Import Dataframe

```
df = read.csv("/path/filename.csv")
library("readxl")
df = read_excel("path/filename.xlsx")
```

Notes:

- to convert all string variables to categories,add:

, stringsAsFactors = TRUE

R Basics

```
install.packages("x")
library(x)

summary(df)
str(df)
View(df)

summary(df$continuous)
table(df$categorical)
prop.table(table(df$categorical))
```

Export Dataframe / Output

```
write.csv(df, file= "name.csv")
write.csv(var, file= "name.csv")
```

Notes:

- can be used to Export Edited df, lists, coefficients of a model, etc

add New Variable to Dataframe

```
df$new_var = x
```

Delete Variable/Column

```
df = df[ , -5]
df = df[ , -5:-10]
df = df[ ,- c(1, 3, 4)]
df$var = Null
```

Notes:

- remove column #5
- remove columns # 5 to 10
- df2 can be used to be new updated data ; df2 = df[,-5]

Outlier Treatment

Find Percentile:

```
quantile(df$var, 0.99)
quantile(df$var, c(0.01, 0.99)
quantile(df$var, seq(0, by= 0.1))
```

Set Cutoff (Capping & Flooring):

```
df$var[(df$var > x)]
df$var[(df$var > x)] = x
```

Notes:

- displays cases meeting condition of crossing the cutoff, then replaces them with a certain value (eg.99th centile, 3×99th centile, mean) or case can be deleted
- on the lower end (< 1st centile) if we want to go 3 times lower we use < and multiply by × 0.3 (not × 3)

Z-Score Standardization

```
x_z = scale(df$x)
```

Plots

Histogram:

```
hist(df$var, main = "label", xlab= "label",
      col= "red2", border= FALSE, breaks = #)
```

Scatterplot:

```
plot(df$var1, df$var2)
pairs(df[c("var1","var2","var3")])

library(psych)
pairs.panels(df[c("var1", "var2", "var3")])
```

Boxplot:

```
boxplot(df$var)
```

Bar chart:

```
barplot(table(df$var))
```

Pie chart:

```
pie(c(0.5, 0.2), labels = c("  ,  "))
```

Notes:

- parameters in histograms are similar in the rest (press F1 and see)
- in the vector, column # can be used instead of variable names (but using variable names is better in case column # is changed)
- the psych package creates a scatterplot of matrices (SPLOM)

Correlation

```
cor(df)
cor(df[ ,1:5])
cor(df[ ,c(1, 3, 4)])
round(cor(df[ ,c(1, 3, 4)]), 2)
```

Notes:

- gets columns 1-5
- gets columns 1,3,4 (var names can be used too)
- rounds to 2 decimals

Missing Data

Find:

```
summary(df)
sum(is.na(df))
which(is.na(df$var))
```

Exclude:

```
mean(df$var, na.rm = TRUE)
```

Impute:

```
df$var[(is.na(df$var))]
= mean(df$var, na.rm = TRUE)
```

Notes:

-if training a model, to avoid ***Data Leakage*** *(including testing data in the training set);* split the data, then impute.

Never impute the whole dataset before splitting.

Dummy Variables

```
library("dummies")
df = dummy.data.frame(df)
```

Notes:

- this will also remove the original converted Categorical Variable
- dummy variables must be 1 less than the # of Categories
- i.e. one Dummy Variable should be discarded
- if a categorical variable has numbers (eg. 1,2,3); it should be converted to a factor using:

```
df$var = as.factor(df$var)
```

In order to be detected by the dummies function

Notes:

- set.seed(#) is a certain pattern of randomization, for replication if needed

If Statement

```
Obese = ifelse(df$BMI >= 30, 1, 0)
```

Group Multiple Categories into One

```
Fat <- df$weight %in% c("Overweight","Obese")
```

Notes:

- %in% is similar to an if statement (i.e. if any of these categories:)
- it returns a boolean (true or false); anyone who is Overweight or Obese will be Fat (TRUE), otherwise will be Fat (FALSE).

Generate a Normal Variable

```
set.seed(0)
rnorm(n, mean= #, sd= #)
```

Notes:

- setting seed is a consistent randomized pattern, for replication if needed

Generate a Categorical Variable

```
set.seed(0)
sample(x = c("Heads", "Tails"),
          prob = c(0.5, 0.5),
          size = #,
          replace = TRUE)
```