Sprint 4 Notes

<https://www.analyticsvidhya.com/blog/2015/12/faster-data-manipulation-7-packages/>

See my powerPoint for above.

Good definition of R tools/libraries/github / etc.

<https://www.datacamp.com/community/tutorials/r-packages-guide>

Starting to look at Statistics Book for R.

The below command happened after I loaded the swirl package.

Type ls() to see a list of the variables in your workspace. Then, type rm(list=ls()) to clear your workspace.

*The normal distribution is one of the most common, well-used distributions in statistics. Normal distributions come in many means and standard deviations, but they all have a signature shape, where the data values fall into a smooth, bell-shaped curve. The data are concentrated in the center, but some of them are more spread out than others. The spread of the distribution is determined by the standard deviation.*

Aug\_2016$Lecture <- gsub('Strongly Agree', '40', Aug\_2016$Lecture)

> View(Aug\_2016)

> Aug\_2016$Lecture <- gsub('Strongly Disagree', '10', Aug\_2016$Lecture)

> Aug\_2016$Lecture <- gsub('Disagree', '20', Aug\_2016$Lecture)

> Aug\_2016$Lecture <- gsub('Agree', '30', Aug\_2016$Lecture)

<https://stackoverflow.com/questions/37707060/converting-data-frame-column-from-character-to-numeric>

If we need only one column to be numeric

yyz$b <- as.numeric(as.character(yyz$b))

But, if all the columns needs to changed to numeric, use lapply to loop over the columns and convert to numeric by first converting it to character class as the columns were factor.

yyz[] <- lapply(yyz, function(x) as.numeric(as.character(x)))

Aug\_2016$Lecture <- as.numeric(as.character(Aug\_2016$Lecture))

**Integer vs. Double**

The two most common numeric classes used in R are integer and double (for double precision floating point numbers). R automatically converts between these two classes when needed for mathematical purposes. As a result, it’s feasible to use R and perform analyses for years without specifying these differences.

**Note for Ben..!**

**Trying to change from Character to Numeric**

> Aug\_2016$Lecture <- as.numeric(as.character(Aug\_2016$Lecture))

Warning message:

NAs introduced by coercion

> Aug\_2016$Responsive <- as.numeric(as.character(Aug\_2016$Responsive))

Warning message:

NAs introduced by coercion

How to Calculate the Mean in R (with ignoring the NA values..!)

<https://www.tutorialspoint.com/r/r_mean_median_mode.htm>

## Applying NA Option

If there are missing values, then the mean function returns NA.

To drop the missing values from the calculation use na.rm = TRUE. which means remove the NA values.

# Create a vector.

x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)

# Find mean.

result.mean <- mean(x)

print(result.mean)

# Find mean dropping NA values.

result.mean <- mean(x,na.rm = TRUE)

print(result.mean)

Mean and Standard Deviation

So for me it is on the Lecture Column

mean(Aug\_2016$Lecture,na.rm = TRUE)

sd(Aug\_2016$Lecture,na.rm = TRUE)

MAYBE VERY USEFUL SOON…!

<https://stackoverflow.com/questions/9690232/writing-multiple-values-in-different-columns-in-a-csv-file-in-r>

Can anyone help me with this: I have calculated 2 different values: standard deviation and z-scores.

I'm trying to figure out how to write these into a csv file in different columns in R. I have about 90 different standard deviation values and 90 zscore values. So I want all my standard deviation values in 1 column and all my z-scores in the 2nd column. I also want to label the column with standard deviation and z-score.

Right now I'm calculating and writing them into separate files in a loop like this:

write.table(st\_dev, file="st\_dev.csv", sep=",", row.names=FALSE, col.names=FALSE, append=TRUE)

write.table(z\_score, file="z.csv", sep=",", row.names=FALSE, col.names=FALSE, append=TRUE)

Edit:

for(i in 1:130)

{

y=test3[i,3:52]

z=as.numeric(y)

average = mean(z)

st\_dev = sd(z)

dfrm <- data.frame(average=average, st\_dev=st\_dev)

write.table(dfrm, file="av\_st.csv", sep=",", row.names=FALSE, col.names=TRUE, append=TRUE)

}

How can I have the column headers just once at the top instead of printing it for each value in the csv file? Thanks

For me to do histogram on a column basis

hist(Aug\_2016$Lecture)

Getting help with Justin:

Remember that "mean" needs numeric data. If you have mixed class data, then use:

numdata<-data[sapply(data, is.numeric)]

sapply(numdata, mean, na.rm = T) # Returns a vector

<https://stackoverflow.com/questions/21807987/calculate-the-mean-for-each-column-of-a-matrix-in-r>

|  |
| --- |
| numdata<-Aug\_2016[sapply(Aug\_2016, is.numeric)]  > sapply(numdata, mean, na.rm = T)  Lecture Responsive Student\_Guide Lab\_Exercises  37.14885 38.02105 34.88273 35.50847  Classroom Computer Network See\_Hear  33.54565 35.35484 35.25532 36.62420  Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details  33.46072 33.66525 34.00428 |
|  |
| |  | | --- | | > | |

Just a copy and paste of R code as of Tuesday night

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aug\_2016 <- rbind(Aug\_2016,data.frame,("MPLS\_All"))  Error in rep(xi, length.out = nvar) :  attempt to replicate an object of type 'closure'  > Aug\_2016[nrow(Aug\_2016) + 1] = MPLS\_All  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, value = c(37.1488469601677, :  new columns would leave holes after existing columns  > Aug\_2016[nrow(Aug\_2016) + 1] = c("MPLS\_All")  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, value = "MPLS\_All") :  new columns would leave holes after existing columns  > Aug\_2016[nrow(Aug\_2016) + 1,] = c("MPLS\_All")  ted  > View(Aug\_2016)  > MPLS\_All  Lecture Responsive Student\_Guide Lab\_Exercises  37.14885 38.02105 34.88273 35.50847  Classroom Computer Network See\_Hear  33.54565 35.35484 35.25532 36.62420  Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details  33.46072 33.66525 34.00428  > Aug\_2016[nrow(Aug\_2016) + 1,] = c(MPLS\_All)  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, , value = c(37.1488469601677, :  replacement has 11 items, need 38  > Aug\_2016[nrow(Aug\_2016) + 1,] = c("MPLS\_All")  There were 24 warnings (use warnings() to see them)  > View(Aug\_2016)  > Aug\_2016[nrow(Aug\_2016) + 1,] = c(MPLS\_All)  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, , value = c(37.1488469601677, :  replacement has 11 items, need 38  > View(Aug\_2016)  > View(MPLS\_All)  > MPLS\_Student\_Guide <- mean(Aug\_2016$Student\_Guide,na.rm = TRUE)  Warning message:  In mean.default(Aug\_2016$Student\_Guide, na.rm = TRUE) :  argument is not numeric or logical: returning NA  > Aug\_2016[nrow(Aug\_2016) + 1,] = c(MPLS\_All)  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, , value = c(37.1488469601677, :  replacement has 11 items, need 38  > View(MPLS\_All)   |  | | --- | | Aug\_2016[nrow(Aug\_2016) + 1,] = c(MPLS\_All)  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, , value = c(37.1488469601677, :  replacement has 11 items, need 38  > View(MPLS\_All)  > Aug\_2016[nrow(Aug\_2016) + 1,] = c("MPLS\_All")  There were 24 warnings (use warnings() to see them)  > View(MPLS\_All)  > Aug\_2016[nrow(Aug\_2016) + 1,] = c("MPLS\_All")  There were 24 warnings (use warnings() to see them)  > View(Aug\_2016)  > | |  | | |  | | --- | | > | | |
|  |
| |  | | --- | |  |   <https://www.r-bloggers.com/difference-between-na-and-nan-in-r/> |

 NaN (“Not a Number”) means 0/0

 NA (“Not Available”) is generally interpreted as a missing value and has various forms – NA\_integer\_, NA\_real\_, etc.

 Therefore, NaN ≠ NA and there is a need for NaN and NA.

 is.na() returns TRUE for both NA and NaN, however is.nan() return TRUE for NaN (0/0) and FALSE for NA.

|  |
| --- |
| numdata<-Aug\_2016[sapply(Aug\_2016, is.numeric)]  > sapply(numdata, mean, na.rm = T)  Lecture Responsive Student\_Guide Lab\_Exercises Classroom Computer Network  37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532  See\_Hear Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details  36.62420 33.46072 33.66525 34.00428  > View(sapply)  > Aug\_2016[nrow(Aug\_2016) + 1,] = sapply(numdata,mean, na.rm = T)  Error in `[<-.data.frame`(`\*tmp\*`, nrow(Aug\_2016) + 1, , value = c(37.1488469601677, :  replacement has 11 items, need 38  > MPLS\_All <- sapply(numdata, mean, na.rm = T)  > View(MPLS\_All) |
|  |
| |  | | --- | | Gives me the following View…!! Getting closer. | |

<https://www.statmethods.net/management/merging.html>

# Merging Data

## Adding Columns

To merge two data frames (datasets) horizontally, use the **merge** function. In most cases, you join two data frames by one or more common key variables (i.e., an inner join).

# merge two data frames by ID  
total <- merge(data frameA,data frameB,by="ID")

# merge two data frames by ID and Country  
total <- merge(data frameA,data frameB,by=c("ID","Country"))

## Adding Rows

To join two data frames (datasets) vertically, use the **rbind** function. The two data frames **must** have the same variables, but they do not have to be in the same order.

total <- rbind(data frameA, data frameB)

If data frameA has variables that data frameB does not, then either:

1. [Delete](https://www.statmethods.net/management/subset.html) the extra variables in data frameA or
2. Create the additional variables in data frameB and [set them to NA](https://www.statmethods.net/input/missingdata.html) (missing)

before joining them with **rbind( )**.

## Going Further

To practice manipulating data frames with the dplyr package, try [this interactive course on data frame manipulation in R.](https://www.datacamp.com/courses/dplyr-data-manipulation-r-tutorial)

> total <- rbind(Aug\_2016, MPLS\_All)

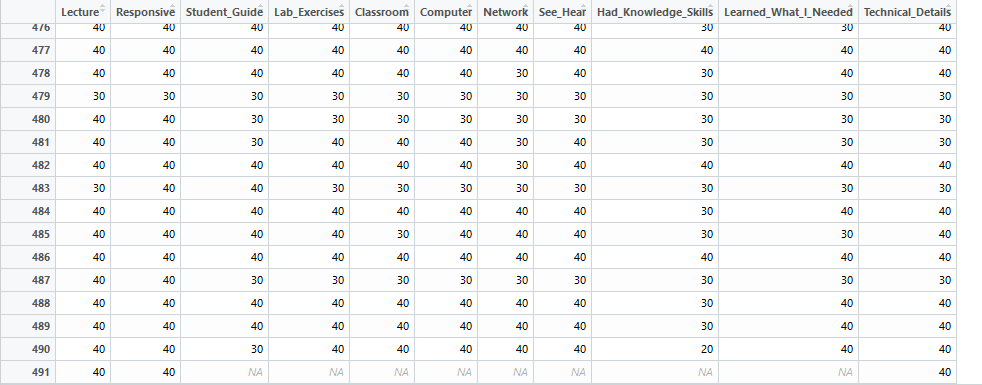
There were 24 warnings (use warnings() to see them)

> View(total)

numdata<-Aug\_2016[sapply(Aug\_2016, is.numeric)]

> View(numdata)

Produced this - closer…!

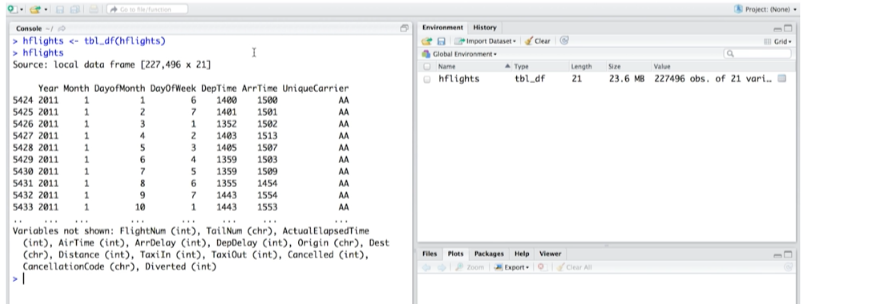


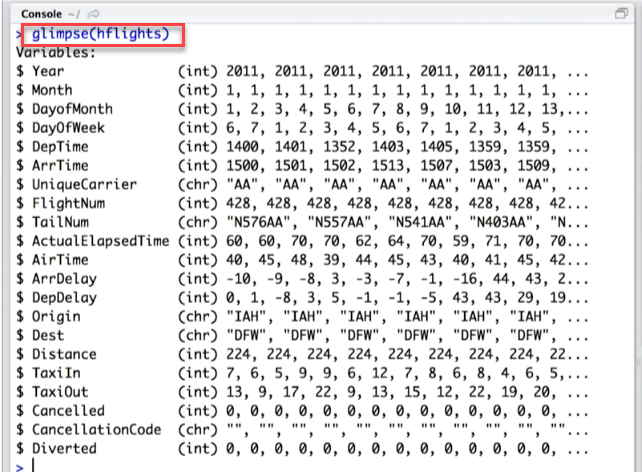
Try this

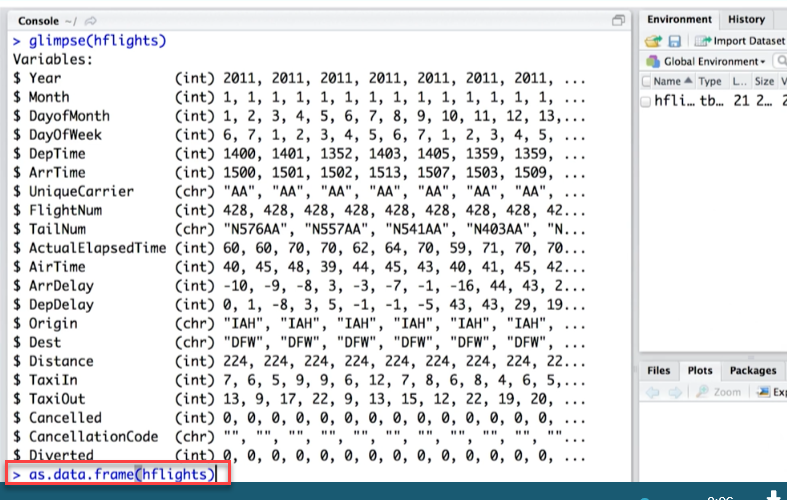
New\_MPLS\_total <- rbind(numdata, Analysis\_Aug\_2016)

|  |
| --- |
|  |
|  |
| |  | | --- | | > | |

During datacamp course:







For my project:

I added the reshape package in order to hopefully help with combining my three datasets together for the SD,Mean, and regular individual scores. It did not work.

|  |
| --- |
| > |
|  |
| |  | | --- | |  | |

For my project:

I used the dplyr package for the bind\_rows command which is what finally worked…!! After I changed the column names of SD\_Lecture and such to Lecture

For my project I am trying to utilize ggplot2 in order to show bar graphs and maybe other data after I learn what is available.

Here is a sample script

# make some data

months <-rep(c("jan", "feb", "mar", "apr", "may", "jun",

"jul", "aug", "sep", "oct", "nov", "dec"), 2)

chickens <-c(1, 2, 3, 3, 3, 4, 5, 4, 3, 4, 2, 2)

eggs <-c(0, 8, 10, 13, 16, 20, 25, 20, 18, 16, 10, 8)

values <-c(chickens, eggs)

type <-c(rep("chickens", 12), rep("eggs", 12))

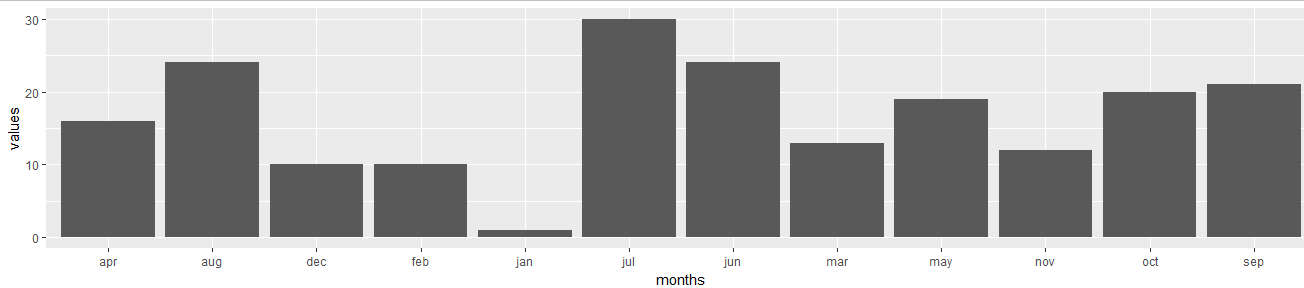
mydata <-data.frame(months, values)

library(ggplot2)

p <-ggplot(mydata, aes(months, values))

p +geom\_bar(stat = "identity")

And it produces this:



Now try and figure out how to get this to work with my code

Thursday morning 1/11/2018

<https://www.statmethods.net/stats/descriptives.html>

sapply(MPLS\_Scores, mean, na.rm=TRUE)

Lecture Responsive Student\_Guide Lab\_Exercises

37.14885 38.02105 34.88273 35.50847

Classroom Computer Network See\_Hear

33.54565 35.35484 35.25532 36.62420

Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details

33.46072 33.66525 34.00428

> for\_bar\_chart <- sapply(MPLS\_Scores, mean, na.rm=TRUE)

> barplot(for\_bar\_chart)

> sapply(Mean\_Analysis\_Aug\_2016)

Error in match.fun(FUN) : argument "FUN" is missing, with no default

> sapply(Mean\_Analysis\_Aug\_2016,mean)

Lecture Responsive Student\_Guide Lab\_Exercises

38.57442 39.01053 37.44136 37.75424

Classroom Computer Network See\_Hear

36.77282 37.67742 32.62766 38.31210

Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details

36.73036 36.83263 37.00214

> for\_bar\_chart <- sapply(Mean\_Analysis\_Aug\_2016, mean)

> barplot(for\_bar\_chart)

Working with Justin

Column\_mean <- sapply(MPLS\_Scores,mean,na.rm=TRUE)

> Column\_SD <- sapply(MPLS\_Scores,sd,na.rm=TRUE)

> Column\_mean

Lecture Responsive Student\_Guide Lab\_Exercises Classroom Computer

37.14885 38.02105 34.88273 35.50847 33.54565 35.35484

Network See\_Hear Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details

35.25532 36.62420 33.46072 33.66525 34.00428

> Column\_SD

Lecture Responsive Student\_Guide Lab\_Exercises Classroom Computer

4.877153 4.194579 5.569788 5.428158 5.682891 5.602965

Network See\_Hear Had\_Knowledge\_Skills Learned\_What\_I\_Needed Technical\_Details

5.602193 4.866840 5.697895 5.635629 5.599661

> bind\_rows(Column\_mean,Column\_SD)

# A tibble: 2 x 11

Lecture Responsive Student\_Guide Lab\_Exercises Classroom Computer Network See\_Hear Had\_Knowledge\_Skills Learned\_What\_I\_Needed

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 37.148847 38.021053 34.882729 35.508475 33.545648 35.354839 35.255319 36.62420 33.460722 33.665254

2 4.877153 4.194579 5.569788 5.428158 5.682891 5.602965 5.602193 4.86684 5.697895 5.635629

# ... with 1 more variables: Technical\_Details <dbl>

> Summary\_stats <- bind\_rows(Column\_mean,Column\_SD)

> View(Summary\_stats)

|  |
| --- |
| as.numeric(Column\_mean)  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > ggplot(Summary\_stats,aes(x=Summary\_Col\_Names, y=as.numeric(Column\_mean)))  Error: Aesthetics must be either length 1 or the same as the data (2): x, y  > unname(unlist(Column\_mean))  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > c(unname(unlist(Column\_mean)))  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > trying <-c(unname(unlist(Column\_mean)))  > barplot(trying)  > barplot(as.numeric(Column\_mean))  > barplot()  Error in barplot.default() :  argument "height" is missing, with no default  > barplot(as.numeric(Column\_SD))  > barplot(Column\_mean)  > Column\_mean\_II <- sapply(MPLS\_Scores,mean,na.rm=TRUE)  > barplot(Column\_mean\_II)  > barplot(as.numeric(Column\_SD))  > barplot(Column\_mean\_II)  > Column\_mean\_II <- sapply(MPLS\_Scores,mean,na.rm=TRUE)  >  > class(Summary\_stats)  [1] "tbl\_df" "tbl" "data.frame"  > class(Column\_mean)  [1] "numeric" |
|  |
| |  | | --- | | > | |

More work with Justin

|  |
| --- |
| as.numeric(Column\_mean)  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > ggplot(Summary\_stats,aes(x=Summary\_Col\_Names, y=as.numeric(Column\_mean)))  Error: Aesthetics must be either length 1 or the same as the data (2): x, y  > unname(unlist(Column\_mean))  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > c(unname(unlist(Column\_mean)))  [1] 37.14885 38.02105 34.88273 35.50847 33.54565 35.35484 35.25532 36.62420  [9] 33.46072 33.66525 34.00428  > trying <-c(unname(unlist(Column\_mean)))  > barplot(trying)  > barplot(as.numeric(Column\_mean))  > barplot()  Error in barplot.default() :  argument "height" is missing, with no default  > barplot(as.numeric(Column\_SD))  > barplot(Column\_mean)  > Column\_mean\_II <- sapply(MPLS\_Scores,mean,na.rm=TRUE)  > barplot(Column\_mean\_II)  > barplot(as.numeric(Column\_SD))  > barplot(Column\_mean\_II)  > Column\_mean\_II <- sapply(MPLS\_Scores,mean,na.rm=TRUE)  >  > class(Summary\_stats)  [1] "tbl\_df" "tbl" "data.frame"  > class(Column\_mean)  [1] "numeric" |
|  |
| |  | | --- | | > | |

For Runjini

<https://www.r-graph-gallery.com/barplot/>

Sunday working with Justin:

|  |
| --- |
| class(Revised\_Barplot\_SD)  [1] "matrix"  > barplot(Column\_SD,col="green", border = "red", main="Standard\_Deviation", ylab="STD", ylim = c(0, 6))  > barplot(Column\_SD,height=Column\_SD, beside = TRUE, col="blue", border = "red", main="Standard\_Deviation", ylab="STD", ylim = c(0, 6))  > barplot(height=Column\_SD, beside = TRUE, col="green", border = "red", main="Standard\_Deviation", ylab="STD", ylim = c(0, 6))  > labels <- list('bar1', 'bar2', 'bar3', 'bar4', 'bar5', 'bar6', 'bar7', 'bar8', 'bar9', 'bar10', 'bar11')  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="green", border = "red", main="Standard\_Deviation", ylab="STD", ylim = c(0, 6))  > labels <- list('Lecture', 'Responsive', 'Guide', 'Labs', 'Classroom', 'Computer', 'Network', 'See\_Hear', 'Had\_Skills', 'Learned', 'Details')  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="green", border = "red", main="Standard\_Deviation", ylab="STD", les=2, ylim = c(0, 6))  Warning messages:  1: In plot.window(xlim, ylim, log = log, ...) :  "les" is not a graphical parameter  2: In axis(if (horiz) 2 else 1, at = at.l, labels = names.arg, lty = axis.lty, :  "les" is not a graphical parameter  3: In title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...) :  "les" is not a graphical parameter  4: In axis(if (horiz) 1 else 2, cex.axis = cex.axis, ...) :  "les" is not a graphical parameter  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="darkgreen", border = "red", main="Standard\_Deviation", ylab="STD",ylim = c(0, 6))  Warning messages:  1: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  2: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  3: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  4: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  5: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  6: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  7: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  8: In doTryCatch(return(expr), name, parentenv, handler) :  "les" is not a graphical parameter  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="darkgreen", border = "red", srt=45, main="Standard\_Deviation", ylab="STD",ylim = c(0, 6))  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="green", border = "red", srt=45, main="Standard\_Deviation", ylab="STD",ylim = c(0, 6))  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="blue", border = "red", main="Standard\_Deviation", ylab="STD", las=2, ylim = c(0, 6))  > barplot(height=Column\_SD, names.arg=labels, beside = TRUE, col="blue", border = "red", main="Standard\_Deviation", ylab="STD", las=2, srt=60, ylim = c(0, 6)) |
|  |
| |  | | --- | | >  https://stackoverflow.com/questions/34146635/r-barplot-put-numbers-on-top-of-bar  <https://stat.ethz.ch/pipermail/r-help/2007-January/123805.html> | |

For another sprint:

<http://datavizproject.com/>