Essential Statistics with R: Cheat Sheet

Important libraries to load

If you don't have a particular package installed already: `install.packages(Tmisc)`. Only **dplyr** and **broom** are strictly required.

```
library(dplyr)    # for filter(), mutate(), %>%, etc. see dplyr lesson.
library(broom)    # for model tidying with tidy(), augment(), glance()
library(ggplot2)  # optional, for making plots in this lesson
library(readr)    # optional, for optimized read with read_csv() instead of read.csv()
library(Tmisc)    # optional, for gg_na() and propmiss()
```

The pipe: %>%

When you load the **dplyr** library you can use **%>%**, the *pipe*. Running **`x %>% f(args)`** is the same as **`f(x, args)`**. If you wanted to run function `f()` on data `x`, then run function `g()` on that, then run function `h()` on that result: instead of nesting multiple functions, **`h(g(f(x)))`**, it's preferable and more readable to create a chain or pipeline of functions: **`x %>% f %>% g %>% h`**. Pipelines can be spread across multiple lines, with each line ending in **%>%** until the pipeline terminates. The keyboard shortcut for inserting **%>%** is Cmd+Shift+M on Mac, Ctrl+Shift+M on Windows.

Functions

Function	Description
`read.csv("path/nhanes.csv")`	Read in `nhanes.csv` in the `path/` folder
`View(df)`	View tabular data frame `df` in a graphical viewer
`head(df)` ; `tail(df)`	Print first and last few rows of data frame `df`
`mean`, `median`, `range`	Descriptive stats. Remember `na.rm=TRUE` if desired
`is.na(x)`	Returns `TRUE`/`FALSE` if `NA`. `sum(is.na(x))` to count `NA`s
`filter(df, ..,)`	Filters data frame according to condition `...` (dplyr)
`t.test(y~grp, data=df)`	T-test mean y across *grp* in data `df`
`wilcox.test(y~grp, data=df)`	Wilcoxon rank sum / Mann-Whitney U test
`lmfit <- lm(y~x1+x2, data=df)`	Fit linear model y against two x's
`anova(lmfit)`	Print ANOVA table on object returned from `lm()`
`summary(lmfit)`	Get summary information about a model fit with `lm()`
`TukeyHSD(aov(lmfit))`	ANOVA Post-hoc pairwise contrasts
`xt <- xtabs(~x1+x2, data=df)`	Cross-tabulate a contingency table
`addmargins(xt)`	Adds summary margin to a contingency table `xt`
`prop.table(xt)`	Turns count table to proportions (remember `margin=1`)
`chisq.test(xt)`	Chi-square test on a contingency table `xt`
`fisher.test(xt)`	Fisher's exact test on a contingency table `xt`
`mosaicplot(xt)`	Mosaic plot for a contingency table `xt`
`relevel(x, ref="control")`	Re-level a factor variable
`glm(y~x1+x2, data=df, family="binomial")`	Fit a logistic regression model
`power.t.test(n, power, sd, delta)`	T-test power calculations
`power.prop.test(n, power, p1, p2)`	Proportions test power calculations
`tidy() augment() glance()`	Model tidying functions in the broom package

ggplot2 basics

Build a plot layer-by-later, starting with a call to `ggplot()`, specifying the data and aesthetic mappings, for instance, to x/y coordinates and color. Continue building a plot by adding layers such as geometric objects (geoms) or statistics, like a trendline. The example below will use `mydata`, plot `xvar` and `yvar` on the x and y axes, plot points colored by levels of `groupvar`, and add a linear model trendline.

```
ggplot(mydata, aes(xvar, yvar)) + geom_point(aes(color=groupvar)) + geom_smooth(method="lm")
```