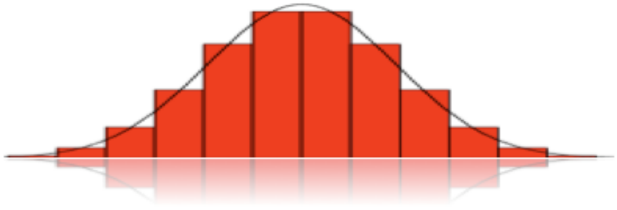
R Reference Card
Introductory Statistics



by Anthony Tanbakuchi. Version 0.5

Used Tom Short's R Quick reference as a template.

†Requires the UsingR package.
Install once with: `install.packages("UsingR")`
Load with: `library(UsingR)`

Getting help

help(topic) documentation on `topic` or function

help.search("phrase") search more generally for a word or phrase

Libraries & Packages

install.packages("package name") install a library / package. Only need to do once

library(name) load library named `name`

Input and output

scan(file) read contents of file with space separated values into a vector.

read.table(file) reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names

read.csv(file,header=TRUE) id. but with defaults set for reading comma-delimited files

save(file,...) saves the specified objects (...) in the XDR platform-independent binary format

load(file) load the datasets written with `save`

save.image(file) saves all objects

write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") prints `x` after converting to a data frame; if `quote` is `TRUE`, character or factor columns are surrounded by quotes ("); `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input

The `file` argument should be a quoted string specifying the file name or replace it with `file.choose(new=FALSE)` to interactively select a file.

source(file) read R source from a file made with `dump(list=..., file=...)`. Often used for web material `source(url("http://..."))`

Data creation

c(2,4,3,...) create vector from comma separated data

data.frame(x1, x2,...) create a data set from comma separated list of vectors. Vectors should be same length.

list(name1=x2, name2=x2,...) create a list data set from name=vector comma separated lists of vectors. Useful for unequal length vectors.

seq(from,to) generates a sequence of numbers, `by=` specifies increment; `length=` specifies desired length

factor(x,levels=) encodes a vector `x` as a factor (levels)

Slicing and extracting data

Indexing vectors

`x[n]`	nth element
`x[x > 3]`	all elements greater than 3
`x[x > 3 & x < 5]`	all elements between 3 and 5
`age[gender == "Male"]`	all ages for "Male" gender (double equal sign)

Accessing variables in data sets (data frames & lists)

`names(D)`	list all variables in data set `D`
`D$x`	access variable `x` in data set `D`
`attach(D)`	make all variables in `D` directly accessible
`detach(D)`	undo attach()

Variable information

ls() list all variables (and other objects)

length(x) number of elements in `x`

names(D) list names of variables in data set `D`.

Data selection and manipulation

sample(x, size) resample randomly and without replacement `size` elements in the vector `x`, the option `replace = TRUE` allows to resample with replacement

rev(x) reverses the elements of `x`

sort(x, decreasing=FALSE) sorts the elements of `x` in increasing order.

cut(x,breaks) divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points

match(x, y) returns a vector of the same length than `x` with the elements of `x` which are in `y` (`NA` otherwise)

which(x == a) returns a vector of the indices of `x` if the comparison operation is true (`TRUE`).

unique(x) if `x` is a vector or a data frame, returns a similar object but with the duplicate elements suppressed

table(x) returns a table with the numbers of the different values of `x` (typically for integers or factors)

subset(x, ...) returns a selection of `x` with respect to criteria (`...`, typically comparisons: `x$V1 < 10`); if `x` is a data frame, the option `select` gives the variables to be kept or dropped using a minus sign

Math

+, -, *, /, ^

factorial(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x,y), log(x), log10(x), exp(x)

sum(x) sum all elements in x x. $\sum_{i=1}^{n} x_i$

diff(x) lagged and iterated differences of vector `x`,

prod(x) product of all elements in `x`. $\prod_{i=1}^{n} x$

round(x, n) rounds elements of `x` to `n` decimals

signif(x, n) rounds elements of `x` to `n` significant digits

log(x, base) computes the logarithm of `x` with base `base`

cumsum(x) a vector which ith element is the sum from `x[1]` to `x[i]`

cumprod(x) id. for the product

cummin(x) id. for the minimum

cummax(x) id. for the maximum

choose(n, k) computes the combinations of k items selected from n total items when order is unimportant = $n!/[(n-k)!k!]$

union(x,y), intersect(x,y), setdiff(x,y), setequal(x,y), is.element(el,set) "set" functions

Excellent health statistics - smokers are less likely to die of age related illnesses.'

Descriptive Statistics: Visual

Univariate quantitative data

`table(cut(x,breaks, include.lowest=FALSE))` frequency table. `break` is the number of classes or a vector of breaks. Set `include.lowest=TRUE` for inclusive lower bounds.

`hist(x)` histogram of the frequencies of `x`

`stem(x)` stem and leaf plot.

`DOTplot(x)` [†]dot plot.

`dotchart(x)` if `x` is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

`plot(x)` plot of the values of `x` (on the *y*-axis) ordered on the *x*-axis

`boxplot(x, range=1.5)` "box-and-whiskers" plot. Set `range=0` for traditional form.

`boxplot(x1 x2)` make a set of box plots for the quantitative variable `x1` in terms of the categorical variable **x2**.

Univariate qualitative data

`t=table(x)` frequency table of vector `x`

`barplot(sort(t, decreasing = TRUE))` Pareto chart

`pie(t)` pie chart

Bivariate quantitative data

`plot(x, y)` scatter plot of `x` and `y`

`plot(t, y, type="b")` time series plot of `t` and `y`. Default for type is "p" so you must set it to "b" to get a line plot with points.

Plotting function optional arguments

`main=" "` main title, must be a variable of mode character

`xlab=" ", ylab=" "` annotates the axes, must be variables of mode character

`type="p"` specifies the type of plot, `"p"`: points, `"l"`: lines, `"b"`: points connected by lines.

`xlim=, ylim=` specifies the lower and upper limits of the axes, for example with `xlim=c(-10, 10)`.

Descriptive Statistics: Numerical

`summary(x)` gives a smart summary of the data in `x`. Output depends on `x`

`max(x)` maximum of the elements of `x`

`min(x)` minimum of the elements of `x`

`range(x)` range of the elements of `x`

`mean(x)` mean of the elements of `x`

`median(x)` median of the elements of `x`

`mode*` to find the mode use `sort(table(x))` to list the frequencies of each value

`var(x)` *sample* variance of `x`

`sd(x)` *sample* standard deviation of `x`

`quantile(x,probs=)` sample quantiles corresponding to the given probabilities (defaults to 0,.25,.5,.75,1)

`rank(x)` ranks of the elements of `x`

Distributions

R has many distributions. The base names for the common ones are: `norm`, `exp`, `gamma`, `pois`, `weibul`, `cauchy`, `beta`, `t`, `f`, `chisq`, `binom`, `geom`, `hyper`, `logis`, `lnorm`, `nbinom`, `unif`, `wilcox`. Prefix the base name with `r` for a random number generator, `d` probability density distribution $f(x)$, `p` cumulative probability distribution $F(x)$, `q` inverse cumulative probability distribution $F^{-1}(a)$ (quantile).

Random number generators

Generates `N` random numbers.

`runif(N, min=0, max=1)` uniform

`rbinom(N, n, p)` binomial

`rnorm(N, mean=0, sd=1)` normal

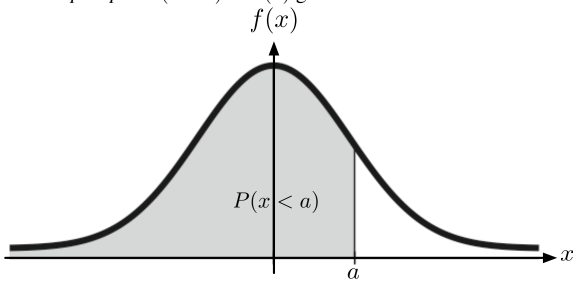
Probability distributions

Returns $p = P(x)$ given x.

`dbinom(x, n, p)` binomial

Cumulative probability

Returns p in $p = P(x < a) = F(a)$ given x.



$$P(x < a) = \int_{-\infty}^{a} f(x)\, dx = F(a)$$

Set optional argument `lower.tail=TRUE` to `FALSE` for area to the right.

`punif(x, min=0, max=1)` uniform

`pbinom(x, n, p)` binomial

`pnorm(x, mean=0, sd=1)` normal

`pt(x, df)` Student's t

`pf(x, df1, df2)` the F

`pchisq(x, df)` the χ^2

Inverse cumulative probability

Solves for a given p in $p = P(x < a) = F(a)$

Set optional argument `lower.tail=TRUE` to `FALSE` if p refers to area to the right, otherwise p must refer to area to the left of a.

`qunif(p, min=0, max=1)` uniform

`qbinom(p, n, p)` binomial

`qnorm(p, mean=0, sd=1)` normal

`qt(p, df)` Student's t

`qf(p, df1, df2)` the F

`qchisq(p, df)` the χ^2

Hypothesis tests

All tests have the optional arguments with defaults:

`alternative="two.sided"` alternatively use "less" or "greater"

`conf.level=0.95` sets confidence level for reported confidence interval, it has no effect on the *p*-value.

One sample

`binom.test(x, n, p)` proportion test for `x` successes in `n` trials with `p`=p_0 null hypothesis of success. Exact test using binomial distribution.

`prop.test(x, n, p)` proportion test for `x` successes in `n` trials with `p`=p_0 null hypothesis of success. Uses normal approximation to the binomial. ($z = \sqrt{\chi^2}$)

`t.test(x, mu=0)` t test with null hypothesis `mu`=μ_0.

Two sample

`prop.test(x, n)` proportion test for `x=c(x1, x2)` successes in `n=c(n1, n2)` trials with null hypothesis that $p_1 = p_2$. Uses normal approximation to the binomial. ($z = \sqrt{\chi^2}$)

`t.test(x1, x2)` t test with null hypothesis $\mu_1 = \mu_2$ for sample vectors `x1` and `x2`.

Testing normality

`qqnorm(x); qqline(x)` plot normal quantiles with normal line

`wilcox.test(x)` Test data in `x` against null hypothesis that `x` is from normal population

Correlation

`cor(x, y)` Linear correlation coefficient for vectors `x` and `y`

`cor.test(x, y)` Test significance of linear correlation

Regression

`results=lm(y~x)` Linear regression of `y` on `x` vectors

`results` View the results

`plot(x, y); abline(results)` Plot regression line on data

`predict(results, newdata=data.frame(x=5), int="pred")` Predict y when $x = 5$ and show the 95% prediction interval.

Contingency Tables

`D=data.frame(c1, c2, c3, ...)` Creates a table of data from vectors of column data `c1, c2, c3, ...`

`chisq.test(D)` Test homogeneity or independence for contingency table `D`

ANOVA: one way

`data=list(x1=x1, x2=x2, ...)` Create data set of treatment levels

`datastack=stack(data)` Make a data stack

`results=aov(values~ind, data=datastack)` Run ANOVA

`summary(results)` Summarize results