R-Guru.com Cheat Sheet for Practical R Tasks

This guide contains basic best practice examples for creating and updating tibble data frames from vectors of the same length or number of records. Examples show common R tasks for importing data, creating data frames, direct variable referencing, piping, conditional and group processing, sql components, character and date operations, variable type conversions, transposing data frames, joining data frames, appending data frames, deriving summary variables, and creating graphs and output files. When possible, base R supplied sample data frames are used in examples.

Mutate() has five features: case_when(), simple expression, summary functions, rowwise(), and group_by()/ungroup() with summary functions. Data utility functions describe and view data frames: View(df), str(df), summary(df), table(vr), print(df, n=), head(df), tail(df), row_number(), nrow(), ncol() and ls(). Tidyverse, DPLYR, STRINGR, READXL, HAVEN, LUBRIDATE and GGPLOT2 packages are required. df#-data frame names, vr# – variable names. Character or numeric variables depend on the function and values.

Import data into data frames: Data frames, CSV, Excel and SAS Datasets

```
install.packages('tidyverse')                  # install package
library(tidyverse)        # load popular data management package
readRDS("df.RDS")                              # read R data frame
read.csv("C:/mydata/my_csv.csv")               # read csv, forward '/'
read_excel("C:/mydata/my_excel.xlsx")  # read excel, missing 'NA'
sdtm <- "c:/my_sdtm"                           # create full path reference
read_sas(file.path(sdtm, "adsl.sas7bdat")) # read dataset, missing ., ''
```

Environmental Setup and Workspace

```
ls()                                           # list all objects
remove(list=ls())                              # remove all objects
# names(adsl)= tolower(names(adsl))     # lower case all variable names
```

Create data frames by combining variables

```
df <- data.frame(vr1, vr2, vr3)                # variable order
```

Derive numeric and character constants to data frame

```
df2 <- cbind(df1, vr1=1, vr2='Drug A')    # to df1, add vr1 and vr2
```

Direct variable reference to select variables and filter records

```
df2 <- df1[c('vr1', 'vr2')]          # combine selected variables by name
df2 <- df1[df1$vr1 == 'male', ]   # row, column reference, subset, all vars
print(df1[df1$vr1 == 'male', c('vr1', 'vr2')])   # print vr1 & vr2 for males
mydata$vr2 <- ifelse(vr1 >= 4, 1, 0)  # derive variable by condition
```

Piping and multiple conditional processing to derive variables

```
df2 <- df1 %>%                          # copy df1 data frame to df2
  mutate(vr2 = case_when(      # derive vr2 based on condition
  vr1 > 20 ~ "label 1",        # ifelse() is two option alternative
  vr1 > 10 ~"label 2",         # ifelse(vr1 < 51, "50 or Below",  "Above 50")
  TRUE ~ "label 3" ) )         # otherwise last case for label 3
```

SQL components (DPLYR) for selecting, filtering, mutating, and arranging

```
myquery <- ToothGrowth %>%       # 1. source df
  select(len, supp, dose) %>%          # 2. select variables, - drop
  filter(tolower(supp) == 'vc') %>%  # 3. subset records, & (and), | (or), ! (not)
  # filter(year %in% c(2010, 2011)) to subset multiple values
  mutate(dose2 = (dose*2)) %>%   # 4. derive variables w/ simple expressions
  arrange(supp, dose)                  # 5. sort records, desc()
```

Character String Operations to combine, remove, subset, and substring

```
vr3 <- str_c(vr1, sep="-", vr2)          # combine two variables as vr1-vr2
vr2 <- str_replace_all(vr1 , "Street" , "St" ) # replace 'Street' with 'St'
vr2 <- str_trim(vr1, side='both')     # remove blanks from left and right sides
vr2 <- str_extract(vr1 , "\\d*" )       # from char vr1, extract all digitss
filter(str_detect( vr1 , "Health"))    # subset records by finding text
vr2 <- str_sub(vr1 , 3 , 6 )   # substring vr1 text from 3rd to 6th position
```

Variable Type Conversion to switch between Numeric & Character Variables

```
vr2 <- as.character(vr1) # number in numeric variable to character variable
vr2 <- as.numeric(vr1) # number in character variable to numeric variable
```

Date Operations: Assignment, Periods, Durations, Intervals, and Formats

Durations - # of seconds, Periods - # of days, weeks, months and years, Intervals - duration between start and end points

```
vr1 <- as.Date("2021-01-25")            # assign date in yyyy-mm-dd format
format(date, format="%m/%d/%y")   # format as mm/dd/yy
+ ddays(1), + dweeks(1), + dmonths(1), + dyears(1)   # add 1 dy, wk, mth or yr
interval(dtvr1, dtvr2) %/% months(1)       # of months between dates
                                  # dates are stored as # of days since 1970
```

Transpose data frames to switch between long (records) & wide (variables)

Long (records) to wide (variables) structure

```
df2 <- df1 %>%                          # vr1 contains new variable name values
  pivot_wider(names_from = vr1, values_from = vr2)  # vr2 contains numbers
```

Wide (variables) to long (records) structure

```
df2 <- df1 %>%          # all other variables in df1 are group by variables
  pivot_longer(c("vr1", "vr2"))    # list all variables to be transposed
```

Group processing to derive summary variables

- ***Summary variables***
- ***First and Last Group By variables***
- ***Descriptive Statistics***

Derive summary variables

```
mtcars_cyl_summary <- mtcars %>%  # final and source data frames
  group_by(cyl) %>%               # without is overall, ignore NA
  summarize(mean_mpg = mean(mpg, na.rm = TRUE))
```

Derive First and Last group by variables

```
first_mpg <- mtcars %>%
  group_by( mpg ) %>%     # group by mpg
  slice(1)        # flag first group by records, distinct() for unique records
  slice(n())      # flag last group by records
  lead(), lag()  # next and previous record values
```

Derive Descriptive Statistics Variable, Left Join to Add to Data Frame

- Across one variable

```
vr1=c(2, 4, 6)        # combine 3 values into one variable
vr2 <- min(vr1)       # one value, max(), sum()
```

- Across variables using rowMins(), rowMax(), rowMeans()

```
df2 <- subset(df1, select=c(vr1, vr2) )         # select variables vr1 and vr2
df3$vr1 <- rowMeans(df2, na.rm=TRUE )    # derive mean of all df2 vars
```

- Across variables using rowwise() with min(), max()

```
df2 <- df1 %>%          # derive min, max variables of vr1 and vr2
  rowwise() %>% mutate(vr3= min(vr1, vr2), vr4= max(vr1, vr2))
```

- Across records using mutate(), min(), max(), mean(), sum(), percent()

```
df2 <- df1 %>%                          # 1. source df
  filter(vr1 != '.') %>%                # 2. subset non-missing records
  group_by(vr1) %>%                     # 3. group by vr1, else by overall
  mutate(vr3 = mean(vr2, .1)) %>%       # 4. derive mean vr2 with rounding
  ungroup()   # 5. ungroup to add back all original variables with vr3
```

Left join data frames to add derived variables

```
df3 <- left_join(df1, df2, by='vr1')              # join by the same by variables
df3 <- left_join(df1, df2, by= c('vr1' = 'vr2'))    # join by different by variables
                    # other joins: right_join(), inner_join(), full_join()
df3 <- crossing(df1, df2) # many-to-many join without by variables
```

Subquery condition in df1 to filter df2 records

```
df3 <- df2 %>%                  # 5. final df3 data frame
  filter(vr1 %in% (             # 4. filter vr1 values in df2 data frame
    df1 %>%                     # 1. lookup df1 data frame
    filter(vr2 == 'male') %>%   # 2. filter vr2=males in df1
    pull(vr1) %>% unique))      # 3. unique vr1 values for df2 filter
```

Append data frame records to end of first data frame records

```
df3 <- bind_rows(df1, df2)    # append data frames with uneven variables
df4 <- rbind(df1, df2, df3)    # append two or more even variables data frames
```

Graphs: Scatterplots, Lines, Boxplots, Bars and Histograms

```
ggplot(df                       # data frame name
, aes(x = vr1, y = vr2          # vr1 for x and vr2 for y axis variables
, fill= , color= , col=, size=)       # valid options with valid values
# one or more required options below, defaults unless options are specified
+ geom_point()       # scatterplot two quantitative variables
+ geom_line          # trend lines over time
+ geom_boxplot()    # boxplot one continuous and one categorical variable
+ geom_bar()         # bar of variables, options: stat=
+ geom_histogram() # histogram of x-axis for counts
+ geom_smooth(method='lm', formula=y~x, se=F) # smooth option
# one or more format options below, defaults unless options are specified
+ theme(), + ggtitle(), + xlab(), + ylab()
```

Output files: Data frames, Text, Excel and SAS Datasets

```
setwd("C:/mydataframes")          # change default working folder
getwd()                           # confirm working folder
saveRDS(df, file = "df.RDS")          # save as permanent data frame
write.table(df,"C:/myoutput/mydm.txt", sep="\t")  # save as text file
write_sas(df, "df.sas7bdat")          # save as SAS dataset
```