             0              0         0
## [42,]             1              0         0
## [43,]             0              1         0
## [44,]             1              0         0
## [45,]             0              0         1
```

```
##  [46,]            0             1         0
##  [47,]            0             0         0
##  [48,]            0             0         1
##  [49,]            0             1         0
##  [50,]            0             1         0
##  [51,]            1             0         0
##  [52,]            0             0         0
##  [53,]            0             0         1
##  [54,]            0             1         0
##  [55,]            0             0         0
##  [56,]            0             0         0
##  [57,]            1             0         0
##  [58,]            0             1         0
##  [59,]            0             0         0
##  [60,]            0             1         0
##  [61,]            0             0         1
##  [62,]            0             0         1
##  [63,]            0             0         0
##  [64,]            0             0         0
##  [65,]            0             1         0
##  [66,]            0             0         0
##  [67,]            0             0         1
##  [68,]            1             0         0
##  [69,]            1             0         0
##  [70,]            0             0         0
##  [71,]            0             1         0
##  [72,]            0             0         1
##  [73,]            1             0         0
##  [74,]            0             0         0
##  [75,]            0             1         0
##  [76,]            0             0         1
##  [77,]            1             0         0
##  [78,]            0             1         0
##  [79,]            0             0         1
##  [80,]            0             0         0
##  [81,]            0             1         0
##  [82,]            1             0         0
##  [83,]            1             0         0
##  [84,]            0             0         0
##  [85,]            0             1         0
##  [86,]            0             0         0
##  [87,]            0             1         0
##  [88,]            0             0         0
##  [89,]            0             1         0
##  [90,]            0             0         0
##  [91,]            0             0         1
##  [92,]            1             0         0
##  [93,]            0             1         0
##  [94,]            0             1         0
##  [95,]            0             0         1
##  [96,]            0             1         0
##  [97,]            0             1         0
##  [98,]            0             1         0
##  [99,]            0             0         0
```

```
## [100,]             0              1         0
```

```
table(X)
```

```
## X
##   0   1
## 229  71
```

- What should the sum of each row be (in English)?

It should be either 1 or 0 because categories are mutually exclusive. We can only put an object into 1 category. None = 0

Verify that.

```
rowSums(X)
```

```
##   [1] 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0
##  [38] 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 0
##  [75] 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1
```

```
table(rowSums(X))
```

```
##
##  0  1
## 29 71
```

- How should the column sum look (in English)?

The sums should be around the expectation, 25 since they are uniformly distributed.

Verify that.

```
colSums(X)
```

```
##  is_infraction is_misdimeanor      is_felony
##             18             34             19
```

- Generate a matrix with 100 rows where the first column is realization from a normal with mean 17 and variance 38, the second column is uniform between -10 and 10, the third column is poisson with mean 6, the fourth column in exponential with lambda of 9, the fifth column is binomial with n = 20 and p = 0.12 and the sixth column is a binary variable with exactly 24% 1's dispersed randomly. Name the rows the entries of the `fake_first_names` vector.

```
n = 100
X = matrix(data = NA, nrow = n, ncol = 6)
X[, 1] = rnorm(n, mean = 17, sd = sqrt(38))
X[, 2] = runif(n, min = -10, max = 10)
X[, 3] = rpois(n, lambda = 6)
X[, 4] = rexp(n, rate = 9)
X[, 5] = rbinom(n, size = 20, prob = 0.12)
```

```r
X[, 6] = sample(c(rep(1, n*.24), rep(0, n*.76)))

fake_first_names = c(
  "Sophia", "Emma", "Olivia", "Ava", "Mia", "Isabella", "Riley",
  "Aria", "Zoe", "Charlotte", "Lily", "Layla", "Amelia", "Emily",
  "Madelyn", "Aubrey", "Adalyn", "Madison", "Chloe", "Harper",
  "Abigail", "Aaliyah", "Avery", "Evelyn", "Kaylee", "Ella", "Ellie",
  "Scarlett", "Arianna", "Hailey", "Nora", "Addison", "Brooklyn",
  "Hannah", "Mila", "Leah", "Elizabeth", "Sarah", "Eliana", "Mackenzie",
  "Peyton", "Maria", "Grace", "Adeline", "Elena", "Anna", "Victoria",
  "Camilla", "Lillian", "Natalie", "Jackson", "Aiden", "Lucas",
  "Liam", "Noah", "Ethan", "Mason", "Caden", "Oliver", "Elijah",
  "Grayson", "Jacob", "Michael", "Benjamin", "Carter", "James",
  "Jayden", "Logan", "Alexander", "Caleb", "Ryan", "Luke", "Daniel",
  "Jack", "William", "Owen", "Gabriel", "Matthew", "Connor", "Jayce",
  "Isaac", "Sebastian", "Henry", "Muhammad", "Cameron", "Wyatt",
  "Dylan", "Nathan", "Nicholas", "Julian", "Eli", "Levi", "Isaiah",
  "Landon", "David", "Christian", "Andrew", "Brayden", "John",
  "Lincoln"
)

rownames(X) = fake_first_names
X
```

```
##                [,1]        [,2] [,3]        [,4] [,5] [,6]
## Sophia    12.6198898  3.33807969    4 0.004147971    2    0
## Emma       8.3204571  1.65488165    3 0.231898635    4    0
## Olivia    24.1362267 -1.40626491    2 0.233821257    1    0
## Ava       24.6236958 -9.33456775    6 0.072828534    1    0
## Mia       17.4678187 -8.42510014   10 0.211463059    0    1
## Isabella  15.0989400 -6.71611276    4 0.003827193    2    1
## Riley      8.2988683 -2.10627119    6 0.196077944    2    0
## Aria      12.5519296 -6.40582307    4 0.107769850    1    1
## Zoe       26.4110883  9.16850392    4 0.065913104    1    0
## Charlotte 18.4521697 -1.39021491    7 0.010109410    2    1
## Lily      12.2584943 -0.19535645   10 0.073187011    2    0
## Layla     13.6590063 -8.88048423    2 0.028233604    5    0
## Amelia    22.6957221 -7.67019673    9 0.064156449    5    0
## Emily     16.6778440 -2.61793616    2 0.081418942    8    1
## Madelyn    5.7294241 -9.64480498    7 0.109297853    0    0
## Aubrey    19.2433760  4.37876909    3 0.332457629    5    0
## Adalyn    17.9284699  9.06825947    4 0.162998871    4    1
## Madison   13.2748475  3.93944626    6 0.217723278    5    0
## Chloe     13.7900619 -3.42930078    3 0.061830994    2    0
## Harper     4.8444751  1.11765497    4 0.039915251    4    0
## Abigail   15.6858615 -4.49187728    6 0.530442326    3    0
## Aaliyah   25.8635693  8.73346201    5 0.050347990    2    0
## Avery     12.4616780  1.14315305    0 0.098389766    3    0
## Evelyn     9.2754733 -2.77486437    5 0.133505153    2    0
## Kaylee    21.7092525  0.48019180    4 0.030592007    4    0
## Ella      20.4163041  5.85581660    5 0.193350737    2    0
## Ellie     14.9209229 -9.09773126    1 0.223502892    2    0
## Scarlett  28.0264400 -2.68731751    3 0.008848331    1    0
```

```
## Arianna   13.7437274  5.44837869    6 0.102990504    1    0
## Hailey     8.6357522  6.98788019    7 0.049492591    4    1
## Nora      17.0692408  6.79845555    6 0.081219923    3    0
## Addison   16.7266463  8.79661008   12 0.121181184    3    0
## Brooklyn  25.2712950 -3.49937209    7 0.284601166    2    0
## Hannah    15.8156532  9.21697626    6 0.035705814    2    0
## Mila       8.7090413 -9.31433444    6 0.015557468    2    0
## Leah      15.6282996 -4.10079385    7 0.250359542    2    0
## Elizabeth  7.7698869  2.79189369    6 0.098167516    3    1
## Sarah     17.0050946 -5.64470716    4 0.195831198    5    0
## Eliana    25.2786666  8.01027023    9 0.044273333    0    0
## Mackenzie 10.0235305 -0.41153574    7 0.003689705    1    1
## Peyton    25.6545410 -1.08227371    6 0.063042616    3    0
## Maria     17.2396485  2.53127669    7 0.076688015    2    1
## Grace     -0.5301344  2.73722549   10 0.111728667    3    1
## Adeline   17.4607190 -8.44667470    5 0.030662186    3    1
## Elena      5.7881638 -4.24080792    6 0.083443270    5    0
## Anna      17.9236221  8.19283561    9 0.147782636    1    0
## Victoria  18.1152863  5.98723963    3 0.067258103    3    0
## Camilla   22.3441339 -3.43083464    8 0.139434021    1    0
## Lillian   16.6723708 -8.27925240    5 0.107807873    2    0
## Natalie   16.9938404  2.55752409    8 0.107622417    0    0
## Jackson   23.4432044 -0.12962677    3 0.014522357    2    1
## Aiden     20.5351581  6.65939393   12 0.015701929    4    0
## Lucas      6.8709441 -9.04275955    7 0.153423010    4    0
## Liam      14.1673267 -0.01780928    3 0.183312448    2    1
## Noah      19.1780926 -3.96080763    6 0.152329010    1    0
## Ethan     12.9209659 -8.05692539    8 0.068715855    0    1
## Mason     15.6981996  3.23295836    9 0.204145948    2    0
## Caden     30.3378697 -2.86549782    4 0.092245323    1    1
## Oliver    14.9490192 -7.61210035    7 0.032592093    1    0
## Elijah    15.8720157  6.10967502    2 0.037572635    2    0
## Grayson   22.0405622  3.79823000    3 0.007380559    3    1
## Jacob     28.5899741 -8.73391831    0 0.055454469    3    0
## Michael   16.4919577  6.63837728    2 0.058367542    3    0
## Benjamin  15.0421128 -6.65545403    3 0.142175645    3    0
## Carter    13.8716194 -1.74341552    3 0.106156431    3    0
## James     13.6322073 -8.68058989    6 0.261191199    3    0
## Jayden    23.8332144  0.22326368    2 0.624920419    2    0
## Logan      9.8782238  8.19142214    5 0.467160884    3    1
## Alexander 21.9117593  3.66419726    8 0.253279569    4    0
## Caleb     21.5600654  5.12028046    5 0.182779074    3    0
## Ryan      21.1267445 -6.14355294    3 0.027747185    3    0
## Luke      19.5549598 -8.76893362    2 0.007472507    1    1
## Daniel    25.8140471  5.74710538    7 0.084306185    4    0
## Jack      17.3221608  4.64997228    6 0.062834128    3    0
## William   12.5559685 -5.44672585   12 0.040982048    1    0
## Owen       1.7059005 -8.06788256    3 0.050229113    1    0
## Gabriel    9.2584450 -4.74206575    4 0.611196490    0    0
## Matthew   14.1473044  4.19284758    7 0.005043386    4    0
## Connor    16.5420033  5.49064781    3 0.041072218    1    0
## Jayce     23.9094189  4.56395020    3 0.100406040    2    0
## Isaac     24.0448556 -4.15192417    8 0.076746330    4    1
## Sebastian 12.8405944 -9.93557224    7 0.133918099    3    0
```

```
## Henry     19.5951801  5.37063703    9 0.086043652    3    0
## Muhammad  11.9078157 -8.06086427    9 0.479115667    1    0
## Cameron   17.7889245 -3.82911862    6 0.103952623    1    0
## Wyatt     32.2859636 -0.41417661   11 0.211787191    0    0
## Dylan     23.0865204  5.41503180   12 0.050517288    1    0
## Nathan     7.4708836 -9.42303729    6 0.105756958    6    0
## Nicholas  22.8640199 -5.95178029    4 0.027224781    1    0
## Julian    19.7184335 -5.75448680   11 0.054664526    1    0
## Eli       23.9596870  8.59343703    7 0.046388076    2    0
## Levi      26.6216149  8.12380432    8 0.003874912    1    0
## Isaiah    17.7580299 -2.68928283    9 0.079709890    1    0
## Landon    12.3505223  3.61187322    9 0.226247864    5    0
## David      7.1210734 -9.39920303    3 0.129695414    4    1
## Christian 12.0973044  8.62928355    2 0.145137158    2    0
## Andrew    16.1634158  0.96245640   14 0.001550903    5    0
## Brayden   20.3326724 -9.65595608    4 0.030071815    4    1
## John       8.2246481 -6.77925705    9 0.288935419    1    1
## Lincoln    6.6280507  2.96715299    7 0.122183234    0    1
```

- Create a data frame of the same data as above except make the binary variable a factor "DOMESTIC" vs "FOREIGN" for 0 and 1 respectively. Print out the top few rows to check if this worked correctly.

```
df = data.frame(X)
df$X6 = factor(df$X6, levels = c(0, 1), labels = c("DOMESTIC", "FOREIGN"))
df
```

```
##                   X1          X2 X3          X4 X5       X6
## Sophia    12.6198898  3.33807969  4 0.004147971  2 DOMESTIC
## Emma       8.3204571  1.65488165  3 0.231898635  4 DOMESTIC
## Olivia    24.1362267 -1.40626491  2 0.233821257  1 DOMESTIC
## Ava       24.6236958 -9.33456775  6 0.072828534  1 DOMESTIC
## Mia       17.4678187 -8.42510014 10 0.211463059  0  FOREIGN
## Isabella  15.0989400 -6.71611276  4 0.003827193  2  FOREIGN
## Riley      8.2988683 -2.10627119  6 0.196077944  2 DOMESTIC
## Aria      12.5519296 -6.40582307  4 0.107769850  1  FOREIGN
## Zoe       26.4110883  9.16850392  4 0.065913104  1 DOMESTIC
## Charlotte 18.4521697 -1.39021491  7 0.010109410  2  FOREIGN
## Lily      12.2584943 -0.19535645 10 0.073187011  2 DOMESTIC
## Layla     13.6590063 -8.88048423  2 0.028233604  5 DOMESTIC
## Amelia    22.6957221 -7.67019673  9 0.064156449  5 DOMESTIC
## Emily     16.6778440 -2.61793616  2 0.081418942  8  FOREIGN
## Madelyn    5.7294241 -9.64480498  7 0.109297853  0 DOMESTIC
## Aubrey    19.2433760  4.37876909  3 0.332457629  5 DOMESTIC
## Adalyn    17.9284699  9.06825947  4 0.162998871  4  FOREIGN
## Madison   13.2748475  3.93944626  6 0.217723278  5 DOMESTIC
## Chloe     13.7900619 -3.42930078  3 0.061830994  2 DOMESTIC
## Harper     4.8444751  1.11765497  4 0.039915251  4 DOMESTIC
## Abigail   15.6858615 -4.49187728  6 0.530442326  3 DOMESTIC
## Aaliyah   25.8635693  8.73346201  5 0.050347990  2 DOMESTIC
## Avery     12.4616780  1.14315305  0 0.098389766  3 DOMESTIC
## Evelyn     9.2754733 -2.77486437  5 0.133505153  2 DOMESTIC
## Kaylee    21.7092525  0.48019180  4 0.030592007  4 DOMESTIC
## Ella      20.4163041  5.85581660  5 0.193350737  2 DOMESTIC
```

```
## Ellie     14.9209229 -9.09773126  1 0.223502892  2 DOMESTIC
## Scarlett  28.0264400 -2.68731751  3 0.008848331  1 DOMESTIC
## Arianna   13.7437274  5.44837869  6 0.102990504  1 DOMESTIC
## Hailey     8.6357522  6.98788019  7 0.049492591  4  FOREIGN
## Nora      17.0692408  6.79845555  6 0.081219923  3 DOMESTIC
## Addison   16.7266463  8.79661008 12 0.121181184  3 DOMESTIC
## Brooklyn  25.2712950 -3.49937209  7 0.284601166  2 DOMESTIC
## Hannah    15.8156532  9.21697626  6 0.035705814  2 DOMESTIC
## Mila       8.7090413 -9.31433444  6 0.015557468  2 DOMESTIC
## Leah      15.6282996 -4.10079385  7 0.250359542  2 DOMESTIC
## Elizabeth  7.7698869  2.79189369  6 0.098167516  3  FOREIGN
## Sarah     17.0050946 -5.64470716  4 0.195831198  5 DOMESTIC
## Eliana    25.2786666  8.01027023  9 0.044273333  0 DOMESTIC
## Mackenzie 10.0235305 -0.41153574  7 0.003689705  1  FOREIGN
## Peyton    25.6545410 -1.08227371  6 0.063042616  3 DOMESTIC
## Maria     17.2396485  2.53127669  7 0.076688015  2  FOREIGN
## Grace     -0.5301344  2.73722549 10 0.111728667  3  FOREIGN
## Adeline   17.4607190 -8.44667470  5 0.030662186  3  FOREIGN
## Elena      5.7881638 -4.24080792  6 0.083443270  5 DOMESTIC
## Anna      17.9236221  8.19283561  9 0.147782636  1 DOMESTIC
## Victoria  18.1152863  5.98723963  3 0.067258103  3 DOMESTIC
## Camilla   22.3441339 -3.43083464  8 0.139434021  1 DOMESTIC
## Lillian   16.6723708 -8.27925240  5 0.107807873  2 DOMESTIC
## Natalie   16.9938404  2.55752409  8 0.107622417  0 DOMESTIC
## Jackson   23.4432044 -0.12962677  3 0.014522357  2  FOREIGN
## Aiden     20.5351581  6.65939393 12 0.015701929  4 DOMESTIC
## Lucas      6.8709441 -9.04275955  7 0.153423010  4 DOMESTIC
## Liam      14.1673267 -0.01780928  3 0.183312448  2  FOREIGN
## Noah      19.1780926 -3.96080763  6 0.152329010  1 DOMESTIC
## Ethan     12.9209659 -8.05692539  8 0.068715855  0  FOREIGN
## Mason     15.6981996  3.23295836  9 0.204145948  2 DOMESTIC
## Caden     30.3378697 -2.86549782  4 0.092245323  1  FOREIGN
## Oliver    14.9490192 -7.61210035  7 0.032592093  1 DOMESTIC
## Elijah    15.8720157  6.10967502  2 0.037572635  2 DOMESTIC
## Grayson   22.0405622  3.79823000  3 0.007380559  3  FOREIGN
## Jacob     28.5899741 -8.73391831  0 0.055454469  3 DOMESTIC
## Michael   16.4919577  6.63837728  2 0.058367542  3 DOMESTIC
## Benjamin  15.0421128 -6.65545403  3 0.142175645  3 DOMESTIC
## Carter    13.8716194 -1.74341552  3 0.106156431  3 DOMESTIC
## James     13.6322073 -8.68058989  6 0.261191199  3 DOMESTIC
## Jayden    23.8332144  0.22326368  2 0.624920419  2 DOMESTIC
## Logan      9.8782238  8.19142214  5 0.467160884  3  FOREIGN
## Alexander 21.9117593  3.66419726  8 0.253279569  4 DOMESTIC
## Caleb     21.5600654  5.12028046  5 0.182779074  3 DOMESTIC
## Ryan      21.1267445 -6.14355294  3 0.027747185  3 DOMESTIC
## Luke      19.5549598 -8.76893362  2 0.007472507  1  FOREIGN
## Daniel    25.8140471  5.74710538  7 0.084306185  4 DOMESTIC
## Jack      17.3221608  4.64997228  6 0.062834128  3 DOMESTIC
## William   12.5559685 -5.44672585 12 0.040982048  1 DOMESTIC
## Owen       1.7059005 -8.06788256  3 0.050229113  1 DOMESTIC
## Gabriel    9.2584450 -4.74206575  4 0.611196490  0 DOMESTIC
## Matthew   14.1473044  4.19284758  7 0.005043386  4 DOMESTIC
## Connor    16.5420033  5.49064781  3 0.041072218  1 DOMESTIC
## Jayce     23.9094189  4.56395020  3 0.100406040  2 DOMESTIC
```

```
## Isaac     24.0448556 -4.15192417  8 0.076746330  4  FOREIGN
## Sebastian 12.8405944 -9.93557224  7 0.133918099  3 DOMESTIC
## Henry     19.5951801  5.37063703  9 0.086043652  3 DOMESTIC
## Muhammad  11.9078157 -8.06086427  9 0.479115667  1 DOMESTIC
## Cameron   17.7889245 -3.82911862  6 0.103952623  1 DOMESTIC
## Wyatt     32.2859636 -0.41417661 11 0.211787191  0 DOMESTIC
## Dylan     23.0865204  5.41503180 12 0.050517288  1 DOMESTIC
## Nathan     7.4708836 -9.42303729  6 0.105756958  6 DOMESTIC
## Nicholas  22.8640199 -5.95178029  4 0.027224781  1 DOMESTIC
## Julian    19.7184335 -5.75448680 11 0.054664526  1 DOMESTIC
## Eli       23.9596870  8.59343703  7 0.046388076  2 DOMESTIC
## Levi      26.6216149  8.12380432  8 0.003874912  1 DOMESTIC
## Isaiah    17.7580299 -2.68928283  9 0.079709890  1 DOMESTIC
## Landon    12.3505223  3.61187322  9 0.226247864  5 DOMESTIC
## David      7.1210734 -9.39920303  3 0.129695414  4  FOREIGN
## Christian 12.0973044  8.62928355  2 0.145137158  2 DOMESTIC
## Andrew    16.1634158  0.96245640 14 0.001550903  5 DOMESTIC
## Brayden   20.3326724 -9.65595608  4 0.030071815  4  FOREIGN
## John       8.2246481 -6.77925705  9 0.288935419  1  FOREIGN
## Lincoln    6.6280507  2.96715299  7 0.122183234  0  FOREIGN
```

- Print out a table of the binary variable. Then print out the proportions of "DOMESTIC" vs "FOREIGN".

```
table(df$X6)
```

```
##
## DOMESTIC  FOREIGN
##       76       24
```

```
table(df$X6)/n
```

```
##
## DOMESTIC  FOREIGN
##     0.76     0.24
```

Print out a summary of the whole dataframe.

```
summary(df)
```

```
##        X1                X2                X3              X4
##  Min.   :-0.5301   Min.   :-9.9356   Min.   : 0.00   Min.   :0.001551
##  1st Qu.:12.5550   1st Qu.:-6.2091   1st Qu.: 3.00   1st Qu.:0.043473
##  Median :16.6751   Median :-0.7482   Median : 6.00   Median :0.085175
##  Mean   :16.6140   Mean   :-0.7155   Mean   : 5.82   Mean   :0.123448
##  3rd Qu.:21.5974   3rd Qu.: 4.5855   3rd Qu.: 7.25   3rd Qu.:0.155817
##  Max.   :32.2860   Max.   : 9.2170   Max.   :14.00   Max.   :0.624920
##        X5             X6
##  Min.   :0.00   DOMESTIC:76
##  1st Qu.:1.00   FOREIGN :24
##  Median :2.00
##  Mean   :2.41
##  3rd Qu.:3.00
##  Max.   :8.00
```

- Let `n = 50`. Create a n x n matrix `R` of exactly 50% entries 0's, 25% 1's 25% 2's. These values should be in random locations.

```
n = 50
R = matrix(sample(c(rep(0, (n^2)*.5), rep(1, (n^2)*.25), rep(2, (n^2)*.25))), nrow = n, ncol = n)
```

- Randomly punch holes (i.e. `NA`) values in this matrix so that an each entry is missing with probability 30%.

```
for(i in 1 : n){
  for(j in 1 : n) {
    if(runif(1) <= 0.3) {
      R[i, j] = NA
    }
  }
}

#Checker (DO NOT RUN THE CODE ABOVE TWICE):
sum(is.na(R)) / (n^2)
```

```
## [1] 0.3032
```

- Sort the rows in matrix `R` by the largest row sum to lowest. Be careful about the NA's!

```
R = R[order(rowSums(R, na.rm = TRUE), decreasing = TRUE), ]

#Test
rowSums(R, na.rm = TRUE)
```

```
##  [1] 39 39 37 33 32 32 32 31 30 30 30 30 30 29 29 29 29 28 28 28 28 26 26 26 26
## [26] 25 25 25 25 24 24 24 24 24 23 23 22 22 22 20 20 20 20 19 18 17 17 14 14 13
```

- We will now learn the `apply` function. This is a handy function that saves writing for loops which should be eschewed in R. Use the apply function to compute a vector whose entries are the standard deviation of each row. Use the apply function to compute a vector whose entries are the standard deviation of each column. Be careful about the NA's! This should be one line.

```
apply(R, 1, sd, na.rm = TRUE)
```

```
##  [1] 0.8583951 0.8849139 0.8819171 0.7816493 0.8219949 0.8868791 0.8530716
##  [8] 0.8337837 0.8427498 0.9051771 0.7766141 0.8086075 0.8451543 0.8560741
## [15] 0.8886408 0.7470874 0.9230931 0.7513429 0.8568215 0.8331372 0.8693637
## [22] 0.7355445 0.8859311 0.8454080 0.7807853 0.7924798 0.8378085 0.8700899
## [29] 0.8339078 0.8866831 0.8667529 0.9055188 0.8835413 0.8393721 0.8333333
## [36] 0.8629652 0.8121186 0.7929275 0.8206017 0.7830650 0.8327955 0.8327955
## [43] 0.7881701 0.7800022 0.7648178 0.7676245 0.7362496 0.8007053 0.6814454
## [50] 0.6825489
```

```r
apply(R, 2, sd, na.rm = TRUE)
```

```
##  [1] 0.8198289 0.7862913 0.9216628 0.7681732 0.8637067 0.9043306 0.8445195
##  [8] 0.9121035 0.8058923 0.8929437 0.6872130 0.7702450 0.8502873 0.8876254
## [15] 0.7702450 0.6333545 0.8480679 0.7017385 0.8995529 0.8138585 0.7675458
## [22] 0.8451543 0.8795559 0.8906612 0.7474705 0.8023076 0.7300911 0.8385856
## [29] 0.8352988 0.8607714 0.8075276 0.8603661 0.8792663 0.7941548 0.8121186
## [36] 0.7766141 0.6671400 0.8208513 0.8669413 0.9625026 0.8206017 0.8637067
## [43] 0.8929437 0.8219949 0.8926508 0.7909747 0.7947194 0.8497042 0.7510676
## [50] 0.8873002
```

- Use the `apply` function to compute a vector whose entries are the count of entries that are 1 or 2 in each column. This should be one line.

```r
apply(R, 2, function(c) sum(!is.na(c) & c > 0))
```

```
##  [1] 20 18 16 13 16 20 20 18 19 16 16 16 12 21 16 11 23 14 16 22 17 20 19 21 15
## [26] 16 15 13 18 20 15  9 19 12 15 22 14 13 14 16 15 16 18 22 21 20 20 19 17 20
```

- Use the `split` function to create a list whose keys are the column number and values are the vector of the columns. Look at the last example in the documentation `?split`.

```r
split(R, col(R, as.factor = TRUE))
```

```
## $`1`
##  [1]  2 NA  2  0  1  0  2 NA  2  0  0  0 NA  2  1  1  2  1  1 NA  0  0 NA NA  0
## [26]  0  1  1  0  0 NA  0 NA  1  2  2 NA  2  1  0  1 NA  1  0  0  0 NA  0  0 NA
##
## $`2`
##  [1] NA  1 NA  1  2  1  2  1  0  1  2  0 NA  0 NA  0  2 NA NA  2  2  0  2  1 NA
## [26]  1  0 NA  0 NA  1  1 NA NA  0 NA NA  0  0  1  0  0  0  0  1  0  0  0  0 NA
##
## $`3`
##  [1]  2 NA  2  2  2 NA NA  1  1 NA NA  0  1 NA  0  0  2  0  2  0  2 NA  1 NA  2
## [26] NA NA  0 NA NA  0  0 NA  2 NA  2 NA  0  0 NA  0 NA  0  0  1 NA NA  2  0  0
##
## $`4`
##  [1]  0  0 NA  0  0  0  1 NA  0 NA NA  0  0  0  2  2  2  0  1  1  0  0  0  1 NA
## [26]  2  1  0 NA NA  0  0  0 NA  0 NA  1  0 NA NA  1  0  2  2 NA  0  0  0 NA  0
##
## $`5`
##  [1]  0  2  1  0  0  2 NA  0  2  0  1  0  0 NA  2  1  0  0  2  0 NA  1  0  0  0
## [26] NA NA  0  0 NA  1 NA NA  0  2 NA  2  2  1 NA  0 NA NA  1 NA NA  0 NA NA  2
##
## $`6`
##  [1]  0  0  2  2 NA  0 NA  0  0  2  1  0 NA  0  2  2 NA  1  2  1  1  1  0  2  2
## [26]  0  0 NA  0 NA  2  2  2 NA  2  0  2  1  0 NA  0  0  0  1  0 NA NA NA  0  0
##
## $`7`
##  [1]  2 NA  2  0  2  0  0  0  2  0  1  1  0 NA  0  0  1 NA NA  1  2  1  2  0  0
## [26] NA  0  1  1  0 NA  0 NA  0 NA NA  2 NA  1  2  2  1  0 NA  1  0 NA NA  2  0
```

```
##
## $`8`
##  [1]  2  2  2  2  2 NA NA  2  1 NA  2  0 NA  2  1 NA  0  0 NA NA  0 NA  2  0  1
## [26]  0  0 NA NA NA NA NA  0  0  0  0  1  0  0  0  2  2  0  1 NA  2  1  0 NA  0
##
## $`9`
##  [1] NA  0  1  0  2 NA  1  2  1  2  0  0 NA  0  0 NA  2  1 NA  2 NA  1 NA  0  1
## [26]  2 NA  2  1  2 NA  0 NA NA NA  0  0  1 NA  0  1 NA  0  1 NA NA NA NA  1 NA
##
## $`10`
##  [1]  1  2  2  0  0  0 NA NA  1 NA  0 NA  2  2  0  2 NA NA NA  2  0  0  0  0 NA
## [26] NA  0 NA  2 NA  1  1 NA  0  1 NA  0 NA  0  1  2 NA  0  0  2  0  2 NA  0 NA
##
## $`11`
##  [1]  2  1  1  1  0 NA  2  0  0  0 NA  1  2  1 NA  1  0  0  2 NA NA NA  0  1 NA
## [26]  0  1  0  1  1  1  0 NA NA NA NA  0  0  0  0  0  0  0  0 NA  0  0  0  1  0
##
## $`12`
##  [1]  0  1 NA  1  0 NA  0  2  1 NA  1  0 NA  0  1  0 NA  2  2  2  0  1 NA  0 NA
## [26] NA  0  0  2 NA  0  2  0  0 NA NA NA  1  1 NA  0  1  0  0  0  0 NA NA  1  0
##
## $`13`
##  [1]  2 NA NA  0  1 NA NA  2 NA  0  1  0 NA  1 NA  0  0 NA  0 NA  1  0  2  2 NA
## [26]  0  2 NA NA NA  0  0  2 NA  2 NA  0  0 NA  0 NA  0 NA  0 NA  0  1  0 NA  0
##
## $`14`
##  [1]  2  2  2  0  2  2 NA  0 NA  2  1 NA NA  2 NA  1  2 NA  0  1 NA  1 NA  0  1
## [26]  2  0 NA NA  2  0  0  0  0 NA  1  0  0 NA  2 NA  1  1  0 NA NA  2  2 NA  0
##
## $`15`
##  [1] NA  0  0  1  1  1  0 NA NA NA  1  2  1  2 NA  0  0  1  2  0  0  1  1  0 NA
## [26] NA  0 NA NA NA NA  0 NA  0  0 NA  2  0  0  0  2 NA NA  2  1  1  0  0  0  0
##
## $`16`
##  [1]  2  2 NA  0 NA  2  1  0 NA  0 NA  1  0 NA NA  1  1 NA  0  0  0  0  0  0  0
## [26]  0  0 NA NA NA  0  0  0  0  0  0  1  1 NA  1  0  0  0  0  0  0  0  1 NA  0
##
## $`17`
##  [1]  1  1  1  0  1 NA  2 NA  2  0 NA  1  1  2  0  0 NA  0  1  0 NA  0  0  0 NA
## [26]  1  2  0 NA  1  2  2  0  2  0 NA  1 NA NA  2  2  2 NA NA NA  2  1  2  0  0
##
## $`18`
##  [1]  1  0  2  1  0  0  1 NA NA  0  0  0  1  0  2  1  2  1  0  0  0  0 NA NA  0
## [26]  1  2  0 NA  0  0 NA  1  0  0  1  0  0 NA NA NA NA  1 NA NA  0  0 NA NA NA
##
## $`19`
##  [1]  2 NA  2 NA  1 NA  2  1  0 NA  0  2  2 NA NA  0 NA NA  2  0  1 NA  0  1  0
## [26] NA NA  0  0  0 NA  0  2  1  2 NA  1  2  0  0 NA NA NA  0 NA  2 NA NA  0 NA
##
## $`20`
##  [1] NA  2  0 NA NA  2  2 NA  2  1  2 NA  1  0  2  0 NA  2  0  0  1  1  1  1 NA
## [26]  1  1  0  2 NA  0 NA NA  1  1  1  2  2 NA NA NA  0 NA NA  2  0 NA NA NA NA
##
## $`21`
```

```
##  [1]  0 NA  1  2  2 NA  0  0 NA  0 NA  0  1  1 NA  1  1  0  1 NA NA NA  2  0 NA
## [26] NA  0  0 NA NA  1  0  0  2  0  0 NA  2  1  1  2 NA  0  0  0  0  1 NA  1 NA
##
## $`22`
##  [1]  1  2  0  1  0 NA  0  2  1  0  0  2  1  0  0  2 NA  1 NA  2 NA  1  1 NA  0
## [26] NA NA  2  0  2 NA  2  0  2  0 NA NA  1  1 NA NA  0  2 NA  0  1 NA NA  0  0
##
## $`23`
##  [1]  2  2  2  1  2  0  1 NA NA  2 NA  0  0 NA NA  0  0  1 NA  2 NA  0  2  2  1
## [26] NA NA NA  1 NA  0 NA  0  0  1  2 NA  0  0  2  0  0 NA  2 NA  0 NA  0  1  1
##
## $`24`
##  [1]  0  2 NA  1  0  2  2  2  0  2  2 NA NA NA  2 NA NA NA  0 NA  0  2 NA  1  1
## [26]  2  2  0 NA NA NA  1  2  1 NA  1  0  1  2  0 NA NA  2  0  0  1  0  0  0  0
##
## $`25`
##  [1] NA NA  1 NA NA  1  0 NA NA NA  1  0  1  0  0  1  0  0 NA  0  2  1  2  0  0
## [26] NA  0  2 NA NA  2 NA NA  0  0 NA NA  0  0  1  0  0  2  1  0  1 NA NA  1  0
##
## $`26`
##  [1] NA NA NA  0  0  0  1  1 NA  1  0  0  0  2  0  1  0  0  1 NA  0  0  2  1  2
## [26] NA  0 NA NA  0  0  0  0  2  0  2 NA  1  2  0 NA NA NA  2  0 NA  1 NA NA  1
##
## $`27`
##  [1]  1  0  0 NA  1  1  0  0  0  1  0  1  0 NA  2 NA  0  2  0  0 NA  0 NA  2  0
## [26]  1 NA NA  2 NA  0  2  1  0 NA NA  0 NA NA  0 NA  0  1  1  0  1  0  0  0  0
##
## $`28`
##  [1] NA NA  0  0 NA NA  1  1  0  0 NA  1  1 NA  0  0 NA  0  1  2  2  0  0 NA NA
## [26]  0  0  2 NA NA NA  0  0 NA  1 NA  0 NA  0 NA NA  2  2 NA NA  2  0  0  0  2
##
## $`29`
##  [1]  1  2  2  2  0  2  1  1 NA  0  0  1  0  2  1  1  2  1  0 NA NA NA  0  0  2
## [26]  0  0 NA  0  2  0  0 NA NA  0 NA  0  0  0  2 NA  1 NA  0  1 NA  0 NA  0  0
##
## $`30`
##  [1]  2  0  1 NA  0  1  2 NA NA  2  0  2  2 NA  2  1  1  2 NA  0  0  2 NA  2  2
## [26] NA  0  1 NA  0 NA NA  1 NA  2  1  1 NA NA  0  1  0  0 NA NA NA  0  0 NA NA
##
## $`31`
##  [1]  0  1  0  0 NA NA  1  0  1  0 NA  1  2  1  2  0 NA NA  0 NA  0  0  0  1 NA
## [26]  1  0 NA NA  0  2 NA  2  2  0  0  0  0  0  0  0  2 NA NA  1 NA NA  0  2 NA
##
## $`32`
##  [1]  1  0  0 NA  0  0 NA NA  2  2 NA NA  0 NA NA  0 NA  2  0  1 NA  2  0  0 NA
## [26] NA  2 NA  0  0 NA  0 NA  0 NA NA NA  2 NA  0  0  2 NA  0  0  0 NA  0  0  0
##
## $`33`
##  [1]  1 NA  1 NA  0  2  0  1  2  0  2  2 NA  1 NA  1 NA  0  2  0  2  0  0 NA  2
## [26]  1  0  2  1 NA  0  2  2  0 NA  2  0  0 NA NA NA NA  0  0 NA NA  1 NA  0  0
##
## $`34`
##  [1]  0  0  0  1 NA  0 NA  0  0  2  2 NA  0 NA  0  0  0  1 NA NA NA NA  0 NA  1
## [26]  0  0  2 NA  2  0  2 NA  1  2 NA  1  0 NA  0 NA NA  1  0  0 NA  0  0 NA  0
```

```
##
## $`35`
##  [1]  1  2  2  1  1 NA  0 NA NA  0  1  2 NA  0 NA  2  0  1  0  1  2  0 NA NA  1
## [26] NA NA  0  2  0  0 NA NA NA  0 NA  0  0  1  0  0  0  0  2 NA NA  0  0  0 NA
##
## $`36`
##  [1]  0  1  2  1  1  0  0  0 NA  2  0 NA NA  0  2  2  1  1  0  1  1 NA  0 NA  1
## [26]  2  1  0  1  0 NA NA  0 NA  0  0  1  2 NA  1  1 NA NA  1  2 NA  0  2  0  0
##
## $`37`
##  [1] NA  0 NA  1 NA  0  0  0 NA  0  1  1 NA NA  0  0  0  1  0  1  0  2  1 NA  1
## [26]  1  0  0  0 NA  2 NA  1  2 NA  1 NA NA  0 NA NA  0  0  0 NA  0  0 NA NA  1
##
## $`38`
##  [1]  0  0 NA  2  1  1  2  1 NA  0 NA  1 NA  0  0  0  0 NA  2 NA  1  0 NA NA  0
## [26]  0  2 NA NA  0 NA NA  2  1 NA  0  0 NA  0  0  0  2  0  0  2 NA  0  0 NA  0
##
## $`39`
##  [1]  2  2  0  2  1  0  1  2  1  0  2  0 NA  1  0  0  2 NA  0  2  0 NA NA NA NA
## [26] NA NA NA  2  0  0  0 NA  1  0  0 NA  0  2  0  0  0 NA  0 NA  0  0  0 NA  0
##
## $`40`
##  [1] NA  2  0  0  1  0  2  2  0  2  1  2  0 NA NA NA  0 NA  0  2  2 NA NA  0  0
## [26] NA  2  2  2  2 NA NA  2 NA  0  2 NA  0 NA NA  0  0  1  0  0 NA  0 NA NA NA
##
## $`41`
##  [1] NA  0 NA NA  1  2 NA  1  1  1  1  0  0 NA  0  2  0 NA  0 NA  2  1  0 NA  1
## [26]  0  2 NA  0  0  2 NA  0 NA NA  0 NA NA  2 NA  0  0  0 NA  2 NA  0  1  0 NA
##
## $`42`
##  [1]  2  0  0  0  0 NA  2 NA  2 NA NA  2  2 NA  0  0  2 NA NA  1 NA  1  0  0  0
## [26]  0  0  0  1  2  1  0 NA NA  1  0  0 NA NA  1  0  1  0 NA NA NA  0  2 NA  2
##
## $`43`
##  [1]  0  1  2  0 NA  2 NA  0  2 NA  0 NA  2  0  1  1 NA NA  0 NA  1  0  2  0  0
## [26]  0 NA NA  0  2  2  0  2 NA  1  2  2 NA NA  1 NA NA NA NA  0  0  1  0  2 NA
##
## $`44`
##  [1]  2  1 NA  1  2  1 NA  0  0 NA  1  1  2  1  0  0 NA NA  1 NA NA  2  2  0 NA
## [26]  2  0  2  0  1  2 NA NA  0  0  1 NA  0 NA  2 NA  1  0  2  0  1  0  0  1  0
##
## $`45`
##  [1]  0  2 NA  2 NA  2 NA  2 NA  1  0 NA  2  2  1 NA  0  2  1  0 NA  0 NA  0  0
## [26]  1  0  1  1  0  2 NA  0  0  2 NA  0  0  2  0  1  2 NA NA  1 NA  2  0 NA  2
##
## $`46`
##  [1] NA NA  0  1 NA  2  0  1  2  0  1  0  0  1  2  1  0  1  0  0 NA  2  0  2 NA
## [26]  1  0  1  0  1  0  2  1  0 NA  2 NA  1  0  0  0 NA  2 NA  0  1  0  0 NA  0
##
## $`47`
##  [1]  0 NA  0  0  0  2  0  0 NA NA  0  1  2  1 NA  0  2  1  0  1  0  0  1  2  1
## [26] NA  2  1 NA  2  0  1  1  1  0  0  0  0  0 NA NA  0  1 NA NA NA  2  2 NA  1
##
## $`48`
```

```
##  [1]  0  2  0  1  2  0  0  2  0  2  0 NA  0 NA  1  1  0 NA  2  0  0  1 NA  2  1
## [26]  0  1  2 NA  1  0  2  0 NA  1 NA  0 NA  2 NA NA  0  1  0  0  0  2  0 NA  0
##
## $`49`
##  [1]  1  0  1 NA NA NA NA  0  0  0  0  2  1  2  0 NA NA NA  0  0  1  1  0  2  0
## [26]  2  1  1  1  1  0  0 NA  2  0  0  0  0  1 NA  2 NA NA  0  1  0  0  0  0  0
##
## $`50`
##  [1]  1  1 NA  2 NA  1 NA  1  1  2  2  0  0 NA  0  0  2  2  0 NA  2  0 NA  0  2
## [26]  1  0  0  2  0 NA  2  0 NA  0 NA  2  0  2  0  0 NA NA NA  0  2 NA NA  1  1
```

- In one statement, use the `lapply` function to create a list whose keys are the column number and values are themselves a list with keys: "min" whose value is the minimum of the column, "max" whose value is the maximum of the column, "pct_missing" is the proportion of missingness in the column and "first_NA" whose value is the row number of the first time the NA appears.

```
lapply(split(R, col(R)), function(c) {
  list("min" = min(c, na.rm = TRUE),
       "max" = max(c, na.rm = TRUE),
       "pct_missing" = sum(is.na(c)) / length(c),
       "first_NA" = which(is.na(c)) [1] )})
```

```
## $`1`
## $`1`$min
## [1] 0
##
## $`1`$max
## [1] 2
##
## $`1`$pct_missing
## [1] 0.24
##
## $`1`$first_NA
## [1] 2
##
##
## $`2`
## $`2`$min
## [1] 0
##
## $`2`$max
## [1] 2
##
## $`2`$pct_missing
## [1] 0.28
##
## $`2`$first_NA
## [1] 1
##
##
## $`3`
## $`3`$min
## [1] 0
```

```
##
## $`3`$max
## [1] 2
##
## $`3`$pct_missing
## [1] 0.38
##
## $`3`$first_NA
## [1] 2
##
##
## $`4`
## $`4`$min
## [1] 0
##
## $`4`$max
## [1] 2
##
## $`4`$pct_missing
## [1] 0.26
##
## $`4`$first_NA
## [1] 3
##
##
## $`5`
## $`5`$min
## [1] 0
##
## $`5`$max
## [1] 2
##
## $`5`$pct_missing
## [1] 0.32
##
## $`5`$first_NA
## [1] 7
##
##
## $`6`
## $`6`$min
## [1] 0
##
## $`6`$max
## [1] 2
##
## $`6`$pct_missing
## [1] 0.22
##
## $`6`$first_NA
## [1] 5
##
##
## $`7`
```

```
## $`7`$min
## [1] 0
##
## $`7`$max
## [1] 2
##
## $`7`$pct_missing
## [1] 0.26
##
## $`7`$first_NA
## [1] 2
##
##
## $`8`
## $`8`$min
## [1] 0
##
## $`8`$max
## [1] 2
##
## $`8`$pct_missing
## [1] 0.3
##
## $`8`$first_NA
## [1] 6
##
##
## $`9`
## $`9`$min
## [1] 0
##
## $`9`$max
## [1] 2
##
## $`9`$pct_missing
## [1] 0.38
##
## $`9`$first_NA
## [1] 1
##
##
## $`10`
## $`10`$min
## [1] 0
##
## $`10`$max
## [1] 2
##
## $`10`$pct_missing
## [1] 0.34
##
## $`10`$first_NA
## [1] 7
##
```

```
##
## $`11`
## $`11`$min
## [1] 0
##
## $`11`$max
## [1] 2
##
## $`11`$pct_missing
## [1] 0.24
##
## $`11`$first_NA
## [1] 6
##
##
## $`12`
## $`12`$min
## [1] 0
##
## $`12`$max
## [1] 2
##
## $`12`$pct_missing
## [1] 0.3
##
## $`12`$first_NA
## [1] 3
##
##
## $`13`
## $`13`$min
## [1] 0
##
## $`13`$max
## [1] 2
##
## $`13`$pct_missing
## [1] 0.4
##
## $`13`$first_NA
## [1] 2
##
##
## $`14`
## $`14`$min
## [1] 0
##
## $`14`$max
## [1] 2
##
## $`14`$pct_missing
## [1] 0.32
##
## $`14`$first_NA
```

```
## [1] 7
##
##
## $`15`
## $`15`$min
## [1] 0
##
## $`15`$max
## [1] 2
##
## $`15`$pct_missing
## [1] 0.3
##
## $`15`$first_NA
## [1] 1
##
##
## $`16`
## $`16`$min
## [1] 0
##
## $`16`$max
## [1] 2
##
## $`16`$pct_missing
## [1] 0.24
##
## $`16`$first_NA
## [1] 3
##
##
## $`17`
## $`17`$min
## [1] 0
##
## $`17`$max
## [1] 2
##
## $`17`$pct_missing
## [1] 0.26
##
## $`17`$first_NA
## [1] 6
##
##
## $`18`
## $`18`$min
## [1] 0
##
## $`18`$max
## [1] 2
##
## $`18`$pct_missing
## [1] 0.3
```

```
##
## $`18`$first_NA
## [1] 8
##
##
## $`19`
## $`19`$min
## [1] 0
##
## $`19`$max
## [1] 2
##
## $`19`$pct_missing
## [1] 0.4
##
## $`19`$first_NA
## [1] 2
##
##
## $`20`
## $`20`$min
## [1] 0
##
## $`20`$max
## [1] 2
##
## $`20`$pct_missing
## [1] 0.38
##
## $`20`$first_NA
## [1] 1
##
##
## $`21`
## $`21`$min
## [1] 0
##
## $`21`$max
## [1] 2
##
## $`21`$pct_missing
## [1] 0.32
##
## $`21`$first_NA
## [1] 2
##
##
## $`22`
## $`22`$min
## [1] 0
##
## $`22`$max
## [1] 2
##
```

```
## $`22`$pct_missing
## [1] 0.3
##
## $`22`$first_NA
## [1] 6
##
##
## $`23`
## $`23`$min
## [1] 0
##
## $`23`$max
## [1] 2
##
## $`23`$pct_missing
## [1] 0.32
##
## $`23`$first_NA
## [1] 8
##
##
## $`24`
## $`24`$min
## [1] 0
##
## $`24`$max
## [1] 2
##
## $`24`$pct_missing
## [1] 0.3
##
## $`24`$first_NA
## [1] 3
##
##
## $`25`
## $`25`$min
## [1] 0
##
## $`25`$max
## [1] 2
##
## $`25`$pct_missing
## [1] 0.34
##
## $`25`$first_NA
## [1] 1
##
##
## $`26`
## $`26`$min
## [1] 0
##
## $`26`$max
```

```
## [1] 2
##
## $`26`$pct_missing
## [1] 0.3
##
## $`26`$first_NA
## [1] 1
##
##
## $`27`
## $`27`$min
## [1] 0
##
## $`27`$max
## [1] 2
##
## $`27`$pct_missing
## [1] 0.26
##
## $`27`$first_NA
## [1] 4
##
##
## $`28`
## $`28`$min
## [1] 0
##
## $`28`$max
## [1] 2
##
## $`28`$pct_missing
## [1] 0.38
##
## $`28`$first_NA
## [1] 1
##
##
## $`29`
## $`29`$min
## [1] 0
##
## $`29`$max
## [1] 2
##
## $`29`$pct_missing
## [1] 0.24
##
## $`29`$first_NA
## [1] 9
##
##
## $`30`
## $`30`$min
## [1] 0
```

```
##
## $`30`$max
## [1] 2
##
## $`30`$pct_missing
## [1] 0.36
##
## $`30`$first_NA
## [1] 4
##
##
## $`31`
## $`31`$min
## [1] 0
##
## $`31`$max
## [1] 2
##
## $`31`$pct_missing
## [1] 0.3
##
## $`31`$first_NA
## [1] 5
##
##
## $`32`
## $`32`$min
## [1] 0
##
## $`32`$max
## [1] 2
##
## $`32`$pct_missing
## [1] 0.4
##
## $`32`$first_NA
## [1] 4
##
##
## $`33`
## $`33`$min
## [1] 0
##
## $`33`$max
## [1] 2
##
## $`33`$pct_missing
## [1] 0.3
##
## $`33`$first_NA
## [1] 2
##
##
## $`34`
```

```
## $`34`$min
## [1] 0
##
## $`34`$max
## [1] 2
##
## $`34`$pct_missing
## [1] 0.34
##
## $`34`$first_NA
## [1] 5
##
##
## $`35`
## $`35`$min
## [1] 0
##
## $`35`$max
## [1] 2
##
## $`35`$pct_missing
## [1] 0.32
##
## $`35`$first_NA
## [1] 6
##
##
## $`36`
## $`36`$min
## [1] 0
##
## $`36`$max
## [1] 2
##
## $`36`$pct_missing
## [1] 0.24
##
## $`36`$first_NA
## [1] 9
##
##
## $`37`
## $`37`$min
## [1] 0
##
## $`37`$max
## [1] 2
##
## $`37`$pct_missing
## [1] 0.34
##
## $`37`$first_NA
## [1] 1
##
```

```
##
## $`38`
## $`38`$min
## [1] 0
##
## $`38`$max
## [1] 2
##
## $`38`$pct_missing
## [1] 0.32
##
## $`38`$first_NA
## [1] 3
##
##
## $`39`
## $`39`$min
## [1] 0
##
## $`39`$max
## [1] 2
##
## $`39`$pct_missing
## [1] 0.28
##
## $`39`$first_NA
## [1] 13
##
##
## $`40`
## $`40`$min
## [1] 0
##
## $`40`$max
## [1] 2
##
## $`40`$pct_missing
## [1] 0.36
##
## $`40`$first_NA
## [1] 1
##
##
## $`41`
## $`41`$min
## [1] 0
##
## $`41`$max
## [1] 2
##
## $`41`$pct_missing
## [1] 0.36
##
## $`41`$first_NA
```

```
## [1] 1
##
##
## $`42`
## $`42`$min
## [1] 0
##
## $`42`$max
## [1] 2
##
## $`42`$pct_missing
## [1] 0.32
##
## $`42`$first_NA
## [1] 6
##
##
## $`43`
## $`43`$min
## [1] 0
##
## $`43`$max
## [1] 2
##
## $`43`$pct_missing
## [1] 0.34
##
## $`43`$first_NA
## [1] 5
##
##
## $`44`
## $`44`$min
## [1] 0
##
## $`44`$max
## [1] 2
##
## $`44`$pct_missing
## [1] 0.26
##
## $`44`$first_NA
## [1] 3
##
##
## $`45`
## $`45`$min
## [1] 0
##
## $`45`$max
## [1] 2
##
## $`45`$pct_missing
## [1] 0.28
```

```
##
## $`45`$first_NA
## [1] 3
##
##
## $`46`
## $`46`$min
## [1] 0
##
## $`46`$max
## [1] 2
##
## $`46`$pct_missing
## [1] 0.2
##
## $`46`$first_NA
## [1] 1
##
##
## $`47`
## $`47`$min
## [1] 0
##
## $`47`$max
## [1] 2
##
## $`47`$pct_missing
## [1] 0.24
##
## $`47`$first_NA
## [1] 2
##
##
## $`48`
## $`48`$min
## [1] 0
##
## $`48`$max
## [1] 2
##
## $`48`$pct_missing
## [1] 0.22
##
## $`48`$first_NA
## [1] 12
##
##
## $`49`
## $`49`$min
## [1] 0
##
## $`49`$max
## [1] 2
##
```

```
## $`49`$pct_missing
## [1] 0.22
##
## $`49`$first_NA
## [1] 4
##
##
## $`50`
## $`50`$min
## [1] 0
##
## $`50`$max
## [1] 2
##
## $`50`$pct_missing
## [1] 0.28
##
## $`50`$first_NA
## [1] 3
```

- Create a vector v consisting of a sample of 1,000 iid normal realizations with mean -10 and variance 100.

```
v = rnorm(1000, mean = -10, sd = sqrt(100))
v
```

```
##    [1] -16.61194508   3.84106904 -17.58668155 -12.93074482 -26.89838858
##    [6] -17.73377488 -11.11490445   9.03053130 -24.11527423 -11.99056646
##   [11] -13.05917617   2.10872542  -7.51481577  11.51398971  -0.37812503
##   [16]  -4.20308131  -7.00385857 -18.88435688 -18.42598014  -9.84762329
##   [21] -20.72015048  -3.01608962 -21.82729404 -18.71274499 -18.74271125
##   [26]  -9.98240684 -19.43288497  -4.68322004 -28.96694352   7.32825924
##   [31]  -9.50300246   1.57792466 -15.96695180  -7.57200440   1.98224882
##   [36] -18.50281418  -8.76485826 -23.75415964   0.63094467  -2.91097394
##   [41] -20.59688627   1.14495526 -18.45097044 -12.24152751 -12.88381101
##   [46] -15.95198812   0.80126502  -9.59238766 -27.11419844 -16.41945260
##   [51] -18.98360529   5.58699160 -18.49556963 -21.87896616 -18.16180758
##   [56]   0.43349240 -23.75984331  -2.62747604  -4.85306926  -3.41746769
##   [61]  -5.30902564  -0.82406937   8.52191109  -7.73881273 -12.41994639
##   [66] -17.44244220  -8.93607828 -13.38430395  -7.74324042   4.11627776
##   [71] -16.30267275  -6.82159634 -28.46062010  -9.98749726  -9.57837175
##   [76] -14.33225219   9.18779688 -13.29049183 -30.69216562 -11.06710065
##   [81] -13.04330527  -7.05279515  11.35158197  -8.62866479  -5.25738838
##   [86]   8.41955034 -30.51920393   8.65222962 -22.92966185 -11.37627025
##   [91]  -1.78937574 -15.92782802  -4.37824263  -5.83222689 -16.19413425
##   [96]  -2.51811313 -32.57645380 -28.36919337  -7.04447431 -12.63092113
##  [101] -13.35133916   0.39595370   6.85425944   1.56111161  -2.97441336
##  [106]  -3.07336522 -18.32676920 -14.18074277   1.42023846   0.14055137
##  [111]  -9.62052924  -4.79893059  -1.06946799   0.20971996  -9.71000078
##  [116]   3.81056276 -15.71019715 -10.03950406 -17.57704813  -5.96962944
##  [121]  -9.22130507 -17.01520969  -1.76996748  -1.91227875   5.80625200
##  [126]  -5.94940001  -7.46585934  -7.68246246 -26.80840398  -2.56040472
##  [131] -37.03377764  -3.39449698  -3.47732134 -11.69655891 -16.59526987
```

```
##  [136] -21.20800108 -25.06494895  -5.37557295  -6.84855569 -13.44088330
##  [141] -30.50019228 -10.28310559  -5.51897329  16.38057394   2.03729961
##  [146] -11.15519287  -9.33393215 -11.57326074 -17.76486287  11.85573410
##  [151] -17.18566676  -7.76643629 -17.79372207  10.08516830  -8.10937735
##  [156]  -7.07906262  17.95977768   1.57080014 -17.68521782  -5.21615664
##  [161] -16.87110096  -6.38843182   7.40482076  -7.99761301   6.84045576
##  [166] -21.24382153 -19.76372779  -5.76010183   1.97380785  -4.45303366
##  [171] -13.79680830  -4.35198775 -27.24684525   2.61731205 -16.94831448
##  [176] -28.11110042 -16.49165594 -11.67736924 -10.50503551  -2.94451704
##  [181] -20.96502649  -4.22317536 -21.86058975  -7.67789383  -3.53958956
##  [186]  -0.47071021  -5.68785721   1.10256363  -4.49458473 -25.58176561
##  [191] -22.66285759 -23.25208504  -7.95585903  -3.18302013  10.10905753
##  [196] -10.15118838  -5.03842245  -1.64647259 -11.59218072 -10.04937018
##  [201] -10.66913382  -1.00637932   0.16072649 -12.64313918   6.63438554
##  [206] -10.38574318 -15.13450142 -14.05569564  11.95001518  -4.42848744
##  [211]  -7.30506038 -15.15886451   5.24664010 -19.27441568 -15.73940327
##  [216]  -5.72200559  -4.80419994 -21.74738805 -19.92369500 -42.68413955
##  [221]  -3.69344771 -18.71387730  -9.22687198 -15.48977677 -22.54356626
##  [226]  -0.99047681   0.20166511 -19.88148539 -20.20106659  -7.24156989
##  [231] -12.09859613  -2.91428449 -15.52435819 -23.57384571   9.01984445
##  [236] -22.80142011  -1.50247198 -25.16931112   2.54443581  -6.36964897
##  [241]  17.80947374   7.83210051 -12.49045387 -10.43031594 -11.72793321
##  [246]  -8.55567505 -11.10693220   1.22612164  -2.24881561 -10.35080922
##  [251]  -6.60213846 -11.20960158 -20.72176286  -8.54128719  -3.05655830
##  [256] -17.77359136   8.55906157   4.03841357  -7.34148030 -16.04159537
##  [261]  -8.34278876 -12.68532185 -14.63560254  -5.13864909 -15.73672587
##  [266]  15.52496942   7.61829040  -6.94221031  -5.49438881 -16.32044121
##  [271] -19.39960488   1.13403527 -19.75530905 -24.72692642  -9.69599528
##  [276] -17.28331422   5.62357149 -24.24529844 -22.05307333 -12.44671457
##  [281]  -2.56143987  -1.57279577 -28.21552948 -31.96899564   2.46998917
##  [286]  -7.92543191  -4.62220104 -31.72505230 -31.80463805  -8.72402560
##  [291]  -7.08433498   8.25732684  -7.20012770  -5.03827779  -1.02130350
##  [296] -10.08792442   5.05442500 -13.79005149 -29.73381118 -18.12421387
##  [301] -14.66202049   4.08031710   2.02670517 -11.43012303 -24.85952517
##  [306] -12.77423259  -3.63461534 -15.62935790 -11.80347947 -13.51367521
##  [311] -12.57751060  -4.48108577   2.32890404 -11.20774093 -15.50260628
##  [316] -33.23421578   0.81040260 -16.46664768 -18.33256236   7.50980666
##  [321] -18.12004433   0.99968444  -2.47818415  13.41481282  -1.96955051
##  [326] -22.15748544   0.56513215 -33.07811788 -13.32452291  12.54108876
##  [331] -20.81833500 -13.71179104 -15.04908292 -13.86781891 -24.38174630
##  [336] -13.08853647  17.36990358   5.67885673 -23.71139700   3.32864446
##  [341]  -0.18640853  -9.32126737   1.62247514  -0.65137262 -13.08462147
##  [346] -15.56955278  -5.72600655 -22.57599671 -10.26698904 -14.86207825
##  [351]  13.60019882   7.68712388  -7.55621993 -22.69955769   9.40785345
##  [356]   6.40690060  -2.09820556   0.61969608   2.85056929  -7.51357453
##  [361] -11.05844073  -7.41921072 -20.84249054  -2.68174944 -22.93397283
##  [366]   0.66333890   4.06138224 -13.39710646  -4.30193079 -12.41590152
##  [371]  -5.57021155 -25.73360208 -16.00247087 -14.54525797   0.48150835
##  [376]  -1.17903618 -16.19984864   9.03566725 -35.97739554  11.64141530
##  [381]  -6.63173924 -11.62084565  -1.31895998 -17.51314266  -9.96000231
##  [386]  -9.54475496  -2.41735000 -11.12307987  -9.18020943  -3.52107419
##  [391] -21.61428266  -7.03704267   2.40197751 -15.03665799  -9.64178827
##  [396]  -3.10072109 -17.77232091 -37.06252015 -23.21426062 -15.38197724
##  [401] -23.79464461  12.84747053 -17.40684345 -23.60959214  -8.84260166
```

```
##  [406]  -1.06165349 -12.75944355 -20.67569294 -14.00201874 -23.90821316
##  [411] -13.63537402   1.99980432 -14.13447334 -14.91310064  -7.27141401
##  [416]  -2.13114149 -14.34569412   5.09651699  -8.00576255  -9.28853274
##  [421] -10.45276546 -17.86010925  15.72658893 -27.95187257 -23.94864693
##  [426] -14.10501169 -23.00795419 -19.28000259  -9.01952091  -2.76603121
##  [431] -18.47051472   0.27865150 -12.34654887  -7.08741873 -13.23854553
##  [436] -17.88124706 -11.02210831 -21.55788728   3.19269457 -21.85003030
##  [441]  -9.06810288  -5.39068009  -3.72462202   0.88428595  -0.84471948
##  [446]  -5.52659895 -13.82666406 -10.13420062   2.87735368 -28.33769582
##  [451]   0.64888272  -8.64402828   2.98501730 -18.20689004   7.50837930
##  [456]   5.67208842 -11.92079549 -26.12235718  -6.98717849 -22.97408870
##  [461] -13.15756903 -13.00199822 -19.52857695  -5.15476574 -11.47397146
##  [466] -15.99813870 -24.53019325  -3.51994710 -24.34166930 -19.95524544
##  [471]   5.05968071  -6.56839680   1.76098365 -27.95838919 -18.81468866
##  [476]  -0.11394084 -15.24031851 -11.52651013  -5.64073118 -19.91743925
##  [481] -18.43545373  -8.77157449 -13.57282443 -11.66156882 -25.28650615
##  [486]  -3.93847062 -41.79810400   1.85619412  -1.89485540 -17.77693617
##  [491]  -3.14634715  -0.75752349 -22.75510035 -26.26096119 -11.75954182
##  [496]  -1.80882353   6.70011211   0.42203331 -19.81206984  -1.59329002
##  [501]  -3.41237755 -10.37691819  -4.22281465  -3.70145901 -21.46661619
##  [506] -14.55841025  -4.39366432  -2.78046622 -20.34180350 -16.56256001
##  [511] -26.01252853 -12.38034564  -8.91289640  -8.67173088 -14.30182717
##  [516]   0.34993432 -30.03126407   9.85632911  18.79314130 -28.44319299
##  [521] -18.32054247  -7.75301489  18.52139591 -13.27809727  -7.52504400
##  [526] -20.80413761  -8.59114634 -15.50442815 -12.33205027 -14.83838539
##  [531] -17.49724399  -9.30125991 -21.81171648 -23.60494218  -7.96797834
##  [536] -10.36876268 -17.04246217  -2.49912733   0.20252424  24.11307295
##  [541] -14.39573103 -12.55837501   2.37460602 -14.72139630  -3.58954018
##  [546]  23.36861079  -6.29270228  -2.58203366 -13.58091715 -18.05440604
##  [551] -23.96029015 -24.22408631 -12.32558198 -21.41706781  -0.04360046
##  [556] -17.69604199  -1.00701169  -8.75678982   8.45079252  -6.63533314
##  [561]  -8.05504144 -24.93297260  -2.49900727 -12.94140851 -18.87969039
##  [566]   7.31633885 -15.56115025  -8.41653504 -16.34875234 -23.59001542
##  [571] -12.81629285 -20.34182242 -22.54739945   8.52238249  -5.29298673
##  [576] -22.64321199 -18.84085318 -28.15105825 -17.53641319   0.07843714
##  [581] -21.95807981 -17.56594403 -14.98595140 -18.93636469  -2.81673242
##  [586]  -6.80897286 -10.02193142  -7.45869962  -7.15105641 -31.16984032
##  [591]  -6.03727114  -2.19237225  -5.71950365 -19.43868901  -8.36193823
##  [596] -20.58100243 -23.22870882  -5.77299716 -18.99043854 -22.98444485
##  [601] -18.67937375  -4.23686840 -10.68217548  -0.48351428 -28.08246350
##  [606]  -4.03108210 -15.96563547 -19.37742663  -5.84051819 -16.98037767
##  [611]  -5.30781912 -17.70739952 -13.69145962 -19.75891529   5.64132831
##  [616] -22.98028826   1.44378058 -13.67061562   3.44065713 -23.32034594
##  [621] -14.95998974  -8.66893079  -8.05903387  -5.25783538 -20.11072078
##  [626]  -5.01002465   1.27872188 -16.15904121  -1.85250066 -20.66764925
##  [631]  -1.12982202  15.31716265  -5.71720775 -11.30334250 -15.70772345
##  [636]   9.18272239  -7.40941849 -13.98975832   2.83128103 -15.87102956
##  [641]   0.88202741 -18.72155442  10.18940816  -6.43447438 -11.72466952
##  [646] -12.17754452  -9.38785109  -5.79742857 -18.51384748   2.61603646
##  [651] -21.23497171  -6.63708855 -22.61871025 -17.90584894 -25.65319042
##  [656]  12.81572276  -6.45777943 -11.08074747  -4.68488340  -6.85875053
##  [661]  -5.27799501 -25.28340196 -10.92271626  -7.21439833 -24.00423026
##  [666]  -4.76003765  -6.07437286  10.01875079 -18.80867696  -7.12729859
##  [671] -13.54032343   2.65395982  -1.70230712  -3.13458201  -9.08330758
```

```
##  [676] -23.77002029   1.48374389   3.93232568  -3.31548548 -11.49633498
##  [681] -19.94897042 -19.75582281 -10.10975495 -13.33240439 -11.25702813
##  [686] -16.82644346 -19.69348995   3.12023196 -15.76994945 -17.91228498
##  [691]  -6.98667057 -19.55269563   1.92312463 -23.00618425 -12.70801517
##  [696]   1.28645822   3.17465239  -2.20682255 -12.21898370 -17.41105498
##  [701] -22.15892534 -13.04488770 -35.86892367 -26.18170028 -16.32310701
##  [706]  -6.21319575  -8.50913507 -20.32901986   5.42380111 -17.13014624
##  [711]   0.20847391   5.76079838  -5.26247785 -17.43427053 -16.36719108
##  [716]   5.50742967  -8.95579858  -3.53621542  -1.76487057 -12.35960496
##  [721]  -2.93037420  -1.92189813 -14.64955007 -12.09145291  -6.07900156
##  [726]  -8.77423680 -15.21478127   0.21658873 -15.44551479  -3.66321473
##  [731] -17.77334865   7.65912271  -7.84012191  -1.13087263 -19.40985556
##  [736] -29.03438309 -12.67782015   5.18223106 -16.66883924   2.65256421
##  [741] -17.68379142 -17.75753007  -5.21447899   0.23825633 -14.96458974
##  [746] -11.26163696 -13.54055303   2.69131555   8.87545743 -28.06703334
##  [751] -20.06689586 -16.26747559 -13.36419630 -12.96471041   3.63463777
##  [756]  -4.29489316  -8.99990708  -3.69309796  -3.31481857  -3.52753672
##  [761]  -0.45250242  -4.56832376 -24.51730636   1.77961280   5.21312679
##  [766] -22.05169004 -14.13835918   3.85330662   2.29467445  -8.00550323
##  [771] -22.21852550  -4.58553791 -10.70160869 -14.97906729  -1.39589349
##  [776]  -2.25426103  -5.32081821 -24.22179551 -14.38718521 -18.36327044
##  [781]   2.79091035 -23.41514501 -19.24332701 -19.35785762 -13.06443373
##  [786] -17.61498971 -11.51182866 -25.75133933 -13.10287360 -22.46829748
##  [791] -41.72402828 -20.63204817   4.09624704 -10.63293944  -6.10465969
##  [796] -14.36398139 -31.18844756  -6.96925635 -13.64269054  -8.50582465
##  [801]  -7.08222608 -21.02650651  -5.87863906 -19.26818013 -15.42216458
##  [806] -22.40546301 -19.40728431  -0.33855743 -10.80919050  -1.09500315
##  [811] -33.05113030  -9.96914713  -3.27191567   0.52158988  -8.03391742
##  [816]  -6.16104518   6.56692083  -6.00964357 -43.65473986  -7.17321395
##  [821]  -3.23414736 -12.69346710 -10.45848064   4.64108542  -1.21757715
##  [826]   2.34968266 -35.98376676 -15.71152902 -10.09259065 -20.29360860
##  [831]  -9.40363723  -2.07409488  -3.72575639 -18.81129017   1.31571527
##  [836] -17.68940288 -14.01777171 -11.38245262 -27.50440686   2.97948935
##  [841] -28.14056765   0.10511821  -7.17728823 -16.51005736 -16.41121324
##  [846]  -8.65404095  -0.25983919  -3.10009111 -28.41900737  -1.75637080
##  [851]   8.03729051   0.75526737 -23.34335503 -10.55576193  -4.22377151
##  [856] -23.28568203 -15.23286793  -6.40522917  -7.42243136 -20.36233708
##  [861]  -9.09287963 -20.20826323 -13.77872830  -6.19799026 -11.49278092
##  [866] -20.21492314 -15.97601150  -4.91015879 -17.63573763  -3.00575500
##  [871] -16.99975571 -17.33094511  -0.77812649 -11.82634425  -7.38678704
##  [876] -13.67690174 -17.75714897   0.81425048 -17.23323583 -25.68903799
##  [881] -11.24500476 -12.41384157  -6.19776148  -7.92808082   1.35199623
##  [886]   2.29337554  -3.62011259 -18.61360621 -10.45018227 -12.01212543
##  [891]  -5.87523266   0.35472987   3.07253461   4.40629561 -10.33592855
##  [896] -22.92015175 -24.33846089 -14.08573547  -2.82027752 -17.17935055
##  [901] -12.53898409 -30.81040398  -0.37338197 -15.32852482   5.35471795
##  [906]  -8.60925000  -4.08214229   5.83870659 -20.80746372  -4.08519546
##  [911]  -5.02481971 -14.57902068 -19.56788526  -3.20916047 -22.98156671
##  [916]  10.37474774 -18.76691238  -6.00387566 -11.26638321 -10.65937006
##  [921]   1.13515681 -11.04067393  -2.62988028  -8.59932853 -11.32585508
##  [926]  -7.23989261  -2.23121446 -12.62645334  -4.54294080   0.02743108
##  [931]  -8.71893257  15.80172439 -18.45416432  -4.09963150 -11.24805270
##  [936]  -0.20225029  -0.11414835 -30.54917094 -21.77644135   0.95800733
##  [941] -10.28001645   0.33811080 -19.73649427   2.14591383 -21.70500508
```

```
##  [946]  -3.13145201 -14.83902074 -12.01494730 -24.27834384   5.05889811
##  [951] -16.98628940 -15.25022548 -11.54090947 -24.40922105 -10.55131635
##  [956]  -6.58674243 -16.07160490 -20.61392014  -7.89875476  -4.91004600
##  [961] -18.30874725 -19.48692514 -15.79510102  -6.02301615 -22.04311390
##  [966] -13.30693410  -8.18627048  -9.21790535  -0.28017027  -1.19499974
##  [971]  -9.39764881   5.56127249 -18.00916886  -8.48593844 -28.90456120
##  [976] -16.75957951  -4.99897590  -6.92850598 -20.53695867 -15.34775902
##  [981]  -8.60128435  -5.45554550  -0.21520267 -13.37805819 -12.24241389
##  [986] -18.11481441 -16.89781183 -24.61880393 -10.45166508   3.66773007
##  [991]  -1.20548885  -8.76861357 -12.77749743 -11.59471214 -12.21703996
##  [996]  -2.72653392  -7.47088267 -15.52875542 -12.39993485 -10.85449862
```

- Create a function `my_reverse` which takes as required input a vector and returns the vector in reverse where the first entry is the last entry, etc. No function calls are allowed inside your function otherwise that would defeat the purpose of the exercise! (Yes, there is a base R function that does this called `rev`). Use `head` on `v` and `tail` on `my_reverse(v)` to verify it works.

```
my_reverse = function(x){
  y = x[length(x):1]
  return(y)
  }

#Test
head(v, 10)
```

```
##  [1] -16.611945   3.841069 -17.586682 -12.930745 -26.898389 -17.733775
##  [7] -11.114904   9.030531 -24.115274 -11.990566
```

```
tail(my_reverse(v), 10)
```

```
##  [1] -11.990566 -24.115274   9.030531 -11.114904 -17.733775 -26.898389
##  [7] -12.930745 -17.586682   3.841069 -16.611945
```

- Create a function `flip_matrix` which takes as required input a matrix, an argument `dim_to_rev` that returns the matrix with the rows in reverse order or the columns in reverse order depending on the `dim_to_rev` argument. Let the default be the dimension of the matrix that is greater.

```
flip_matrix = function(mat, dim_to_rev = NULL){
  if(is.null(dim_to_rev)){
    if(length(mat[1, ]) > length(mat[ , 1]))
      mat[nrow(mat):1, ]
    else
      mat[ , ncol(mat):1]
  }

  else{
    dim_to_rev = tolower(dim_to_rev)
    if(dim_to_rev == "row" || dim_to_rev == "r")
      mat[ , ncol(mat):1]
    else if(dim_to_rev == "col" || dim_to_rev == "column" || dim_to_rev == "c" )
      mat[nrow(mat):1, ]
  }
```

```
}

#Test
test_matrix = matrix(data = rnorm(15), nrow = 3, ncol = 5)
test_matrix
```

```
##            [,1]       [,2]       [,3]        [,4]       [,5]
## [1,]  0.7940468  1.4146624  0.2164673 -0.03445341 -1.6248695
## [2,] -0.1183017  0.2914973 -0.8440554 -0.33376066 -1.7552902
## [3,]  0.1960262 -0.9212267  0.8079331 -1.62964745 -0.4570013
```

```
flip_matrix(test_matrix)
```

```
##            [,1]       [,2]       [,3]        [,4]       [,5]
## [1,]  0.1960262 -0.9212267  0.8079331 -1.62964745 -0.4570013
## [2,] -0.1183017  0.2914973 -0.8440554 -0.33376066 -1.7552902
## [3,]  0.7940468  1.4146624  0.2164673 -0.03445341 -1.6248695
```

- Find the average of `v` and the standard error of `v`.

```
mean(v, na.rm = TRUE)
```

```
## [1] -9.930764
```

```
sd(v, na.rm = TRUE)
```

```
## [1] 10.33153
```

- Find the 5%ile of `v` and use the `qnorm` function to compute what it theoretically should be. Is the estimate about what is expected by theory?

```
quantile(v, .05)
```

```
##       5%
## -25.7644
```

```
qnorm(.05, mean = -10, sd = sqrt(100))
```

```
## [1] -26.44854
```

#The estimate is about the same as what is expected by theory.

- What is the percentile of `v` that corresponds to the value 0? What should it be theoretically? Is the estimate about what is expected by theory?

```
quantile(v, 0)
```

```
##        0%
## -43.65474
```

```r
qnorm(0, mean = -10, sd = sqrt(100))
```

```
## [1] -Inf
```

The value should be theoretically be negative infinity. The estimate gives -43.30308 which is not close to negative infinity.

- Create a list named `my_list` with keys "A", "B", ... where the entries are arrays of size 1, 2 x 2, 3 x 3 x 3, etc. Fill the array with the numbers 1, 2, 3, etc. Make 8 entries.

```r
my_list = list("A" = matrix(1, nrow=1, ncol=1),
               "B" = matrix(2, nrow=2, ncol=2),
               "C" = matrix(3, nrow=3, ncol=3),
               "D" = matrix(4, nrow=4, ncol=4),
               "E" = matrix(5, nrow=5, ncol=5),
               "F" = matrix(6, nrow=6, ncol=6),
               "G" = matrix(7, nrow=7, ncol=7),
               "H" = matrix(8, nrow=8, ncol=8))
```

Run the following code:

```r
lapply(my_list, object.size)
```

```
## $A
## 224 bytes
##
## $B
## 248 bytes
##
## $C
## 344 bytes
##
## $D
## 344 bytes
##
## $E
## 416 bytes
##
## $F
## 504 bytes
##
## $G
## 608 bytes
##
## $H
## 728 bytes
```

```r
?object.size()
```

```
## starting httpd help server ... done
```

Use `?object.size` to read about what these functions do. Then explain the output you see above. For the later arrays, does it make sense given the dimensions of the arrays?

The object.size function returns an estimate for the memory allocation of the parameter. It makes sense that the larger n x n arrays require more memory allocation compared to smaller sized arrays.

Now cleanup the namespace by deleting all stored objects and functions:

```
rm(list = ls())
```

A little about strings

- Use the `strsplit` function and `sample` to put the sentences in the string `lorem` below in random order. You will also need to manipulate the output of `strsplit` which is a list. You may need to learn basic concepts of regular expressions.

```
lorem = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi posuere varius volutpat. Morbi
sample(unlist(strsplit(lorem,"\\.")))
```

```
##  [1] " Integer dapibus mi lectus, eu posuere arcu ultricies in"
##  [2] " Mauris at sodales augue"
##  [3] "Lorem ipsum dolor sit amet, consectetur adipiscing elit"
##  [4] " Morbi posuere varius volutpat"
##  [5] " Aenean nulla ante, iaculis sed vehicula ac, finibus vel arcu"
##  [6] " Curabitur est augue, congue eget quam in, scelerisque semper magna"
##  [7] " Donec vehicula sagittis nisi non semper"
##  [8] " "
##  [9] " Donec at tempor erat"
## [10] " Cras suscipit id nibh lacinia elementum"
## [11] " Morbi faucibus ligula id massa ultricies viverra"
```