Lab 1

Tyron Samaroo

11:59PM February 8, 2020

You should have RStudio installed to edit this file. You will write code in places marked "TO-DO" to complete the problems. Some of this will be a pure programming assignment. The tools for the solutions to these problems can be found in the class practice lectures. I want you to use the methods I taught you, not for you to google and come up with whatever works. You won't learn that way.

To "hand in" the homework, you should compile or publish this file into a PDF that includes output of your code. Once it's done, push by the deadline to your repository in a directory called "labs".

- Print out the numerical constant pi with ten digits after the decimal point using the internal constant `pi`.

```r
options(digits = 10)
pi
```

```
## [1] 3.141592654
```

- Sum up the first 100 terms of the series 1 + 1/2 + 1/4 + 1/8 + ...

```r
1/2 ^ (0:99) #Generates 1 + 1/2 ...
```

```
##   [1] 1.000000000e+00 5.000000000e-01 2.500000000e-01 1.250000000e-01
##   [5] 6.250000000e-02 3.125000000e-02 1.562500000e-02 7.812500000e-03
##   [9] 3.906250000e-03 1.953125000e-03 9.765625000e-04 4.882812500e-04
##  [13] 2.441406250e-04 1.220703125e-04 6.103515625e-05 3.051757812e-05
##  [17] 1.525878906e-05 7.629394531e-06 3.814697266e-06 1.907348633e-06
##  [21] 9.536743164e-07 4.768371582e-07 2.384185791e-07 1.192092896e-07
##  [25] 5.960464478e-08 2.980232239e-08 1.490116119e-08 7.450580597e-09
##  [29] 3.725290298e-09 1.862645149e-09 9.313225746e-10 4.656612873e-10
##  [33] 2.328306437e-10 1.164153218e-10 5.820766091e-11 2.910383046e-11
##  [37] 1.455191523e-11 7.275957614e-12 3.637978807e-12 1.818989404e-12
##  [41] 9.094947018e-13 4.547473509e-13 2.273736754e-13 1.136868377e-13
##  [45] 5.684341886e-14 2.842170943e-14 1.421085472e-14 7.105427358e-15
##  [49] 3.552713679e-15 1.776356839e-15 8.881784197e-16 4.440892099e-16
##  [53] 2.220446049e-16 1.110223025e-16 5.551115123e-17 2.775557562e-17
##  [57] 1.387778781e-17 6.938893904e-18 3.469446952e-18 1.734723476e-18
##  [61] 8.673617380e-19 4.336808690e-19 2.168404345e-19 1.084202172e-19
##  [65] 5.421010862e-20 2.710505431e-20 1.355252716e-20 6.776263578e-21
##  [69] 3.388131789e-21 1.694065895e-21 8.470329473e-22 4.235164736e-22
##  [73] 2.117582368e-22 1.058791184e-22 5.293955920e-23 2.646977960e-23
##  [77] 1.323488980e-23 6.617444900e-24 3.308722450e-24 1.654361225e-24
##  [81] 8.271806126e-25 4.135903063e-25 2.067951531e-25 1.033975766e-25
##  [85] 5.169878828e-26 2.584939414e-26 1.292469707e-26 6.462348536e-27
##  [89] 3.231174268e-27 1.615587134e-27 8.077935669e-28 4.038967835e-28
##  [93] 2.019483917e-28 1.009741959e-28 5.048709793e-29 2.524354897e-29
##  [97] 1.262177448e-29 6.310887242e-30 3.155443621e-30 1.577721810e-30
```

```r
sum(1/2 ^ (0:99))
```

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

- Find the product of the first 20 terms of `1/3 * 1/6 * 1/9 * ...`

```
#
prod(1 / seq(3, 60, by = 3))
```

```
## [1] 1.178827582e-28
```

- Find the product of the first 500 terms of `1 * 1/2 * 1/4 * 1/8 *` ...

```
prod(1/2 ^ (0:499))
```

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

Is this answer *exactly* correct?

No because will give wrong computation. Not precise answer. The answer is almost 0 but not exactly 0

- Figure out a means to express the answer more exactly. Not compute exactly, but express more exactly.

```
#Use Ln to help.
log(1/2 ^ (0:499))
```

```
##   [1]    0.0000000000   -0.6931471806   -1.3862943611   -2.0794415417
##   [5]   -2.7725887222   -3.4657359028   -4.1588830834   -4.8520302639
##   [9]   -5.5451774445   -6.2383246250   -6.9314718056   -7.6246189862
##  [13]   -8.3177661667   -9.0109133473   -9.7040605278  -10.3972077084
##  [17]  -11.0903548890  -11.7835020695  -12.4766492501  -13.1697964306
##  [21]  -13.8629436112  -14.5560907918  -15.2492379723  -15.9423851529
##  [25]  -16.6355323334  -17.3286795140  -18.0218266946  -18.7149738751
##  [29]  -19.4081210557  -20.1012682362  -20.7944154168  -21.4875625974
##  [33]  -22.1807097779  -22.8738569585  -23.5670041390  -24.2601513196
##  [37]  -24.9532985002  -25.6464456807  -26.3395928613  -27.0327400418
##  [41]  -27.7258872224  -28.4190344030  -29.1121815835  -29.8053287641
##  [45]  -30.4984759446  -31.1916231252  -31.8847703058  -32.5779174863
##  [49]  -33.2710646669  -33.9642118474  -34.6573590280  -35.3505062086
##  [53]  -36.0436533891  -36.7368005697  -37.4299477502  -38.1230949308
##  [57]  -38.8162421114  -39.5093892919  -40.2025364725  -40.8956836530
##  [61]  -41.5888308336  -42.2819780142  -42.9751251947  -43.6682723753
##  [65]  -44.3614195558  -45.0545667364  -45.7477139170  -46.4408610975
##  [69]  -47.1340082781  -47.8271554586  -48.5203026392  -49.2134498198
##  [73]  -49.9065970003  -50.5997441809  -51.2928913614  -51.9860385420
##  [77]  -52.6791857226  -53.3723329031  -54.0654800837  -54.7586272642
##  [81]  -55.4517744448  -56.1449216254  -56.8380688059  -57.5312159865
##  [85]  -58.2243631670  -58.9175103476  -59.6106575282  -60.3038047087
##  [89]  -60.9969518893  -61.6900990698  -62.3832462504  -63.0763934310
##  [93]  -63.7695406115  -64.4626877921  -65.1558349726  -65.8489821532
##  [97]  -66.5421293338  -67.2352765143  -67.9284236949  -68.6215708754
## [101]  -69.3147180560  -70.0078652366  -70.7010124171  -71.3941595977
## [105]  -72.0873067782  -72.7804539588  -73.4736011394  -74.1667483199
## [109]  -74.8598955005  -75.5530426810  -76.2461898616  -76.9393370422
## [113]  -77.6324842227  -78.3256314033  -79.0187785838  -79.7119257644
## [117]  -80.4050729450  -81.0982201255  -81.7913673061  -82.4845144866
## [121]  -83.1776616672  -83.8708088478  -84.5639560283  -85.2571032089
## [125]  -85.9502503894  -86.6433975700  -87.3365447506  -88.0296919311
## [129]  -88.7228391117  -89.4159862922  -90.1091334728  -90.8022806534
## [133]  -91.4954278339  -92.1885750145  -92.8817221950  -93.5748693756
## [137]  -94.2680165562  -94.9611637367  -95.6543109173  -96.3474580978
## [141]  -97.0406052784  -97.7337524590  -98.4268996395  -99.1200468201
## [145]  -99.8131940006 -100.5063411812 -101.1994883618 -101.8926355423
## [149] -102.5857827229 -103.2789299034 -103.9720770840 -104.6652242646
## [153] -105.3583714451 -106.0515186257 -106.7446658062 -107.4378129868
## [157] -108.1309601674 -108.8241073479 -109.5172545285 -110.2104017090
## [161] -110.9035488896 -111.5966960702 -112.2898432507 -112.9829904313
## [165] -113.6761376118 -114.3692847924 -115.0624319730 -115.7555791535
## [169] -116.4487263341 -117.1418735146 -117.8350206952 -118.5281678758
## [173] -119.2213150563 -119.9144622369 -120.6076094174 -121.3007565980
## [177] -121.9939037786 -122.6870509591 -123.3801981397 -124.0733453202
## [181] -124.7664925008 -125.4596396814 -126.1527868619 -126.8459340425
## [185] -127.5390812230 -128.2322284036 -128.9253755841 -129.6185227647
## [189] -130.3116699453 -131.0048171258 -131.6979643064 -132.3911114869
## [193] -133.0842586675 -133.7774058481 -134.4705530286 -135.1637002092
## [197] -135.8568473897 -136.5499945703 -137.2431417509 -137.9362889314
## [201] -138.6294361120 -139.3225832925 -140.0157304731 -140.7088776537
## [205] -141.4020248342 -142.0951720148 -142.7883191953 -143.4814663759
## [209] -144.1746135565 -144.8677607370 -145.5609079176 -146.2540550981
## [213] -146.9472022787 -147.6403494593 -148.3334966398 -149.0266438204
## [217] -149.7197910009 -150.4129381815 -151.1060853621 -151.7992325426
## [221] -152.4923797232 -153.1855269037 -153.8786740843 -154.5718212649
## [225] -155.2649684454 -155.9581156260 -156.6512628065 -157.3444099871
## [229] -158.0375571677 -158.7307043482 -159.4238515288 -160.1169987093
## [233] -160.8101458899 -161.5032930705 -162.1964402510 -162.8895874316
## [237] -163.5827346121 -164.2758817927 -164.9690289733 -165.6621761538
## [241] -166.3553233344 -167.0484705149 -167.7416176955 -168.4347648761
## [245] -169.1279120566 -169.8210592372 -170.5142064177 -171.2073535983
## [249] -171.9005007789 -172.5936479594 -173.2867951400 -173.9799423205
## [253] -174.6730895011 -175.3662366817 -176.0593838622 -176.7525310428
## [257] -177.4456782233 -178.1388254039 -178.8319725845 -179.5251197650
## [261] -180.2182669456 -180.9114141261 -181.6045613067 -182.2977084873
## [265] -182.9908556678 -183.6840028484 -184.3771500289 -185.0702972095
## [269] -185.7634443901 -186.4565915706 -187.1497387512 -187.8428859317
## [273] -188.5360331123 -189.2291802929 -189.9223274734 -190.6154746540
## [277] -191.3086218345 -192.0017690151 -192.6949161957 -193.3880633762
## [281] -194.0812105568 -194.7743577373 -195.4675049179 -196.1606520985
## [285] -196.8537992790 -197.5469464596 -198.2400936401 -198.9332408207
## [289] -199.6263880013 -200.3195351818 -201.0126823624 -201.7058295429
## [293] -202.3989767235 -203.0921239041 -203.7852710846 -204.4784182652
## [297] -205.1715654457 -205.8647126263 -206.5578598069 -207.2510069874
## [301] -207.9441541680 -208.6373013485 -209.3304485291 -210.0235957097
## [305] -210.7167428902 -211.4098900708 -212.1030372513 -212.7961844319
## [309] -213.4893316125 -214.1824787930 -214.8756259736 -215.5687731541
## [313] -216.2619203347 -216.9550675153 -217.6482146958 -218.3413618764
## [317] -219.0345090569 -219.7276562375 -220.4208034181 -221.1139505986
## [321] -221.8070977792 -222.5002449597 -223.1933921403 -223.8865393209
## [325] -224.5796865014 -225.2728336820 -225.9659808625 -226.6591280431
## [329] -227.3522752237 -228.0454224042 -228.7385695848 -229.4317167653
## [333] -230.1248639459 -230.8180111265 -231.5111583070 -232.2043054876
## [337] -232.8974526681 -233.5905998487 -234.2837470293 -234.9768942098
## [341] -235.6700413904 -236.3631885709 -237.0563357515 -237.7494829321
## [345] -238.4426301126 -239.1357772932 -239.8289244737 -240.5220716543
## [349] -241.2152188349 -241.9083660154 -242.6015131960 -243.2946603765
## [353] -243.9878075571 -244.6809547377 -245.3741019182 -246.0672490988
## [357] -246.7603962793 -247.4535434599 -248.1466906405 -248.8398378210
## [361] -249.5329850016 -250.2261321821 -250.9192793627 -251.6124265433
## [365] -252.3055737238 -252.9987209044 -253.6918680849 -254.3850152655
## [369] -255.0781624461 -255.7713096266 -256.4644568072 -257.1576039877
## [373] -257.8507511683 -258.5438983489 -259.2370455294 -259.9301927100
## [377] -260.6233398905 -261.3164870711 -262.0096342517 -262.7027814322
## [381] -263.3959286128 -264.0890757933 -264.7822229739 -265.4753701545
## [385] -266.1685173350 -266.8616645156 -267.5548116961 -268.2479588767
## [389] -268.9411060573 -269.6342532378 -270.3274004184 -271.0205475989
## [393] -271.7136947795 -272.4068419601 -273.0999891406 -273.7931363212
## [397] -274.4862835017 -275.1794306823 -275.8725778629 -276.5657250434
## [401] -277.2588722240 -277.9520194045 -278.6451665851 -279.3383137657
## [405] -280.0314609462 -280.7246081268 -281.4177553073 -282.1109024879
## [409] -282.8040496685 -283.4971968490 -284.1903440296 -284.8834912101
## [413] -285.5766383907 -286.2697855713 -286.9629327518 -287.6560799324
## [417] -288.3492271129 -289.0423742935 -289.7355214741 -290.4286686546
## [421] -291.1218158352 -291.8149630157 -292.5081101963 -293.2012573769
## [425] -293.8944045574 -294.5875517380 -295.2806989185 -295.9738460991
## [429] -296.6669932797 -297.3601404602 -298.0532876408 -298.7464348213
## [433] -299.4395820019 -300.1327291825 -300.8258763630 -301.5190235436
## [437] -302.2121707241 -302.9053179047 -303.5984650853 -304.2916122658
## [441] -304.9847594464 -305.6779066269 -306.3710538075 -307.0642009881
## [445] -307.7573481686 -308.4504953492 -309.1436425297 -309.8367897103
## [449] -310.5299368909 -311.2230840714 -311.9162312520 -312.6093784325
## [453] -313.3025256131 -313.9956727937 -314.6888199742 -315.3819671548
## [457] -316.0751143353 -316.7682615159 -317.4614086965 -318.1545558770
## [461] -318.8477030576 -319.5408502381 -320.2339974187 -320.9271445993
## [465] -321.6202917798 -322.3134389604 -323.0065861409 -323.6997333215
## [469] -324.3928805021 -325.0860276826 -325.7791748632 -326.4723220437
## [473] -327.1654692243 -327.8586164049 -328.5517635854 -329.2449107660
## [477] -329.9380579465 -330.6312051271 -331.3243523077 -332.0174994882
## [481] -332.7106466688 -333.4037938493 -334.0969410299 -334.7900882105
## [485] -335.4832353910 -336.1763825716 -336.8695297521 -337.5626769327
## [489] -338.2558241133 -338.9489712938 -339.6421184744 -340.3352656549
## [493] -341.0284128355 -341.7215600161 -342.4147071966 -343.1078543772
## [497] -343.8010015577 -344.4941487383 -345.1872959189 -345.8804430994
```

```r
sum(log(1/2 ^ (0:499)))
```

```
## [1] -86470.11077
```

- Create the sequence `x = [Inf, 20, 18, ..., -20]`.

```r
#Use c to concat
x = c(Inf,seq(from = 20, to = -20, by = -2))
x
```

```
##  [1] Inf  20  18  16  14  12  10   8   6   4   2   0  -2  -4  -6  -8 -10
## [18] -12 -14 -16 -18 -20
```

Create the sequence `x = [log_3(Inf), log_3(100), log_3(98), ... log_3(-20)]`.

```r
x = c(Inf,seq(from = 100, to=-20, by = -2))
# Getting NaN no good
x = log(x, base = 3)
```

```
## Warning: NaNs produced
```

```r
x
```

```
##  [1]          Inf 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536         -Inf          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

Comment on the appropriateness of the non-numeric values.

Nan come from log(-num). Inf come from log3(Inf and log -inf come from log3(0)

- Create a vector of booleans where the entry is true if `x[i]` is positive and finite.

```
x
```

```
##  [1]          Inf 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536         -Inf          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```
x > 0 & x != Inf & !is.nan(x)
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [34]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
is_pos_real_bool = x > 0 & x != Inf & !is.nan(x)
is.finite(x) #Another way
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [34]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
is_pos_real_bool
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [34]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

- Locate the indices of the non-numbers in this vector. Hint: use the `which` function.

```
#Theres are values that arent finite
which(!is_pos_real_bool)
```

```
##  [1]  1 52 53 54 55 56 57 58 59 60 61 62
```

- Locate the indices of the infinite quantities in this vector. Hint: use the `which` function.

```
x
```

```
##  [1]          Inf 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536         -Inf          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```
which(x == Inf | x == -Inf)
```

```
## [1]  1 52
```

```
is.infinite(x) # Return vector location of -inf and inf
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
which(is.infinite(x)) # return index of those location
```

```
## [1]  1 52
```

- Locate the indices of the min and max in this vector. Hint: use the `which.min` and `which.max` functions.

```
x[is.infinite(x)] = NA
x
```

```
##  [1]           NA 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536           NA          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```r
which.min(x)
```

```
## [1] 51
```

```r
which.max(x)
```

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

- Count the number of unique values in x.

```r
#TO-DO
x
```

```
##  [1]           NA 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536           NA          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```r
length(unique(x))
```

```
## [1] 52
```

- Cast x to a factor. Do the number of levels make sense?

```r
factor(x)
```

```
##  [1] <NA>              4.19180654857877  4.1734172518943  
##  [4] 4.15464876785729  4.13548512895119  4.11590933734319 
##  [7] 4.09590327428938  4.07544759935851  4.05452163806914 
## [10] 4.03310325630434  4.01116871959141  3.98869253500376 
## [13] 3.96564727304425  3.94200336638929  3.91772888178973 
## [16] 3.89278926071437  3.86714702345081  3.84076143030548 
## [19] 3.81358809221559  3.78557852142874  3.75667961082847 
## [22] 3.72683302786084  3.69597450568212  3.66403300987579 
## [25] 3.63092975357146  3.59657702661571  3.56087679500731 
## [28] 3.52371901428583  3.48497958377173  3.44451784578705 
## [31] 3.40217350273288  3.3577627814323   3.31107361281783 
## [34] 3.26185950714291  3.20983167673402  3.15464876785729 
## [37] 3.09590327428938  3.03310325630434  2.96564727304425 
## [40] 2.89278926071437  2.8135880922156   2.72683302786084 
## [43] 2.63092975357146  2.52371901428583  2.40217350273288 
## [46] 2.26185950714291  2.09590327428938  1.89278926071437 
## [49] 1.63092975357146  1.26185950714291  0.630929753571457
## [52] <NA>              NaN               NaN              
## [55] NaN               NaN               NaN              
## [58] NaN                NaN                NaN               
## [61] NaN                NaN               
## 51 Levels: 0.630929753571457 1.26185950714291 ... NaN
```

```
#as.factor(x) # Same
```

- Cast x to integers. What do we learn about R's infinity representation in the integer data type?

```
x
```

```
##  [1]           NA 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536           NA          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```
as.integer(x)
```

```
##  [1] NA  4  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3  3  3  3  3  3  3
## [24]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2
## [47]  2  1  1  1  0 NA NA NA NA NA NA NA NA NA NA NA
```

- Use x to create a new vector y containing only real numbers.

```
x
```

```
##  [1]           NA 4.1918065486 4.1734172519 4.1546487679 4.1354851290
##  [6] 4.1159093373 4.0959032743 4.0754475994 4.0545216381 4.0331032563
## [11] 4.0111687196 3.9886925350 3.9656472730 3.9420033664 3.9177288818
## [16] 3.8927892607 3.8671470235 3.8407614303 3.8135880922 3.7855785214
## [21] 3.7566796108 3.7268330279 3.6959745057 3.6640330099 3.6309297536
## [26] 3.5965770266 3.5608767950 3.5237190143 3.4849795838 3.4445178458
## [31] 3.4021735027 3.3577627814 3.3110736128 3.2618595071 3.2098316767
## [36] 3.1546487679 3.0959032743 3.0331032563 2.9656472730 2.8927892607
## [41] 2.8135880922 2.7268330279 2.6309297536 2.5237190143 2.4021735027
## [46] 2.2618595071 2.0959032743 1.8927892607 1.6309297536 1.2618595071
## [51] 0.6309297536           NA          NaN          NaN          NaN
## [56]          NaN          NaN          NaN          NaN          NaN
## [61]          NaN          NaN
```

```
y = x[is.finite(x)]
y
```

```
##  [1] 4.1918065486 4.1734172519 4.1546487679 4.1354851290 4.1159093373
##  [6] 4.0959032743 4.0754475994 4.0545216381 4.0331032563 4.0111687196
## [11] 3.9886925350 3.9656472730 3.9420033664 3.9177288818 3.8927892607
## [16] 3.8671470235 3.8407614303 3.8135880922 3.7855785214 3.7566796108
## [21] 3.7268330279 3.6959745057 3.6640330099 3.6309297536 3.5965770266
## [26] 3.5608767950 3.5237190143 3.4849795838 3.4445178458 3.4021735027
## [31] 3.3577627814 3.3110736128 3.2618595071 3.2098316767 3.1546487679
## [36] 3.0959032743 3.0331032563 2.9656472730 2.8927892607 2.8135880922
## [41] 2.7268330279 2.6309297536 2.5237190143 2.4021735027 2.2618595071
## [46] 2.0959032743 1.8927892607 1.6309297536 1.2618595071 0.6309297536
```

```
length(y) #Different size of vector
```

```
## [1] 50
```

- Use the left rectangle method to numerically integrate x^2 from 0 to 1 with rectangle size 1e-6.

```
sum(seq(0, 1 - 1e-6, by = 1e-6) ^ 2) * 1e-6
```

```
## [1] 0.3333328333
```

- Calculate the average of 100 realizations of standard Bernoullis in one line using the `sample` function.

```
mean(sample(c(0,1) ,size = 100, replace = TRUE))
```

```
## [1] 0.57
```

- Calculate the average of 500 realizations of Bernoullis with p = 0.9 in one line using the `sample` function.

```
#TO-DO
#Probability of 1 is 0.9
# realization_bernoullis = sample(c(0, 1),
#                                   size = 500,
#                                   replace = TRUE,
#                                   prob = c(0.1, 0.9))
# mean(realization_bernoullis)


mean(sample(c(0,1),size = 500, replace = TRUE, prob = c(0.1, 0.9)))
```

```
## [1] 0.902
```

- 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.

```
x_3 = sample(c("none", "infraction", "misdimeanor", "felony"),
             size = 100,
             replace = TRUE,
             prob = c(.25, .25, .25, .25))
factor(x_3)
```

```
##   [1] felony      none        infraction  none        infraction 
##   [6] misdimeanor felony      felony      none        infraction 
##  [11] felony      infraction  infraction  infraction  infraction 
##  [16] infraction  felony      none        infraction  none       
##  [21] none        infraction  misdimeanor infraction  misdimeanor
##  [26] felony      none        infraction  none        felony     
##  [31] felony      felony      misdimeanor none        infraction 
##  [36] felony      felony      felony      none        infraction 
##  [41] felony      none        none        felony      none       
##  [46] misdimeanor none        none        felony      none       
##  [51] infraction  infraction  misdimeanor felony      infraction 
##  [56] none        none        infraction  misdimeanor felony     
##  [61] none        none        misdimeanor none        felony     
##  [66] none        infraction  felony      infraction  felony     
##  [71] infraction  infraction  none        felony      infraction 
##  [76] none        misdimeanor infraction  misdimeanor infraction 
##  [81] infraction  misdimeanor infraction  none        none
##  [86] none        misdimeanor felony      felony      infraction
##  [91] none        misdimeanor felony      none        misdimeanor
##  [96] none        infraction  misdimeanor infraction  none
## Levels: felony infraction misdimeanor none
```

- Use `x_3` to create `x_3_bin`, a binary feature where 0 is no crime and 1 is any crime.

```
x_3
```

```
##   [1] "felony"      "none"        "infraction"  "none"        "infraction"
##   [6] "misdimeanor" "felony"      "felony"      "none"        "infraction"
##  [11] "felony"      "infraction"  "infraction"  "infraction"  "infraction"
##  [16] "infraction"  "felony"      "none"        "infraction"  "none"
##  [21] "none"        "infraction"  "misdimeanor" "infraction"  "misdimeanor"
##  [26] "felony"      "none"        "infraction"  "none"        "felony"
##  [31] "felony"      "felony"      "misdimeanor" "none"        "infraction"
##  [36] "felony"      "felony"      "felony"      "none"        "infraction"
##  [41] "felony"      "none"        "none"        "felony"      "none"
##  [46] "misdimeanor" "none"        "none"        "felony"      "none"
##  [51] "infraction"  "infraction"  "misdimeanor" "felony"      "infraction"
##  [56] "none"        "none"        "infraction"  "misdimeanor" "felony"
##  [61] "none"        "none"        "misdimeanor" "none"        "felony"
##  [66] "none"        "infraction"  "felony"      "infraction"  "felony"
##  [71] "infraction"  "infraction"  "none"        "felony"      "infraction"
##  [76] "none"        "misdimeanor" "infraction"  "misdimeanor" "infraction"
##  [81] "infraction"  "misdimeanor" "infraction"  "none"        "none"
##  [86] "none"        "misdimeanor" "felony"      "felony"      "infraction"
##  [91] "none"        "misdimeanor" "felony"      "none"        "misdimeanor"
##  [96] "none"        "infraction"  "misdimeanor" "infraction"  "none"
```

```
x_3_bin = ifelse(x_3 == "none", 1, 0)
x_3_bin
```

```
##   [1] 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0
##  [36] 0 0 0 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0
##  [71] 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1
```

- Use `x_3` to create `x_3_ord`, an ordered, nominal factor variable. Ensure the proper ordinal ordering.

```
sample_criminality= sample(c("none", "infraction", "misdimeanor", "felony"),
             size = 100,
             replace = TRUE)
x_3_ord = factor(sample_criminality, levels = c("none", "infraction", "misdimeanor", "felony"), ordered
x_3_ord
```

```
##   [1] infraction  infraction  misdimeanor infraction  none
##   [6] none        misdimeanor infraction  infraction  misdimeanor
##  [11] infraction  felony      misdimeanor misdimeanor felony
##  [16] felony      misdimeanor felony      infraction  none
##  [21] felony      misdimeanor none        infraction  none
##  [26] felony      felony      felony      none        felony
##  [31] none        misdimeanor felony      misdimeanor none
##  [36] none        infraction  misdimeanor none        misdimeanor
##  [41] felony      infraction  misdimeanor none        misdimeanor
##  [46] none        infraction  misdimeanor none        felony
##  [51] felony      misdimeanor none        none        felony
##  [56] none        none        felony      none        misdimeanor
##  [61] infraction  misdimeanor infraction  misdimeanor none       
##  [66] misdimeanor misdimeanor misdimeanor felony      none       
##  [71] none        infraction  infraction  infraction  none       
##  [76] misdimeanor felony      infraction  none        infraction 
##  [81] felony      none        none        infraction  misdimeanor
##  [86] none        felony      misdimeanor misdimeanor infraction 
##  [91] infraction  felony      none        infraction  infraction 
##  [96] infraction  felony      misdimeanor misdimeanor misdimeanor
## Levels: none < infraction < misdimeanor < felony
```