R Cheat Sheet

Dainish Jabeen

May 1, 2023

1 Data Types

Focus is on numerical types: Scalers, Vectors, Matrices, Data Frames and Lists.

Listing 1: General language

```
1
2       x <- 28 #Initalise var
3
4       vect <- c(1,10,100) #Initalise vector
5
6       vect < 0 # Auto does per component iteration
7
8       vect[(vect<0)] #Outputs all values < 0, () creates index
9
10      (cond && cond ..) #Create multi condition index
```

1.1 Matrix

Listing 2: Matrix language

```
1
2       matrix(data,nrow,ncol,byrow=T/F) #byrow if filled left to right
3
4       a <- matrix(1:10,byrow=TRUE,nrow=5) #Creates 5x2 matrix
5
6       cbind(matrix,vector) #Combine matrix and vectors
7
8       m[1,] #all elements of first row
9       m[,1] #all elements of first col
```

1.2 Data Frames

Listing 3: Data Frames language

```
1
2       data.frame(df,stringsAsFactors=T/F) # Converts strings to factors
3
```

```
4         df[row,col]
5         df[,c("col id 1","col id 2")]
6
7         df$newcol <- newcol #Add new col
8
9         subsetdf <- subset(df,subset=cond) #Create subset df
10
11        df <- df[order(df$col),] #Arrange by col order
12
13        df[,colnames(df) %in% list] #Output all col names in list
```

Listing 4: Data Frames Functions language

```
1
2         select(df,cols) #Choose desired cols
3         filter(df,cond)
4
5         table(df) #Cont numb of observations per level
```

Tibble data frames

- Can have list in cols
- Auto extends to match col rows

1.3 Lists

Can hold any other data type in an array type.

2 R Factor

Categorize and store data, in categorical variables.

$$factor(x = character(), levels, labels = levels, added = is.ordered(x))$$

Stores strings into categories to be used in ML tasks.

3 Dplyr

Listing 5: Combine data Caption above the listing language

```
1
2         left_join() #keep data from original
3         right_join() #keep data from destination data
4
5         inner_join() #exclude unmatched cols
6         full_join() #keep everything
7
8         mutate(df,var=condition,..) #Create new var
9         na.omit() #Remove all NA items
10
11        na.rm = True #Ignore missing vals
12
```

```
13          mutate(col_name=ifelse(condition,val,col)) #If condition met will replace col val
```

4 Tidyr

Listing 6: Manipulate data language

```
1
2           gather() #Convert from wide to long (change inner vals into a new col)
3           spread() #Convert from long to wide
4
5           separate() #Split data in a col to multiple cols
6           unite() #Combine data from cols into a col
7
8           diff() #Return lagged and iterated difference
```

5 Functions

Listing 7: Functions language

```
1
2           function.name <- function(arg)
3           {
4
5           }
6
7           rm{func} #removes func
```

Listing 8: Useful functions language

```
1
2           ls(environment()) #Print currently global vars and funcs
3
4           apply(x,margin=1/2,func) #Apply func on all rows or cols (1:row,2:col)
5           lapply(x,func) #Apply func output list
6           sapply(x,func) #Output matrix
7           tapply(x,index,func) #
8
9           #Impute example
10
11          new_df <- data.frame(
12                  sapply(df,function(col) ifelse(is.na(col),mean(x,na.rm=True),col))
13          )
14
15          summarise() #Basic stats
16
17          group_by(col) #Group df by chosen id
18          nth() #Goto nth val
19
20          arrange() #Sort works with pipelines
```

6 Loops

Python like.

Listing 9: Loops language

```
1
2       for item in list{
3
4       }
```

7 Pipeline

Use the operator % > % to allow operations on the same data through stages.

Listing 10: Pipeline language

```
1
2       New_df <- df %>%
3       step_1 %>%
4       step_2 ...
```