03-05 Sampling

1 - Purpose

- Creating a uniform random sample
- Creating a normal sample
- Creating repeatable pseudo-random values
- Saving and loading list objects

2 - Concepts

3 - Script for this lesson

You can get the script for this lesson here. The lesson goes in order on the script. You can comment and uncomment multiple lines in the script using Control-Shift-C in R (Command-Shift-C on a MAC).

4 - Uniform random sampling

There are times when you want to take random samples from a large database of values, like temperatures from multiple years and locations. One possible reason is to evaluate a suite of potential multiple potential weather scenarios. In this example, we will take random values without any weighting. In other words, all values have the same probability of being chosen.

4.1 - sample of numbers

We can use the ***sample()*** function to extract a random sample from a set of values.

sample() has three parameters:

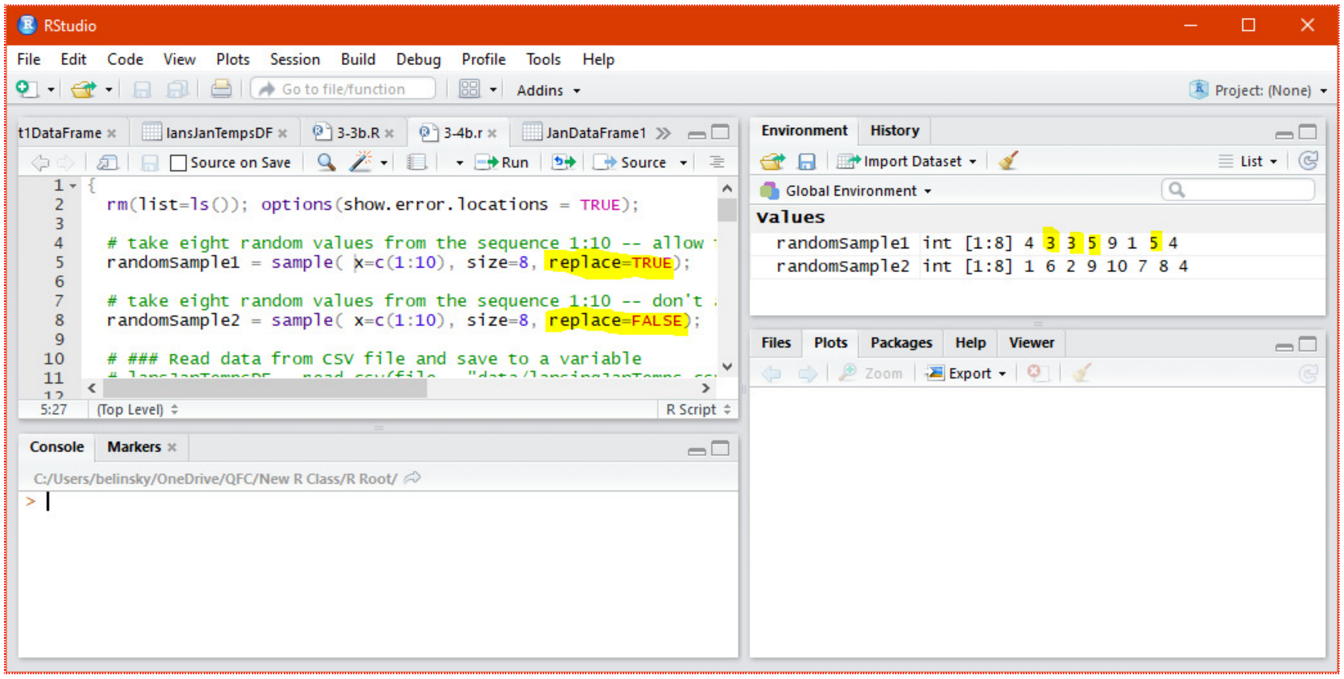
- ***x:*** the values we are taking a random sample from (in this case, a sequence vector)
- ***size***: The number of samples we will take.
- ***replace***: Whether we allow for a value to be sampled multiple times (***replace=TRUE*** means we allow resampling of the same value)

In the following code, the parameters for ***sample()*** are set so that:

- The data set we are using is the sequence 1-10: ***c(1:10)***
- We are taking eight random samples: ***size***=8
- We allow repeat values (***replace***) in ***randomSample1*** but not in ***randomSample2***

```
randomSample1 = sample( c(1:10), size=8, replace=TRUE);
randomSample2 = sample( c(1:10), size=8, replace=FALSE);
```

In *Fig 1*, ***randomSample1*** has two values that were repeated (3 and 5), whereas ***randomSample2*** has no repeated values. *Running the script again will produce different values* but ***randomSample2*** will always have unique values because ***replace=FALSE***.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  # take eight random values from the sequence 1:10 -- allow
  randomSample1 = sample( x=c(1:10), size=8, replace=TRUE);

  # take eight random values from the sequence 1:10 -- don't
  randomSample2 = sample( x=c(1:10), size=8, replace=FALSE);

  # ### Read data from CSV file and save to a variable
```

```
Values
  randomSample1  int [1:8] 4 3 3 5 9 1 5 4
  randomSample2  int [1:8] 1 6 2 9 10 7 8 4
```

Fig 1: Random sampling of a vector with replacement and without replacement

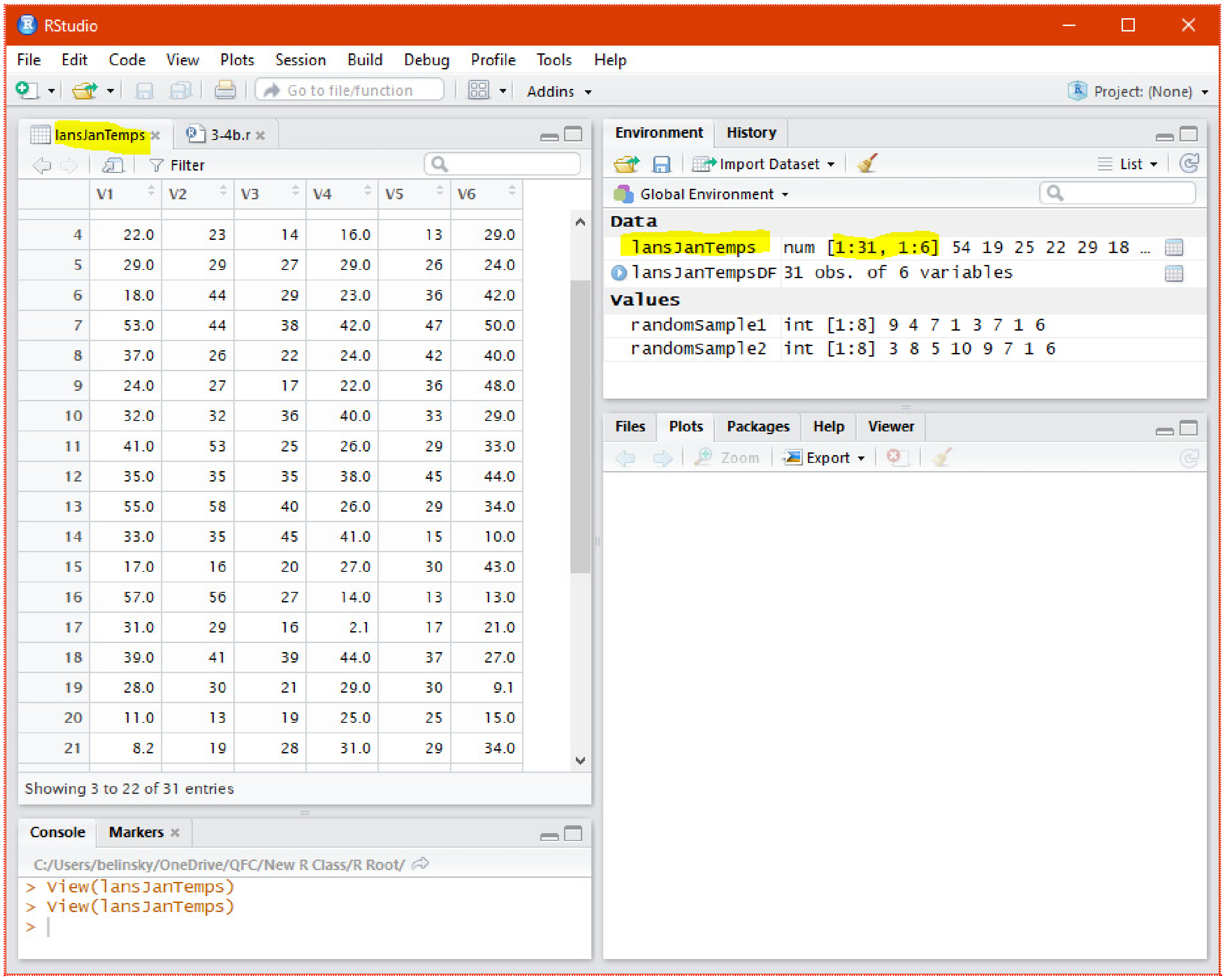
4.2 - Sampling a data vector

We can also use ***sample()*** to choose random values from a matrix. Remember that *a matrix is a reshaped vector with rows and columns*. In this case, we will get random values from the January temperature matrix from the previous lessons.

First we will get the data from the CSV file we create in lesson 3-2 and then we need to convert the data frame into a matrix.

```
lansJanTempsDF = read.csv(file = "data/lansingJanTemps.csv");
lansJanTemps = as.matrix(x=lansJanTempsDF);
```

Remember that ***lansJanTemps*** contain 6 years worth of January temperature. So, 31*6 = 186 temperature values in all.



	V1	V2	V3	V4	V5	V6
4	22.0	23	14	16.0	13	29.0
5	29.0	29	27	29.0	26	24.0
6	18.0	44	29	23.0	36	42.0
7	53.0	44	38	42.0	47	50.0
8	37.0	26	22	24.0	42	40.0
9	24.0	27	17	22.0	36	48.0
10	32.0	32	36	40.0	33	29.0
11	41.0	53	25	26.0	29	33.0
12	35.0	35	35	38.0	45	44.0
13	55.0	58	40	26.0	29	34.0
14	33.0	35	45	41.0	15	10.0
15	17.0	16	20	27.0	30	43.0
16	57.0	56	27	14.0	13	13.0
17	31.0	29	16	2.1	17	21.0
18	39.0	41	39	44.0	37	27.0
19	28.0	30	21	29.0	30	9.1
20	11.0	13	19	25.0	25	15.0
21	8.2	19	28	31.0	29	34.0

Showing 3 to 22 of 31 entries

```
Data
  lansJanTemps     num [1:31, 1:6] 54 19 25 22 29 18 ...
  lansJanTempsDF 31 obs. of 6 variables
Values
  randomSample1  int [1:8] 9 4 7 1 3 7 1 6
  randomSample2  int [1:8] 3 8 5 10 9 7 1 6
```

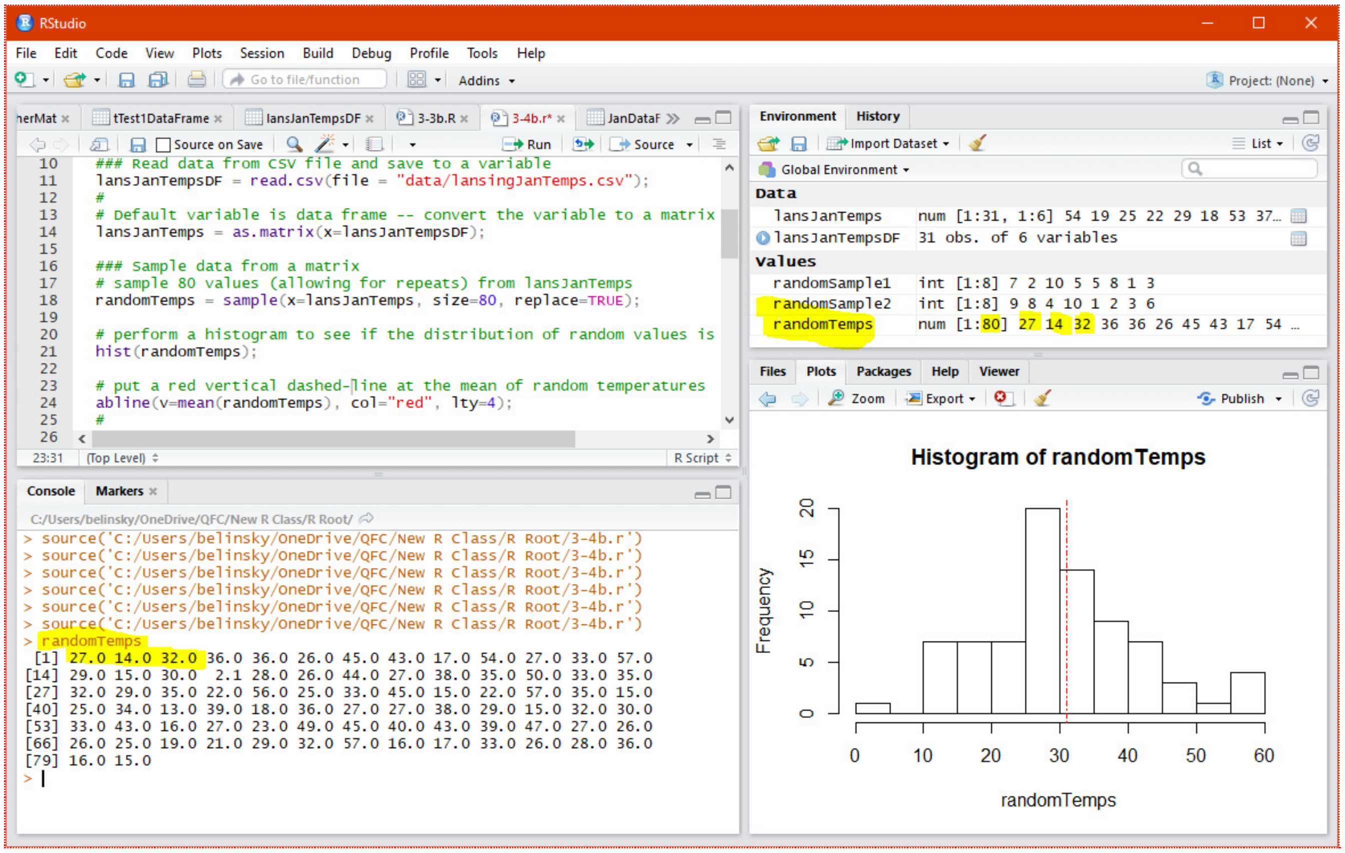
```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> View(lansJanTemps)
> View(lansJanTemps)
> 
```

*Fig 2: The **lansJanTemp** matrix with six years of January temperatures.*

We will take 80 random samples from ***lansJanTemps*** allowing for repeats and create a histogram of the random values. We will also add a vertical line to the histogram that represents the mean value.

```
randomTemps = sample(x=lansJanTemps, size=80, replace=TRUE);
hist(x=randomTemps);
abline(v=mean(randomTemps), col="red", lty=4);
```

Note: all 80 values in ***randomTemps*** can be displayed in the Console Window by typing "lansJanTemps " and hitting Enter.



```
### Read data from CSV file and save to a variable
lansJanTempsDF = read.csv(file = "data/lansingJanTemps.csv");
#
# Default variable is data frame -- convert the variable to a matrix
lansJanTemps = as.matrix(x=lansJanTempsDF);

### Sample data from a matrix
# sample 80 values (allowing for repeats) from lansJanTemps
randomTemps = sample(x=lansJanTemps, size=80, replace=TRUE);

# perform a histogram to see if the distribution of random values is
hist(randomTemps);

# put a red vertical dashed-line at the mean of random temperatures
abline(v=mean(randomTemps), col="red", lty=4);
#
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> randomTemps
 [1] 27.0 14.0 32.0 36.0 36.0 26.0 45.0 43.0 17.0 54.0 27.0 33.0 57.0
[14] 29.0 15.0 30.0  2.1 28.0 26.0 44.0 27.0 38.0 35.0 50.0 33.0 35.0
[27] 32.0 29.0 35.0 22.0 56.0 25.0 33.0 45.0 15.0 22.0 57.0 35.0 15.0
[40] 25.0 34.0 13.0 39.0 18.0 36.0 27.0 27.0 38.0 29.0 15.0 32.0 30.0
[53] 33.0 43.0 16.0 27.0 23.0 49.0 45.0 40.0 43.0 39.0 47.0 27.0 26.0
[66] 26.0 25.0 19.0 21.0 29.0 32.0 57.0 16.0 17.0 33.0 26.0 28.0 36.0
[79] 16.0 15.0
> 
```

```
Data
  lansJanTemps     num [1:31, 1:6] 54 19 25 22 29 18 53 37...
  lansJanTempsDF   31 obs. of 6 variables
Values
  randomSample1    int [1:8] 7 2 10 5 5 8 1 3
  randomSample2    int [1:8] 9 8 4 10 1 2 3 6
  randomTemps      num [1:80] 27 14 32 36 36 26 45 43 17 54 ...
```

*Fig 3: Random sampling of temperature values from the **lansJanTemps** matrix*

5 - Sampling from a normal distribution

sample() creates a random sample where each value has the same chance of being picked (i.e., uniform probability). If we want to create a set of values that comes from a normal distribution with a specified mean and variance we use the function ***rnorm()***.

rnorm() has 3 parameters:

- ***n***: the number of values to sample
- ***mean***: the mean of the distribution
- ***sd***: the standard deviation of the distribution

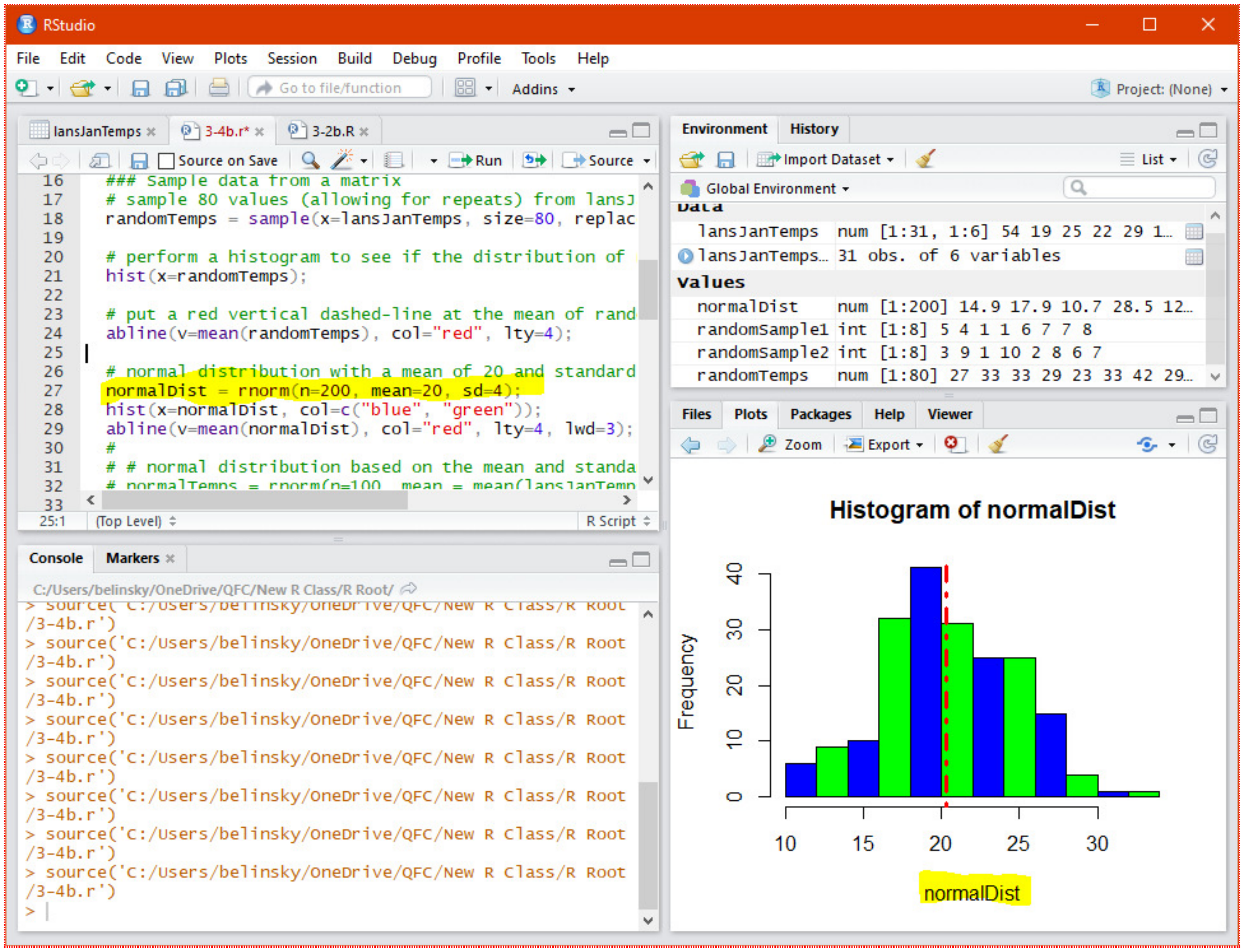
The following code samples and summarizes **200** values from a normal distribution with a mean of **20** and a standard deviation of **4**.

```
normalDist = rnorm(n=200, mean=20, sd=4);
```

The values in ***normalDist*** can be display visually in a histogram using ***hist()***. I have also put a vertical line in to represent the mean value of the ***normalDist*** using ***abline()***:

```
hist(x=normalDist, col=c("blue", "green"));
abline(v=mean(normalDist), col="red", lty=4, lwd=3); # line type, line width
```

The histogram (*Fig 4*) shows a fairly normal distribution of values and the mean is close to 20. Of course, the more samples you use, the more likely the distribution will look normal and the mean will be at 20.



```
### Sample data from a matrix
# sample 80 values (allowing for repeats) from lansJ
randomTemps = sample(x=lansJanTemps, size=80, replac

# perform a histogram to see if the distribution of
hist(x=randomTemps);

# put a red vertical dashed-line at the mean of rand
abline(v=mean(randomTemps), col="red", lty=4);

# normal distribution with a mean of 20 and standard
normalDist = rnorm(n=200, mean=20, sd=4);
hist(x=normalDist, col=c("blue", "green"));
abline(v=mean(normalDist), col="red", lty=4, lwd=3);
#
# # normal distribution based on the mean and standa
# normalTemps = rnorm(n=100, mean = mean(lansJanTemp
```

```
Data
  lansJanTemps      num [1:31, 1:6] 54 19 25 22 29 1…
  lansJanTemps…     31 obs. of 6 variables
Values
  normalDist        num [1:200] 14.9 17.9 10.7 28.5 12…
  randomSample1     int [1:8] 5 4 1 1 6 7 7 8
  randomSample2     int [1:8] 3 9 1 10 2 8 6 7
  randomTemps       num [1:80] 27 33 33 29 23 33 42 29…
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root
/3-4b.r')
>
```

*Fig 4: Normal distribution: **200** samples with mean **20** and standard deviation **4***

Note: the values are being picked at random so the histogram will look different each time you execute the script.

5.1 - Normal temperature values

Instead of using hardcoded values (***mean* = 20, *sd* = 4**) for a normal distribution, we are going to sample from a normal distribution with the same mean and standard deviation as the ***lansJanTemps*** matrix.

We use ***mean()*** and ***sd()*** to get the mean and standard deviation from the ***lansJanTemps*** matrix and use these values as the sample mean and standard deviation in ***rnorm():***

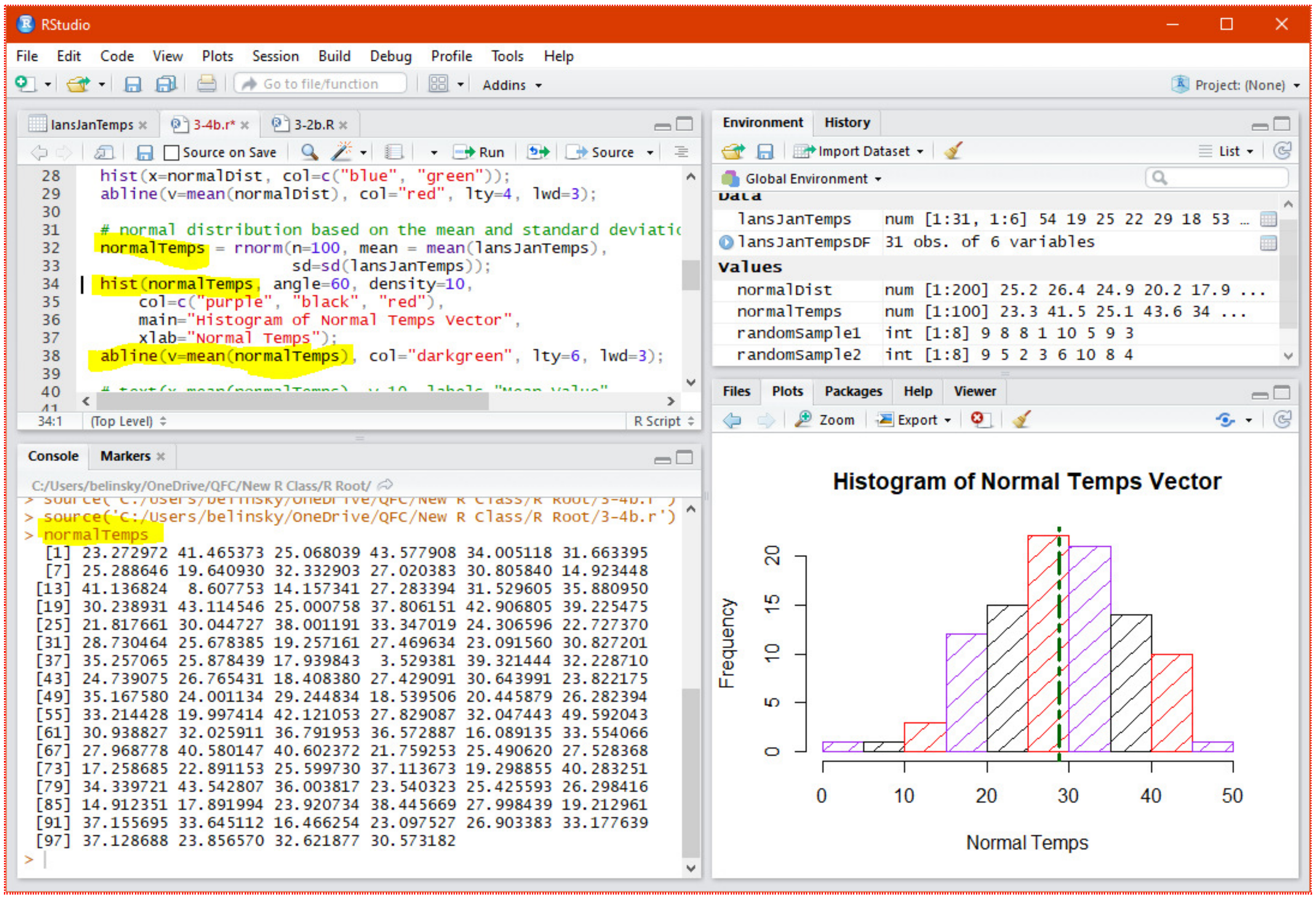
```
normalTemps = rnorm(n=100, mean = mean(lansJanTemps),
                    sd=sd(lansJanTemps));
```

So, in the code above, we generate 100 temperature values from a normal distribution based on the mean and standard deviation of ***lansJanTemps.*** The data generated are saved in ***normalTemps***.

Then we use a histogram to make a visual representation of ***normalTemp*** (using ***hist()***) and, to the histogram, we add a vertical line at the mean value (using ***abline()***):

```
hist(normalTemps, angle=60, density=10,
     col=c("purple", "black", "red"),
      main="Histogram of Normal Temps Vector",
      xlab="Normal Temps");
# lty: line type, lwd: line width
abline(v=mean(normalTemps), col="darkgreen", lty=6, lwd=3);
```

We can display all 100 values in the Console Window by typing in "normalTemps" and hitting Enter. The distribution of the 100 temperature values is shown in the histogram.



```
hist(x=normalDist, col=c("blue", "green"));
abline(v=mean(normalDist), col="red", lty=4, lwd=3);

# normal distribution based on the mean and standard deviatio
normalTemps = rnorm(n=100, mean = mean(lansJanTemps),
                         sd=sd(lansJanTemps));
hist(normalTemps, angle=60, density=10,
     col=c("purple", "black", "red"),
     main="Histogram of Normal Temps Vector",
     xlab="Normal Temps");
abline(v=mean(normalTemps), col="darkgreen", lty=6, lwd=3);

# text(x=mean(normalTemps), y=10, labels="Mean Value"
```

```
Data
  lansJanTemps     num [1:31, 1:6] 54 19 25 22 29 18 53 ...
  lansJanTempsDF   31 obs. of 6 variables
Values
  normalDist       num [1:200] 25.2 26.4 24.9 20.2 17.9 ...
  normalTemps      num [1:100] 23.3 41.5 25.1 43.6 34 ...
  randomSample1    int [1:8] 9 8 8 1 10 5 9 3
  randomSample2    int [1:8] 9 5 2 3 6 10 8 4
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> normalTemps
  [1] 23.272972 41.465373 25.068039 43.577908 34.005118 31.663395
  [7] 25.288646 19.640930 32.332903 27.020383 30.805840 14.923448
 [13] 41.136824  8.607753 14.157341 27.283394 31.529605 35.880950
 [19] 30.238931 43.114546 25.000758 37.806151 42.906805 39.225475
 [25] 21.817661 30.044727 38.001191 33.347019 24.306596 22.727370
 [31] 28.730464 25.678385 19.257161 27.469634 23.091560 30.827201
 [37] 35.257065 25.878439 17.939843  3.529381 39.321444 32.228710
 [43] 24.739075 26.765431 18.408380 27.429091 30.643991 23.822175
 [49] 35.167580 24.001134 29.244834 18.539506 20.445879 26.282394
 [55] 33.214428 19.997414 42.121053 27.829087 32.047443 49.592043
 [61] 30.938827 32.025911 36.791953 36.572887 16.089135 33.554066
 [67] 27.968778 40.580147 40.602372 21.759253 25.490620 27.528368
 [73] 17.258685 22.891153 25.599730 37.113673 19.298855 40.283251
 [79] 34.339721 43.542807 36.003817 23.540323 25.425593 26.298416
 [85] 14.912351 17.891994 23.920734 38.445669 27.998439 19.212961
 [91] 37.155695 33.645112 16.466254 23.097527 26.903383 33.177639
 [97] 37.128688 23.856570 32.621877 30.573182
> 
```

*Fig 5: Generating normalized temperature data based on the temperature values from the **lansJanTemps** vector*

6 - Psuedo-Random values

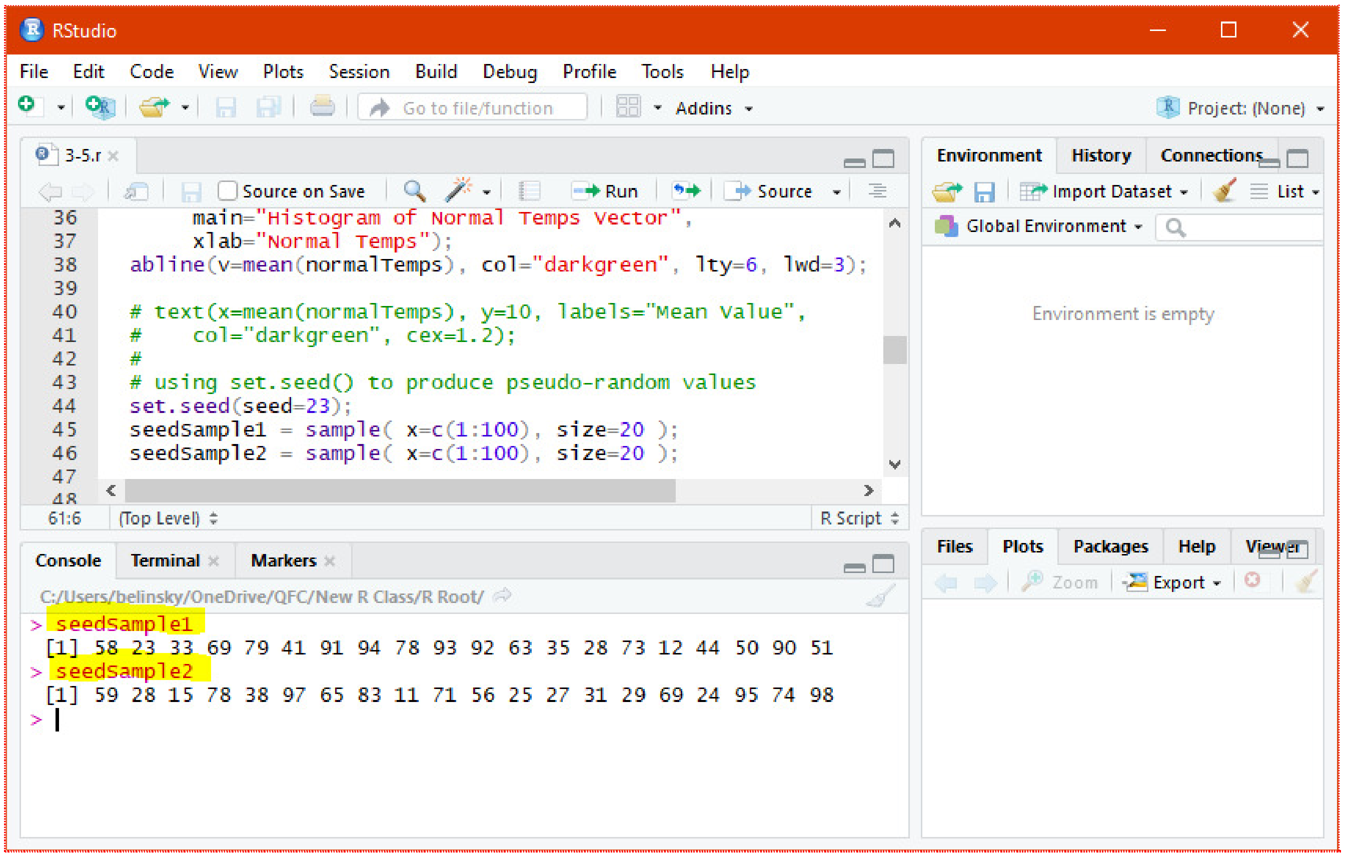
Every time your script is executed, a new random set of values is produced. If you want to replicate your results there is a function called ***set.seed().*** ***set.seed()*** will produce the same "random" numbers every time using a seed number. In other words, ***set.seed()*** produces values that look random but the values do not change each time the script is executed.

6.1 - Seed value

Here is a really good video on PBS Infinite Series about random number generators, pseudo-random number generators, and seed numbers. The discussion on seed numbers starts at 2:15.

set.seed() takes one parameter and that is the seed number. The seed number is the starting point for a complex formula that produces a series of random-looking, or pseudo-random, values. Here we set the seed number to **23** and ran a couple samples. Each time you execute the code, ***seedSample1*** and ***seedSample2*** will have the same value (although they are different from each-other).

```
set.seed(seed=23);
seedSample1 = sample( x=c(1:100), size=20 );
seedSample2 = sample( x=c(1:100), size=20 );
```



```
     main="Histogram of Normal Temps Vector",
     xlab="Normal Temps");
abline(v=mean(normalTemps), col="darkgreen", lty=6, lwd=3);

# text(x=mean(normalTemps), y=10, labels="Mean Value",
#     col="darkgreen", cex=1.2);
#
# using set.seed() to produce pseudo-random values
set.seed(seed=23);
seedSample1 = sample( x=c(1:100), size=20 );
seedSample2 = sample( x=c(1:100), size=20 );
```

```
> seedSample1
 [1] 58 23 33 69 79 41 91 94 78 93 92 63 35 28 73 12 44 50 90 51
> seedSample2
 [1] 59 28 15 78 38 97 65 83 11 71 56 25 27 31 29 69 24 95 74 98
> 
```

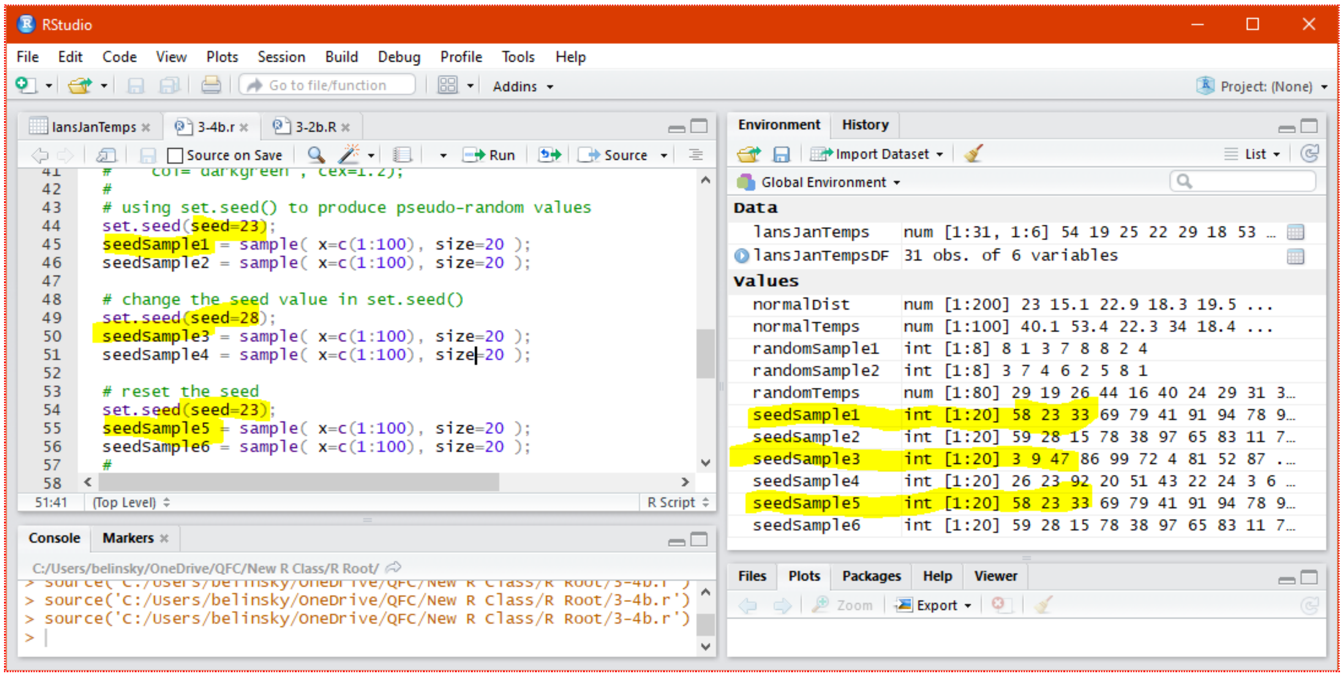
Fig 6: Pseudo-random values produced by a seed number.

6.2 - Changing the seed value

Each time we change the seed value, we "reset" the pseudo-random number generator. ***seedSample3*** and ***seedSample4*** will always have the same 20 values based on a generator seeded with the value **28**. ***seedSample5*** and ***seedSample6*** will be identical to ***seedSample1*** and ***seedSample2*** because the seed value was reset to **23**.

```
set.seed(seed=28);
seedSample3 = sample( x=c(1:100), size=20 );
seedSample4 = sample( x=c(1:100), size=20 );

set.seed(seed=23);   # same seed value as above
seedSample5 = sample( x=c(1:100), size=20 );  # will be same as seedSample1
seedSample6 = sample( x=c(1:100), size=20 );  # will be same as seedSample2
```



```
#    col="darkgreen", cex=1.2);
#
# using set.seed() to produce pseudo-random values
set.seed(seed=23);
seedSample1 = sample( x=c(1:100), size=20 );
seedSample2 = sample( x=c(1:100), size=20 );

# change the seed value in set.seed()
set.seed(seed=28);
seedSample3 = sample( x=c(1:100), size=20 );
seedSample4 = sample( x=c(1:100), size=20 );

# reset the seed
set.seed(seed=23);
seedSample5 = sample( x=c(1:100), size=20 );
seedSample6 = sample( x=c(1:100), size=20 );
#
```

```
Data
  lansJanTemps    num [1:31, 1:6] 54 19 25 22 29 18 53 ...
  lansJanTempsDF  31 obs. of 6 variables
Values
  normalDist      num [1:200] 23 15.1 22.9 18.3 19.5 ...
  normalTemps     num [1:100] 40.1 53.4 22.3 34 18.4 ...
  randomSample1   int [1:8] 8 1 3 7 8 8 2 4
  randomSample2   int [1:8] 3 7 4 6 2 5 8 1
  randomTemps     num [1:80] 29 19 26 44 16 40 24 29 31 3...
  seedSample1     int [1:20] 58 23 33 69 79 41 91 94 78 9...
  seedSample2     int [1:20] 59 28 15 78 38 97 65 83 11 7...
  seedSample3     int [1:20] 3 9 47 86 99 72 4 81 52 87 ....
  seedSample4     int [1:20] 26 23 92 20 51 43 22 24 3 6 ...
  seedSample5     int [1:20] 58 23 33 69 79 41 91 94 78 9...
  seedSample6     int [1:20] 59 28 15 78 38 97 65 83 11 7...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
>
```

*Fig 7: Using **set.seed()** to create pseudo-random values*

Note: if you want to stop using the pseudo-random generator in your script, then call:

```
set.seed(NULL);  # not in the lesson's script
```

6.3 - Pseudo-random values from a matrix

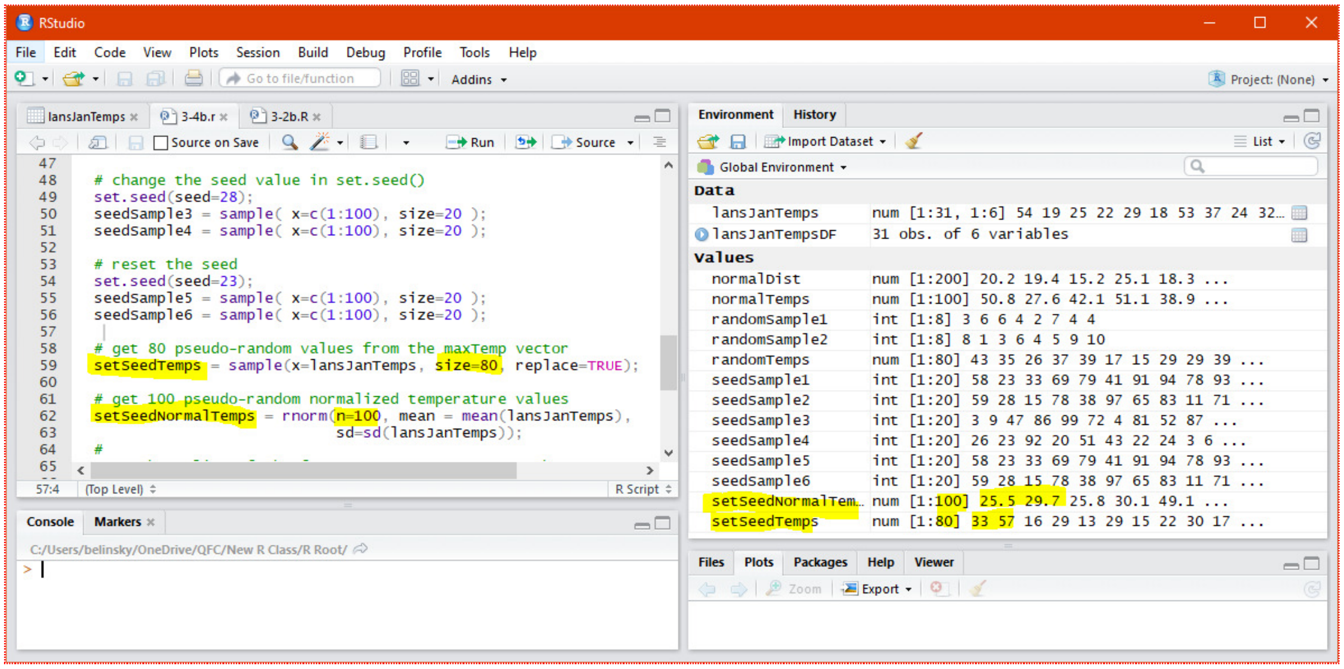
set.seed() has been executed in our script and a seed value has been established. So any time we ask for random values in the script, pseudo-random values will be generated based on the seed value (in this case, ***23)***.

The following code *will produce the same* 80 pseudo-random values from the ***lansJanTemps*** matrix each time:

```
setSeedTemps = sample(x=lansJanTemps, size=80, replace=TRUE);
```

And the following code *will produce the same* 100 normalized pseudo-random values based on the mean and standard deviation from ***lansJanTemps***.

```
setSeedNormalTemps = rnorm(n=100, mean = mean(lansJanTemps),
                                  sd=sd(lansJanTemps));
```


```

# change the seed value in set.seed()
set.seed(seed=28);
seedSample3 = sample( x=c(1:100), size=20 );
seedSample4 = sample( x=c(1:100), size=20 );

# reset the seed
set.seed(seed=23);
seedSample5 = sample( x=c(1:100), size=20 );
seedSample6 = sample( x=c(1:100), size=20 );

# get 80 pseudo-random values from the maxTemp vector
setSeedTemps = sample(x=lansJanTemps, size=80, replace=TRUE);

# get 100 pseudo-random normalized temperature values
setSeedNormalTemps = rnorm(n=100, mean = mean(lansJanTemps),
                           sd=sd(lansJanTemps));
#
```

```
Data
  lansJanTemps        num [1:31, 1:6] 54 19 25 22 29 18 53 37 24 32...
  lansJanTempsDF      31 obs. of 6 variables
Values
  normalDist          num [1:200] 20.2 19.4 15.2 25.1 18.3 ...
  normalTemps         num [1:100] 50.8 27.6 42.1 51.1 38.9 ...
  randomSample1       int [1:8] 3 6 6 4 2 7 4 4
  randomSample2       int [1:8] 8 1 3 6 4 5 9 10
  randomTemps         num [1:80] 43 35 26 37 39 17 15 29 29 39 ...
  seedSample1         int [1:20] 58 23 33 69 79 41 91 94 78 93 ...
  seedSample2         int [1:20] 59 28 15 78 38 97 65 83 11 71 ...
  seedSample3         int [1:20] 3 9 47 86 99 72 4 81 52 87 ...
  seedSample4         int [1:20] 26 23 92 20 51 43 22 24 3 6 ...
  seedSample5         int [1:20] 58 23 33 69 79 41 91 94 78 93 ...
  seedSample6         int [1:20] 59 28 15 78 38 97 65 83 11 71 ...
  setSeedNormalTem…   num [1:100] 25.5 29.7 25.8 30.1 49.1 ...
  setSeedTemps        num [1:80] 33 57 16 29 13 29 15 22 30 17 ...
```

*Fig 8: Using **set.seed()** to create pseudo-random temperature data.*

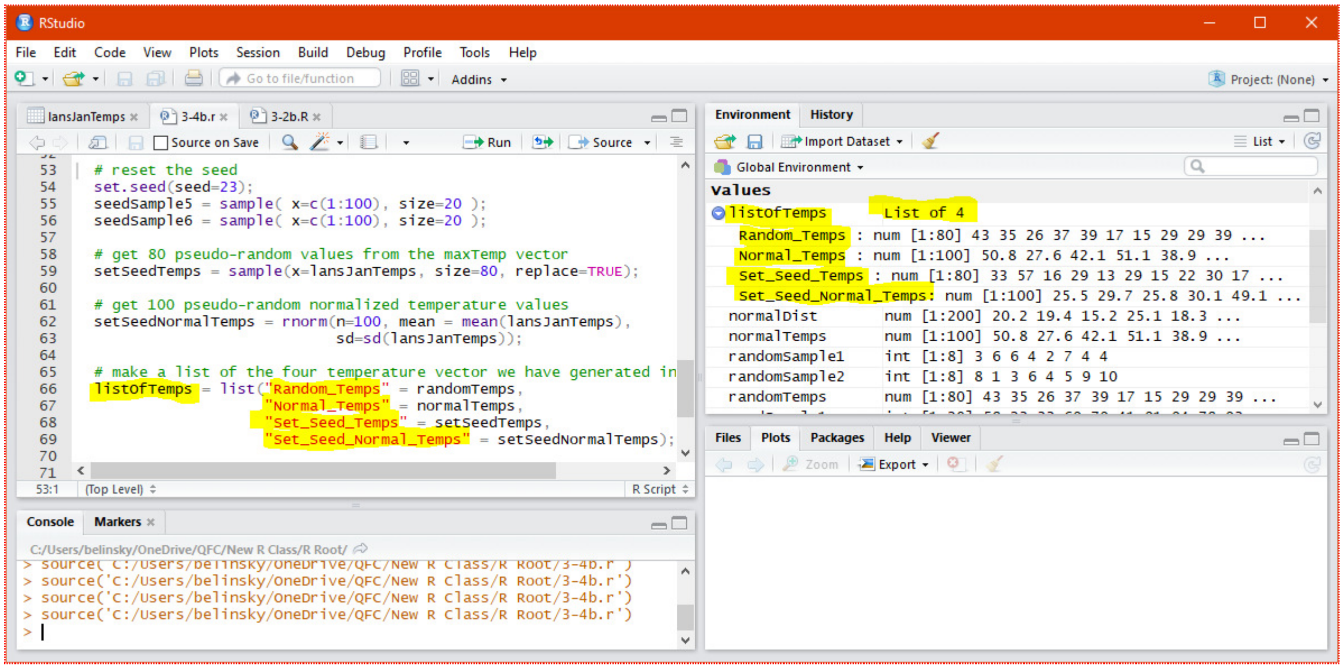
Try this out by running the code multiple times without changing the seed number. Then change the seed number and see what happens.

7 - Saving the data you generated

We have generated four different vectors with temperature data in this lesson: ***randomTemps***, ***normalTemps***, ***setSeedTemps, and setSeedNormalTemp***. We might want to save these vectors for later use -- perhaps in another script. However, we cannot save the four vectors to a data frame because the vectors have different sizes (i.e., they have a different number of values). If we want to put the four vectors into one structure, we need to use a ***list.*** We can create a list using the ***list()*** function.

```
listOfTemps = list("Random_Temps" = randomTemps,
          "Normal_Temps" = normalTemps,
          "Set_Seed_Temps" = setSeedTemps,
          "Set_Seed_Normal_Temps" = setSeedNormalTemps);
```

In this case, the quoted values become the vector's name in the list.



```
# reset the seed
set.seed(seed=23);
seedSample5 = sample( x=c(1:100), size=20 );
seedSample6 = sample( x=c(1:100), size=20 );

# get 80 pseudo-random values from the maxTemp vector
setSeedTemps = sample(x=lansJanTemps, size=80, replace=TRUE);

# get 100 pseudo-random normalized temperature values
setSeedNormalTemps = rnorm(n=100, mean = mean(lansJanTemps),
                           sd=sd(lansJanTemps));

# make a list of the four temperature vector we have generated in
listOfTemps = list("Random_Temps" = randomTemps,
                   "Normal_Temps" = normalTemps,
                   "Set_Seed_Temps" = setSeedTemps,
                   "Set_Seed_Normal_Temps" = setSeedNormalTemps);
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-4b.r')
>
```

```
Values
listOfTemps         List of 4
  Random_Temps : num [1:80] 43 35 26 37 39 17 15 29 29 39 ...
  Normal_Temps : num [1:100] 50.8 27.6 42.1 51.1 38.9 ...
  Set_Seed_Temps : num [1:80] 33 57 16 29 13 29 15 22 30 17 ...
  Set_Seed_Normal_Temps: num [1:100] 25.5 29.7 25.8 30.1 49.1 ...
normalDist          num [1:200] 20.2 19.4 15.2 25.1 18.3 ...
normalTemps         num [1:100] 50.8 27.6 42.1 51.1 38.9 ...
randomSample1       int [1:8] 3 6 6 4 2 7 4 4
randomSample2       int [1:8] 8 1 3 6 4 5 9 10
randomTemps         num [1:80] 43 35 26 37 39 17 15 29 29 39 ...
```

*Fig 9: Creating a **list** of the four temperature vectors we created in the script.*

7.1 - *Saving and loading a list object*

We will save the list to a file, called ***weatherList.rdata***, so we can access the values from another script.

```
save(listOfTemps, file = "data/weatherList.rdata");
```

The file ***weatherList.rdata*** now exists in your data folder and can be called from another script.

7.2 - Opening the list from another script

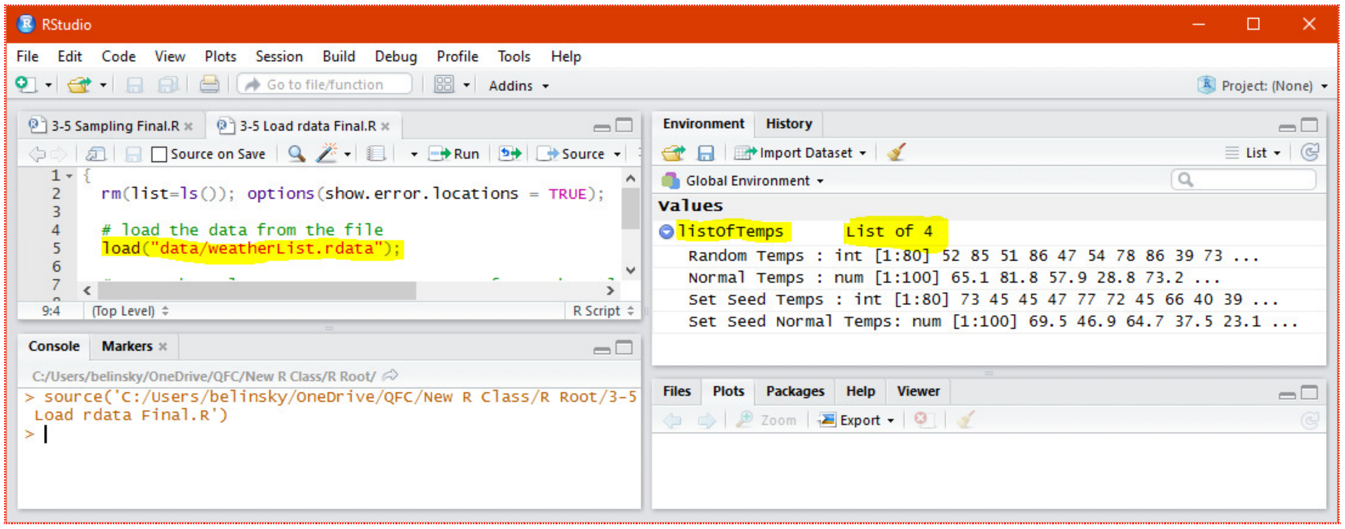
From another script, we can load the data from the file ***weatherList.rdata***, which resides in your R data directory.

Note: the code in the rest of this lesson is not in the lesson's script.

We use ***load()*** to load the data from the ***weatherList.rdata*** file for use in the current script.

```
load("data/weatherList.rdata"));
```

Notice that this loaded ***listOfTemps*** that was saved from the other script -- and all the values are the same as before.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  # load the data from the file
  load("data/weatherList.rdata");
```

```
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-5 Load rdata Final.R')
>
```

```
Values
listofTemps        List of 4
  Random Temps : int [1:80] 52 85 51 86 47 54 78 86 39 73 ...
  Normal Temps : num [1:100] 65.1 81.8 57.9 28.8 73.2 ...
  Set Seed Temps : int [1:80] 73 45 45 47 77 72 45 66 40 39 ...
  Set Seed Normal Temps: num [1:100] 69.5 46.9 64.7 37.5 23.1 ...
```

*Fig 10: Loading the **list** that was saved from the another script file.*

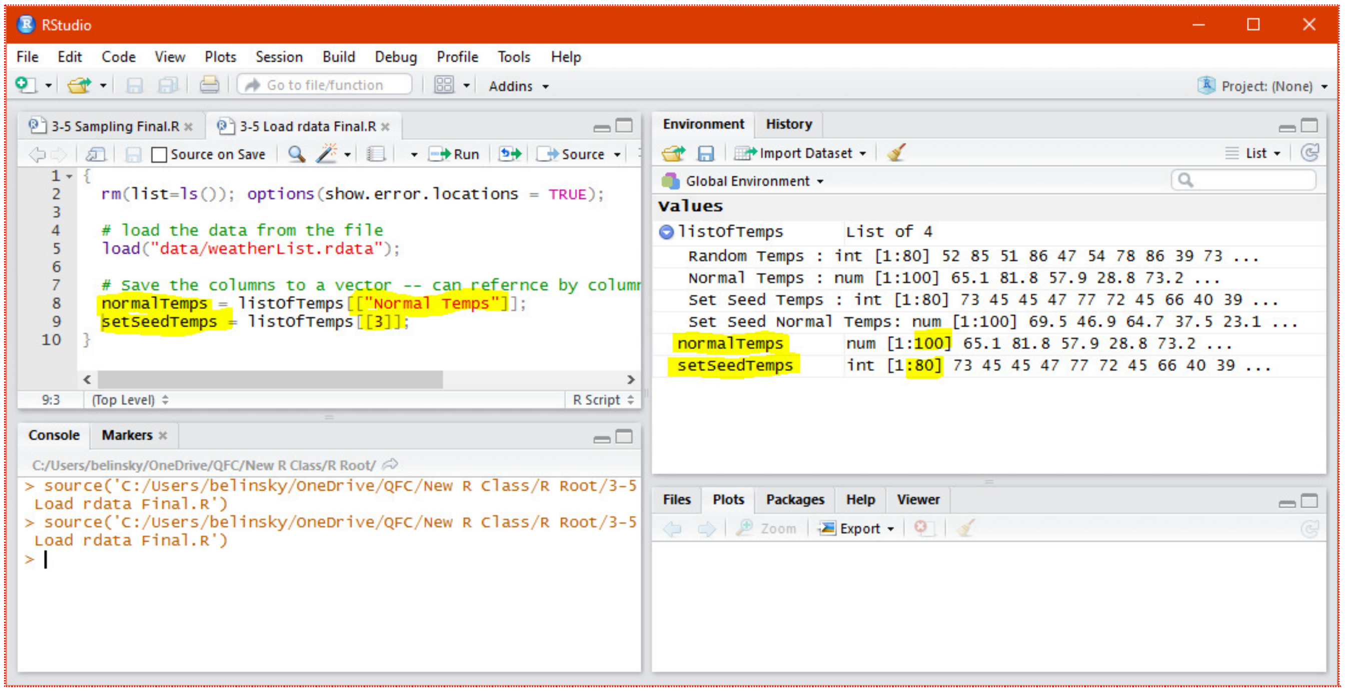
If we want to take elements out of the list and save them to a vector we need to use double brackets ***[[ ]]*** instead of single brackets.

We can refer to the column by its name:

```
normalTemps = listofTemps[["Normal Temps"]];
```

Or by the column number

```
setSeedTemps = listofTemps[[3]];
```



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  # load the data from the file
  load("data/weatherList.rdata");

  # Save the columns to a vector -- can refernce by colum
  normalTemps = listofTemps[["Normal Temps"]];
  setSeedTemps = listofTemps[[3]];
}
```

```
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-5 Load rdata Final.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-5 Load rdata Final.R')
>
```

```
Values
listofTemps        List of 4
  Random Temps : int [1:80] 52 85 51 86 47 54 78 86 39 73 ...
  Normal Temps : num [1:100] 65.1 81.8 57.9 28.8 73.2 ...
  Set Seed Temps : int [1:80] 73 45 45 47 77 72 45 66 40 39 ...
  Set Seed Normal Temps: num [1:100] 69.5 46.9 64.7 37.5 23.1 ...
normalTemps        num [1:100] 65.1 81.8 57.9 28.8 73.2 ...
setSeedTemps       int [1:80] 73 45 45 47 77 72 45 66 40 39 ...
```

*Fig 11: Saving columns from a **list** to a vector.*

8 - Application

1) Using the data from lesson 3-1 (saved as "data/formattedLansingWeather.csv")
- get 50 samples of the average wind speed (AWND) and allow for repeats
- get 30 samples of precipitation (PRCP) and do not allow for repeats

2) Create a set of wind speed values that has a normal distribution with mean and standard deviation of the mean and standard deviation equal to the wind speed data from lesson 3-1. Create distributions with n=20, 50, and 200 values. Create a histogram for each of the distributions and show where the mean value is.
3) Repeat #2 using a seed value of 15.