

Essential Statistics with R: Cheat Sheet

Important libraries to load

If you don't have a particular package installed already: `install.packages(Tmisc)`.

```
library(readr)      # for optimized read with read_csv() instead of read.csv()
library(dplyr)      # for filter(), mutate(), %>%, etc. see dplyr lesson.
library(ggplot2)    # for making plots in this lesson
library(broom)      # OPTIONAL: for model tidying with tidy(), augment(), glance()
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
<code>read_csv("path/nhanes.csv")</code>	Read <code>nhanes.csv</code> in the <code>path/</code> folder (readr)
<code>View(df)</code>	View tabular data frame <code>df</code> in a graphical viewer
<code>head(df) ; tail(df)</code>	Print first and last few rows of data frame <code>df</code>
<code>mean, median, range</code>	Descriptive stats. Remember <code>na.rm=TRUE</code> if desired
<code>is.na(x)</code>	Returns TRUE/FALSE if NA. <code>sum(is.na(x))</code> to count NAs
<code>filter(df, ...)</code>	Filters data frame according to condition ... (dplyr)
<code>t.test(y~grp, data=df)</code>	T-test mean <code>y</code> across <code>grp</code> in data <code>df</code>
<code>wilcox.test(y~grp, data=df)</code>	Wilcoxon rank sum / Mann-Whitney <i>U</i> test
<code>lmfit <- lm(y~x1+x2, data=df)</code>	Fit linear model <code>y</code> against two <code>x</code> 's
<code>anova(lmfit)</code>	Print ANOVA table on object returned from <code>lm()</code>
<code>summary(lmfit)</code>	Get summary information about a model fit with <code>lm()</code>
<code>TukeyHSD(aov(lmfit))</code>	ANOVA Post-hoc pairwise contrasts
<code>xt <- xtabs(~x1+x2, data=df)</code>	Cross-tabulate a contingency table
<code>addmargins(xt)</code>	Adds summary margin to a contingency table <code>xt</code>
<code>prop.table(xt)</code>	Turns count table to proportions (remember <code>margin=1</code>)
<code>chisq.test(xt)</code>	Chi-square test on a contingency table <code>xt</code>
<code>fisher.test(xt)</code>	Fisher's exact test on a contingency table <code>xt</code>
<code>mosaicplot(xt)</code>	Mosaic plot for a contingency table <code>xt</code>
<code>factor(x, levels=c("wt", "mutant"))</code>	Create factor specifying level order
<code>relevel(x, ref="wildtype")</code>	Re-level a factor variable
<code>glm(y~x1+x2, data=df, family="binomial")</code>	Fit a logistic regression model
<code>power.t.test(n, power, sd, delta)</code>	T-test power calculations
<code>power.prop.test(n, power, p1, p2)</code>	Proportions test power calculations
<code>tidy() augment() glance()</code>	Model tidying functions in the broom package

ggplot2 basics

Build a plot layer-by-layer, 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")
```