DR. CLARK'S REFERENCE CARD FOR GETTING STARTED WITH R

Object ***classes***: Very important!! Determine how R will handle that object: `as.numeric()`, `as.integer()`, `as.character()`, `as.factor()`

Operators

`<-`	Assignment operator.
`> , < , >= , <= , !=`	Greater, less than, not equal to
`#`	Comment symbol
`" " or ' '`	Use to surround text strings
`,`	(comma) Separator between items
`NA`	Missing data. If your data contain "NA"s they may affect calculations. Many functions accept the argument `na.rm=T`
`1:3`	the vector 1, 2, 3
`+, -, /, *, ^`	Typical math notation for addition, subtraction, division, multiplication, exponents
`~`	Formula symbol to use instead of equal sign in formulas (e.g. y ~ x)

General functions

`c()`	Create a vector of the specified elements inside
`which()`	Find elements inside a vector that satisfy a condition
`str()`	Learn more about the structure of an object
`head() or tail()`	View the first or last 6 entries in a data.frame
`help()`	Access the help documentation for a function
`install.package()`	Install a new package for the first time
`library()`	Load an installed package
`summary()`	Output depends on the nature of the object provided
`sqrt()`	Take the square root
`log()`	Takes the natural log ln
`class()`	Get information about or set the "class" of an object
`data()`	Load a provided dataset
`View(m1)`	view data frame m1

Indexing

`m1[r1, c1]`	view entry at row 1, column 1, where r1 and c1 are numbers
`m1[ , c1]`	view entirety of column 1
`m1[ , 1:3]`	Select or view first three columns
`m1$a1`	Also view entirety of column 1, where a1 is a column name
`m1[ , "a1"]`	Also selects column named a1, where a1 is a name
`m1[ , c("a1", "a2", "a3")]`	Select columns a1, a2, and a3 by name
`m1[ which(m1$a1 == "thing"), "a2"]`	For all rows of column a1 that equal "thing," display value for entry in column a2

Plotting functions

`hist()`	Frequency histogram
`plot(y ~ x), plot(x, y)`	Scatter and line plots
`plot(y ~ x, type = "l")`	Line plot without points
`plot(y ~ x, type = "b")`	Lines and points plotted
`points()`	Add points to existing plot (overlay)
`abline()`	Lines from a to b.
`barplot()`	Barplots
`boxplot(y ~ x), boxplot(y ~ x * z)`	Boxplots
`axis(side=1, at=1:3, tick=TRUE, c("label 1", "label 2", "label 3")`	Create a "custom" plot axis
`mtext()`	Add text to the margins of a plot
`par(mfrow = c(1, 2))`	Create a 2-panel figure with 1 row and 2 columns
`legend("bottomright", fill=c("red", "blue"), legend = c("first thing", "second thing"))`	Add a legend to the plot
`pdf("filename.pdf", height=5, width=4) ... # plot drawing commands dev.off()`	Will save your plot as a pdf with dimensions that you specify.

Graphical parameters – add as arguments to plotting functions above

`xaxt`	If xaxt="n" the x-axis is set but not drawn (useful in conjunction with axis(side=1, …)
`main`	Main title
`xlab, ylab`	Label for x axis or y axis
`xlim, ylim`	Axis limits for x and y axis
`col`	Color. Check colorbrewer.org for good color schemes.

`lwd`	line width
`pch`	Symbol shape
`cex`	Symbol size

Descriptive Statistics

`max(), min(), mean(), median(), sum(), var(), sd(), range()`	Refer to Guidelines for what each of these means
`sd(x) / sqrt(length(x))`	Calculate the standard error of the mean (no built-in function for this)
`summary(data.frame)`	Summary information for all columns in a data frame
`tapply(x1, list1, function1)`	Apply function to x1 by list1

Comparative Statistics

`aov(y1 ~ x1 * x2, data=m1), anova(y1 ~ x1 * x2, data=m1)`	Two-way Analysis of Variance of response variable y1 as it relates to factors x1 and x2, including the interaction term.
`summary(aov.object) or summary(aov(y~x))`	Returns statistical results of Analysis of Variance object
`TukeyHSD(aov(y~x))`	Post-hoc pairwise Tukey test for the Anova specified
`lm(y1 ~ x1, data=m1)`	Linear regression of response variable y1 as related to continuous predictor variable x1
`summary(lm.object) or summary(lm(y~x))`	Returns statistical results of linear regression object
`t.test(y ~ x, data=m1) or t.test(m1$y, m2$y)`	Unpaired, two-sided t-test. See guidelines for one-sided tests.
`t.test(y ~ x, data=m1, paired="true")`	Paired t-test
`chisq.test()`	Method for calculating a chi-squared statistic in R. Recommend you calculate this by hand instead.

Advanced Methods: Line plots for 2 or more lines that include points, lines, and error bars. Assumes each treatment group is in its own dataset named "tx1" etc, which contains a column that indicates the x-axis position called "xvar" and another column with the measurement of interest called "yvar"

```
library("tidyverse") # run install.packages() the first time to
install this package

# Calculate standard deviation for first treatment group.
Repeat process for subsequent treatment groups.

tx1sd <- tx1 %>%
  group_by(xvar) %>%
  summarise(meany = mean(yvar), sdy = sd(yvar))

# Start plot:
plot(tx1sd$meany ~ tx1sd$xvar, ...)

# Overlay next lines for next treatment group:
points(tx2sd$meany ~ tx2sd$xvar, ...)

# Add standard error bars for each treatment group:
arrows(tx1sd$xvar, tx1sd$meany - tx1sd$sdy, tx1sd$xvar,
tx1sd$meany + tx1sd$sdy, length = 0.05, angle = 90, code = 3)
```