R at a Glance:

Useful functions and syntax for Psych 252

Getting started

Install library: `install.packages('mylibrary')`
Load library: `library(mylibrary)`
Useful libraries to start with:

Load data	Clean up data	Statistics	Share results
xlsx	plyr	car	ggplot2
R.Matlab	dplyr	nlm	rmarkdown
jsonlite	tidyr	lme4	

Set working directory: `setwd('~/path/to/my/data')`
Get current working directory: `getwd()`

Search for a function in the R documentation:
By its exact name: `?functionName`
By part of its name: `??fun`

Data frames

Reading and writing data

Combine elements into a vector	`c(1,2,3,4)`
Create number sequence	`seq(start, end)`
Combine vectors by rows	`rbind(vectr1, vectr2)`
Combine vectors by columns	`cbind(vectr1, vectr2)`
Create data frame from vectors	`data.frame(tag = value)`
Load an existing data set[1]	`read.csv('mydata.csv')` `read.xlsx('mydata.xls')`

Exploring datasets

View topmost rows	`head(data)`
View structure of data	`str(data)`
View summary of data[2]	`summary(data)`

Data types

(In the examples below, fill in `datatype` with the type of data you want, such as `factor`, `character`, `numeric`, or `logical`.)

Converting to other data types	`as.datatype(vectr)`
Check data type of a vector	`is.datatype(vectr)`
View/set levels of a factor	`levels(data$columnName)`

Selecting and extracting data

View all column names	`names(data)`
Get column by name	`data$columnName`
Get i-th row	`data[i, ]`
Get j-th column	`data[, j]`
Get element at row i, column j	`data[i, j]`
Select rows using logical[3]	`data[data$group == 'A', ]`

Notes:
[1] Make sure you are in the correct working directory!
[2] `summary` returns different results depending on the object you want to summarize. Try it out with data frames, models, statistical tests, etc., to see what information it gives you.
[3] Don't forget the comma! This will return all rows that match the condition in the brackets. You can use this syntax to subset your data frame by *any* criterion, such as subjects that are above a certain age, all measurements taken in a drug trial before treatment, etc.

Basic math & statistics

	Name	Example
Arithmetic	Variable assignment	`x <- 10` `x = 10`
	Addition	`x + y`
	Subtraction	`x - y`
	Multiplication	`x * y`
	Exponent	`x ** y` `x^y`
	Modulus	`x %% y`
Comparison	Less than	`x < y`
	Less or equal to	`x <= y`
	Greater than	`x > y`
	Greater or equal to	`x >= y`
	Equal to	`x == y`
	Not equal to	`x != y`
Logic	NOT x	`!x`
	x OR y	`x | y`
	x AND y	`x & y`
	x IN y	`x %in% y`
Other operations	Exponential	`exp(x)`
	Logarithm	`log(x)`
	Square root	`sqrt(x)`
	Round	`round(x)`
	Absolute value	`abs(x)`
	Sum	`sum(vectr)`
	Scale & center	`scale(vectr)`
	Length of vector	`Length(vectr)`
Statistics	Maximum	`max(vectr)`
	Minimum	`min(vectr)`
	Mean	`mean(vectr)`
	Median	`median(vectr)`
	Std. dev.	`sd(vectr)`
	Variance	`var(vectr)`
	Correlation	`cor(vectr1, vectr2)`
	Covariance	`cov(vectr1, vectr2)`
	T-Test	`t.test(y ~ x ,data)` `t.test(vectr1, vectr2)`
	Chi-squared test	`chisq.test(table)`
	ANOVA	`aov(y ~ x, data = d)` `lm(y ~ x, data)`

More statistics

Sampling from distributions

For the commands below, use `norm` to sample from the normal distribution with mean 0 and s.d. 1, or substitute `norm` with the name of another distribution.

View all available distributions	`?Distributions`
Get probability of quantile x[1]	`pnorm(x)`
Get quantile with probability p[1]	`qnorm(p)`
Get n samples from distribution	`rnorm(n)`

[1] By default, `pnorm(x)` will return $P(X \leq x)$, the probability of drawing values that are less than or equal to x, and `qnorm(p)` will return some x that satisfies $P(X \leq x) = p$. To instead compute $P(X > x)$, use `pnorm(x, lower.tail = FALSE)`.

Modeling datasets

Simple linear model	`lm(y ~ x, data)`
Logistic regression	`glm(y ~ x, data, family = "binomial")`
Mixed-effects model	`lmer(y ~ x + (int|slope), data = mydata)`
Summary of model	`summary(mymodel)`
Compare model fits	`anova(model1, model2)`

Programming basics

Comments

```
# this is a comment! the computer will
# ignore it, but the humans reading your
# code will appreciate it.
```

Defining functions

```
myfunction <- function(input1, input2, … ) {
   statements
   return(output)
}
```

Control statements

If-else statements:

```
if (chk1) {
  # run this if chk1 is true
} ifelse (chk2) {
  # run this if chk2, but not chk1, is true
} else {
  # if all else fails, run this
}
```

For loops:

```
for (item in sequence) {
  # carry out these same instructions for
  # each item in the vector sequence
}
```

Useful commands

Combine strings of text	`paste(str1, str2)`
'Attach' data frame to environment	`attach(mydata)`
'Detach' data frame[	`detach(mydata)`
Evaluate expression using contents of data frame	`with(mydata, expr, …)`

Plotting with ggplot2

Making any plot with ggplot follows the same basic steps:

1. Choosing a **dataset** to plot
2. Using **geoms** to specify what kinds of marks (such as lines, dots, or bars) will appear on the plot
3. Using **aesthetic mappings** to specify how different properties of the dataset will appear on the plot. The most basic of these is choosing which variables will appear on the x and y axis.
4. Changing the look of the plot with **custom settings**.

Basic syntax

Use + to add elements, layers, and custom options.

```
ggplot(data, aes(x=IV, y=DV, color=cond))[1]+
          geom_point() +
          geom_smooth(method = 'lm')[2] +
          xlab('Time') +
          ylab('Score')
```

[1] `color` color-codes lines and points according to the factor of your choice (here, 'cond'). `fill` color-codes bars in bar graphs.
[2] Each geom has custom options available that can be specified as arguments to the geom function. Check the documentation!

Geoms

In the examples below:

```
myplot <- ggplot(data, aes(x = IV, y  = DV))
```

Plot type	Usage & example

Histogram	Sorts values in x into bins, shows number of elements in each bin on the y-axis.
	`ggplot(data, aes(x=age)) +`
	`  geom_histogram(binwidth=5)`

Bar graph	Here, x is a factor, and y is a numeric vector of bar heights. `myplot + geom_bar(stat='identity')`

Scatter plot	`geom_jitter` moves points around to avoid overplotting.
	`myplot + geom_point()`
	`myplot + geom_jitter()`

Line graph	`myplot + geom_line()`

Error bars	Use `ymin` and `ymax` to set the bounds and `width` to set the width of the bars.
	`myplot +`
	`  geom_errorbar(aes(ymin = lower,`
	`  ymax = upper), width = 0.1)`

Customization

Adding a title	`ggtitle('My Plot')`
Label x-axis	`xlab('Condition')`
Label y-axis	`ylab('Response')`
Faceting	
By row	`facet_grid(gender ~ .)`
By column	`facet_grid(. ~ gender)`
By row & column	`facet_grid(age ~ gender)`
Wrap facets to page	`facet_wrap(. ~ gender)`