R vs Python: Cheat sheet of Basic Data Wrangling Syntax

using example of Iris dataset

R (dplyr,tidyr)

Import data and get basic info
- Iris<-read_csv(“iris.csv”) //readr
- dim(iris)
- names(iris)/ colnames(iris)

Select columns
- iris$Species
- iris[,c(“Id”, “SepalLengthCm”)]
- iris %>% select(c(Id, SepalLengthCm))

Select rows with conditions
- iris[iris$SepalWidthCm > 4.0 & iris$Species == “Iris=setosa”,]
- iris %>% filter (iris$SepalWidthCm > 4.0 & iris$Species == “Iris=setosa”)

Drop columns
- iris %>% select(-c(Id, SepalLengthCm))

Find number of missing values for each column
- iris %>% is.na() %>% colSums()

Drop rows with missing values
- iris[complete.cases(iris),]

Impute missing values
- iris %>% mutate(SepalLengthCm = replace_na(SepalLengthCm, mean(SepalLengthCm)), ...)

Sort values
- iris %>% arrange(desc(SepalLengthCm),PetalWidthCm)

Python(pandas)

Import data and get basic info
- iris = pd.read_csv(“iris.csv”)
- iris.shape
- iris.columns

Select columns
- iris[[“Id”, “SepalLengthCm”]]

Select rows with conditions
- iris[(iris$SepalWidthCm > 4.0) & (iris$Species == “Iris=setosa”)]

Drop columns
- iris.drop([“Id”, “SepalLengthCm”], axis = 1)

Find number of missing values for each column
- iris.isnull().sum()

Drop rows with missing values
- iris.dropna(axis = 0)

Impute missing values
- iris[“SepalLengthCm”] = iris[“SepalLengthCm”].fillna(0)
- iris.fillna(value = {'SepalLengthCm':np.mean(iris ['SepalLengthCm']), … })

Sort values
- iris.sort_values(by = [“SepalLengthCm”, “PetalWidthCm”], ascending = [False, True])

R (dplyr,tidyr)

Map values
- iris%>%mutate(new_col, case_when(condition1 ~ value1, condition2 ~ value2, …))

Wide format to long format
- iris %>% pivot_longer(2:5, names_to = "measurement", values_to = "Length")

//could also use gather(key = "Measurement", value = "Length",2:5)

Long format to wide format
- iris %>% pivot_wider(id_cols= c("Id","Species"), names_from = "Measurement",values_from = "Length")

//could also use spread(key =”Measurement”, value = “Length”)

Grouping and aggregating
- iris %>% group_by(Species) %>% summarise(count=n(),min_len=mean (SepalLengthCm))

Joining
- **inner join**: inner_join(x=df1, y=df2, by = “col”)
- **left join**: left_join(x=df1, y=df2, by = “col”)

// could also use baser merge() and specifying all.x=True for left join

Binding
- **row bind**: rbind(df1, df2)
- **column bind**: cbind(df1, df2)

Python(pandas)

Map values
- iris[‘new_col’] = iris[‘original_col’].map(lambda x: mapping_function(x))

//after defining your mapping function

Wide format to long format
- iris.melt(id_vars=['Id','Species'], value_vars=[col1, col2,…],var_name='measurement', value_name='length')

Long format to wide format
- iris_long.pivot(index = ['Id','Species'], columns=["measurement"], values="length")

Grouping and aggregation
- iris.groupby(["Species"]).agg(count = ("Id","count"), mean_len = ("SepalLengthCm","mean")).reset_index()

// reset_index helps make index into columns for easier use further

Joining
- **inner join**: pd.merge(df1, df2, on = “col”, how = “inner”)
- **left join**: pd.merge(df1, df2, on = “col”, how = “left”)

Binding
- **row bind:** pd.concat([df1,df2])
- **column bind**: pd.concat([df1,df2], axis=1)