Many useful Excel and R functions

Notes: NA = not applicable; 1:, 2: ... = alternative methods to do the same thing; # ... = comments if necessary

Description	**Excel (Calc usually too)**	**R**
Getting started		
Install program	you probably already have it	http://cran.r-project.org/

Update program	spend money	1: http://cran.r-project.org/
		2: Within R (Windows):
		# install, move, update, quit:
		if(!require(installr))
		{ install.packages("installr");require(installr)}
		updateR(F, T, T, F, T, F, T)

Run command C	=C # "=" everywhere below too	C # no need for "=" before it
Get help on function x	1: click *fx* symbol, find function, double-click it, click 函數說明 2: search the web	1: ?x # Needs exact match 2: help("x") # same as ?x 3: ??x # Fuzzy match 4: search the web # Usage shows syntax and # defaults; Arguments shows # input; Value shows output
Get help on general function F that works differently for object types A vs. B	NA	?F.A vs. ?F.B # For example: ?summary.lm ?summary.glm ?summary.aov ?summary.table ?plot.table
Put value V into variable x	type V into cell x	1: x = V 2: x <- V 3: V -> x
Put value V into both x and y	type V into cell x, drag to y	x = y = V # Cool!
Load tab-delimited file "F" into data frame D, first row as variable names	1: copy/paste from text file 2: open file within Excel	1: D = read.delim("F") 2: D = read.table("F",T)
Load tab-delimited file on the web at http://www/F into data frame D	use File/Open, then write/paste http://www /F	D = read.delim("http://www /F ")
Load space-delimited file "F" into data frame D, first row as variable names	same as above, but then split by space (空格)	1: D = read.table("F",T) 2: D = read.delim("F",sep=" ")
Load comma-delimited file "F" into data frame D, first raw as variable names	1: open within Excel, splitting columns by "," 2: copy/paste from text file, then split columns by ","	1: D = read.csv("F") 2: D = read.table("F", sep=",", header=T)
Object x inside object O (e.g. data frame)	click appropriate row or column	1: O$x 2: attach(O); x; detach(O) # "$" also applies to function # outputs, e.g.: summary(lm(y~x))$residuals
Show the local file directory	NA	dir()
Timing script S	NA	now = proc.time() S proc.time() - now

Vectors, matrices, lists and data frames		
Create the vector of numbers x, y, z	x, y, z in adjacent cells (vertical or horizontal)	c(x,y,z)
Omit NA (not available) data in object O	Math functions automatically ignore strings like "NA"	na.omit(O)
Create number series 1, 2, ... n	type 1 & 2, select, then drag lower right corner	1:n
Create number sequence 1, 3, 5,..., n	type 1 & 3, then drag corner	seq(1,n,by=2)
Repeat number x for n times	drag cell with x from corner	rep(x,n)
Add 1 to the numbers 2,5,7 to get 3,6,8	NA	1+c(2,5,7) # grammatical!
Number of values in vector x	COUNT(x) # only numbers	length(x) # numbers or strings (all same type)
Number of values in vector x that are greater than y	COUNTIF(x,">y")	length(x[x>y])
Convert number x into string "x"	TEXT(x,"#.##") # 2 decimals	as.character(x)
Look up x in table T, find what's in column C (in x's row)	VLOOKUP(x,T,C,FALSE)	C[T==x]
Create data frame D with columns x & y	NA	D = data.frame(x,y)
Create data frame D1 that's a subset of data frame D, such that x > 1	NA	D1 = subset(D,D$x>1)
Count number of rows in data frame D	COUNT(D) # select a column	nrow(D)
Count number of columns in data frame D	COUNT(D) # select a row	ncol(D)
Put columns x and y side by side	copy/paste them as you like	cbind(x,y)
Put rows x and y one on top of the other	copy/paste them as you like	rbind(x,y)
Create a vector V with N zeros	type 0, drag corner	V = numeric(N)
Create an empty matrix M with C columns and R rows (all "NA" = not available)	type "NA", drag corner down to make column, then drag again rightward (or vice versa)	M = matrix(ncol = C, nrow=R)
Create the matrix a c b d	type a, b, c, d into the appropriate cells	1: matrix(c(a,b,c,d),nrow=2) 2: matrix(c(a,b,c,d),ncol=2)
Flip (transpose) n × m matrix M (n rows, m columns) into m × n matrix M'	copy matrix, paste in new place using Paste Special (選擇性貼上) and Transpose (轉置)	t(M)
Add column names "A" & "B" to two-column matrix M	NA	colnames(M) = c("A","B")
Add row names "A" & "B" to two-row matrix M	NA	rownames(M) = c("C","D")
Show column and row names of matrix M	NA	colnames(M); rownames(M)
Show column names in data frame D	NA	1: names(D) 2: colnames(D)
For vector x, find the ith position	click on the appropriate cell	x[i]
For the data frame (or matrix) x, find the ith row and jth column	click on the appropriate cell	x[i,j]
All values in data frame D on row x	click row number x	D[x,]
All values in data frame D in ith column named "x"	click column letter "x"	1: D[,i] # Using number 2: D[,"x"] # Using name
Show first six rows of data frame D	scroll to the top of the sheet	head(D)

<table>
<tr><td>Show last six rows of data frame D</td><td>scroll to the bottom of the sheet</td><td>tail(D)</td></tr>
<tr><td>Sort column x into alphanumerical order</td><td>use A→Z dialog box</td><td>sort(x)</td></tr>
<tr><td>Sort columns x and y in data frame D into the order defined by x</td><td>use A→Z dialog box</td><td>1: D[order(D$x),]
2: library(dplyr)
arrange(D, x)</td></tr>
<tr><td>Remove repeats in vector A</td><td>sort column A, then in B2: IF(A2=A1,"",A2), drag down, copy/paste value, sort column B</td><td>unique(A)</td></tr>
<tr><td>Combine tables D1 and D2 by matching same-named column x</td><td>use VLOOKUP cleverly</td><td>merge(D1,D2)</td></tr>
<tr><td>Combine smaller item, subject, and response files I, S, R into one large file</td><td>use VLOOKUP cleverly</td><td>merge(R,I) # Same item IDs
merge(R,S) # Same subj IDs</td></tr>
<tr><td>Create a list with number 1 and string "a"</td><td>just type/paste into cells</td><td>list(1,"a") # c() can't do this</td></tr>
<tr><td>Create a list L of vectors (1,2) and (3,4,5)</td><td>NA</td><td>L = list(c(1,2),c(3,4,5))</td></tr>
<tr><td>Second element in first element in list L</td><td>NA</td><td>L[[1]][2] # e.g. = 2 for above</td></tr>
<tr><td>Split vector or data frame X by factor F</td><td>NA</td><td>split(X,F) # Outputs a list</td></tr>
<tr><td>Check if vector x elements are in vector y</td><td>NA</td><td>is.element(x,y)</td></tr>
<tr><td>Remove elements that are in vector x from vector y</td><td>NA</td><td>1: y[y!=x]
2: setdiff(y,x)</td></tr>
<tr><td>Cut continuous values in vector x into n equal-sized bins, creating new factor B</td><td>NA</td><td>B = cut(x,n)</td></tr>
<tr><td colspan="3">Logic</td></tr>
<tr><td>True</td><td>TRUE</td><td>1: TRUE
2: T # never "true" or "t"</td></tr>
<tr><td>False</td><td>FALSE</td><td>1: FALSE
2: F # never "false" or "f"</td></tr>
<tr><td>If x is true then value y, otherwise value z</td><td>IF(x,y,z)</td><td>if (x) {y} else {z}</td></tr>
<tr><td>If x is true then command y, otherwise command z</td><td>NA</td><td>if (x) {y} else {z}</td></tr>
<tr><td>x equals y (true or false)</td><td>x=y</td><td>x==y</td></tr>
<tr><td>x doesn't equal y (true or false)</td><td>x<>y</td><td>x!=y</td></tr>
<tr><td>x and y (true only if both x and y are true)</td><td>AND(x,y)</td><td>x & y</td></tr>
<tr><td>x or y (true if either x and/or y is true)</td><td>OR(x,y)</td><td>x | y</td></tr>
<tr><td>Convert logical x into 0 (F) or 1 (T)</td><td>if(x,1,0)</td><td>1: 1*x
2: as.numeric(x)</td></tr>
<tr><td colspan="3">Functions and packages</td></tr>
<tr><td>Add comment y after R code line x</td><td>NA</td><td>x # y</td></tr>
<tr><td>Run command x, then command y</td><td>NA</td><td>1: x
y
2: x; y</td></tr>
<tr><td>Repeat command x for n times (for-loop)</td><td>NA</td><td>for (i in 1:n) {x} # 1:n is vector!</td></tr>
<tr><td>Print out "JM" one letter at a time</td><td>NA</td><td>for (i in c("J","M")) {print(i)}</td></tr>
<tr><td>Create a new function Fun that takes argument x and outputs value y</td><td>need to use VBA to create a macro (search web for help)</td><td>Fun = function(x) {
return(y)
}</td></tr>
<tr><td>Compute means of rows in matrix M</td><td>AVERAGE(row), drag down</td><td>apply(M,1,mean)</td></tr>
<tr><td>Find sum of columns in matrix M</td><td>AVERAGE(col), drag right</td><td>apply(M,2,sum)</td></tr>
<tr><td>Compute by-subj means for variable x in data set D
or any one-argument function (e.g. sum)</td><td>AVERAGE(x) # assumes subj defines rows (columns) and x values are in a matrix</td><td>1: apply(Dx,Dsubj,mean)
2: library(dplyr)
summarize(group_by(D, subj),
mean(x))</td></tr>
<tr><td>Compute means for variable y when another variable x > 23</td><td>AVERAGEIFS(y,x,">23")
assumes x and y are columns like in R</td><td>mean(y[x>23])</td></tr>
</table>

Compute means for variable y for factors A (A1 vs. A2) & B (B1 vs. B2), and put them in a table	DAVERAGE(database,field, criteria) # "database" = R-style data # "field" = factor name # "criteria" = minitable like: A B (factor names) A1 B2 (one level each)	tapply(y,list(A,B),mean) # more factors and more levels also work
Compute means for variable y for all levels of factor A & B and their interaction, and put into data frame D with columns y, A, B	N	D=aggregate(y ~ A * B, mean)
Read and run R code in local File	NA	1: File/Source R code... menu 2: source("File")
Read and run R code at http://www/File	NA	source("http://www/File ")
Install package P from http://www/P	NA	1: Packages/Install package(s) menu 2: install.packages("http://www/P")
Load package P	NA	1: library(P) # Error if no P 2: require(P) # FALSE if no P
Strings		
Concatenate strings "x" & "y" into "xy"	"" & "y"	paste("x","y",sep="")
Number of characters in string x	LEN(x)	nchar(x)
First n characters in string x	LEFT(x,n)	substring(x,1,n)
Last n characters in string x	RIGHT(x,n)	substring(x,nchar(x)-n+1, nchar(x))
n characters in string x starting at a	MID(x,a,n)	substring(x,a,a+n-1)
Characters a to b in string x	MID(x,a,b-a+1)	substring(x,a,b)
Split string S at space " "	Menu: Data / Text to columns	unlist(strsplit(S," "))
Replace a with b everywhere in x	1: Menu: Home / Find / Replace 2: SUBSTITUTE(x,a,b)	gsub(a,b,x)

Handling Unicode in Windows	NA	library(readr)
		read_lines() # instead of readLines()
		read_delim() # instead of read.delim()

Basic math		
Round number x to y decimal places	ROUND(x,y)	round(x,y)
Round x down to nearest integer	ROUNDDOWN(x,0)	floor(x)
Round x up to nearest integer	ROUNDUP(x,0)	ceiling(x)
Min, max, sum of vector x	MIN(x), =MAX(x), =SUM(x)	min(x); max(x); sum(x)
Square root of x ($\sqrt{x}$)	SQRT(x)	sqrt(x)
Range of vector x (min & max)	MIN(x), =MAX(x)	range(x) # Output is vector
Square of x (x^2)	x^2	x^2
Logarithm of x, base 10	LOG(x)	log10(x)
Natural log of x (base *e* = 2.718...)	LN(x)	log(x)
e^x (inverse of natural log)	EXP(x)	exp(x) # exp(log(x)) == x
Graphs		
Make a graph	1: poke around chart menu (depends on Excel version) 2: search the web (ditto below)	1: plot, boxplot, etc 2: install ggplot2 package: library(ggplot2) qplot(...) # Simple plots ggplot(...) # Complex plots
Get help with graphs	1: poke around chart menu 2: search the web	1: ?plot, ?boxplot, ?par 2: search the web
Make a scatterplot of vectors x and y	poke around chart menu	plot(x,y)
Save a graph to a file	export the Excel file as HTML, which puts graphs into a folder	when graph window is open, use File/Save as menu

Put six plots into a 2-row by 3-column arrangement	lots of plotting	1:par(mfrow=c(2,3)) for(i in 1:6){plot(runif(10))} 2:layout(matrix(1:6,nrow=2)) for(i in 1:6){plot(runif(10))}

Make a scatterplot of vectors x and y with x-axis label "Age", y-axis label "Accuracy", with x values from 0.5 to 1 and y values from 0 to 0.5
# Same methods work for most plots, including histograms 	
	poke around chart menu	plot(x,y,
		  xlab = "Age",
		  ylab = "Accuracy",
		  xlim=c(0.5,1),
		  ylim=c(0,0.5)
		)

Make x & y scatterplot with no numbers	poke around chart menu	plot(x,y,xaxt="n",yaxt="n")
Add horizontal line to existing plot at y=3	NA	abline(h=3)
Add vertical line to existing plot at x = 7	NA	abline(v=7)

Make a line plot of x (on x-axis) and y (on y-axis)	poke around chart menu	plot(x,y,type="l")
		# Make sure x is sorted first!
		# type="l": line;
		  # type="p" (points) default;
		  # ?plot for other types
		  # ?points pch for other dot
		  # shapes

Make a bar graph of crossed values of Y = a,b,c,d as a function of factors F= F1,F2 and G = G1, G2 in matrix M, with y-axis starting at zero, and M and barplot like so:
M: Barplot: [* G1]
      F1   F2   * #   * #   [# G2]
G1   a   c   a b   c d
G2   b   d   F1   F2
	adjust Excel's automatic y-axis to start it at zero (recommended by many statisticians to make scale clear); R acts like Excel here, in treating columns (F) as the main label (in names.arg) and rows (G) as legend label (in legend.test)	1:M = matrix(c(a,b,c,d),
		      nrow=2)
		  barplot(M, beside=T,
		    names.arg=c("F1","F2"),
		    legend.text=c("G1","G2"),
		    ylab = "Y")
		2: search web or books for help on using ggplot2

Make same bar graph as above, but use y-axis range a to b, where a is not zero	poke around chart menu	barplot(M, beside=T,
		    legend.text=c("F1","F2"),
		    ylab = "Y", ylim=c(a,b),
		    xpd=F, # Keep bars inside
		    xaxt="n") # No x label yet
		axis(side=1, at=c(2,5),
		    labels=c("G1", "G2"))
		box(bty="l") # lower-case L

Plot standard histogram of sample S (S = vector of numbers)	Analysis toolbox: 直方圖 # use about 10 equal-sized bins	hist(S)
Change number/size of bars in histogram for S (remember for histograms, bar area is what matters, not bar height)	Analysis toolbox: 直方圖 # enter different bins	1: hist(S, breaks=3) # 3 bars 2: hist(S, breaks=c(0,10)) # Breaks at points 0 and 10
Make box (and whiskers) plot	NA	boxplot(...) # cf. ?boxplot
Plot density of sample S	NA	plot(density(S))

Make line plot with solid line for variable x1 and dashed line for variable x2 with dependent variable y	poke around chart menu	plot(x1,y,type="l")
		lines(x2,y,lty=2)
		  # lty is line type
		  # lty=1 (solid) is default
		  # lty=2 is dashed
		  # lty=3 is dotted
		  # lwd=2 is wider

Add a legend at the top right for a line plot showing that the solid line represents Cats and the dashed line represents Dogs	poke around chart menu	legend("topright",
		  lty=c(1,2),
		  legend=c("Cats","Dog"))

<table>
<tr>
<td>Add upper + lower error bars to bar plot B with n means M (vector), where each half of the error bar has length E (e.g., E = 1 sd, or E = SE, or E = one half of the 95% confidence interval)</td>
<td>make bar plot, search menu for error bars, enter values you want</td>
<td>1:source("http://www.ccunix.
ccu.edu.tw/~lngproc/
errorbar_Rcode.txt")
E.bars = rep(E,n)
error.bar(B,M,E.bars)
2: library(ggplot2)
B + geom_errorbar(
aes(ymin=M-E,
ymax=M+E))</td>
</tr>
<tr>
<td>Add linear regression line to scatter plot (x on x-axis, y on y-axis)</td>
<td>right-click dots, choose 加上趨勢線, then keep 線性 default</td>
<td>plot(x,y)
abline(lm(y~x))</td>
</tr>
<tr>
<td>Add local regression line to scatter plot (x on x-axis, y on y-axis)</td>
<td>right-click dots, choose 加上趨勢線, then choose 移動平均</td>
<td>plot(x,y)
lines(predict(loess(y~x)))
# sort x first</td>
</tr>
<tr>
<td>Make trellis plot for scatterplot y ~ x1 * x2 (y, x1, x2 all numerical, and you want to visualize the x1 × x2 interaction) in data frame D, with linear best-fit lines for each</td>
<td>sort data by x1, divide x1 into a few (3-6) subsets, plot y~x2 for each subset (like method 3 for R)</td>
<td>1: library(lattice)
x1.eq = equal.count(D$x1)
xyplot(D$y ~ D$x2 | x1.eq,
panel = function(x, y) {
panel.xyplot(x, y)
panel.abline(lm(y~x))
}
)
2: library(ggplot2)
D$x1cuts = cut(D$x1, 7)
qplot(y, x2, data=D,
facets=~x1cuts) +
stat_smooth(method
="lm")
3: par(mfrow=c(2,3))#6 plots
D = D[order(D$x1),]
N = nrow(D)
n = ceiling(N/6)
rangey = range(D$y)
rangex2 = range(D$x2)
for (i in 1:6) {
minx1 = D$x1[n*(i-1)+1]
maxx1=D$x1[min(n*i,N)]
D.i = subset(D,
(D$x1 >= minx1 &
D$x1 <= maxx1))
plot(D.i$x2, D.i $y,
xlab="x2", ylab="y",
main = paste("x1:
from",minx1,
"to",maxx1))
abline(lm(y~x2,
data=D.i))
}</td>
</tr>
</table>

<table>
<tr><td>Make trellis plot for scatterplot of y ~ x with linear best-fit lines, with grouping unit g (y & x numerical) in data frame D
Useful for LME and GLMM too</td><td>sort data by g, plot y~x for each g (like method 3 for R)</td><td>1: library(lattice)
xyplot(y ~ x | factor(g),
data = D)
2: library(ggplot2)
qplot(x, y, data=D,
facets = ~g) +
stat_smooth(method
="lm")
3: par(mfrow=c(n,m))
# n & m divide up g neatly
rangex = range(D$x)
rangey = range(D$y)
for (i in 1:length(g)) {
D.i=subset(D,D[D$g==i])
plot(Dx, Dy, main = i,
xlim = rangex,
ylim=rangey)
}</td></tr>
<tr><td>Make 3D scatterplot (x on x-axis, y on y-axis, z on z-axis)</td><td>NA</td><td>library(rgl)
plot3d(x,y,z)</td></tr>
<tr><td>Make 3D scatterplot, split into a series of 2D scatterplots (x & y = independent variables, z = dependent variable)</td><td>NA</td><td>library(ggplot2)
y.cut = cut(y, 7)
qplot(z, x, facets = ~y.cut)</td></tr>
<tr><td>Make mosaic plot of contingency table T</td><td>NA</td><td>mosaicplot(T)</td></tr>
<tr><td>Plot logistic regression model L for y~x</td><td>sort data by x, divide y into bins, within each bin convert y to logits:
LN(AVERAGE(y)/
(1-AVERAGE(y))
make scatterplot of logit(y)~x,
right-click dots, choose 加上趨勢線, then keep 線性 default
Like method 2 for R</td><td>1: plot(x,y)
curve(predict(L,
data.frame(x=x),
type="response"),
add=T)
2: bins = cut(x,10) # Or more
logit.bin = function(x){
prob1 = mean(c(x,0,1))
prob0 = 1-prob1
return(log(prob1/prob0))
}
meanx = tapply(x, bins,
mean)
logity = tapply(y, bins,
logit.bin)
plot(meanx, logity)
abline(lm(logity~meanx))</td></tr>
<tr><td colspan="3">Descriptive statistics</td></tr>
<tr><td>Make a frequency table for sample S</td><td>1: Analysis toolbox: 直方圖
2: see handout for word frequency example</td><td>1: xtabs(~S)
2: table(S)</td></tr>
<tr><td>Make a frequency table cross-classified by factors x and y (x = row, y = columns)</td><td>basically do it by hand</td><td>1: xtabs(~x+y)
2: table(x,y)</td></tr>
<tr><td>Mean of sample S</td><td>1: AVERAGE(S)
2: SUM(S)/COUNT(S)</td><td>1: mean(S)
2: sum(S)/length(S)</td></tr>
<tr><td>Median of sample S</td><td>MEDIAN(S)</td><td>median(S)</td></tr>
<tr><td>Mode of sample S</td><td>MODE(S)</td><td>as.numeric(names(sort(
-table(S)))[1]) # Handout code doesn't work (sorry)</td></tr>
<tr><td>Sample standard deviation of sample S</td><td>STDEV(S)</td><td>1: sd(S)
2: sqrt(sum((S-mean(S))^2)/
(length(S)-1))</td></tr>
<tr><td>Sample variance of sample S</td><td>1: VAR(S)
2: STDEV(S)^2</td><td>1: var(S)
2: sd(S)^2</td></tr>
</table>

Randomness and permutations		
Reset randomizer	Microsoft won't say	set.seed(1) # or any number
Given x things, calculate how many ways to choose y things	COMBIN(x,y)	choose(x,y)
Randomly select x values between 0 and 1 with equal probability	copy and paste RAND() x times	runif(x)
Randomly select x values from a normal (Gaussian) distribution with mean M and standard deviation s	Analysis toolbox: 亂數產生器	rnorm(x,M,s)
Distributions		
Plot normal distribution with mean M and standard deviation s from SD = -3 to +3	create close *z* values from -3 to +3, use NORMDIST(*z*, 1 ,0, FALSE) to get density instead of probability, make line graph	1: curve(dnorm(x), -3, 3) 2: plot(function(z) dnorm(z),-3, 3)
z score of item x in sample S with mean M and standard deviation s	STANDARDIZE(x,M,s)	1: (x-M)/s 2: scale(S)[S==x]
Probability of getting at most x heads in y coin flips (50% probability, x < y/2)	BINOMDIST(x,y,0.5,TRUE)	pbinom(x,y,0.5)
Area to the left of z score in standard normal distribution (mean = 0, SD = 1)	NORMSDIST(z)	pnorm(z)
z score that marks area p to its left in standard normal distribution	NORMSINV(p)	qnorm(p)
Make quantile-quantile norm plot of sample S	sort S from smallest to largest, number from i = 1 to n; define expected normal curve E with NORMSINV((i-0.5)/n); make a scatterplot of S vs. E	qqnorm(S)
Add line to QQ-norm plot (Excel and R don't add quite the same type of line)	right click any dot in QQ-plot, select 加上趨勢線 # adds best-fit line	qqline(S) # draws line between first and third quantiles of ideal
One-tailed *p* value for given t value and df	T.DIST(ABS(t), df, TRUE)	pt(-abs(t), df) # pt assumes a *negative* t!
Two-tailed *p* value for given t value and df	2*TDIST(ABS(t), df, TRUE)	2*pt(-abs(t), df)
Plot t distribution with given df	T.DIST(t,df,FALSE) plus cleverness	curve(dt(x,df),-3,3)
One-tailed *p* value for given F, df1 & df2 (as used in ANOVA and ratio tests)	FDIST(F,df1,df2) # Right side	1: pf(F,df1,df2,lower.tail=F) 2: 1-pf(F,df1,df2)
Plot F distribution with given df1 & df2	FDIST and cleverness	curve(df(x,df1,df2),0,5)
One-tailed p value for given χ^2 & df (as used in chi-squared tests and elsewhere)	CHIDIST(χ^2,df)	1: pchisq(χ^2,df,lower.tail=F) 2: 1- pchisq(χ^2,df)
Plot χ^2 distribution with given df	CHIDIST and cleverness	curve(dchisq(x,df),0,10)
One-tailed p value for at *most* x heads in n fair coin flips (binomial distribution)	BINOMDIST(x,n,0.5,TRUE)	pbinom(x,n,0.5)
Plot binomial distribution for above n	BINOMDIST(x,n,0.5,FALSE) and cleverness	plot((0:n),dbinom((0:n), size=n, prob=0.5))
Factors		
Convert vector S into a factor # Crucial to do this before # repeated-measures ANOVA using aov	NA	as.factor(S)

Convert factor F into an ordinal factor	NA	1: F = ordered(F) 2: F = factor(F, ordered = T) # Creates polynomial coding: # F.L = linear component; # F.Q = F^2 (quadratic) # F.C = F^3 (cubic)
Relevel factor F (levels A and B) so that B is the reference level (0 in dummy coding)	NA	F = relevel(F,"B")
Convert factor F (levels A, B, C) into effect (sum) coding, splitting F into FB (A=0, B=1, C=-1) and FC (A=0, B=-1, C=1) # Effect coding is better if you want to test interactions with F	NA	contrasts(F) = contr.sum(levels(F))
Convert factor F (levels A, B) into effect (sum) coding, changing F into FA (A=1, B=-1)	NA	contrasts(F) = contr.sum(levels(F))
Convert factor F (levels A, B) into effect (sum) coding, changing F into FB (A=-1, B=1) # Safer than above, in my experience	NA	FB = 2*(F=="B")-1
***z*, *t*, and *F* tests**		
Two-tailed p value for one-sample *z* test with population μ and σ and sample S	z = (AVERAGE(S)-μ) / (σ/SQRT(COUNT(S))) p = 2*NORMSDIST(-ABS(z))	z = (mean(S)-μ) / (σ/sqrt(length(S))) p = 2*pnorm(-abs(z))
Two-tailed p value for one-sample *t* test with population μ and sample S	t = (AVERAGE(S)- μ)/ (STDEV(S)/ SQRT(COUNT(S))) p =TDIST(ABS(t), COUNT(S)-1,2)	1: t = (mean(S)-μ) / (sd(S)/sqrt(length(S))) p = 2*pt(-abs(t), df=length(S)-1) 2: t.test(S,mu=μ)
Unpaired *t* test assuming equal variance (homoscedastic) for a vs. b (levels of factor X, with dependent variable Y)	Analysis toolbox: t 檢定：兩個母體平均數差的檢定，假設變異數相等	1: t.test(a, b, var.equal=T) 2: t.test(Y~X, var.equal=T)
Unpaired *t* test not assuming equal variance (heteroscedastic) for a vs. b (levels of X, with dependent variable Y)	Analysis toolbox: t 檢定：兩個母體平均數差的檢定，假設變異數不相等	1: t.test(a,b) 2: t.test(Y~X)
Paired *t* test for a vs. b (levels of factor X, with dependent variable Y)	Analysis toolbox: t 檢定：成對母體平均數差異檢定	1: t.test(a, b, paired=T) 2: t.test(Y~X, paired=T)
One-tailed *p* value for a certain *F* value and $df_{numerator}$ and $df_{denominator}$	FDIST(F, df_n, df_d)	1: 1 - pf(F, df_n, df_d) 2: pf(F, df_n, df_d, lower.tail=F)
One-tailed *F* test to test if samples a and b come from populations with equal variances, where $s_a > s_b$	Analysis toolbox: F 檢定：兩個常態母體變異數的檢定 (a must be to the left b)	1: 1 - pf(F, df_a, df_b) 2: pf(F, df_a, df_b, lower.tail=F)
Two-tailed *F* test to test if samples a and b come from populations with equal variances	FTEST(a, b)	var.test(x, y)
95% confidence interval for *t* tests	Run analysis toolbox, get critical value and variance to compute using handout formulas	t.test(...) gives upper and lower value of confidence interval automatically; to use in graph, must find half its range: (max-min)/2

x% confidence interval for *t* tests	Run analysis toolbox using alpha = 1-x/100, get critical value and variance to compute using handout formulas	t.test(..., conf.level = x/100) gives x% confidence interval automatically
Correlation and linear regression analysis		
Pearson's correlation coefficient *r* (for variables x and y)	CORREL(x,y)	cor(x,y)
Test significance of Pearson's correlation coefficient (between x and y)	use **correl-sig.xls** or search the Web for tools	1: cor.test(x, y) 2: summary(lm(y~x))
Multiple linear regression (y = dep; x1, x2 = indeps), with data in D	Analysis toolbox: 迴歸	summary(lm(y~x1+x2, data=D)) # data argument also used below
Likelihood ratio test for fit of simpler model L0 vs. fit of more complex L1	NA	anova(L0,L1) # L0 and L1 created by lm(...)
Test significance of indep x1 in linear model y ~ x1 + x2	Analysis toolbox: 迴歸	1: summary(lm(y~x1+x2)) 2: anova(M.no_x1,M.has_x1)
Stepwise regression for y~x1+x2 in dataframe D	NA	attach(D) base.lm = lm(y~1) summary(step(base.lm,y~x1+x2))
Test independent variables x1, x2, x3 for collinearity in dataframe D (dependent variable = y)	Analysis toolbox: 迴歸 Then compute R^2 for x1~x2+x3, then use VIF formula in handout	1: library(car) vif(lm(y~x1+x2+x3)) # < 5 is good 2: kappa(D[c("x1","x2","x3")]) # < 30 is good 3: library(languageR) collin.fnc(D[c("x1", "x2", "x3")])$cnumber
Get predictions (y-hat) of simple linear model predicting y from x for new data x'	FORECAST(x',y,x) # x' is just one value	predict(lm(y~x), newdata = data.frame(x')) # x' is a vector; also works for multiple regression
Get residuals of a linear model L for dependent variable Y	Analysis toolbox: 迴歸, then use coefficients to write equation to predict y-hat, then subtract y-hat from real values Y	1: resid(L) 2: Y-predict(L)
Standardize regression coefficients for regression model y ~ x1 + x2 (also works for generalized linear models and mixed-effects models)	1: Use STANDARDIZE on x1 & x2 then Analysis toolbox: 迴歸 on these *z* scores 2: Analysis toolbox: 迴歸, then for x1 coefficient B1: B1 * STDEV(x1) / STDEV(y) # Same for x2	1: summary(lm(y ~ scale(x1) + scale(x2)) 2: For x1 coefficient B1: B1*sd(x1)/sd(y) # Same for x2

Repeated-measures regression y~x with grouping variable g in data frame D (also applies to logistic & Poisson regression), where B= by-unit coefficients (e.g., B = B0 for intercept, or B = B1 for x slope)	Analysis toolbox: 迴歸
	AVERAGE(B) # Coef.
	STDEV(B) # SE
	# t, df, p from one-sample
	# t test (see above)
		B.coef = numeric(length(g))
		for (i in 1:length(g)) {
		D.i = subset(D,D[D$g==i])
		lm.i = lm(y~x, data=D.i)
		B.coef[i] = summary(lm.i)
		$coefficients["B","Estimate"]
		}
		t.test(B.coef) # gives all but SE
		# I'll add SE info after HW3...

ANOVA		
One-way independent-measures ANOVA (y = dependent; x = independent)	Analysis toolbox: 單因子變異數分析	1: summary(aov(y ~ x)) 2: anova(lm(y ~ x))

Two-way independent-measures ANOVA (y = dependent; x1, x2 = independents)	Analysis toolbox: 雙因子變異數分析：重複試驗	1: summary(aov(y ~ x1*x2)) 2: anova(lm(y ~ x1*x2))

One-way repeated-measures ANOVA (y = dependent; x = indep; S= grouping unit)	Analysis toolbox:
	雙因子變異數分析：無重複試驗
		S = as.factor(S) # Make sure!
		summary(aov(y ~ x + Error(S/x)))

Two-way repeated-measures ANOVA (y = dep; x1, x2 = indeps; grouped by S)	use repeated-measures regression by hand	summary(aov(y ~ x1*x2 +Error(S/(x1*x2))))
One type of mixed ANOVA (y = dep; x1 = between-group indep; x2 = within-group indep, grouped by S)	probably NA	1: summary(aov(y ~ x1*x2 +Error(S/x2))) 2: library(ez) # Likewise above ezANOVA(dv = y, wid = S, within = x2, between = x1)
Tukey HSD test [formula = any ANOVA formula, e.g. y~x, or y~x+Error(S/x)]	use equation in handout and find table of Studentized range statistic q on the Web	1: TukeyHSD(aov(formula)) 2: library(emmeans) emmeans(aov(formula), list(pairwise~x), adjust="tukey")
Correct for sphericity violations in repeated-measures ANOVA in factors with more than two levels (y = dep; x = within-group indep with 3 or more levels; S = grouping unit; df = $df_{denominator}$)	NA	library(ez) ezANOVA(dv = y, wid = S, within = x) # HFe = Huynh-Feldt epsilon # p[HF] = its p value # correct df = original df * HFe
Compute minF' for independent variable x, using the following ANOVA results: By-participant ANOVA: x.F1: F value for x x.dfn1: df for x levels (numerator) x.dfd1: df for random (denominator) By-item ANOVA: x.F2: F value for x x.dfn2: df for x levels (numerator) x.dfd2: df for random (denominator) Then you get the following: minF.F: minF' minF.dfn: df for x levels minF.dfd: df for random minF.p: p value	minF.F = (x.F1*x.F2/(x.F1+x.F2) minF.dfn = x.dfn1 # (= xdfn2) minF.dfd = (x.F1+x.F2)^2 / (x.F1^2/x.dfd2 + x.F2^2/x.dfd1) minF.p = FDIST(minF.F, minF.dfn, minF.dfd)	minF.F = (x.F1*x.F2/(x.F1+x.F2) minF.dfn = x.dfn1 # (= xdfn2) minF.dfd = (x.F1+x.F2)^2 / (x.F1^2/x.dfd2 + x.F2^2/x.dfd1) minF.p = pf(minF.F, minF.dfn, minF.dfd, lower.tail=F)
Contingency tables (and other simple categorical tests)		
one-tailed p value for binomial test on getting at most x in n binary events	BINOMDIST(x,n,0.5,TRUE)	1: pbinom(x, n, 0.5) 2: binom.test(x,n,alternative="left")
One-way chi-squared test on vector V, where H_0: all counts the same	CHITEST(observed,expected) # Must compute expected first	chisq.test(V)
One-way chi-squared test on vector V, where H_0: counts = vector W	CHITEST(observed,expected) # Also, only gives p value	chisq.test(V, p = W)
Two-way chi-squared test for column × row interaction in 2 × 2 matrix M	CHITEST(observed,expected) # Doesn't use Yate's correction	1: chisq.test(M) # With Yate's 2: summary(as.table(M)) # Without Yate's correction
Two-way chi-squared test for column × row interaction in larger matrix M	CHITEST(observed,expected) # Basically, forget this method	1: chisq.test(M) 2: summary(as.table(M)) # Same: Yate's irrelevant
Two-tailed p value testing for column × row interaction in contingency table M	NA	fisher.test(M)
Exact McNemar test for paired binary data, with a (1,0) pairs and b (0,1) pairs	BINOMDIST(MIN(a,b), a+b, 0.5, TRUE)	pbinom(min(a,b), a+b, 0.5)

<table>
<tr><th colspan="3">Logistic regression (and other generalized linear models)</th></tr>
<tr><td>Convert probability P into log odds (logit)</td><td>LN(P/(1-P))</td><td>1: ln(P/(1-P))
2: library(gtools)
   logit(P)</td></tr>
<tr><td>Convert log odds L into probability</td><td>EXP(L)/(1+EXP(L))</td><td>1: exp(L)/(1 + exp(L))
2: library(gtools)
   inv.logit(L)</td></tr>
<tr><td>Logistic regression model y ~ x1+x2 (y is binary variable, all data are independent), with data in data frame D</td><td>NA</td><td>glm(y~x1+x2,
  family=binomial, data = D)
# data argument also below</td></tr>
<tr><td>Show coefficients table for logistic regression model L</td><td>NA</td><td>summary(L)
p-values based on Wald test</td></tr>
<tr><td>Predict log odds from logistic regression model L</td><td>NA</td><td>predict(L)</td></tr>
<tr><td>Predict binary observations (0 vs. 1) from logistic regression model L</td><td>NA</td><td>predict(L, type="response")</td></tr>
<tr><td>Likelihood ratio test for simpler generalized linear regression model L0 vs. more complex L1 (applies to both logistic regression and Poisson regression)</td><td>NA</td><td>anova(L0, L1 test="Chisq")</td></tr>
<tr><td>Test parameter x1 of logistic regression model y~x1+x2 using likelihood ratio test</td><td>NA</td><td>L1 = glm(y~x1+x2,
  family=binomial)
L0 = glm(y~x2,
  family=binomial)
anova(L0, L1 test="Chisq")</td></tr>
<tr><td>Ordinal logistic regression y~x (y is ordinal variable, all data are independent)</td><td>NA</td><td>library(MASS)
summary(polr(y ~ x)
# Table compares each level
# with next level</td></tr>
<tr><td>Multinomial logistic regression y~x (y has three nominal values "A", "B", "C", all data independent)</td><td># Wald test only:
z = B/SE
p = 2*NORMSDIST(
-ABS(z))
# Likewise below</td><td>library(nnet)
summary(multinom(y ~ x))
# Table treats A as baseline
# Wald test for each row:
z = B/SE # B = coefficient
p = 2*pnorm(-abs(z))</td></tr>
<tr><td>Poisson regression y ~ x (y is count data)</td><td>NA</td><td>summary(glm(y~x, family=poisson))</td></tr>
<tr><th colspan="3">Mixed-effects modeling (linear and generalized linear)</th></tr>
<tr><td>Maximal one-random-factor LME:
  y = dependent (continuous, normal)
  x1, x2 = independent
  g = grouping unit (x1 grouped by g)</td><td>NA</td><td>1: library(nlme)
   lme(y~x1+x2, random = ~x1|g)
2: library(lme4) # Assumed elsewhere below
   lmer(y~x1+x2 + (x1|g))</td></tr>
<tr><td>Show results of LME model L</td><td>NA</td><td>summary(L)
lme shows p, lmer doesn't
Always build/name model first, before using summary
because the model may take a long time to build</td></tr>
<tr><td>Get p values for LME model derived from formula structure y ~ x + (x|g)</td><td>NA</td><td>1: Trust lme output (controversial)
2: 2*pnorm(-abs(t)) # Claims t = z (needs large N)
3: library(afex) # Loads lme4 for you
   L = mixed(y ~ x + (x|g)) # Kenward-Roger p
   summary(L)
4: library(afex) # Loads lme4 for you
   # Likelihood ratio tests (needs large N)
   L = mixed(y ~ x + (x|g), method="LRT")
   summary(L)
# method="PB" not working for afex's mixed function;
# Forget about lmerTest (worse than Kenward-Roger,
# changes summary.lmer behavior)</td></tr>
</table>

<table>
<tr><td>Maximal two-random-factor additive LME (recommended by Barr et al., 2013):
y = dependent (continuous, normal)
x1, x2 = independent
g1 = grouping unit for x1 (random effect)
g2 = grouping unit for x2 (random effect)</td><td>NA</td><td>1: lmer(y~x1+x2 + (x1|g1) + (x2|g2))
2: lmer(y~x1+x2 + (1+x1|g1) + (1+x2|g2))
R assumes the intercepts automatically</td></tr>
<tr><td>Maximal one-random-factor LME with interaction:
y = dependent (continuous, normal)
x1, x2 = independent
g = grouping unit for x1 & x2</td><td>NA</td><td>lmer(y~x1*x2 + ((x1*x2)|g))</td></tr>
<tr><td>Likelihood ratio test to compare fit of simper LME model L0 vs. complex L1</td><td>NA</td><td>anova(L0,L1)</td></tr>
<tr><td>Likelihood ratio test for above to see if random g2 variable is really necessary (not recommended by Barr et al., 2013, but cf. Raaijmakers et al., 1999)</td><td>NA</td><td>L.1.2 = lmer(y~x1+x2 + (x1|g1) + (x2|g2))
L.1 = lmer(y~x1+x2 + (x1|g1))
anova(L.1, L.1.2)</td></tr>
<tr><td>LME without random intercepts (if maximal model fails to converge)
This and below also work for GLMM</td><td>NA</td><td>lmer(y~x + (x|g)) # Maximal model
lmer(y~x + (0+x|g)) # Next-best model</td></tr>
<tr><td>LME without random intercept × slope interaction (if above also fails)</td><td>NA</td><td>lmer(y~x + (0+x|g) + (1|g)) # Slope & intercept separate</td></tr>
<tr><td>LME without random slopes (if all fails)</td><td>NA</td><td>lmer(y~x + (1|g)) # Worst LME model (Barr et al., 2013)</td></tr>
<tr><td>Maximal one-random-factor mixed-effects logistic regression (a kind of GLMM):
y = dependent (binary)
x1, x2 = independent
g = grouping unit (x1 grouped by g)</td><td>NA</td><td>1: library(MASS)
glmmPQL(y~x1+x2, random=~x1|g, family=binomial)
2: library(lme4) # Assumed elsewhere below
glmer(y~x1+x2+(x1|g), family=binomial)</td></tr>
<tr><td>Maximal two-random-factor mixed-effects logistic regression:
y = dependent (binary)
x1, x2 = independent
g1 = grouping unit for x1 (random effect)
g2 = grouping unit for x2 (random effect)</td><td>NA</td><td>glmer(y~x1+x2 + (x1|g1) + (x2|g2), family=binomial)</td></tr>
<tr><td>Likelihood ratio test to compare fit of simper GLMM model L0 vs. more complex L1</td><td>NA</td><td>anova(L0, L1, test="Chisq")</td></tr>
</table>