Translation Syntax
(SPSS, Stata, SAS, R, and python)

The Basics

The following conventions are used in this document:

• Bold font indicates code or other text that should be typed literally.

• Un-bolded font shows code or text that should be replaced with user-supplied values (i.e., your own variable names and other environment details).

Calling in a data set

SPSS	**GET FILE='P:\QAC\qac201\Studies\study name\filename.sav'.**
STATA	**use "**P:\QAC\qac201\Studies\study name\filename**"**
SAS	**LIBNAME** mydata **"**P:\QAC\QAC201\study name**";** **DATA new; set** mydata**.**filename**;**
R	**load ("**filename-including-path**.Rdata")** myData_orig **<-** name-of-object-that-loaded-in-your-workspace If calling in from a tab-delimited text file: myData_orig **<- read.table(file = "**filename-including-path.txt**", sep = "\t", header = TRUE**)

PYTHON	**import pandas**
	import numpy
	myData **= pandas.read_csv(**'nesarc_pds.csv'**)**

Selecting variables you want to examine

SPSS	**/KEEP** VAR1 VAR2 VAR3 VAR4 VAR5 VAR6 VAR7 VAR8. (Must follow the **SAVE OUTFILE=**'dataname' command)
STATA	**use** VAR1 VAR2 VAR3 VAR4 VAR5 VAR6 VAR7 VAR8 /// **using** "P:\QAC\qac201\Studies\study name\filename"**, clear**
SAS	**KEEP** VAR1 VAR2 VAR3 VAR4 VAR5 VAR6 VAR7 VAR8**;**
R **(base)**	var.keep **<- c(**"VAR1"**,** "VAR2"**,** "VAR3"**,** "VAR4"**,** "VAR5"**,** "VAR6"**,** "VAR7"**,** "VAR8"**)** myData **<-** myData_orig **[,**var.keep**]**
R **(Tidyverse)**	myData <- myData_orig **%>% select(**VAR1, VAR2, VAR3, VAR4, VAR5, VAR6, VAR7, VAR8**)** or, if these variables are in consecutive columns: myData <- myData_orig **%>% select(VAR1:VAR8)**
PYTHON	myData = myData_orig**[[**'VAR1',VAR2','VAR3','VAR4','VAR5','VAR6','VAR7', 'VAR8'**]]**

Outputting your abbreviated data set

SPSS	**SAVE OUTFILE=** **'**Drive:\folder\folder\title_of_new_data_set**'.**
STATA	**save** filename
SAS	**Data libname**.title_of_new_data_set; **set** dataname; **by** unique_id**;**
R	To open in excel: **write.table(**myData**, file =** "filename.txt"**, sep = "\t", row.names = FALSE)** To use in R: **save (**myData**, file =** "filename.Rdata"**")**

PYTHON	**pandas.DataFrame.to_csv(**myData,'filename.csv'**)**

Sorting the data

SPSS	**SORT CASES BY** unique_id.
STATA	**sort** unique_id
SAS	**proc sort; by** unique_id**;**
R **(base)**	myData **<-** myData**[order(**myData**$**unique_id**, decreasing = FALSE),]**

R **(Tidyverse)**	myData <- myData %>% **arrange**(unique_id)
	
	or in descending order
	
	myData <- myData %>% arrange(**desc**(unique_id))

PYTHON	myData = myData.sort_values(by='unique_id')

Displaying frequency tables

SPSS	**FREQUENCIES VARIABLES**=VAR1 VAR2 VAR3 **/ORDER=ANALYSIS.**
STATA	**tab1** VAR1 VAR2 VAR3
SAS	**PROC FREQ; tables** VAR1 VAR2 VAR3**;**

R **(base)**	**library(descr)**
	freq(as.ordered(myData**$**VAR1**))**
	freq(as.ordered(myData**$**VAR2**))**
	freq(as.ordered(myData**$**VAR3**))**

R **(Tidyverse)**	myData %>% **group_by**(VAR1) %>% **tally()** myData %>% **group_by**(VAR2) %>% **tally()**

	or
	myData **%>%**
	  group_by(VAR1) **%>%**
	  tally() %>%
	  mutate(pct = n/sum(n)*100,
	        cumpct = cumsum(pct))

PYTHON	c1 **= myData['**VAR1**'].value_counts(sort=False, dropna=False)** **print**(c1**)**

Data Management

Basic operations

SPSS	EQ or =	>= or GE	<= or LE	> or GT	< or LT	NE
STATA	==	>=	<=	>	<	!=
SAS	EQ or =	>= or GE	<= or LE	> or GT	< or LT	!= or NE
R	==	>=	<=	>	<	!=
PYTHON	==	>=	<=	>	<	!= or <>

Examples

1. Need to identify missing data

Often, you must define the response categories that represent missing data. For example, if the number 9 is used to represent a missing value, you must either designate in your program that this value represents missingness or else you must recode the variable into a missing data character that your statistical software recognizes. If you do not, the 9 will be treated as a real/meaningful value and will be included in each of your analyses.

SPSS	**RECODE** VAR1 **(**9=**SYSMIS).**

STATA	**replace** VAR1**=. if** VAR1**==9**
SAS	**if** VAR1=9 **then** VAR1=.**;**
R (base)	myData**$**VAR1**[**myData**$**VAR1 **==** 9 **]** **<- NA** *if multiple values need to be set to missing* myData**$**VAR1**[**myData**$**VAR1 **%in%** c(8, 9) **]** **<- NA**
R (Tidyverse)	*Set single value in a single variable to missing* myData <- myData %>% **mutate**(VAR1 = **na_if**(VAR1, 9)) *Set multiple values in a single variable to missing* myData <- myData %>% **mutate**(VAR1 = **ifelse**(VAR1 **%in%** c(8, 9), NA, VAR1) *Set multiple values to missing across multiple variables* **vlist <- c("**VAR1", "VAR2", "VAR3"**)** **set.to.na <- function(x) ifelse(x %in% c(8,9), NA, x)** **myData <- myData %>% mutate(across(vlist, set.to.na))** *Change a 999 for ALL numeric variables into missing* myData <- myData %>% mutate(across(where(is.numeric), ~na_if(., 999))) *General method to recode values of a single variable:* ***Syntax: recode(df$variable, “old” = “new”)*** #For variable OUTCOME convert PASS to 0 and FAIL to 1 Df$OUTCOME <- recode(df$OUTCOME, “PASS” = 0, “FAIL = 1) #For variable SMOKER convert y to Smoker and n to Non-Smoker df$SMOKER <- recode(df$SMOKER, "y" = “Smoker”, "n" = "Non-Smoker")
PYTHON	myData['VAR1']= myData['VAR1'].replace(9, numpy.nan)

2. Need to recode responses to "no" based on skip patterns

There are a number of skip outs in some data sets. For example, if we ask someone whether or not they have ever used marijuana, and they say "no", it would not make sense to ask them more detailed questions about their marijuana use (e.g. quantity, frequency, onset, impairment, etc.). When analyzing more detailed questions regarding marijuana (e.g. have you ever smoked marijuana daily for a month or more?), those individuals that never used the substance may show up as missing data. Since they have never used marijuana, we can assume that their answer is "no", they have never smoked marijuana daily. This would need to be explicitly recoded. Note that we commonly code a no as 0 and a yes as 1.

SPSS	**RECODE** VAR1 **(SYSMIS**=7**).**
STATA	**replace** VAR1**=7 if** VAR1**==.**
SAS	**if** VAR1**=. then** VAR1=7**;**
R (base)	myData**$**VAR1**[is.na(**myData**$**VAR1**)] <-** 7
R (Tidyverse)	myData <- myData %>% **mutate**(VAR1 = ifelse(**is.na**(VAR1), 7, VAR1)
PYTHON	myData**[**'VAR1'**]**.**fillna**(7, **inplace=True**)

3. Creating a new variable by recoding a string variables into numeric variable

It is important when preparing to run statistical analyses in most software packages, that all variables have response categories that are numeric rather than "string" or "character" (i.e. response categories are actual strings of characters and/or symbols). All variables with string responses must therefore be recoded into numeric values. These numeric values are known as dummy codes in that they carry no direct numeric meaning.

SPSS	**RECODE** TREE **('**Maple**'**=1**)** **('**Oak**'**=2**)** **INTO** TREE_N**.**
STATA	**generate** TREE_N**=.** **replace** TREE_N=1 **if** TREE=="Maple" **replace** TREE_N=2 if TREE=="Oak" OR by using the encode command **encode** TREE**, gen(**TREE_N**)**

SAS	**IF** TREE**='**Maple**' then** TREE_N=1**;**
	else if TREE**= '**Oak**' then** TREE_N**=**2**;**

R	Not necessary in R.

PYTHON	def TREE_N (row) :
	        if row['TREE'] == 'Maple' :
	                return 1
	        if row['TREE'] == 'Oak' :
	                return 2
	
	myData['TREE_N'] = myData.apply(lambda row: TREE_N (row), axis = 1)

4. Creating a new variable by collapsing response categories

If a variable has many response categories, it can be difficult to interpret the statistical analyses in which it is used. Alternatively, there may be too few subjects or observations identified by one or more response categories to allow for a successful analysis. In these cases, you would need to collapse across categories. For example, if you have the following categories for geographic region, you may want to collapse some of these categories:

Region: New England=1, Middle Atlantic=2, East North Central=3, West North Central=4, South Atlantic=5, East South Central=6, West South Central=7, Mountain=8, Pacific=9.
New_Region: East=1, West=2.

SPSS	**COMPUTE** new_region**=**2**.** **IF (**region=1| region=2|region=3| region=5|region=6**)** new_region=1**.**
STATA	**generate** new_region =2 **replace** new_region=1 **if** region==1| region==2|region==3| region==5|region==6 OR by using the recode command **recode region** (1/3 5 6=2) **gen(**new_region**)**

SAS	**if r**egion=1 or region=2 or region=3 or region=5 or region=6 **then** new_region=1**;**
	else if region=4 or region=7 or region=8 or region=9 **then** new_region=2**;**

R **(base)**	myData$new_region <- NA myData**$**new_region**[**myData**$**region %in% c(1,2,3,5,6)**]** **<-** 1 myData**$**new_region**[**myData**$**region %in% c(4,7,8,9)**]** **<-2** or myData$new_region <- **ifelse(**region %in% c(1,2,3,5,6), **1**, **2**)
R **(Tidyverse)**	*If the new variable only has 2 categories:* myData %>% **mutate**(new_region =**ifelse**(region %in% c(1,2,3,5,6), **1**, **2**)) *If the new variable has more than 2 categories:* myData %>% **mutate**(new_region = **recode**(region, "2" = "1", “3” = “1”, “5” = “1”, “6”=”1”, “4”=”2”, “7” = “2”, “8”=”3”, “9”=”3”))

PYTHON	def new_region (row) :
	if row['region'] == 1 or row['region'] == 2 or row['region'] == 3 or row['region']

	row['region'] ==5 or row['region'] == 6 :
	return 1
	elif row['region'] == 4 or row['region'] == 7 or row['region'] == 7 or row['region']
	row['region']== 8 or row['region'] == 9 :
	return 2
	
	myData['new_region'] = myData.**apply(lambda row:** new_region **(row), axis = 1)**

5. Creating a new variable by aggregating across variables

In many cases, you will want to combine multiple variables into one. For example, while NESARC assesses several individual anxiety disorders, I may be interested in anxiety more generally. In this case I would create a general anxiety variable in which those individuals who received a diagnosis of social phobia, generalized anxiety disorder, specific phobia, panic disorder, agoraphobia, or obsessive compulsive disorder would be coded "yes" and those who were free from all of these diagnoses would be coded "no".

Syntax shown on next page.

5. Creating a new variable by aggregating across variables (continued)

SPSS	**IF (**socphob=1|gad=1|specphob=1| panic=1|agora=1|ocd=1**) anxiety=1.** **RECODE** anxiety **(SYSMIS**=0**).**
STATA	**gen** anxiety=1 if socphob==1|gad==1|specphob==1| panic==1|agora==1|ocd==1 **replace** anxiety=0 if anxiety==.

SAS	**if** socphob=1 or gad=1 or specphob=1 or panic=1 or agora=1 or ocd=1 **then** anxiety=1**;**
	else anxiety=0**;**

R (base)	myData**$**anxiety **<- rep(**0, **nrow(**myData**))** myData**$**anxiety**[**myData**$**socphob == 1 \| myData**$**gad==1 | myData**$**panic == 1 \| myData**$**agora==1 | myData**$**ocd == 1**] <- 1** myData**$**anxiety**[is.na(**myData**$**socphob**) & is.na(**myData**$**gad**) &** **is.na(**myData**$**panic**) & is.na(**myData**$**agora**) &** **is.na(**myData**$**ocd**)] <- NA**
R (Tidyverse)	*Tidyverse way: specify to do operations by row (**rowwise**), count number of non-missing symptoms (**n.symptoms**), count number of missing values per row (**n.miss**), create **anxiety** using **ifelse**, overwrite anxiety if all specific symptoms were missing.* myData <- myData %>% rowwise() %>%

	mutate(n.symptoms = sum(c(socphob==1, gad==1, panic==1, agora==1, ocd==1), na.rm=TRUE),
	n.miss = sum(is.na(socphob), is.na(gad), is.na(panic), is.na(agora), is.na(ocd)),
	anxiety = ifelse(n.symptoms > 0, 1, 0),
	
	anxiety = ifelse(nmiss==5, NA, anxiety)

PYTHON	def anxiety (row) :
	if row['socphob'] == 1 or row['gad'] == 1 or row['panic'] == 1 or row['agora'] == 1 or row['ocd'] == 1 :
	return 1
	else :
	return 0
	
	myData['anxiety'] = data.apply(lambda row: anxiety (row), axis = 1)

6. Need to create quantitative variables

If you are working with a number of items that represent a single construct, it may be useful to create a composite variable/score. For example, I want to use a list of nicotine dependence symptoms meant to address the presence or absence of nicotine dependence (e.g. tolerance, withdrawal, craving, etc.). Rather than using a dichotomous variable (i.e. nicotine dependence present/absent), I want to examine the construct as a dimensional scale (i.e. number of nicotine dependence symptoms). In this case, I would want to recode each symptom variable so that yes=1 and no=0 and then sum the items so that they represent one composite score.

SPSS	**COMPUTE** nd_sum**=sum(**nd_symptom1 nd_symptom2 nd_symptom3 nd_symptom4)**.**
STATA	**egen** nd_sum=**rsum(**nd_symptom1 nd_symptom2 nd_symptom3 nd_symptom4**)**
SAS	nd_sum**=sum (of** nd_symptom1 nd_symptom2 nd_symptom3 nd_symptom4**);**
R **(base)**	myData$nd_sum <- NA myData$nd_sum **<-** myData**$**nd_symptom1 **+** myData**$**nd_symptom2 **+** myData**$**nd_symptom3 **+** myData**$**nd_symptom4
R **(Tidyverse)**	myData **<-** myData %>% rowwise() %>% mutate(nd_sum **=** **sum(**nd_symptom1, nd_symptom2, nd_symptom3, nd_symptom4)) *Can be shortened if the variables are in consecutive columns:*

	myData **<-** myData %>% rowwise() %>% mutate(nd_sum **=** **sum(**nd_symptom1: nd_symptom4))
PYTHON	myData['nd_sum'] = myData['nd_symptom1'] + myData['nd_symptom2'] + myData['nd_symptom3'] + myData['nd_symptom4']

7. Labeling variables

Given the often cryptic names that variables are given, it can sometimes be useful to label them.

SPSS	**VARIABLE LABELS** VAR1 **'**label**'.**
STATA	**label variable** VAR1 **"**label**"**
SAS	**LABEL** VAR1**='**label**';**
R	For frequency tables: **library (**Hmisc**)** **label(**myData**$**VAR1**) <- "**label**"**
PYTHON	**N/A**

8. Renaming variables

Given the often cryptic names that variables are given, it can sometimes be useful to give a variable a new name (something that is easier for you to remember or recognize).

SPSS	**COMPUTE** newvarname**=**VAR1**.**
STATA	**rename** VAR1 newvarname
SAS	**RENAME** VAR1=newvarname**;**
R (base)	**names(**myData**)[names(**myData**)== "**VAR1**"] <- "**newvarname**"**
R (Tidyverse)	myData <- myData %>% **rename**(newvar == oldvar)
PYTHON	myData = myData**.rename('**oldvar**':'**newvar**', axis='columns')**

9. Labeling variable responses/values

Given that nominal and ordinal variables have, or are given numeric response values (i.e. dummy codes), it can be useful to label those values so that the labels are displayed in your output.

SPSS	**VALUE LABELS** VAR1 0 **'**value0label**'** 1 **'**value1label**'** 2 **'**value2label**'** 3 **'**value3label**'.**
STATA	**label define** VAR1 0 "value0label" 1 "value1label" 2 "value2label" 3 "value3label" **label values** VAR1 newvarname
SAS	Set up format before the data step. **proc format; VALUE** FORMATNAME 0**="**value0label**"** 1**="**value1label**"** 2**="**value2label**"** 3**="**value3label**";** Before the end of the data step, tell SAS which variables you would like to format with these values. **format** VAR1 FORMATNAME**.**
R **(base)**	Because the function doesn't rearrange the order of the levels (the default is alphabetical), make sure you write the new labels in the order that they currently appear in the data set. This shows you the existing levels in the current order. If VAR1 is a character variable, this will return NULL **levels(**myData**$**VAR1**)** If VAR1 is already a factor, you can apply the labels like this: **levels(**myData**$**VAR1**) <- c("**value0label**", "**value1label**", "**value2label**", "**value3label**")** If VAR1 is not yet a factor, you can convert it to a factor, and apply the labels like this. Again, the order you write the labels must match the order that is currently in the data. myData$VAR1 <- factor(myData$VAR1, levels = **c("**value0label**", "**value1label**", "**value2label**", "**value3label**"))**
R **(Tidyverse)**	*To reorder factor levels:* myData$VAR1 <- myData$VAR1 %>% fct_relevel(“value1label”, “value2label”, “value3label”, “value0label”) *To entirely reverse the label ordering* myData$Var1 <- fct_rev(myData$Var1) *To reorder the factor levels based on the frequency they show up in the data* myData$Var1 <- fct_infreq(myData$Var1)
PYTHON	Because the function doesn't name the existing levels, make sure you have them all in the right order.

	myData['VAR1']= myData['VAR1'].astype('category') myData['VAR1']= myData['VAR1'].cat.rename_categories(["value0label", "value1label", "value2label", "value3label"])

10. Need to further subset the sample

When using large data sets, it is often necessary to subset the data so that you are including only those observations that can assist in answering your particular research question. In these cases, you may want to select your own sample from within the survey's sampling frame. For example, if you are interested in identifying demographic predictors of depression among Type II diabetes patients, you would plan to subset the data to subjects endorsing Type II Diabetes.

SPSS	**/SELECT=**diabetes2 EQ 1 (must be added as a command option)
STATA	**if** diabetes2==1 (put this after the command)
SAS	**if** diabetes2=1**;** (put in the data step before sorting the data)
R (base)	title_of_subsetted_data **<-** myData**[**myData**$**diabetes2 **==** 1**,]**
R (Tidyverse)	title_of_subsetted_data <- myData %>% **filter**(diabetes2 ==1)
PYTHON	title_of_subsetted_data = myData[myData.diabetes2 == 1]

11. Need to create groups that will be compared to one another

Often, you will need to create groups or sub-samples from the data set for the purpose of making comparisons. It is important to be certain that the groups that you would like to compare are of adequate size and number. For example, if you were interested in comparing complications of depression in parents who had lost a child through miscarriage vs. parents who had lost a child in the first year of life, it would be important to have large enough groups of each. It would not be appropriate to attempt to compare 5000 observations in the miscarriage group to only 9 observations in the first year group.

Refer to other data management syntax examples.

Graphing and Data Visualization

1. Univariate

Code for Univariate Output (Categorical):

SPSS	**FREQUENCIES VARIABLES=**CategVar1 CategVar2 CategVar3 **/ORDER=ANALYSIS.**
STATA	**tab1** CategVar1 CategVar2 CategVar3
SAS	**PROC FREQ; tables** CategVar1 CategVar2 CategVar3**;**

R **(base)**	table(myData$CategVar1)
	
	or
	
	library(descr)
	freq(as.ordered(myData**$**CategVar1**))**
	freq(as.ordered(myData**$**CategVar2**))**
	freq(as.ordered(myData**$**CategVar3**))**

R **(Tidyverse)**	myData %>% group_by(CategVar1) %>% tally()
PYTHON	c1 = myData['CategVar'**].value_counts(sort=False)** **print (**c1**)**

Code for Univariate Graph (Categorical):

SPSS	Use graphical user interface (GUI)
STATA	**histogram** BinaryVar
SAS	**Proc GCHART; VBAR** CategVar **/ Discrete type=PCT** **Width=**30**;**

R	**library(ggplot2)**
	ggplot(data=myData**)+**
	  geom_bar(aes(x=CategVar**))+**
	  ggtitle("Descriptive Title Here"**)**

PYTHON	**import seaborn**
	seaborn.countplot(x="CategVar**", data=**myData**)**
	plt.xlabel("Label for CategVar**")**
	plt.title("Descriptive Title**")**

Code for Univariate Output (Quantitative):

SPSS	**DESCRIPTIVES VARIABLES=**QuantVar1 QuantVar2 QuantVar3 **/STATISTICS=MEAN STDDEV.**
STATA	**summarize** QuantVar1 QuantVar2 QuantVar3
SAS	**proc means; var** QuantVar1 QuantVar2 QuantVar3**;**

R (base)	Repeat for each variable.
	summary(myData**$**QuantVar1**)**
	mean(myData**$**QuantVar1**, na.rm = TRUE)**
	sd(myData**$**QuantVar1**, na.rm = TRUE)**

R (Tidyverse)	summarize can calculate multiple summary stats at the same time. myData %>% **summarize**(mean_var1 = mean(QuantVar1, na.rm=TRUE), sd_var1 = sd(QuantVar1, na.rm=TRUE))
PYTHON	Repeat for each variable. desc1 = myData['QuantVar1'].dropna().describe() print (desc1)

Code for Univariate Graph (Quantitative):

SPSS	Use graphical user interface (GUI)
STATA	**histogram** QuantVar
SAS	**Proc GCHART; VBAR** QuantVar**;**
R	**ggplot(data=**myData**)+** **geom_histogram(aes(x=**QuantVar**))+** **ggtitle(**“Descriptive Title Here”**)**

	appropriate geometries: geom_histogram, geom_density, geom_boxplot

PYTHON	**import seaborn**
	seaborn.distplot(myData**["**QuantVar**"].dropna(), kde=False)**
	plt.xlabel("Label for QuantVar**")**
	plt.title("Descriptive Title Here**")**

2. Bivariate

Code for Bivariate Output (Categorical Explanatory Variable and Categorical Response Variable):

SPSS	**CROSSTABS** **/TABLES=**CategResponseVar **by** CategExplanatoryVar **/CELLS=COUNT ROW COLUMN TOTAL.**
STATA	**tab** CategResponseVar CategExplanatoryVar**, row column cell**
SAS	**Proc freq; tables** CategResponseVar*****CategExplanatoryVar**;**
R (base)	tab1 **<- table (**myData**$**CategResponseVar**,** myData**$**CategExplanatoryVar**)** tab1 # to output the table tab1_colProp **<- prop.table(**tab1**, 2)** # column proportions tab1_rowProp **<- prop.table(**tab1**, 1)** # row proportions tab1_cellProp **<- prop.table(**tab1**)** # cell proportions tab1_colProp tab1_rowProp tab1_cellProp
R (Tidyverse)	% of response var within levels of explanatory var. Corresponds to row proportions myData %>% group_by(CategResponseVar, CategExplanatoryVar) %>% summarise(n=n()) %>% na.omit() %>% group_by(CategExplanatoryVar) %>% mutate(prop = n/sum(n)) % of explanatory var within levels of response var. Corresponds to column proportions myData %>% group_by(CategResponseVar, CategExplanatoryVar) %>%

	summarise(n=n()) %>% na.omit() %>% group_by(**Categ**ResponseVar) %>% mutate(prop = n/sum(n)) Cell proportions myData %>% group_by(CategResponseVar, CategExplanatoryVar) %>% summarise(n=n()) %>% na.omit() %>% **ungroup()** %>% mutate(prop = n/sum(n))
PYTHON	**print (pandas.crosstab(**myData**[**'CategResponseVar'**],** myData**[**'CategExplanatoryVar'**], margins=True))** # get column proportions **print (pandas.crosstab(**myData**[**'CategResponseVar'**],** myData**[**'CategExplanatoryVar'**], margins=True, normalize='columns'))** # get row proportions **print (pandas.crosstab(**myData**[**'CategResponseVar'**],** myData**[**'CategExplanatoryVar'**], margins=True, normalize='index'))** # get cell proportions **print (pandas.crosstab(**myData**[**'CategResponseVar'**],** myData**[**'CategExplanatoryVar'**], margins=True, normalize='all'))**

Code for Bivariate Bar Graph (Categorical Explanatory Variable and Bivariate Categorical Response Variable – should be dummy coded 0/1):

SPSS	Use graphical user interface (GUI)
STATA	**graph bar (mean)** CategResponseVar**, over(**CategExplanatoryVar**)**
SAS	**Proc GCHART; vbar** CategExplanatoryVar **/discrete type=mean sumvar=**CategResponseVar**;**

R (base)	**ggplot(data=**myData**)+**
	  stat_summary(aes(x=CategExplanatoryVar**, y=**CategResponseVar**),**
	        fun=mean, geom="bar")+
	  ggtitle("Descriptive Title Here"**)**

R (Tidyverse)	Alternative if you do not code your response variable as 0/1.
	
	Data.to.plot <- myData %>%
	   group_by(CategResponseVar, CategExplanatoryVar) %>%
	   summarise(n=n()) %>% na.omit() %>%
	   group_by(CategExplanatoryVar) %>%
	   mutate(prop = n/sum(n))
	
	ggplot(Data.to.plot) +
	   geom_col(aes(x= CategExplanatoryVar, fill= CategResponseVar, y=prop), position="dodge")

PYTHON	**seaborn.catplot(x="**CategExplanatoryVar**", y="**QuantResponseVar**", data=**myData**, kind="bar", ci=None)**
	plt.xlabel('Label for CategExplanatoryVar**')**
	plt.ylabel('Label for QuantResponseVar**')**
	plt.title('Descriptive Title Here**')**

Note: Your graph will display the proportion of observational units within a particular category who exhibit the indicated level of the response variable.

Code for Bivariate Output (Categorical Explanatory Variable and Quantitative Response Variable):

SPSS	**MEANS TABLES=** CategExplanatoryVar **by** QuantResponseVar **/CELLS MEAN COUNT STDDEV.**
STATA	**bys** CategExplanatoryVar**: su** QuantResponseVar
SAS	**proc sort; by** CategExplanatoryVar**;** **proc means; var** QuantResponseVar**; by** CategExplanatoryVar**;**

R **(base)**	**by(**myData**$**QuantResponseVar**,** myData**$**CategExplanatoryVar**, mean, na.rm = TRUE)**
	by(myData**$**QuantResponseVar**,** myData**$**CategExplanatoryVar**, sd, na.rm = TRUE)**
	by(myData**$**QuantResponseVar**,** myData**$**CategExplanatoryVar**, length)**

R **(Tidyverse)**	myData %>% **group_by**(CategExplanatoryVar) %>%
	    summarize(mean_var1 = mean(QuantVar1, na.rm=TRUE),
	                    sd_var1 = sd(QuantVar1, na.rm=TRUE),
	                    n_var1 = n())

PYTHON	desc1 = myData['QuantResponseVar'].groupby(myData['CategExplanatoryVar']).describe() print (desc1)

Code for Bivariate Graphs (Categorical Explanatory Variable and Quantitative Response Variable):

SPSS	Use graphical user interface (GUI)
STATA	**graph box** QuantResponseVar**, over(**CategExplanatoryVar**)**
SAS	**Proc GCHART; vbar** CategExplanatoryVar **/discrete type=mean sumvar=**QuantResponseVar**;**
R	##Below is code for bar graph **ggplot(data=**myData**)+** **stat_summary(aes(x=**CategExplanatoryVar**, y=**QuantResponseVar**),** **fun.y=mean, geom=”bar”)** as.character(myData$ CategExplanatoryVar) ##Below is code for boxplots **ggplot(data=**myData**)+** **geom_boxplot(aes(x=**CategExplanatoryVar**, y=**QuantResponseVar**))+** **ggtitle(**“Descriptive Title Here”**)** ##Below is code for density plots **ggplot(data=**myData**)+** **geom_density(aes(x=**QuantResponseVar, color=CategExplanatoryVar**))+** **ggtitle(**“Descriptive Title Here)
PYTHON	scat1 **= seaborn.catplot(x= '**CategExplanatoryVar ', **y= '**QuantResponseVar ', **data=**myData, **kind="bar", ci=None)** **print(**scat1**)**

Code for Bivariate Scatterplot (Quantitative Explanatory Variable and Quantitative Response Variable):

SPSS	**GRAPH /scatterplot(bivar)=**QuantExplanatoryVar **with** QuantResponseVar**.**
STATA	**twoway (scatter** QuantResponseVar QuantExplanatoryVar**) (lfit** QuantResponseVar QuantExplanatoryVar**)**
SAS	**Proc GPLOT; Plot** QuantResponseVar *****QuantExplanatoryVar**;**

R	**ggplot(data=**myData**)+**
	  geom_point(aes(x=QuantExplanatoryVar**, y=**QuantResponseVar**))+**
	  geom_smooth(aes(x=QuantExplanatoryVar**, y=**QuantResponseVar**), method="lm")**
	
	### Note, you can also add a 3rd variable to a scatterplot by using the color argument:
	ggplot(data=myData**)+**
	  geom_point(aes(x=QuantExplanatoryVar**, y=**QuantResponseVar, **,**
	                color=CategThirdVar**))+**
	  **geom_smooth(aes(x=**QuantExplanatoryVar**, y=**QuantResponseVar,
	                color=CategThirdVar**)), method="lm")**

PYTHON	**seaborn.regplot(x=**"QuantExplanatoryVar", **y=**"QuantResponseVar", **fit_reg=False, data**=myData**)**
	plt.xlabel('Label for QuantExplanatoryVar'**)**
	plt.ylabel('Label for QuantResponseVar'**)**
	plt.title('Descriptive Title Here**')**

3. Adding a Third Variable

Code for Output with a Third Variable (Categorical Explanatory Variable, Quantitative Response Variable, Categorical 3rd VAR):

SPSS	**MEANS TABLES=** QuantResponseVar **BY** CategExplanatoryVar **BY** CategThirdVar **/CELLS MEAN COUNT STDDEV.**
STATA	**bys** CategExplanatoryVar CategThirdVar**: su** QuantResponseVar
SAS	**proc sort; by** CategExplanatoryVar CategThirdVar**;** **proc means; var** QuantResponseVar**; by** CategExplanatoryVar CategThirdVar**;**
R **(base)**	**ftable(by(**myData**$**QuantResponseVar**,** **list(**myData**$**CategExplanatoryVar**,** myData**$**CategThirdVar**),** **mean, na.rm = TRUE))**
R **(Tidyverse)**	Summary statistics of quantitative response variable within cross-combination of two categorical variables myData %>% **group_by**(CategExplanatoryVar, CategThirdVar) %>% **summarize**(mean = mean(QuantResponseVar, na.rm=TRUE), sd = sd(QuantResponseVar, na.rm=TRUE), n = n())

PYTHON	#graphing code
	seaborn.catplot(x="CategExplanatoryVar"**, y=**"QuantResponseVar"**,**
	hue="CategThirdVar"**, data=**myData**, kind='bar', ci=None)**
	plt.xlabel('Label for CategExplanatoryVar'**)**
	plt.ylabel('Label for QuantResponseVar'**)**
	plt.title('Descriptive Title Here**')**

Code for Output with a Third Variable (Categorical Explanatory Variable and Categorical Response Variable, Categorical 3rd VAR):

SPSS	**CROSSTABS** **/TABLES=**CategResponseVar **BY** CategExplanatoryVar **BY** CategThirdVar**.**
STATA	**bys** CategExplanatoryVar CategThirdVar**: tab** CategResponseVar
SAS	**proc sort; by** CategThirdVar**;** **proc freq; tables** CategResponseVar*****CategExplanatoryVar**; by** CategThirdVar**;**
R (base)	tab1 **<- ftable(**myData**$**CategResponseVar**,** myData**$**CategExplanatoryVar**,** myData**$**CategThirdVar**)** tab1 tab1_colProp **<- prop.table(**tab1**, 2)** tab1_colProp
R (Tidyverse)	myData %>% **group_by**(CategThirdVar , CategResponseVar , CategExplanatoryVar) %>% tally()

PYTHON	seaborn.catplot(x="CategExplanatoryVar", y="CategResponseVar", hue="CategThirdVar", data=myData, kind="bar", ci=None)
	plt.xlabel('Label for CategExplanatoryVar'**)**
	plt.ylabel('Label for CategResponseVar'**)**
	plt.title('Descriptive Title Here**')**

Note: If your 3rd variable is quantitative, for graphing purposes, create meaningful categories and then use the code above.

Bivariate Analysis

ANOVA

SPSS	**UNIANOVA** QuantResponseVar **BY** CategExplanatoryVar**.**
STATA	**oneway** QuantResponseVar CategExplanatoryVar**, tabulate**

SAS	**proc anova;**
	class CategExplanatoryVar**;**
	model QuantResponseVar **=** CategExplanatoryVar**;**
	means CategExplanatoryVar**;**

R	myAnovaResults **<- aov(**QuantResponseVar **~** CategExplanatoryVar**, data =** myData**)** **summary(**myAnovaResults**)**

PYTHON	import statsmodels.formula.api as smf
	import statsmodels.stats.multicomp as multi
	
	model1 = smf.ols(formula='QuantResponseVar ~ C(CategExplanatoryVar)', data=myData)
	results1 = model1.fit()
	print (results1.summary())

Pearson Correlation

SPSS	**CORRELATIONS** **/VARIABLES=** QuantResponseVar QuantExplanatoryVar **/STATISTICS DESCRIPTIVES.**
STATA	**corr** QuantResponseVar QuantExplanatoryVar r OR **pwcorr** QuantResponseVar QuantExplanatoryVar**, sig**
SAS	**Proc corr; var** QuantResponseVar QuantExplanatoryVar**;**
R	**cor.test(**myData**$**QuantResponseVar, myData**$**QuantExplanatoryVar**)**
PYTHON	**import scipy**

	sub1= myData[['QuantResponseVar', 'QuantExplanatoryVar']].dropna() print ('association between QuantExplanatoryVar and QuantResponseVar') print (scipy.stats.pearsonr(sub1['QuantResponseVar'], sub1['QuantExplanatoryVar']))

Chi-Square Test

SPSS	**CROSSTABS** **/TABLES=** CategResponseVar **by** CategExplanatoryVar **/STATISTICS=CHISQ.**
STATA	**tab** CategResponseVar CategExplanatoryVar **, chi2 row col**
SAS	**Proc freq; tables** CategResponseVar*****CategExplanatoryVar**/ chisq;**
R	myChi **<- chisq.test(**myData**$**CategResponseVar**,** myData**$**CategExplanatoryVar**)** myChi myChi**$observed** # for actual, observed cell counts **prop.table(**myChi**$observed, 2)** # for column percentages **prop.table(**myChi**$observed, 1)** # for row percentages

PYTHON	**import scipy.stats**
	ct1=**pandas.crosstab(**myData**['**CategResponseVar**],** myData**['**CategExplanatoryVar**'])**
	print ('chi-square value, p value, degrees of freedom, expected counts')
	cs1= **scipy.stats.chi2_contingency**(ct1)
	print (cs1)
	# column percentages
	colsum=ct1**.sum(axis=0**)
	colpct=ct1/**colsum**
	print(colpct)

POST HOC TESTS WITHIN ANOVA

SPSS	**UNIANOVA** QuantResponseVar **BY** CategExplanatoryVar **/POSTHOC=**CategExplanatoryVar **(TUKEY)** **/PRINT=ETASQ DESCRIPTIVE.**
STATA	**oneway** QuantResponseVar CategExplanatoryVar**, sidak**

SAS	**Proc anova;**
	class CategExplanatoryVar**;**
	model QuantResponseVar**=**CategExplanatoryVar**;**
	means CategExplanatoryVar**/duncan;**

R	myAnovaResults **<- aov(**QuantResponseVar **~** CategExplanatoryVar, **data =** myData**)** **TukeyHSD(**myAnovaResults**)**

PYTHON	model1 = smf.ols(formula='QuantResponseVar ~ C(CategExplanatoryVar)', data=myData)
	results1 = model1.fit()
	print (results1.summary())
	
	sub1 = myData[['QuantResponseVar', 'CategExplanatoryVar']].dropna()
	mc1 = multi.MultiComparison(sub1['QuantResponseVar'], sub1['CategExplanatoryVar'])
	res1 = mc1.tukeyhsd()
	print(res1.summary())

POST HOC TESTS FOR CHI SQUARE (must subset data in order to conduct 2X2 comparisons)

SPSS	**TEMPORARY.** **SELECT IF** CategExplanatoryVar**=1 OR** CategExplanatoryVar **=3.** **CROSSTABS** **/TABLES=** CategResponseVar CategExplanatoryVar **/STATISTICS=CHISQ.**
STATA	**tab** CategResponseVar CategExplanatoryVar **If** CategExplanatoryVar**==1 |** CategExplanatoryVar**==3, chi2**

SAS	**IF (**CategExplanatoryVar **= 1) OR (**CategExplanatoryVar **= 3);** (in data step)
	Proc freq; tables CategResponseVar*CategExplanatoryVar**/ chisq;**

R	## You do not need to subset the data in R pairwise.prop.test(table(myData$CategResponseVar, myData$CategExplanatoryVar), p.adjust.method=”bonferroni”)

PYTHON	#for each Chi Sq pair data subset (code below compares group 1 to group 2)
	recode1 = {1: 1, 2: 2}
	myData['COMP1v2']= myData['CategExplanatoryVar'**].map**(recode1)
	ct1=**pandas.crosstab**(myData['CategResponseVar], myData['COMP1v2']
	cs1= **scipy.stats.chi2_contingency**(ct1)
	print (cs1)

Statistical Interactions: Testing for Moderation

Moderation: ANOVA

In these analyses, the third variable must be categorical.

SPSS	**SORT CASES BY** CategThirdVar**.** **SPLIT FILE LAYERED BY** CategThirdVar**.** **ONEWAY** QuantResponseVar **BY** CategExplanatoryVar **/ STATISTICS DESCRIPTIVES** **/ POSTHOC = BONFERRONI ALPHA (0.05).** **SPLIT FILE OFF.**
STATA	**bys** CategThirdVar**: oneway** QuantResponseVar CategExplanatoryVar**, tab**
SAS	**Proc sort; by** CategThirdVar**;** **Proc anova; class** CategExplanatoryVar**;** **model** QuantResponseVar**=**CategExplanatoryVar**;** **means** CategExplanatoryVar**; by** CategThirdVar**;**

R	**by(**myData**,** myData**$**CategThirdVar**, function(x)**
	    list(
	        aov(QuantResponseVar **~** CategExplanatoryVar**, data = x),**
	        summary(aov(QuantResponseVar **~** CategExplanatoryVar**, data = x))))**

PYTHON

```
#subset by categorical 3rd variable
sub2=myData[(myData['CategThirdVar']=='Group 1')]
sub3=myData[(myData['CategThirdVar']=='Group 2')]

import statsmodels.api
import statsmodels.formula.api as smf
model2 = smf.ols(formula='QuantResponseVar ~ C(CategExplanatoryVar)',
data=sub2).fit()
print (model2.summary())

model3 = smf.ols(formula= formula='QuantResponseVar ~ C(CategExplanatoryVar)',
data=sub3).fit()
print (model3.summary())
```

Moderation: PEARSON CORRELATION

In these analyses, the third variable must be categorical

SPSS

```
SORT CASES BY CategThirdVar.
SPLIT FILE LAYERED BY CategThirdVar.
CORRELATIONS
/VARIABLES= QuantResponseVar QuantExplanatoryVar
/STATISTICS DESCRIPTIVES.
SPLIT FILE OFF.
```

STATA

```
bys CategThirdVar: corr QuantResponseVar QuantExplanatoryVar
OR
bys CategThirdVar: pwcorr QuantResponseVar QuantExplanatoryVar, sig
```

SAS

```
Proc sort; by CategThirdVar;
Proc corr; var QuantResponseVar QuantExplanatoryVar; by CategThirdVar;
```

R

```
by(myData, myData$CategThirdVar, function(x)
    cor.test(x$QuantResponseVar, x$QuantExplanatoryVar))
```

PYTHON

```
#subset by categorical 3rd variable
sub1=myData[(myData[‘CategThirdVar’]== 1)]
sub2=myData[(myData[‘CategThirdVar’]== 2)]
sub3=myData[(myData[‘CategThirdVar’]== 3)]

print (scipy.stats.pearsonr(sub1['QuantResponseVar'], sub1['QuantExplanatoryVar']))
print (scipy.stats.pearsonr(sub2['QuantResponseVar'], sub2['QuantExplanatoryVar']))
print (scipy.stats.pearsonr(sub3['QuantResponseVar'], sub3['QuantExplanatoryVar']))
```

Moderation: CHI-SQUARE TEST

In these analyses, the third variable must be categorical.

SPSS

```
CROSSTABS
/TABLES = CategResponseVar by CategExplanatoryVar
by CategThirdVar
/CELLS = COUNT ROW
/STATISTICS = CHISQ.
```

STATA

```
bys CategThirdVar: tab CategResponseVar CategExplanatoryVar,
chi2 row
```

SAS

```
Proc sort; by CategThirdVar;
Proc freq; tables CategResponseVar*CategExplanatoryVar/chisq; by CategThirdVar;
```

R

```
by(myData, myData$CategThirdVar, function(x)
    list(
        chisq.test(x$CategResponseVar, x$CategExplanatoryVar),
        chisq.test(x$CategResponseVar, x$CategExplanatoryVar)$observed,
        prop.table(chisq.test(x$CategResponseVar, x$CategExplanatoryVar)$observed,
        2))) # column %s
```

PYTHON

```
#subset by categorical 3rd variable
sub2=myData[(myData['CategThirdVar']=='Group 1')]
sub3=myData[(myData['CategThirdVar']=='Group 2')]

ct2=pandas.crosstab(sub2['CategResponseVar'], sub2['CategExplanatoryVar'])
print (ct2)
ct3=pandas.crosstab(sub3['CategResponseVar'], sub3['CategExplanatoryVar'])
print (ct3)
```

Regression Analyses

In these analyses, treat ordinal variables (e.g., variables scaled Strongly Disagree (1) to Strongly Agree (5)) as quantitative. Dummy code (0 = no, 1 = yes) all multi-level categorical variables.

Adding variables other than your response variable and your explanatory variable, as shown by the use of ThirdVar in the following syntax, lets you test for confounding.

Multivariate Regression

MULTIPLE REGRESSION

SPSS	**REGRESSION** **/DEPENDENT** QuantResponseVar **/METHOD ENTER** ExplanatoryVar ThirdVar1 ThirdVar2**.**
STATA	**reg** QuantResponseVar ExplanatoryVar ThirdVar1 ThirdVar2

SAS	Code binary variables as yes = 1 and no = 2.
	Proc glm; class CategExplanatoryVar CategThirdVar;
	model QuantResponseVar**=** CategExplanatoryVar CategThirdVar QuantThirdVar
	/solution;

R	my.lm **<- lm(**QuantResponseVar **~** ExplanatoryVar **+** ThirdVar1 **+** ThirdVar2**, data =** myData**)** **summary(**my.lm**)**

PYTHON	**import statsmodels.api**
	import statsmodels.formula.api as smf
	#note that categorical explanatory/third variables have to be entered as C(CategVar)
	lm1 = **smf.ols('**QuantResponseVar ~ ExplanatoryVar + **C(**CategThirdVar1**)** + QuantThirdVar**', data**=myData**).fit()**
	print (lm1**.summary())**

LOGISTIC REGRESSION

SPSS	**LOGISTIC REGRESSION** BinaryResponseVar **with** ExplanatoryVar ThirdVar1 ThirdVar2**.**
STATA	**logistic** BinaryResponseVar ExplanatoryVar ThirdVar1 ThirdVar2

SAS	Code your binary variables as yes = 1 and no = 2.
	Proc logistic;
	class CategExplanatoryVar CategThirdVar**;**
	model BinaryResponseVar**=**CategExplanatoryVar CategThirdVar QuantThirdVar**;**

R	my.logreg **<- glm(**BinaryResponseVar **~** ExplanatoryVar **+** ThirdVar1 **+** ThirdVar2**, data =** myData**, family = "binomial")** **summary(**my.logreg**)** # for p-values **exp(my.logreg$coefficients)** # for odds ratios **exp(confint(**my.logreg**))** # for confidence intervals on the odds ratios

PYTHON	**import statsmodels.api**
	import statsmodels.formula.api as smf
	# logistic regression

```
lreg1 = smf.logit(formula = 'BinaryResponseVar ~ ExplanatoryVar + C(CategThirdVar) + QuantThirdVar', data = myData).fit()
print (lreg1.summary())

# odd ratios with 95% confidence intervals
params = lreg1.params
conf = lreg1.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (numpy.exp(conf))
```