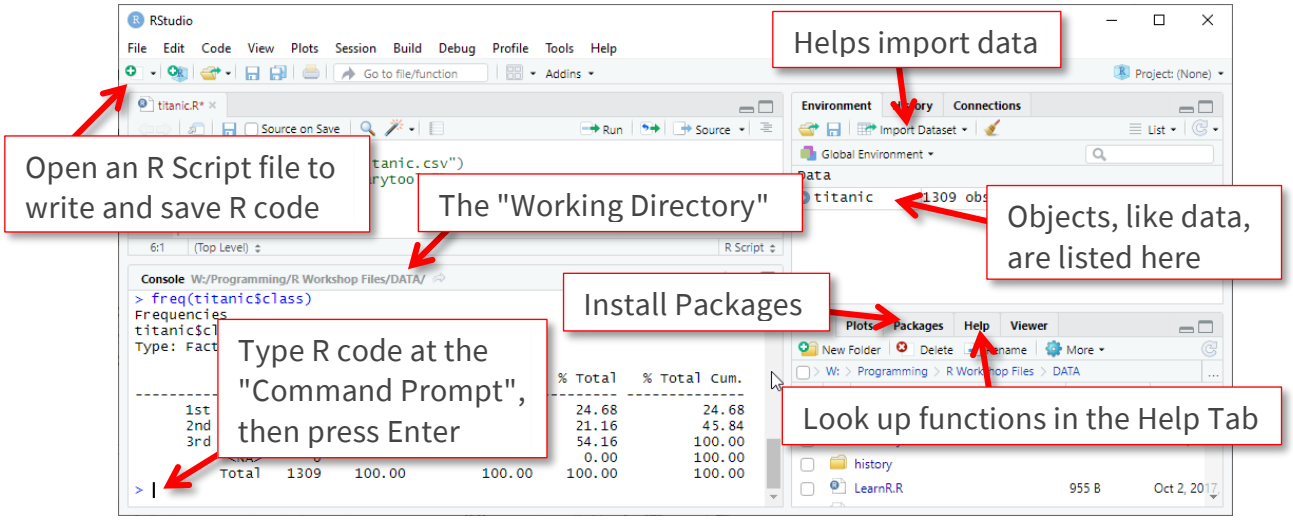
Starter R: The Basics

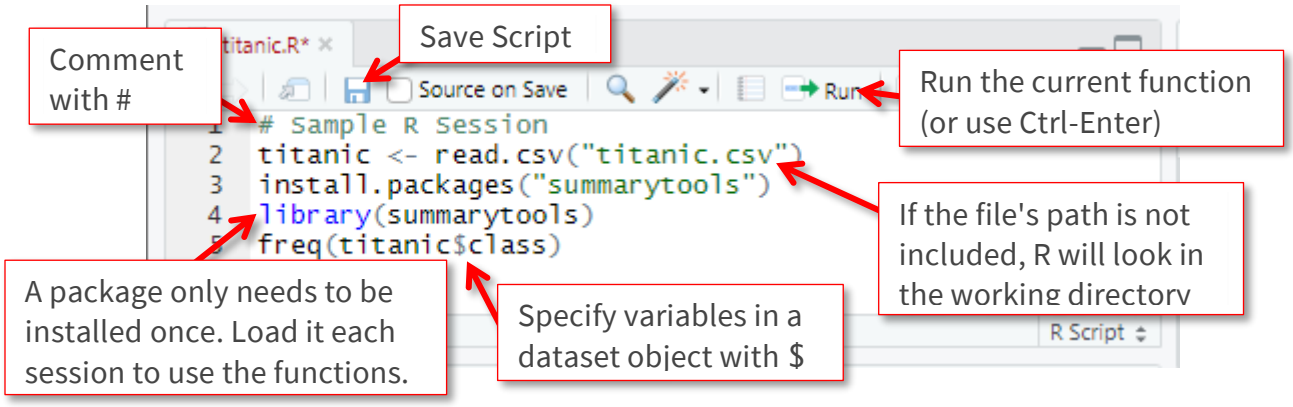
1. Download and Install **R** and **RStudio Desktop**

   R: https://cran.r-project.org/bin/windows/base/ or https://cran.r-project.org/bin/macosx/
   RStudio: https://rstudio.com/products/rstudio/download/

2. Open **RStudio**



3. Use the **Script** file to write instructions for R. Click Run or press Ctrl-Enter to execute code.



4. Write **functions** to accomplish tasks. Use the help to see what **arguments** can be used.

```
mydata <- read_excel("data.xlsx", trim_ws = TRUE, na = c("NA",""))
```

- Object: Objects can be data, lists, or single values
- Assignment Operator: Here, it creates an object
- Function Name
- Positional Argument: ()'s hold a functions' arguments
- Named Argument: Commas separate arguments
- Named Argument: TRUE is Yes, FALSE is No
- c() function makes a list one "thing"
- Strings are quoted, "" is blank

Starter R: Import Data

This uses functions from the **tidyverse**. Other options include data.table or base R.

Prepare your Environment

```
install.packages("tidyverse")      # also installs haven and readxl
install.packages("summarytools")   # only need to install these once

library(tidyverse)  # Load dplyr ggplot2 readr titbble stringr & forcats
setwd("L:/MyProject") # Set the working directory (where your files are)
```

Load Data

From Spreadsheets (pick one read_ function)

```
mydata <- read_csv("mydata.csv")           # CSV
mydata <- read_excel("mydata.xlsx")        # Excel, 1st sheet
mydata <- read_excel("mydata.xlsx", sheet = "data", na = c("", "NA"))
```

From Statistical Software (pick one read_ function)

```
library(haven)
mydata <- read_sav("mydata.sav")           # SPSS
mydata <- read_dta("mydata.dta")           # Stata
mydata <- read_sas("mydata.sas7bdat")      # SAS
mydata <- as_factor(mydata)   # turn vars with labeled values to factors
```

If specifying the file name is not working, replace it with file.choose() like this:

```
mydata <- read_csv( file.choose() )
```

Prepare Data

Check that your data loaded correctly and that each variable is the correct type: ***<type>***

Categorical variables must be *Factors* or *fct*, and quantitative variables must be *int* or *num*

```
glimpse(mydata)     # Or, click the blue arrow in the Environment tab
```

To fix the type, or make other modifications, use functions starting with **as.** , **fct_** , or **str_** . Search help to see choices, arguments can tweak the results. Save data with a new name.

```
df <- mutate( mydata,
     class = as.factor(class),     # Arguments allow specifing labels
     class = fct_rev(class),       # Reverses the ordering of the levels
     emb_c = str_detect(emb, "C")  # Will be TRUE if there is a C
     )
df <- filter( df, age >= 18, state == "VA", good != 0 ) # Remove rows
df <- drop_na(df)                    # Listwise deletion, from tidyr
```

Starter R: Descriptive Statistics

- Using *package*`::`***function***() avoids having to load a package with library(*package*).
- Use **%>%** to pass the left side as the 1st argument for the function on the right. This helps to avoid nested functions and/but can lead to long chains of code.
- This is for a data frame object **df**, which has the varaibles **Age** and **Class.**
- Install relevant packages and load tidyverse as needed: `library(dplyr)`

Quick Explore

```
df <- select(mydata, Class, Sex, Age)      # Keep only vars of interest

ggquickeda::run_ggquickeda(df)             # Point and Click Interface
esquiesse::esquisser(df)                   # ggplot2 builder
GGally::ggpairs(df)                        # Bivariate
```

Descriptive Statistics

```
library(summarytools)
```

Frequency Table for a Categorical Variable

```
freq(df, Class)
freq(df, Class, cumul = FALSE, order = "freq")
```

Summary Statistics for Quantitative Variables

```
descr(df)
descr(df, Age, stats = "fivenum", transpose = TRUE)
```

More to Try

```
library(summarytools)
dfSummary(df) %>% view      # Univariate
dfSummary(df, varnumbers = FALSE, valid.col = FALSE) %>% view
```

Focusing on numeric variables

```
GGally::ggscatmat(df, color = "species")
stargazer::stargazer(df, type = "text", median = TRUE)
```

Crosstabulation for Two Categorical Variables

```
ctable(df$Freq, df$Class, prop = "Columns")         # base R
with( df, ctable(Freq, Class, prop = "Columns"))    # base R & with
df %>% select(Sex,Class) %>% ctable(prop = "Rows", useNA = "ifany") # dplyr
```

Starter R: Merge and Restructure

Combine Two Files:

Append: Same Variables, Different Cases

```
mydata <- dplyr::union(wave1, wave2)
```

Join: Same Cases, Different Variables

Keeps everything in trials, but only matching rows in demos. Also inner_join (only matching rows) and full_join (all rows) as needed

```
mydata <- dplyr::left_join(trials, demos, by = id)
```

Reshape One File:

Older versions of tidyr use the functions gather() and spread() with different arguments.

Wide to Long: Variables to Cases

Example with arguments, cols can be lists like `apple:zebra` or `-id` (all but id)

```
df <- tidyr::pivot_longer(mydata, cols = starts_with("w_"),
             names_to = words, values_to = rt,
             names_prefix = "w_", values_drop_na = TRUE
      )
```

Long to Wide: Cases to Variables

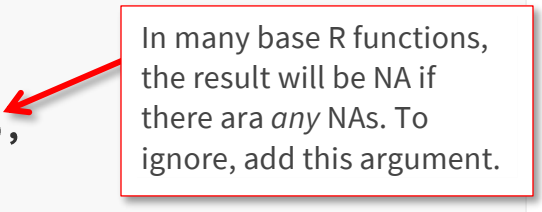
id_cols need not be specified if it is all columns except for those in …_from

```
df <- tidyr::pivot_wider(mydata, id_cols = id,
             names_from = words, values_from = rt,
             names_prefix = "w_", values_fill = NA
      )
```

Aggregate: Collapse Values Across Rows

Examples summary statistics.

```
library(dplyr)
df <- mydata %>% group_by(state) %>%
      summarize(
             country = first(country),
             total = mean(cost, ra.rm=TRUE),
             count = n()
      )
```



Starter R: Statistical Analysis

Step 1: Create the formula

ex. reaction_time ~ color + age + gender + color:gender + (1 | subject) + (color | word)

Core Components:

Y ~ X	Y predicted from X i.e., Y is the Dependent Variable. If no DV, just use " ~ X"
+ X	include other IVs, will dummy code if a factor variable
+ X:Z	include an interaction between X and Z
+ X*Z	factorial (both variables and interaction), shorthand for "+ X + Z + X:Z"

For Repeated Measures, Fixed Effects, Random Slopes, and Random Intercepts :

+ (1|Z)	Random intercept / Fixed Effects / Repeated Measures / (2-Level Models)- Control for Z by allowing each value to generate it's own Y intercept
+ (1|Z/P)	Add Nested Fixed Effects (3 level models) - Also control for P within Z, ex. "(1 | school/teacher)"
+ X + (X|Z)	Add a correlated random-effects slope - each value of Z can have a different slope/coefficient for X.
+ X + (X||Z)	Add an uncorrelated (think "||" is separated) random-effects slope for each value of Z on X

Temporary Change Variable Types

+ factor(Z)	If Z is not a factor variable, and you want to include dummy codes
+ as.numeric(Z)	If Z is an ordered factor variable, and you want to treat it as interval

Step 2: Choose the function and arguments

Linear **M**odel , **G**eneral **L**inear **M**odel, & **L**inear **M**ixed **E**ffects **R**egression

```
linear <- lm(formula, data = mydata )
logit  <- glm(formula, data = mydata, family = "binomial" )
anova  <- afex::aov_4(formula, data = mydata )
mixed  <- lme4::lmer(formula, data = mydata, REML = FALSE)
```

Unless you know what REML is, set to FALSE. Otherwise, you cannot compare models with different fixed efects.

Step 3: Display results

Use whatever name you saved the model as below

```
summary(model)
stargazer(model, model2, model3, type="text")   # not for ANOVA
```

Step 4: Model Diagnostics

```
# This will produce multiple plots after you press <Return>
plot(model)
# Compare models statistically
anova(model_null,model)
```