Week 4 - Simulation and Profiling

Contents

1 Week 4 - Simulation & Profiling

1.1 The str Function

str: Completly display the internal structyure of an R object

- A diagnostic function and an alternative to 'summary'
- It is especially well suited to compactly display the (abbreviated) contents of (spossibly nester) lists.
- Roughly one line per basic object

```
str(str)
```

```
## function (object, ...)
```

```
str(lm)
```

```
## function (formula, data, subset, weights, na.action, method = "qr", model = TRUE,
##     x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL,
##     offset, ...)
```

```
x <- rnorm(100, 2, 4)
x
```

```
##   [1] -0.312118937 -5.619057850  5.579319236  1.627084514  3.503555774
##   [6]  0.390286977  6.425381046  3.990880761  0.465288103  2.688933420
##  [11] -3.121574057  5.680190520  7.498814183  3.089195725  0.468557557
##  [16] -0.039917516  7.076667654  1.819032763 -1.231957795  4.520287313
##  [21]  3.421960250  8.486084712  0.190746615  3.981293725  1.383674291
##  [26]  1.771462789 -0.871515172  1.059194519 -2.149355055  4.809957138
```

```
##  [31] -1.428933559  1.434414787 10.798339162 -0.127084930  1.701156216
##  [36] -1.822315205  4.164725187  1.412761897 -0.225046458  1.963507251
##  [41]  3.947241612  0.596202773  2.197630211 -5.234225102  9.746092635
##  [46]  6.337932885 -2.524069429  6.948984557 -3.389341942 -4.900351570
##  [51]  2.491328206 -2.834012557 -1.422408791  1.271737992  4.728854357
##  [56]  2.864701878 -0.753596993  2.818317098 -1.403465517  2.182883484
##  [61] -3.033955523  1.175762721  5.551114354 -1.830031679 -2.247787266
##  [66] -0.040912799  7.085911228  1.036536831  2.443138750 -1.949937749
##  [71]  1.830434869 -0.003794044  5.276924444 -0.654973576  3.338336588
##  [76]  4.894377263 10.881200880 -3.713692907  3.262133305 -1.861118051
##  [81]  1.487341024 -5.703888709  2.723227018  0.661802059  4.256379364
##  [86]  9.566704891  3.660849535 -8.520101095 -0.307584178  7.085330783
##  [91]  7.367296356 -2.204555592  1.718733162 -4.276604005  0.683630326
##  [96]  3.438309299  6.221416728  2.582234689  1.803482284  2.775281874
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.5201 -0.6796  1.7099  1.7458  3.9837 10.8812
```

```
str(x)
```

```
##  num [1:100] -0.312 -5.619 5.579 1.627 3.504 ...
```

x is a numeric vector, has 100 elements in it, and gives first 5 elements.

```
f <- gl(40,10)
str(f)
```

```
##  Factor w/ 40 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(f)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40
## 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

```
library(datasets)
head(airquality)
```

Ozone	Solar.R	Wind	Temp	Month	Day
41	190	7.4	67	5	1
36	118	8.0	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
NA	NA	14.3	56	5	5
28	NA	14.9	66	5	6

```
str(airquality)
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
```

```
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
m <- matrix(rnorm(100,10,10))
str(m)
```

```
##  num [1:100, 1] -6.83 30.54 13.02 28.58 13.2 ...
```

```
m[,1]
```

```
##   [1]  -6.82960549  30.54448932  13.02225643  28.58175974  13.20034850
##   [6]  12.97357766  21.15839522  11.82066073   5.28502561  -2.99130548
##  [11]   9.08316115  13.32364707  12.55437859  25.23689638   8.48852401
##  [16]  27.83409110  18.73880340  13.37388569  16.19116858   6.06653675
##  [21]  16.86044202  16.16705404  -4.58937204  27.38157852  21.37339698
##  [26]  22.13949871  13.28522605 -11.47213296  31.35660943  12.29908547
##  [31]  19.38850721   1.12412871   1.98442147   8.29462010  27.31445678
##  [36]  14.57021409  17.90893256  29.12754369   1.62011188  10.11325888
##  [41]   4.98953924   6.93593045  12.48102326   8.74434830   2.24174967
##  [46]   8.20921439 -19.94311085  15.04918746   4.35813392  17.58128590
##  [51]   7.57843765  10.22159509 -10.47220024  18.44986894   2.33866377
##  [56]   1.61225869  17.48504958  10.03259578   1.82575791  13.56974138
##  [61]  19.18940255  -4.37655552   1.40846803   7.36040441   4.74428322
##  [66]  13.59284894  10.19343085   0.08415636  12.56929636 -19.33952803
##  [71]  -0.18713860   3.57434460  13.75478358   9.96742926 -10.54888991
##  [76]   2.63642191   8.47190541   7.07102279   2.27988553   1.62090934
##  [81]  -2.31157752   6.75099813  -6.84529039  10.57567174  23.25519801
##  [86]   3.48209372   3.36123761   9.31945279   8.32275083  15.80888430
##  [91]   5.22142479  12.21129691 -11.72670892 -15.43750412  21.95871969
##  [96]  27.54797458   9.41561152  25.25529292   0.34890560   2.72154893
```

```
s <- split(airquality, airquality$Month)
s
```

```
## $`5`
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
## 11     7      NA  6.9   74     5  11
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
## 15    18      65 13.2   58     5  15
## 16    14     334 11.5   64     5  16
## 17    34     307 12.0   66     5  17
## 18     6      78 18.4   57     5  18
## 19    30     322 11.5   68     5  19
## 20    11      44  9.7   62     5  20
## 21     1       8  9.7   59     5  21
```

```
## 22    11     320 16.6   73     5  22
## 23     4      25  9.7   61     5  23
## 24    32      92 12.0   61     5  24
## 25    NA      66 16.6   57     5  25
## 26    NA     266 14.9   58     5  26
## 27    NA      NA  8.0   57     5  27
## 28    23      13 12.0   67     5  28
## 29    45     252 14.9   81     5  29
## 30   115     223  5.7   79     5  30
## 31    37     279  7.4   76     5  31
##
## $`6`
##    Ozone Solar.R Wind Temp Month Day
## 32    NA     286  8.6   78     6   1
## 33    NA     287  9.7   74     6   2
## 34    NA     242 16.1   67     6   3
## 35    NA     186  9.2   84     6   4
## 36    NA     220  8.6   85     6   5
## 37    NA     264 14.3   79     6   6
## 38    29     127  9.7   82     6   7
## 39    NA     273  6.9   87     6   8
## 40    71     291 13.8   90     6   9
## 41    39     323 11.5   87     6  10
## 42    NA     259 10.9   93     6  11
## 43    NA     250  9.2   92     6  12
## 44    23     148  8.0   82     6  13
## 45    NA     332 13.8   80     6  14
## 46    NA     322 11.5   79     6  15
## 47    21     191 14.9   77     6  16
## 48    37     284 20.7   72     6  17
## 49    20      37  9.2   65     6  18
## 50    12     120 11.5   73     6  19
## 51    13     137 10.3   76     6  20
## 52    NA     150  6.3   77     6  21
## 53    NA      59  1.7   76     6  22
## 54    NA      91  4.6   76     6  23
## 55    NA     250  6.3   76     6  24
## 56    NA     135  8.0   75     6  25
## 57    NA     127  8.0   78     6  26
## 58    NA      47 10.3   73     6  27
## 59    NA      98 11.5   80     6  28
## 60    NA      31 14.9   77     6  29
## 61    NA     138  8.0   83     6  30
##
## $`7`
##    Ozone Solar.R Wind Temp Month Day
## 62   135     269  4.1   84     7   1
## 63    49     248  9.2   85     7   2
## 64    32     236  9.2   81     7   3
## 65    NA     101 10.9   84     7   4
## 66    64     175  4.6   83     7   5
## 67    40     314 10.9   83     7   6
## 68    77     276  5.1   88     7   7
## 69    97     267  6.3   92     7   8
```

```
## 70    97     272  5.7   92     7   9
## 71    85     175  7.4   89     7  10
## 72    NA     139  8.6   82     7  11
## 73    10     264 14.3   73     7  12
## 74    27     175 14.9   81     7  13
## 75    NA     291 14.9   91     7  14
## 76     7      48 14.3   80     7  15
## 77    48     260  6.9   81     7  16
## 78    35     274 10.3   82     7  17
## 79    61     285  6.3   84     7  18
## 80    79     187  5.1   87     7  19
## 81    63     220 11.5   85     7  20
## 82    16       7  6.9   74     7  21
## 83    NA     258  9.7   81     7  22
## 84    NA     295 11.5   82     7  23
## 85    80     294  8.6   86     7  24
## 86   108     223  8.0   85     7  25
## 87    20      81  8.6   82     7  26
## 88    52      82 12.0   86     7  27
## 89    82     213  7.4   88     7  28
## 90    50     275  7.4   86     7  29
## 91    64     253  7.4   83     7  30
## 92    59     254  9.2   81     7  31
##
## $`8`
##     Ozone Solar.R Wind Temp Month Day
## 93     39      83  6.9   81     8   1
## 94      9      24 13.8   81     8   2
## 95     16      77  7.4   82     8   3
## 96     78      NA  6.9   86     8   4
## 97     35      NA  7.4   85     8   5
## 98     66      NA  4.6   87     8   6
## 99    122     255  4.0   89     8   7
## 100    89     229 10.3   90     8   8
## 101   110     207  8.0   90     8   9
## 102    NA     222  8.6   92     8  10
## 103    NA     137 11.5   86     8  11
## 104    44     192 11.5   86     8  12
## 105    28     273 11.5   82     8  13
## 106    65     157  9.7   80     8  14
## 107    NA      64 11.5   79     8  15
## 108    22      71 10.3   77     8  16
## 109    59      51  6.3   79     8  17
## 110    23     115  7.4   76     8  18
## 111    31     244 10.9   78     8  19
## 112    44     190 10.3   78     8  20
## 113    21     259 15.5   77     8  21
## 114     9      36 14.3   72     8  22
## 115    NA     255 12.6   75     8  23
## 116    45     212  9.7   79     8  24
## 117   168     238  3.4   81     8  25
## 118    73     215  8.0   86     8  26
## 119    NA     153  5.7   88     8  27
## 120    76     203  9.7   97     8  28
```

```
## 121   118     225  2.3   94     8  29
## 122    84     237  6.3   96     8  30
## 123    85     188  6.3   94     8  31
##
## $`9`
##     Ozone Solar.R Wind Temp Month Day
## 124    96     167  6.9   91     9   1
## 125    78     197  5.1   92     9   2
## 126    73     183  2.8   93     9   3
## 127    91     189  4.6   93     9   4
## 128    47      95  7.4   87     9   5
## 129    32      92 15.5   84     9   6
## 130    20     252 10.9   80     9   7
## 131    23     220 10.3   78     9   8
## 132    21     230 10.9   75     9   9
## 133    24     259  9.7   73     9  10
## 134    44     236 14.9   81     9  11
## 135    21     259 15.5   76     9  12
## 136    28     238  6.3   77     9  13
## 137     9      24 10.9   71     9  14
## 138    13     112 11.5   71     9  15
## 139    46     237  6.9   78     9  16
## 140    18     224 13.8   67     9  17
## 141    13      27 10.3   76     9  18
## 142    24     238 10.3   68     9  19
## 143    16     201  8.0   82     9  20
## 144    13     238 12.6   64     9  21
## 145    23      14  9.2   71     9  22
## 146    36     139 10.3   81     9  23
## 147     7      49 10.3   69     9  24
## 148    14      20 16.6   63     9  25
## 149    30     193  6.9   70     9  26
## 150    NA     145 13.2   77     9  27
## 151    14     191 14.3   75     9  28
## 152    18     131  8.0   76     9  29
## 153    20     223 11.5   68     9  30
```

```
str(s)
```

```
## List of 5
##  $ 5:'data.frame':   31 obs. of  6 variables:
##   ..$ Ozone  : int [1:31] 41 36 12 18 NA 28 23 19 8 NA ...
##   ..$ Solar.R: int [1:31] 190 118 149 313 NA NA 299 99 19 194 ...
##   ..$ Wind   : num [1:31] 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##   ..$ Temp   : int [1:31] 67 72 74 62 56 66 65 59 61 69 ...
##   ..$ Month  : int [1:31] 5 5 5 5 5 5 5 5 5 5 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 6:'data.frame':   30 obs. of  6 variables:
##   ..$ Ozone  : int [1:30] NA NA NA NA NA NA 29 NA 71 39 ...
##   ..$ Solar.R: int [1:30] 286 287 242 186 220 264 127 273 291 323 ...
##   ..$ Wind   : num [1:30] 8.6 9.7 16.1 9.2 8.6 14.3 9.7 6.9 13.8 11.5 ...
##   ..$ Temp   : int [1:30] 78 74 67 84 85 79 82 87 90 87 ...
##   ..$ Month  : int [1:30] 6 6 6 6 6 6 6 6 6 6 ...
##   ..$ Day    : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 7:'data.frame':   31 obs. of  6 variables:
```

```
##   ..$ Ozone  : int [1:31] 135 49 32 NA 64 40 77 97 97 85 ...
##   ..$ Solar.R: int [1:31] 269 248 236 101 175 314 276 267 272 175 ...
##   ..$ Wind   : num [1:31] 4.1 9.2 9.2 10.9 4.6 10.9 5.1 6.3 5.7 7.4 ...
##   ..$ Temp   : int [1:31] 84 85 81 84 83 83 88 92 92 89 ...
##   ..$ Month  : int [1:31] 7 7 7 7 7 7 7 7 7 7 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 8:'data.frame':   31 obs. of  6 variables:
##   ..$ Ozone  : int [1:31] 39 9 16 78 35 66 122 89 110 NA ...
##   ..$ Solar.R: int [1:31] 83 24 77 NA NA NA 255 229 207 222 ...
##   ..$ Wind   : num [1:31] 6.9 13.8 7.4 6.9 7.4 4.6 4 10.3 8 8.6 ...
##   ..$ Temp   : int [1:31] 81 81 82 86 85 87 89 90 90 92 ...
##   ..$ Month  : int [1:31] 8 8 8 8 8 8 8 8 8 8 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 9:'data.frame':   30 obs. of  6 variables:
##   ..$ Ozone  : int [1:30] 96 78 73 91 47 32 20 23 21 24 ...
##   ..$ Solar.R: int [1:30] 167 197 183 189 95 92 252 220 230 259 ...
##   ..$ Wind   : num [1:30] 6.9 5.1 2.8 4.6 7.4 15.5 10.9 10.3 10.9 9.7 ...
##   ..$ Temp   : int [1:30] 91 92 93 93 87 84 80 78 75 73 ...
##   ..$ Month  : int [1:30] 9 9 9 9 9 9 9 9 9 9 ...
##   ..$ Day    : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
```

1.2 Simulation - Generating Random Numbers

Functions for probability distributions in R

· rnorm: generate random Normal variates with a given mean and standard deviation · dnorm: evaluate the Normal probability density (with a given mean/SD) at a point (or vector of points) · pnorm: evaluate the cumulative distribution function for a Normal distribution · rpois: generate random Poisson variates with a given rate

Probability distribution functions usually have four functions associated with them. The functions are prefixed with a · d for density · r for random number generation · p for cumulative distribution · q for quantile function

Working with the Normal distributions requires using these four functions

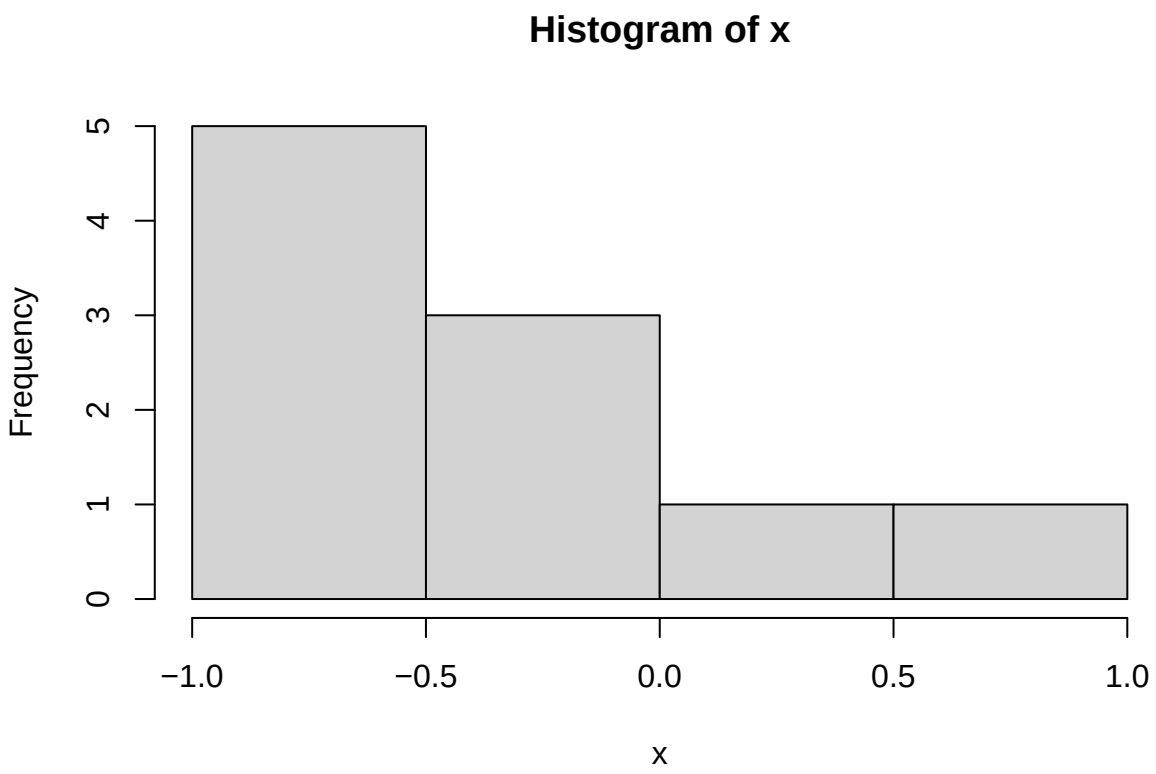
```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

If Φ is the cumulative distribution function for a standard Normal distribution, then pnorm(q) = Φ(q) and qnorm(p) = Φ−1 (p).

```
x <- rnorm(10)
x
```

```
##  [1] -0.77718098 -0.43616995  0.62084419  0.32032108 -0.90479986 -0.63977207
##  [7] -0.09944947 -0.94166052 -0.68483322 -0.11472052
```

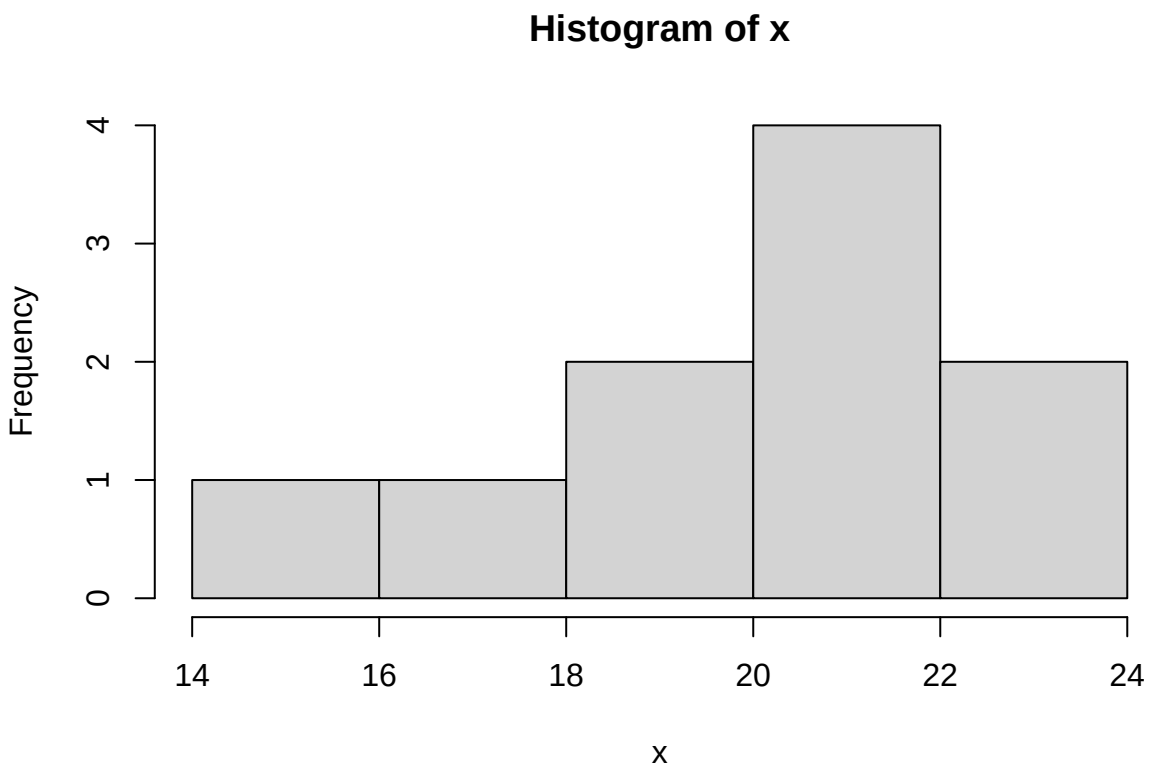
```r
hist(x)
```




```r
x <- rnorm(10,20,2)
x
```

```
##  [1] 20.61342 21.52248 23.13157 18.95772 21.57169 23.45541 15.85809 19.52111
##  [9] 21.35115 17.29110
```

```r
hist(x)
```




```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.86   19.10   20.98   20.33   21.56   23.46
```

Setting the random number seed with set.seed ensures reproducibility

```
set.seed(1)
rnorm(5)
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

```
rnorm(5)
```

```
## [1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.3053884
```

```
set.seed(1)
rnorm(5)
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

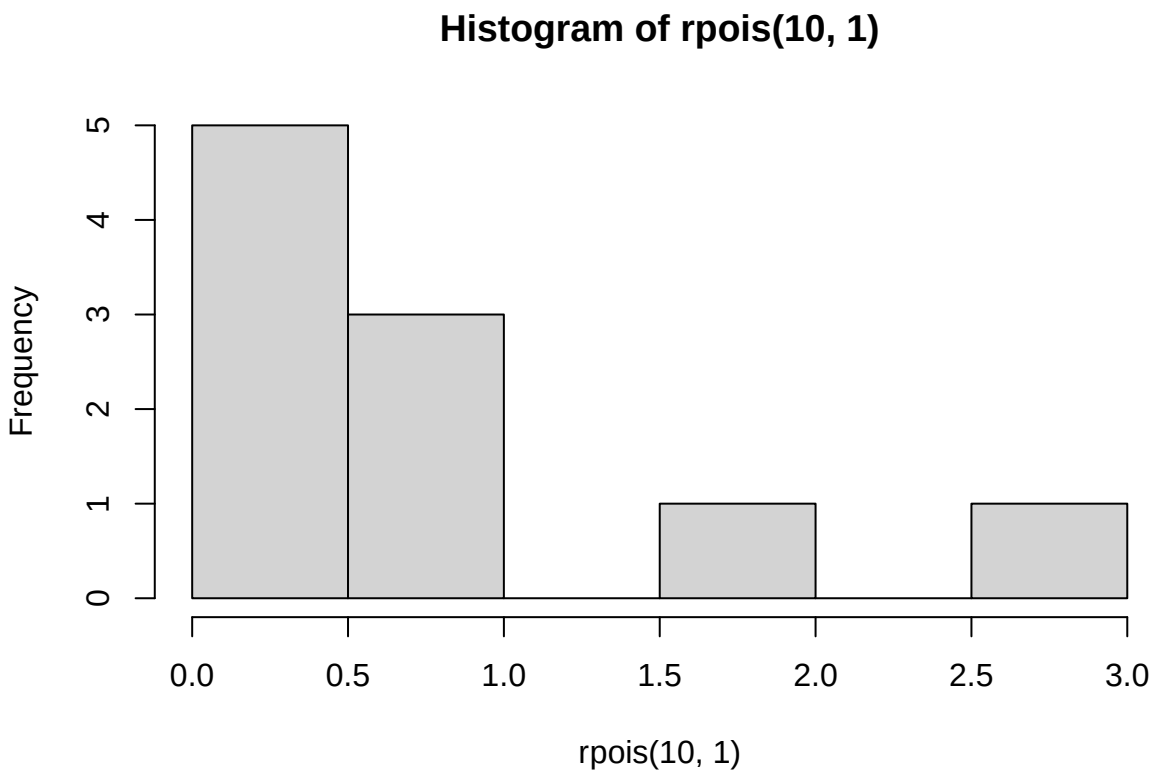
Always set the random number seed when conducting a simulation!

Generating Poisson data

```
rpois(10, 1)
```

```
##  [1] 0 0 1 1 2 1 1 4 1 2
```

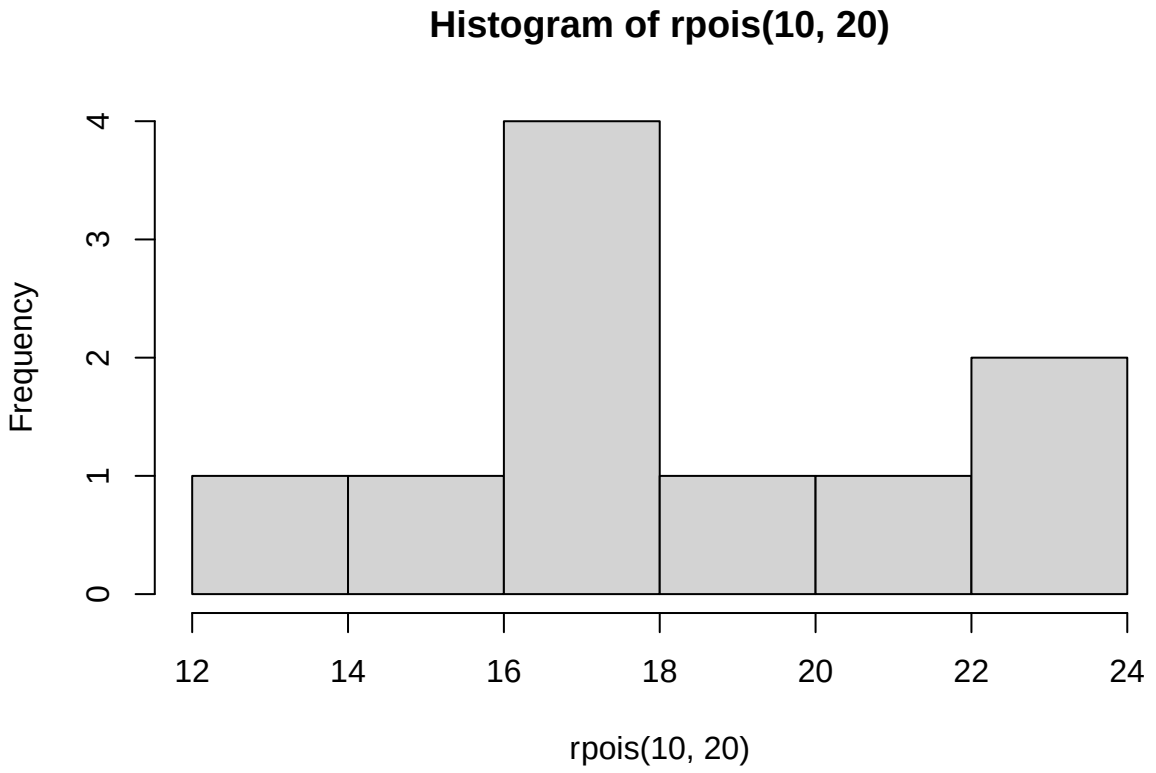
```
hist(rpois(10, 1))
```




```
rpois(10, 20)
```

```
##  [1] 19 19 24 23 22 24 23 20 11 22
```

```
hist(rpois(10, 20))
```



```r
ppois(2, 2) ## Cumulative distribution
```

```
## [1] 0.6766764
```

Pr(x <= 2)

```r
ppois(6, 2)
```

```
## [1] 0.9954662
```

Pr(x <= 6)

```r
ppois(4, 2)
```

```
## [1] 0.947347
```

Pr(x <= 4)

1.2.1 Generating Random Numbers From a Linear Model

Suppose we want to simulate from the following linear model

y= 0 + 1x+

where N(0,22). Assume x N(0,12), 0 = 0.5 and 1 = 2.

```r
set.seed(20)
x <- rnorm(100)
e <- rnorm(100, 0, 2)
y<-0.5+2*x+e
```

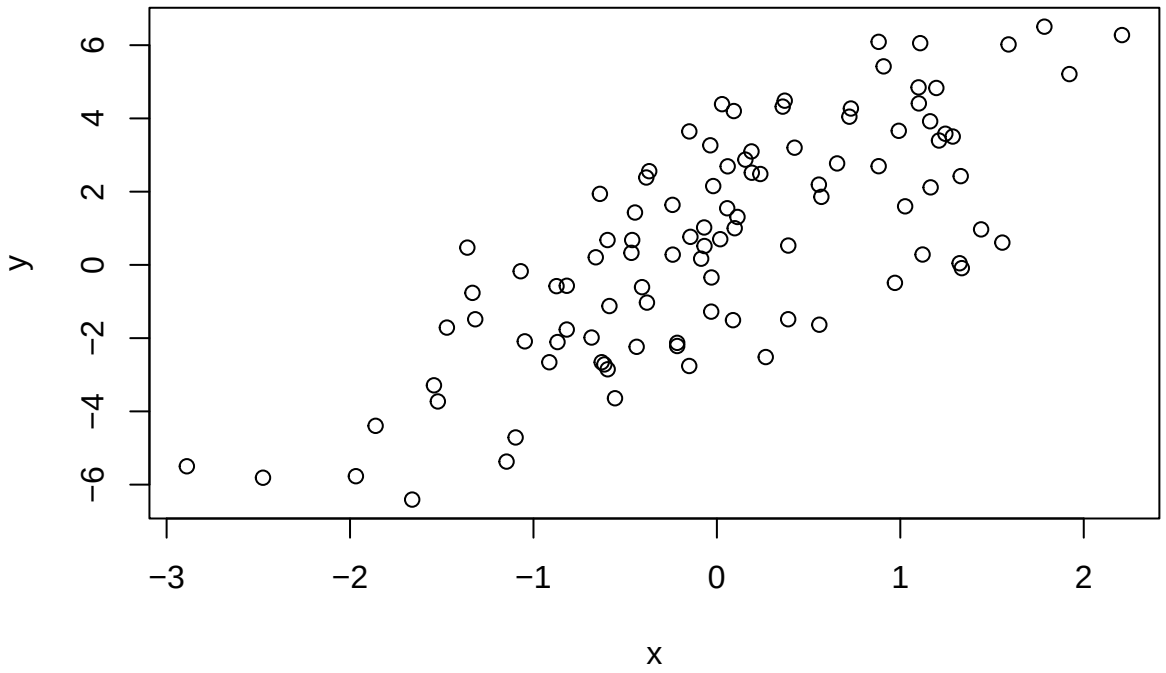
```r
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.4084 -1.5402  0.6789  0.6893  2.9303  6.5052
```

```r
y
```

```
##   [1]  3.92145671 -1.12306154  6.50516206 -0.76334657  1.43080733  1.85824578
##   [7] -5.49942000 -2.10701153  0.67867568 -3.64118829  2.15300374  3.64495611
##  [13] -2.65751679  0.04553572 -3.72883351 -2.23668198 -0.49039654  4.38887620
##  [19]  0.16890419  0.52829153  2.48205656  0.76639727  4.04713057  4.48476280
##  [25]  1.63952492 -1.71150396  0.67914445 -5.37113780 -5.81140592 -2.71828488
##  [31] -2.12810590  6.02056165  0.60895238  6.05517659 -4.71161217 -4.39467489
##  [37] -2.65861895  3.58184381 -1.50917314  3.20076678 -1.76509940 -3.28807288
##  [43]  2.19238717  2.55958594 -2.08720395  0.70189574  2.69534679  6.09139163
##  [49]  1.59929020 -1.02936781  4.84828910 -1.27631332  2.51707303 -0.08880639
##  [55]  4.27076653  1.54675806  2.42362336 -0.60750392 -0.57008476  4.32189399
##  [61]  2.69160962  1.02230623  2.38639291 -0.58044205  4.83151620 -6.40843800
##  [67]  2.11701166 -0.17304283  5.42106533 -1.48483428  2.77113330  0.20767705
##  [73]  3.66166652 -1.98363440  0.28360591 -1.48370020 -2.76032486  0.32931957
##  [79]  0.47189300 -0.34351008  0.28078066  5.21035478  1.00142931  4.20158959
##  [85]  3.09853964  1.93979202  3.26573037  1.30890176  3.39689480  0.96975660
##  [91]  6.27609452 -5.76983364 -1.63342938 -2.21466670  0.51530957 -2.51624067
##  [97]  3.50519076  4.40940115 -2.85087510  2.87426339
```

```r
plot(x, y)
```



What if x is binary?

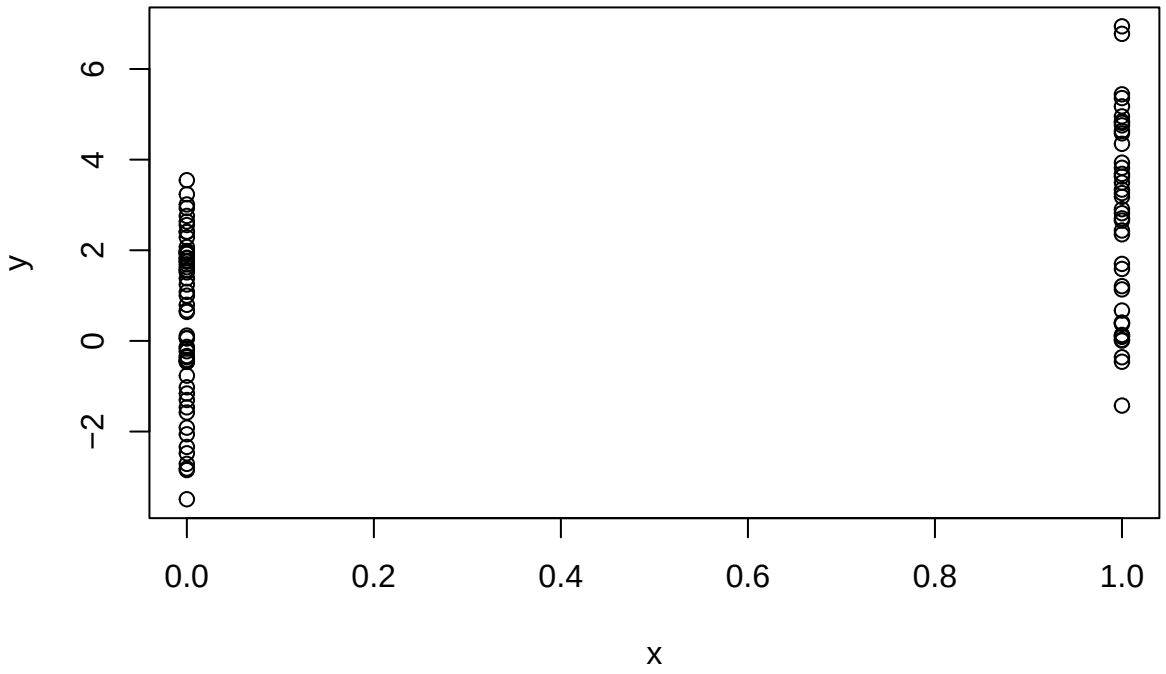
```
set.seed(10)
x <- rbinom(100, 1, 0.5)
e <- rnorm(100, 0, 2)
y<-0.5+2*x+e
y
```

```
##   [1]  1.698724906 -0.169113130  3.235907906  6.775534207  1.511638529
##   [6]  2.072684768 -1.304423888  1.565793985  1.208211493  1.081974977
##  [11]  0.024811062  1.587647450 -1.160645309  3.180231287  2.632752791
##  [16]  2.932251676  1.971381315 -0.462417235  1.625489526  0.007360576
##  [21]  3.261844425 -0.360854506  0.403108990  0.062992899 -2.479872473
##  [26]  4.845412562 -0.459654043 -0.360775632  0.396722716  3.545172688
##  [31]  3.685656109  0.054676982  1.925788552  3.933201667  1.380483729
##  [36]  2.817661243  3.819528277  6.941039326  0.132109852  2.352088331
##  [41] -0.332709350  0.117035312  0.639089563  4.810696636  1.689914694
##  [46] -2.339290217 -2.713354491  2.285851799  0.796335910  4.954056780
##  [51] -1.023608678  3.338750812 -1.579886729  1.923147932 -0.766426030
##  [56]  3.626349329  1.821973372 -2.816101715  2.556335954  4.755907228
##  [61] -2.060309207  2.757736455 -0.428269054 -0.131520419  4.348586294
##  [66]  2.654289448  4.579847210  1.983772413  3.011089717  2.401837933
##  [71] -0.462731215  2.905763556  2.436520512  0.108839399  1.747362474
##  [76]  0.670391033  0.997516015  0.374754414 -0.227964494 -1.913989707
##  [81]  5.358425563  1.766871782 -3.493631235  1.136335654 -0.420110959
##  [86] -1.466138388  3.490663426  1.951635000  1.834597464  2.409572873
##  [91] -2.850664359  0.089629215 -1.426504978  5.441504620  1.244944677
##  [96]  4.631758668  1.561299737  2.703966892  5.175564932  0.674469537
```

```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.4936 -0.1409  1.5767  1.4322  2.8397  6.9410
```

```
plot(x,y)
```



Suppose we want to simulate from a Poisson model where

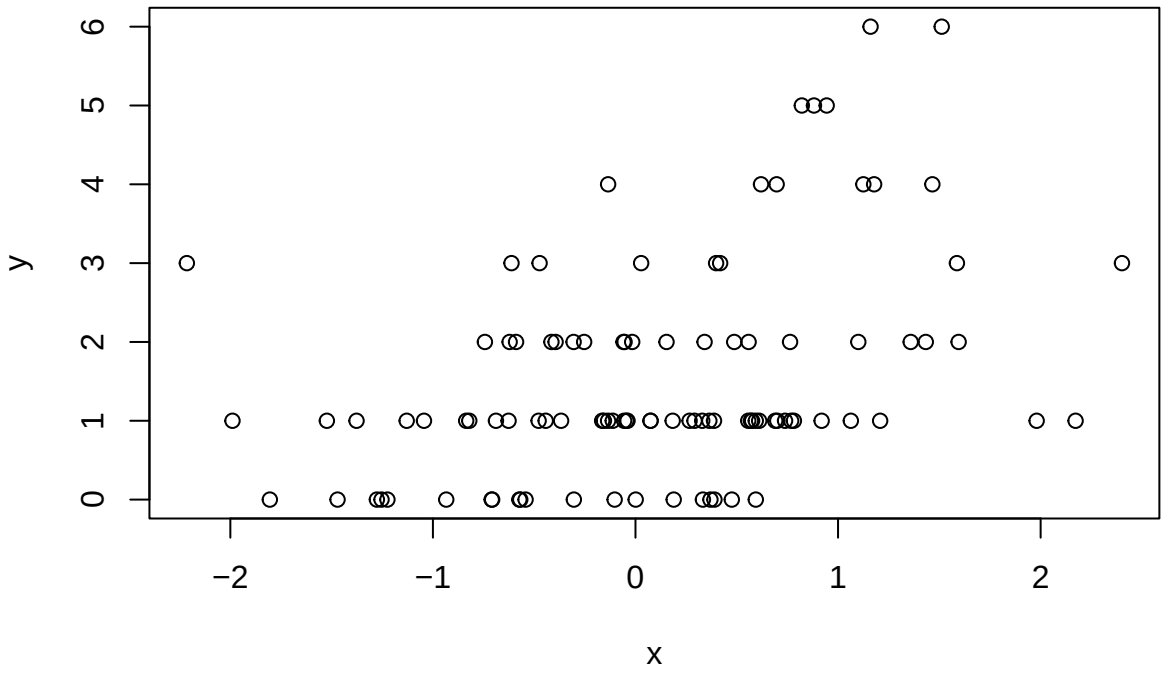
Y ~ Poisson( ) log = 0 + 1x

and 0 = 0.5 and 1 = 0.3. We need to use the rpois function for this

```
set.seed(1)
x <- rnorm(100)
log.mu<-0.5+0.3*x
y <- rpois(100, exp(log.mu))
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    1.00    1.55    2.00    6.00
```

```
plot(x,y)
```



1.2.2 Random Sampling

The sample function draws randomly from a specified set of (scalar) objects allowing you to sample from arbitrary distributions.

```
set.seed(1)
sample(1:10, 4)
```

```
## [1] 9 4 7 1
```

```
sample(1:10, 4)
```

```
## [1] 2 7 3 6
```

```
sample(letters, 5)
```

```
## [1] "r" "s" "a" "u" "w"
```

```
sample(1:10) ## permutation
```

```
##  [1] 10  6  9  2  1  5  8  4  3  7
```

```
sample(1:10)
```

```
##  [1]  5 10  2  8  6  1  4  3  9  7
```

```
sample(1:10, replace = TRUE) ## Sample w/replacement
```

```
##  [1]  3  6 10 10  6  4  4 10  9  7
```

Summary · Drawing samples from specific probability distributions can be done with r* functions · Standard distributions are built in: Normal, Poisson, Binomial, Exponential, Gamma, etc. · The sample function can be used to draw random samples from arbitrary vectors · Setting the random number generator seed via set.seed is critical for reproducibility

1.3 R Profiler

1.3.1 Why is My Code So Slow?

Profiling is a systematic way to examine how much time is spend in different parts of a program · Useful when trying to optimize your code · Often code runs fine once, but what if you have to put it in a loop for 1,000 iterations? Is it still fast enough? · Profiling is better than guessing

1.3.2 On Optimizing Your Code

· Getting biggest impact on speeding up code depends on knowing where the code spends most of its time · This cannot be done without performance analysis or profiling

We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil –Donald Knuth

1.3.3 General Principles of Optimization

· Design first, then optimize · Remember: Premature optimization is the root of all evil · Measure (collect data), don't guess. · If you're going to be scientist, you need to apply the same principles here!

1.3.4 Using system.time()

· Takes an arbitrary R expression as input (can be wrapped in curly braces) and returns the amount of time taken to evaluate the expression · Computes the time (in seconds) needed to execute an expression - Ifthere'sanerror,givestimeuntiltheerroroccurred · Returns an object of class proc_time - user-time:timechargedtotheCPU(s)forthisexpression - elapsedtime:"wallclock"time

· Usually, the user time and elapsed time are relatively close, for straight computing tasks · Elapsed time may be greater than user time if the CPU spends a lot of time waiting around · Elapsted time may be smaller than the user time if your machine has multiple cores/processors (and is capable of using them) - Multi-threadedBLASlibraries(vecLib/Accelerate,ATLAS,ACML,MKL) - Parallelprocessingviatheparallelpackage

```
## Elapsed time > user time
system.time(readLines("http://www.jhsph.edu"))
```

```
##    user  system elapsed
##   0.045   0.010   2.877
```

```
## Elapsed time < user time
hilbert <- function(n) {
  i <- 1:n
  1 / outer(i - 1, i, "+")
}

x <- hilbert(1000)
system.time(svd(x))
```

```
##    user  system elapsed
##   1.828   0.013   1.842
```

1.3.5 Timing Longer Expressions

```
system.time({
  n <- 1000
  r <- numeric(n)
  for (i in 1:n) {
    x <- rnorm(n)
```

```
    r[i] <- mean(x)
  }
})
```

```
##    user  system elapsed
##   0.040   0.001   0.042
```

`user`: The CPU time charged for the execution of user instructions of the calling process. This is the time spent by the CPU executing the code.

`system`: The CPU time charged for execution by the system on behalf of the calling process. This often refers to time spent on system-level tasks such as memory allocation.

`elapsed`: The total elapsed (wall-clock) time in seconds. This is essentially the real-world time it took for the code to run start-to-finish. It's usually the sum of user and system times, but can be greater due to reasons like if your process is running on a multitasking system and gets paused to allow other tasks to run.

1.3.6 Beyond system.time()

- Using system.time() allows you to test certain functions or code blocks to see if they are taking excessive amounts of time
- Assumes you already know where the problem is and can call system.time() on it
- What if you don't know where to start?

1.3.7 The R Profiler

· The Rprof() function starts the profiler in R - Rmustbecompiledwithprofilersupport(butthisisusuallythecase) · The summaryRprof() function summarizes the output from Rprof() (otherwise it's not readable) · DO NOT use system.time() and Rprof() together or you will be sad

· Rprof() keeps track of the function call stack at regularly sampled intervals and tabulates how much time is spend in each function · Default sampling interval is 0.02 seconds · NOTE: If your code runs very quickly, the profiler is not useful, but then you probably don't need it in that case

```
##lm(y~x)

sample.interval=10000
```

1.3.8 Using summaryRprof()

· The summaryRprof() function tabulates the R profiler output and calculates how much time is spend in which function · There are two methods for normalizing the data · "by.total" divides the time spend in each function by the total run time · "by.self" does the same but first subtracts out time spent in functions above in the call stack

1.3.9 By Total

```
$by.total
```

```
## Error: <text>:1:1: unexpected '$'
## 1: $
##     ^
```

```
$by.self
```

```
## Error: <text>:1:1: unexpected '$'
## 1: $
```

```
##     ^
```

```
sample.interval
```

```
## [1] 10000
```

Summary · Rprof() runs the profiler for performance of analysis of R code · summaryRprof() summarizes the output of Rprof() and gives percent of time spent in each function (with two types of normalization) · Good to break your code into functions so that the profiler can give useful information about where time is being spent · C or Fortran code is not profiled