Doing Basic Math for Stats in R

Nathan L. Brouwer | brouwern@gmail.com

September 2017

Doing Math in R

- R can do everything a scientific calculator or spreadsheet can do. * This includes basic functions like +, - , /, sqrt, etc,
- Also basic functions like mean(), median(), sd() for the standard deviation and var() for the variance.
- In general in this course we will focus on having R do most of the calculations for our stats.

- However, it's important to understand some of the underlying math.
- Today, to practice doing math in R and to learn about basic statsitcal functions, we'll do calculations of the mean, variance, "sum of squares", and standard deviation "by hand" in R.

- We'll then compare them to the output R produces.

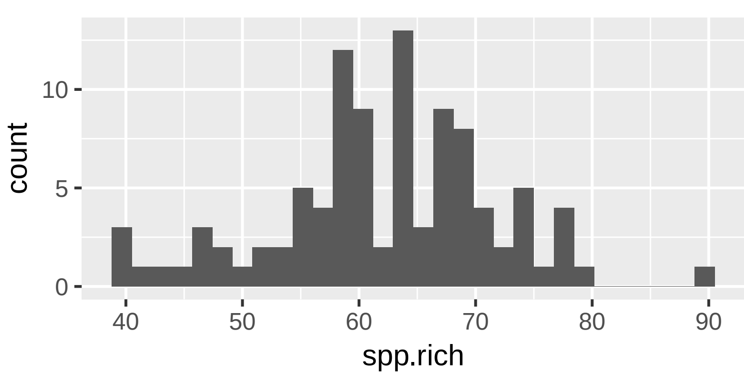
The USGS Breeding Bird Survey (BBS) data

Below are the number of species observed during a recent year of surveys on all routes in PA

Distribution of species richness values

The distribution of species richness values looks like this

```
library(ggplot2)
qplot(spp.rich,
      dat = dat)
```



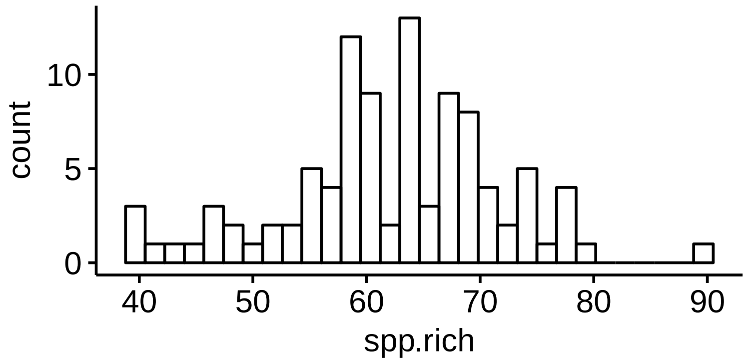
A nice package for making graphs with ggplot is ggpubr. This lots of "dependencies" so you'll see lots of red text fly by and it will take a minute to download.

```
#download the packages
# install.packages("ggpubr")
# install.packages("dplyr")

library(ggpubr)
```

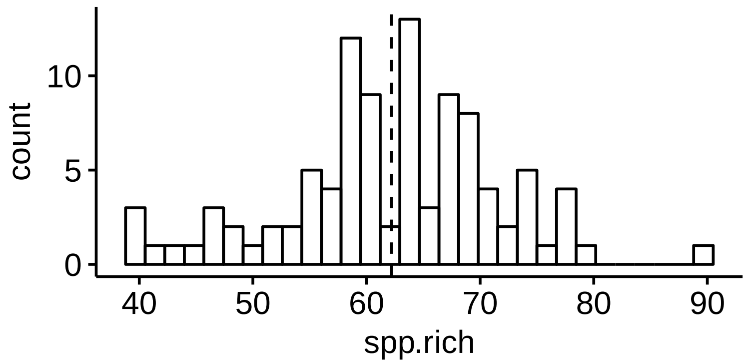
A basic ggpubr histogram using the function gghistogram(). (Ignore the red text about "bind = 30")

```
gghistogram(data = dat,
            x = "spp.rich")
```



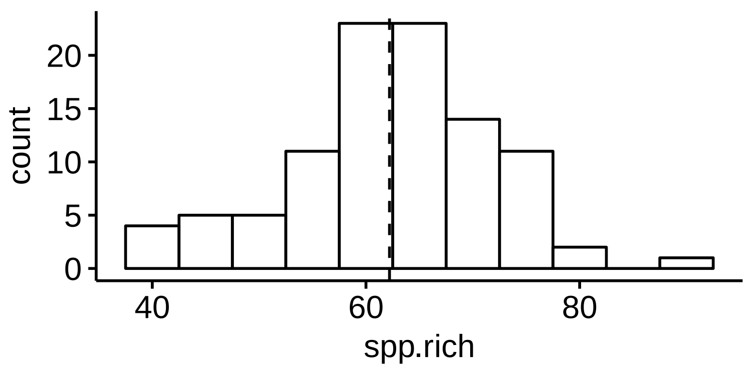
Add the mean as a verticle line using the add = "mean" arguement

```
gghistogram(data = dat,
            x = "spp.rich",
            add = "mean")
```



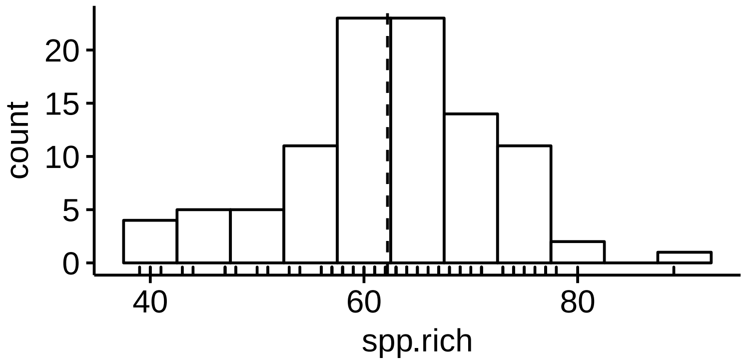
Change the "binwidth"

```
gghistogram(data = dat,
            x = "spp.rich",
            add = "mean",
            binwidth = 5)
```



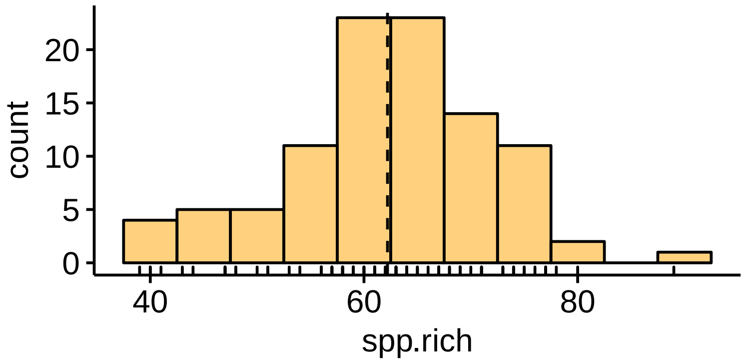
Add a "rug" on the bottom of the plot where each datapoint is exactly.

```r
gghistogram(data = dat,
            x = "spp.rich",
            add = "mean",
            binwidth = 5,
            rug = TRUE)
```



Add a color to fill the bars.

```r
gghistogram(data = dat,
            x = "spp.rich",
            add = "mean",
            binwidth = 5,
            fill = "orange",
            rug = TRUE)
```



Summary stats

- All your basic stats.
- Note meaninful for "route" b/c it is actually a categorical variable

```r
summary(dat)
```

```
##      route          spp.rich
##  Min.   :  2.0   Min.   :39.0
##  1st Qu.: 30.5   1st Qu.:57.5
##  Median : 63.0   Median :63.0
##  Mean   :152.1   Mean   :62.2
##  3rd Qu.:165.5   3rd Qu.:69.0
```

```
## Max.   :911.0   Max.   :89.0
```

Accessing a column in a dataframe

Look at just the spp.rich column using $spp.rich

```r
dat$spp.rich
```

```
##  [1] 69 58 69 65 65 61 74 57 61 61 56 40 67 56 58 69 73 59 63 51 77 71 59
## [24] 77 63 64 74 74 89 60 64 71 77 78 80 62 61 67 69 69 57 58 69 64 61 61
## [47] 69 68 66 57 64 67 64 67 60 63 64 58 67 63 50 60 43 73 58 56 59 70 57
## [70] 48 76 56 58 75 64 59 54 64 68 67 56 41 59 63 75 69 47 53 62 48 71 59
## [93] 47 39 47 67 51 44 40
```

Basic math stuff in R

sum()

The summed of the spp richness of all routes. Not really meaninful on its own.

```r
sum(dat$spp.rich)
```

```
## [1] 6158
```

The min, max, etc.

```r
min(dat$spp.rich)
```

```
## [1] 39
```

```r
max(dat$spp.rich)
```

```
## [1] 89
```

Mean()

The mean number of species observed per route.

```r
mean(dat$spp.rich)
```

```
## [1] 62.20202
```

The varinace var()

- On its own this value is hard to interpret. But he larger the variance, the more variation there is between each route.
- If all the routes had the exact same number of species, var = 0.0

```r
var(dat$spp.rich)
```

```
## [1] 90.55061
```

The standard deviation (sd)

The standard deviation is a very common way to represent variability

```
sd(dat$spp.rich)
```

```
## [1] 9.515808
```

Variance vs. standard deviation

- The standard deviation is just the square root of the of the variance.
- We get the square root using sqrt()

```
#SD using R's function
sd(dat$spp.rich)
```

```
## [1] 9.515808
```

```
#SD as the sqrt of the variance
sqrt(var(dat$spp.rich))
```

```
## [1] 9.515808
```

Doing math for stats "by hand""

Calculating the mean by hand

The numerator

```
my.sum <- sum(dat$spp.rich)
```

The denomintor

Using the length() function to get the number of rows in the dataframe

```
my.N <- length(dat$spp.rich)
```

The mean

Doing division using "/"

```
my.mean <- my.sum/my.N
my.mean
```

```
## [1] 62.20202
```

In one step Same thing to get the mean, just in 1 step

```
my.mean <- sum(dat$spp.rich)/length(dat$spp.rich)
my.mean
```

```
## [1] 62.20202
```

Checking our answer using == We can check our answer like this using the "==" function which asks R "are these two objects EXACTLy the same?"

```
my.mean == mean(dat$spp.rich)
```

```
## [1] TRUE
```

We can do the same thing to check that the standard deviation is indeed the square root of the mean

```
sd(dat$spp.rich) == sqrt(var(dat$spp.rich))
```

```
## [1] TRUE
```

If the were not the same, R would say "FALSE"

- Because R is very very very very precise any rounding errors will result in R saying that 2 things are NOT exactly the same. Sometimes some R functions do some rounding so you have to check "FALSE" answers sometimes.

Calcualting variance & standard deviation by hand, step by step

- Variance and standard deviation are fundamental quantities in statistics.
- We'll step through each part of their calculation

"Deviations" between the mean and each observation

Calculate the difference between each observation and the mean of all observation that you calculated above.

```
Yi.deviations <- dat$spp.rich-my.mean
```

Start making a dataframe (df)

- We can keep track of the math by making a spreadsheet-like object in R called a dataframe using the data.frame command.

```
my.df <- data.frame(spp.rich = dat$spp.rich,
               Yi.deviations)
```

Look at the matrix. Note that some of the deviations are positive and some are negative.

```
head(my.df)
```

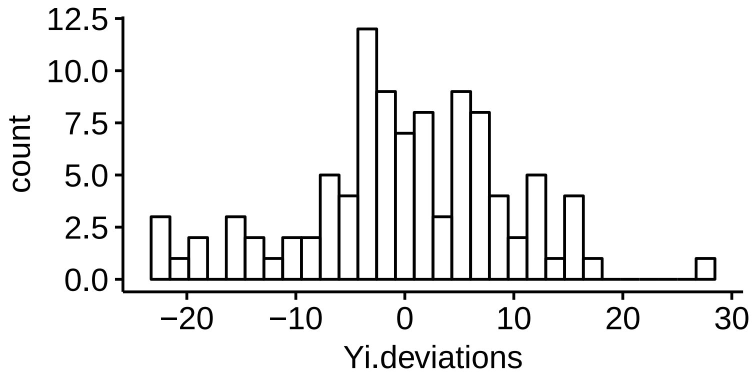
```
##   spp.rich Yi.deviations
## 1       69       6.79798
## 2       58      -4.20202
## 3       69       6.79798
## 4       65       2.79798
## 5       65       2.79798
## 6       61      -1.20202
```

```
tail(my.df)
```

```
##    spp.rich Yi.deviations
## 94       39     -23.20202
## 95       47     -15.20202
## 96       67       4.79798
## 97       51     -11.20202
## 98       44     -18.20202
## 99       40     -22.20202
```

Look at a histogram of these values. We won't focus on this now, but these values are called the "residuals"

```
gghistogram(data = my.df,
            x = "Yi.deviations")
```



Calculate the“Squared deviations” between the mean and each observation

Take your set of deviations and square them using “^2”

Here, we’ve done math on a whole list of numbers at the same time.

```
Yi.deviations.square <- Yi.deviations^2
```

- Squaring is key b/c it makes deviations greater than the mean and less than the mean to have the same magnitude

- That is, we go from “deviations” that are positive and negative to “squared deviations” that are all positive

Add the square deviations to the dataframe

- We can add stuff to our dataframe using the function cbind()
- cbind() stands for “column bind”

```
my.df <- cbind(my.df, Yi.deviations.square)
```

Look at your expanding matrix

```
head(my.df)
```

```
##   spp.rich Yi.deviations Yi.deviations.square
## 1       69       6.79798            46.212529
## 2       58      -4.20202            17.656974
## 3       69       6.79798            46.212529
## 4       65       2.79798             7.828691
## 5       65       2.79798             7.828691
## 6       61      -1.20202             1.444853
```

Sum of squares between the mean & each deviation

- we can now calcualte the “sum of square deviations”
- AKA the “sum of squares” (SS)
- This is the numerator (the thing on top) in the variance & standard deviation equations.*
- This is an important intermediary step in the process

```
my.sum.of.squares<- sum(Yi.deviations.square)
```

This is a rather big number and I'm glad we don't have to calcualte it by hand :)

```
my.sum.of.squares
```

```
## [1] 8873.96
```

And now…the variance (var)

- We can now use the sum of squares(SS) to calcuate the variance.
- This is is the SS divided by the sample size minus one.
- That is SS/(n-1)
- Recall that we calcualted the sample size (N) above and put it in an object called "my.N".

- We subtract 1 from N to get what is known as the "Degrees of freedom" (DF). More on this later…

Equation for the variance

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

```
#The sample size
my.N
```

```
## [1] 99
```

```
#degrees of freedom
dfs <- my.N - 1

#The var
my.var <- my.sum.of.squares/dfs

#Could also do it more directly
my.var <- my.sum.of.squares/(my.N - 1)
```

Check our var vs. R's var

```
my.var == var(dat$spp.rich)
```

```
## [1] TRUE
```

Got it!

And now…the standard deviation (SD)

- The standard deviation is just the square root of the variance.
- We get the square root using the sqrt() function

Equation for Standard Deviation Where x = a column of data in a dataframe

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

R-ish notation for SE

$$SD = \sqrt{var(x)}$$

```
my.sd <- sqrt(my.var)
```

We can check if our results are the same as R

```
#Variance
var(dat$spp.rich)
```

```
## [1] 90.55061
```

```
my.var
```

```
## [1] 90.55061
```

```
#stdev
sd(dat$spp.rich)
```

```
## [1] 9.515808
```

```
my.sd
```

```
## [1] 9.515808
```

We can check this also like this usig "=="

```
#stdev
sd(dat$spp.rich) == my.sd
```

```
## [1] TRUE
```

The standard error (SE)

- The standard error (SE) is a fundamental quantity in stats.

- It is used as a measure of the precision of the data
- It is closely related to the concept of a 95% confidence interval (CI)
- Is is calculated the standard deviation divided by the square root of the sample size.

Equation for Standard error (SE) Where x = a column of data in a dataframe

$$SE = \frac{SD}{\sqrt{N}}$$

R-ish notation for SE

$$SE = \frac{SD(x)}{sqrt(length(x))}$$

```
my.se <- my.sd/sqrt(my.N)
```

Calculating the SE directly

- For some reason the basic R set up doesn't have a function for the SE.
- There is a function for the SE, std.error, in the package plotrix

```
#Download plotrix
# install.packages("plotrix")
library(plotrix)

std.error(dat$spp.rich)
```

```
## [1] 0.9563747
```

Advanced Pro trick: write your own function

- If there wasn't a function in for the SE, you could write your own like this.
- This is an advanced topic and is only shown for those who are curious.

```
#with all the math
my.se.fnxn <- function(x){sqrt(var(x))/sqrt(length(x))}

#with less  math
my.se.fnxn <- function(x){sd(x)/sqrt(length(x))}


my.se.fnxn(dat$spp.rich)
```

```
## [1] 0.9563747
```

95% Confidence intervals (95% CIs)

- The standard error is related closely to the 95% Confidence interval
- There are precise equations for calculating 95% CIs for different circumstances
- Roughtly, a 95% CIs extend on either side of the mean to indicate a plausible range of values the should contain the real value you are trying to estimate with your eamn
- As an approximatation, 95% CI is mean +/- 1.98*SE
- The upper bound of the CI is mean + 1.98*SE
- The lower bound of the CI is mean - 1.98*SE

```
my.CI.upper <- my.mean + my.se*1.96
```

Background info

https://www.r-bloggers.com/standard-deviation-vs-standard-error/