

36-350 Statistical Computing R/Python Cheatsheet

stdout	<pre>print("R code!")</pre>	<pre>print("Python code!")</pre>
find type	<pre>typeof(x)</pre>	<pre>type(x)</pre>
removing vars	<pre>rm(x) exists('x')</pre>	<pre>del x</pre>
vectors		
instantiation	<pre>x <- c(0, 0, 0, 0, 0) x <- rep(0, 5) x <- vector("integer", 5) x <- integer(5) x <- seq(1, 5, by = 1) x <- seq(1, 5, length.out = 5) x <- 1:5</pre>	<pre>x = np.zeros(5, dtype = int) x = np.ones(5, dtype = float) x = np.full(5, 5.43) x = np.arange(0, 10, 2) x = np.linspace(1, 5, 5) x = np.array(["a", "b", "c", "d", "e"])</pre>
usage	<pre>length(x) x[1] # 1-indexed x[1:2] # first 2 x[-y] # x without indices y rev(x) sort(x) order(x) as.vector(x) # cleaning attrs sum(x, na.rm=TRUE) unique(x) table(x) union(x, y) intersect(x, y) setdiff(x, y) setequal(x, y) is.element(x, y) # x[i] in y? match(x, y)</pre>	<pre>x.size x[0] # 0-indexed x[b:e:s] # [begin, end) with step, accepts negatives. WARNING: returns a "view," like a pointer instead of a copy np.delete(x, y) np.sort(x); x.sort() # sorted copy; in-place np.argsort(x) # indices that would sort an array np.nansum(x) np.unique(x) np.unique(x, return_counts=True) np.union1d(x, y) np.intersect1d(x, y) np.setdiff1d(x, y) set(x) == set(v) np.in1d(u, v)</pre>
rand	<pre>set.seed(5) runif(8)</pre>	<pre>np.random.seed(5) np.random.random(8)</pre>
logical subsetting	<pre># let x: vector[int] x > 0: vector[bool] x[x>0 & x<0.4] # elements of x where x>0 and x<0.4 which(x < 0)</pre>	<pre># let x: np.array[int] x > 0: np.array[bool] # can do like R with much parentheses, but consider np.logical_... (https://stackoverflow.com/questions/33384529/difference-between-numpy-logical-and-and) np.where(x < 0)</pre>
edge case data types	<pre>NA; is.na() # missing data NULL; is.null() # fns that return nothing NaN; is.nan() # e.g., 0/0 Inf; -Inf; is.infinite() # e.g., 1/0 # NAN IS STRICT SUBSET OF NA</pre>	<pre># no NA in numpy None; is None # fns that return nothing np.nan; np.isnan() # e.g., 0/0 np.inf; np.isinf() # e.g., 1/0</pre>
lists/dicts	<pre>list(foo=1:5, bar=c("a", "b")) x[[2]] # column 2: "a" "b" x[["bar"]] # equivalent unlist(x) # list elements -> a vector data.frame(u = 1:2, v = c("a", "b")) # dfs are square lists matrix(1:6, nrow=2) # matrices are dfs w/ cols of same type # ^ filled col by col x[2, 1]; x[2,] # matrix indexing is standard</pre>	<pre>{"foo": np.arange(1, 6), "bar": np.array(["a", "b"])} # use pandas for this use case.</pre>