Common R functions - Cheat Sheet

## Base functions

|  |  |  |
| --- | --- | --- |
| Function | Description | Comments |
| install.packages(“*packageName*”)  install.packages(*c(“a”, “b”, “c”)*) | Install single package  Install multiple packages |  |
| library(*package*) | Load package |  |
| str(*object*) | See object info (variable names and types, etc.) | Mnemonic: Short for *str*ucture. |
| is.data.frame(*object*)  is.character(*object*)  is.double(*object*)  is.integer(*object*)  is.logical(*object*) | Test whether objects are of this type. |  |
| is.na(*object*) | Test if element is NA. | Test if NA with is.na() not with == |
| data.frame(*objects*) | Create df from columns |  |
| as.character(*object*)  as.double(*object*)  as.integer(*object*)  as.logical(*object*) | Coerce objects to each of these types if possible. |  |
| head(*df,* ***n*** *= 6*) | Get first **n** rows of df. | Default **n** = 6. |
| format(*DateObject, "%B-%d,%Y"*) | Change Date format | %d, %a, %A, %m, %b, %B, %y, %Y |
| paste(*strings, objects,* ***sep****="xy")* | Concatenate strings | Strings in quotes. Default **sep** = “ “ |

## Functions from external packages

Note: Packages need to be loaded with library() function in each R session before using any of the package’s functions.

|  |  |  |  |
| --- | --- | --- | --- |
| Package | Function | Description | Comments |
| rio | import(“*path/file.ext*”)  export(“*path/file.ext*”)  convert(“*path/file.ext1*”, “*path/file.ext2*”) | Import and export dfs or convert between file types. | Notes: Always use forward slashes in file paths (use **/** not \). |
| RODBC | odbcDriverConnect (  ***'driver={SQL Server};******server****=SERVER\\instance;* ***dsn****/****dbname/database****=*  *DBNAME;* ***trusted\_connection****=yes’*) | Connect to remote server. | For SQL Server implementations.  Use either **dsn**, **dbname**, or **database** (just one); which works depends on specific SQL implementation.  If **server** name has \, replace with \\.  If not using Windows Authentication, do not use **trusted\_connection**, use **uid**=username and **pwd**=password. |
| sqlQuery(*con, “sql\_statement”)* | Submit *sql\_statement* through connection. |  |
| sqldf | sqldf(“*sql\_statement*”) | Perform SQL with R dfs |  |
| dplyr | select(*df, +Kcols/-Dcols*) | Keep (+)/Drop (-) cols. | Prefix column name with dash to drop. |
| filter(*df, cond1, cond2, ...*) | Filter rows by logic conds | Can use &/comma or | for logical ops. |
| mutate(*df,*  *col1=computation1 ,...*) | Create/change columns | Change/add new computed columns. |
| replace(*col, cond, subs*) | Replace values in column | Usually used within mutate(). |
| arrange(*df,v1,****desc****(v2),..)* | Sort by asc/desc order | asc by default. **desc**(var): descending |
| rename(*df,new1=old1,...*) | Rename columns |  |
| distinct(*df, var1, var2, ...*) | Distinct obs. of var. vals. | See distinct var. value combinations |
| group\_by(*df, V1, V2,* ...) | Group df by listed vars. | Each Vn is a layer. Use w/ summarize(). |
| ungroup(*df*) | Ungroup df by all vars. | ungroup() undoes group\_by(). |
| summarize(*gdf,new=****n****()*) | Get aggregate measures | Grouped input. Use **n**(),**sum**(V) or both. |
| left\_join(*a,b,****by*** *=“var”*) | Left join datasets | Or **by**=c("aV1"="bV1","aV2"="bV2",...) |
| right\_join(*a,b,****by*** *=“var”*) | Right join datasets | Or **by**=c("aV1"="bV1","aV2"="bV2",...) |
| inner\_join(*a,b,****by*** *=“var”*) | Inner join datasets | Or **by**=c("aV1"="bV1","aV2"="bV2",...) |
| full\_join(*a,b,****by*** *=“var”*) | Full outer join datasets | Or **by**=c("aV1"="bV1","aV2"="bV2",...) |
| magrittr | %>% | f(a,b) is equivalent to  a %>% f(b) | magrittr extends where piping will work to most, but not ALL, scenarios. Can use many pipes in series. |
| lubridate | ymd(*dateStrings*)  ydm(*dateStrings*)  mdy(*dateStrings*)  myd(*dateStrings*)  dmy(*dateStrings*)  dym(*dateStrings*) | Convert character dates to Date objects. | dateStrings can be either column of character dates or single/multiple dates in quotes. Output is a Date object. |
| year(*DateObject*)  month(*DateObject*)  day(*DateObject*) | Get year, month, or day of Date object. |  |
| hour(*DateTimeObject*)  minute(*DateTimeObject*)  second(*DateTimeObject*) | Get hour, minute, or second of DateTime object. |  |
| ymd\_hms(*datetimeString*, ***tz****=”UTC”*) | Convert chr datetimes to DateTime objects. | **tz** = "America/Indiana/Indianapolis"  Use OlsonNames() for other timezones |
| tidyr | gather(*df, key, value, ...cols...,* ***na.rm*** *= F*) | Wide to long format | *key*: column name for new variable  *value*: column name for variable values  *...cols...*: names of columns to gather  *na.rm*: option to remove rows w/ NAs |
| spread(*df, key, value,*  ***fill*** *= NA*) | Long to wide format | *key*: column values to convert to multiple columns  *value*: single column values to convert to multiple columns’ values  *fill*: value to substitute for missing key and column combination values |
| gridExtra | grid.table(*df,* ***rows****=NULL*) | Create nice image of a data table and much more. | Best to use in R Markdown docs. |