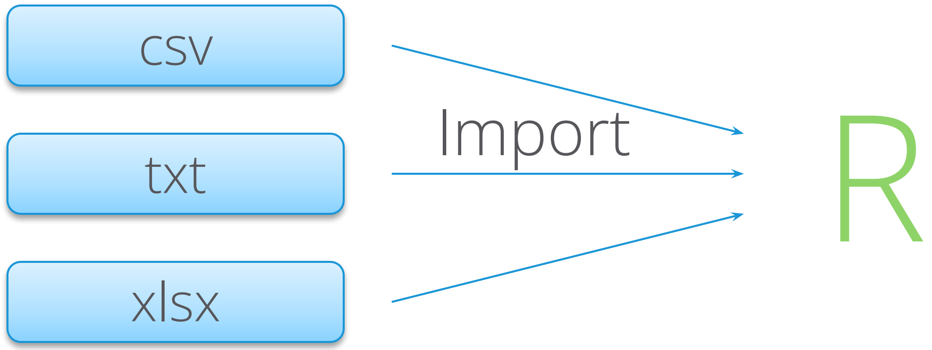
Importing & Exporting Data

(CSV, TXT and XLSX, SAS, STATA, SPSS, MySQL, PostgreSQL and Oracle)

Contents

About Importing Data

- To do any kind of data analysis in R, you first need to have a data in a file somewhere, either locally or on the Web. If the file is on the web, download and save the data in your system and load it into R to work on it.
- In R, the general term used for loading data is **Importing data.**
- The data is stored in the form of files and files can be in any format (csv, txt, xlxs,etc). There are several ways to import these files in R which we will learn in this tutorial.
- Let's first understand what are the most commonly used file formats to store data.



Common File Formats

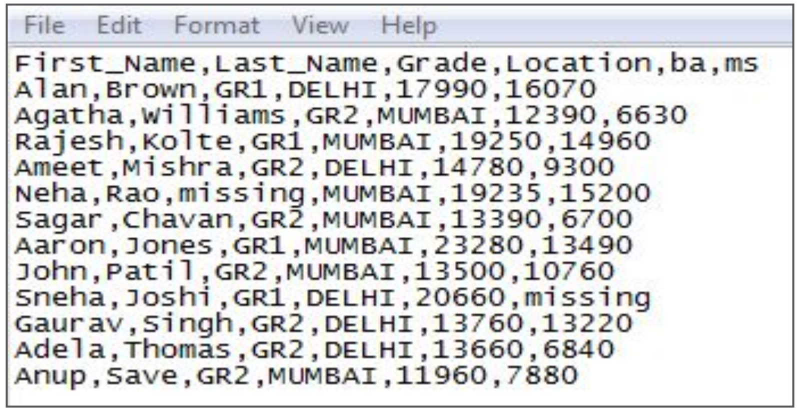
The most commonly used file formats for storing data

CSV:

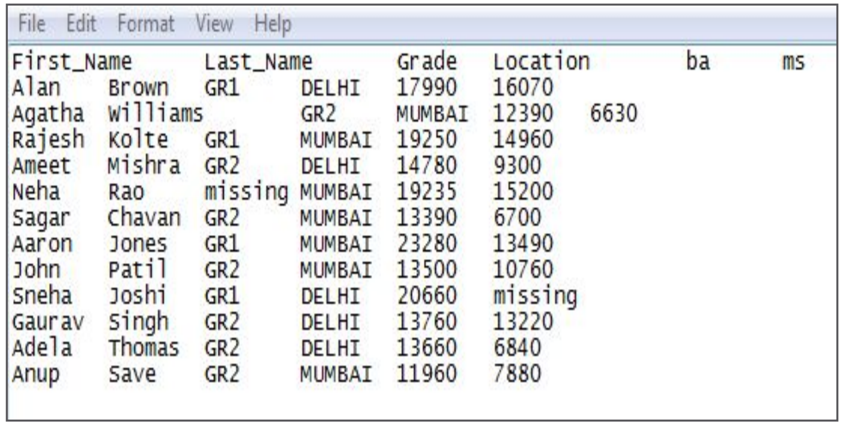
Comma Separated Values file. Allows data to be saved in a table structured format; with commas acting as separators.

TXT:

Text file. Stores data in plain text format, separated by different types of delimiters such as blank space, tab (“\t”), comma (“,”), pipe (“|”), etc.



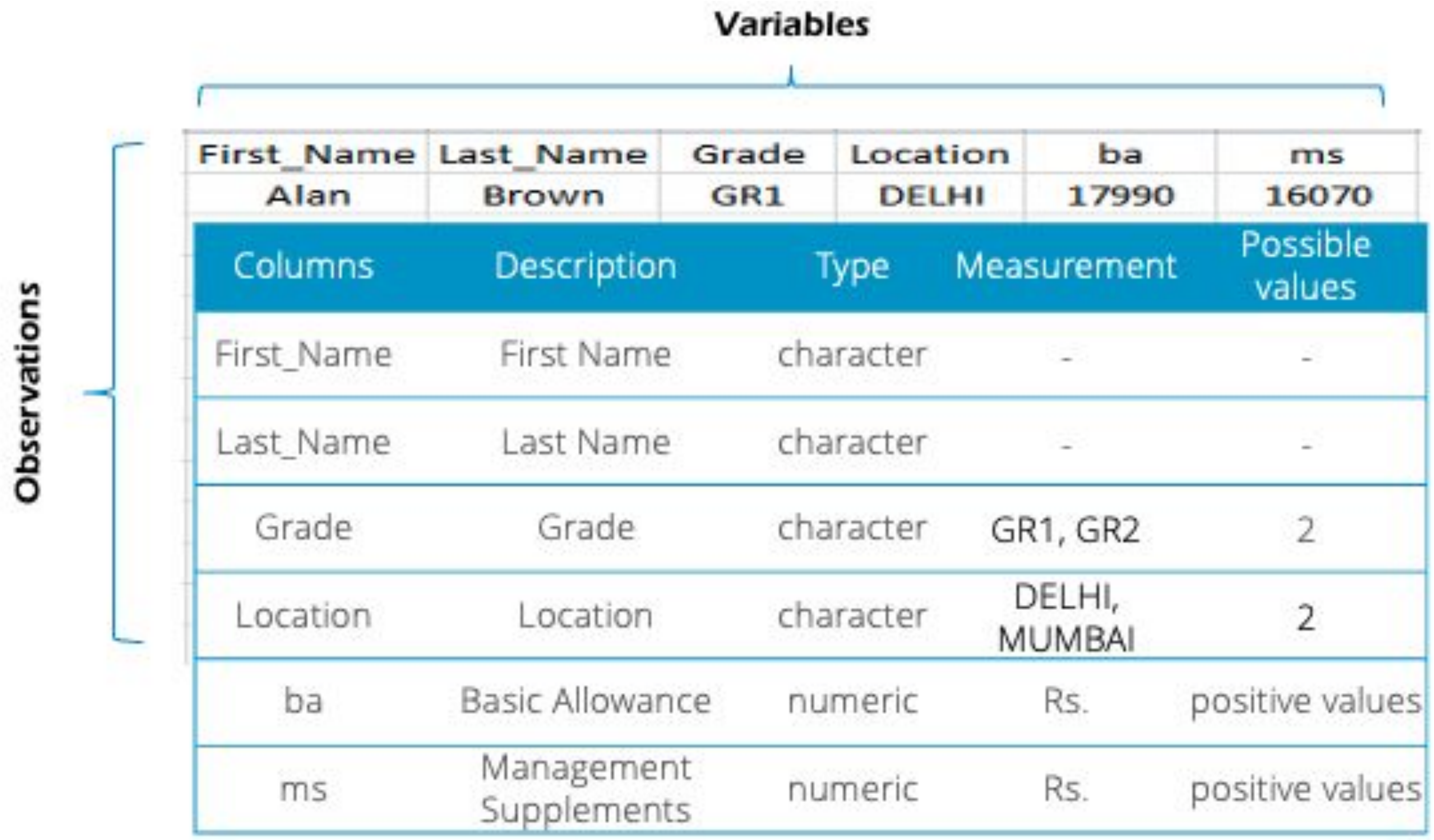
```
First_Name,Last_Name,Grade,Location,ba,ms
Alan,Brown,GR1,DELHI,17990,16070
Agatha,Williams,GR2,MUMBAI,12390,6630
Rajesh,Kolte,GR1,MUMBAI,19250,14960
Ameet,Mishra,GR2,DELHI,14780,9300
Neha,Rao,missing,MUMBAI,19235,15200
Sagar,Chavan,GR2,MUMBAI,13390,6700
Aaron,Jones,GR1,MUMBAI,23280,13490
John,Patil,GR2,MUMBAI,13500,10760
Sneha,Joshi,GR1,DELHI,20660,missing
Gaurav,Singh,GR2,DELHI,13760,13220
Adela,Thomas,GR2,DELHI,13660,6840
Anup,Save,GR2,MUMBAI,11960,7880
```



```
First_Name      Last_Name       Grade   Location        ba      ms
Alan    Brown   GR1     DELHI   17990   16070
Agatha  Williams        GR2     MUMBAI  12390   6630
Rajesh  Kolte   GR1     MUMBAI  19250   14960
Ameet   Mishra  GR2     DELHI   14780   9300
Neha    Rao     missing MUMBAI  19235   15200
Sagar   Chavan  GR2     MUMBAI  13390   6700
Aaron   Jones   GR1     MUMBAI  23280   13490
John    Patil   GR2     MUMBAI  13500   10760
Sneha   Joshi   GR1     DELHI   20660   missing
Gaurav  Singh   GR2     DELHI   13760   13220
Adela   Thomas  GR2     DELHI   13660   6840
Anup    Save    GR2     MUMBAI  11960   7880
```

Data Snapshot

basic_salary data consist salary of each employee with it’s Location & Grade.


Variables
Observations

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

read.csv() Function

One way of reading csv files is through **`read.csv()`.**

```
salary_data <- read.csv("C:/Users/Documents/basic_salary.csv")
```

read.csv() assumes **header = TRUE** and **sep** = “,” by default.

*

First locate your data file, whether it is saved in the default working directory of R or any other location in your system. If it is not stored in default working directory then you will have to give its path for importing it into R. If you copy file path from the folder, ensure it uses forward slash (/). Do not forget to accurately write the file name and extension.

read.table() Function

Importing a .csv file

`read.table()` is almost identical to **`read.csv()`** and differ from it in two aspects: **`sep`**, & **`header`** arguments.

```
salary_data <-read.table("C:/Users/Documents/basic_salary.csv",
header = TRUE, sep = ",")
```

- ❑ **header = TRUE** (logical) indicates that the first row of the file contains the names of the columns. Default is **header = FALSE**
- ❑ **sep = ","** specifies that the data is separated by comma. Without this command, R imports data in a single column.

read.table() Function

Importing a .txt (tab delimited)file

```
salary_data <-read.table("C:/Users/Documents/basic_salary.txt",
header = TRUE, sep = "\t", fill= TRUE)
```

fill= TRUE implicitly adds blank fields, in case the rows have unequal length

- Apart from the functions in the base package, R also has several additional packages to carry out efficient data import like **`readr`**, **`data.table`**, **`readxl`**.

How does R Handle Missing Observations?

- Your data may contain some missing value(s) that need(s) to be handled effectively to reduce bias and to produce correct results.
- There are many ways of handling missing values in R. Missing values in R appears as NA.
- NA is not a string or a numeric value, but an indicator of missing values.
- While importing files using **`read.table()`** and **`read.csv()`** missing values are automatically filled with NA.

* Sometimes data may contain random values which are considered as missing values. It is important to detect those values and overcome them. Therefore, we have a special tutorial for Handling missing values where you will learn to deal with different types of missing data.

Package readr

- The package **readr** has functions for importing files and ensures faster imports, proving useful when the data under study is large.

```
# Install and load readr
```

```
install.packages(“readr”)

library(readr)
```

```
# Importing a .csv file
```

```
salary_data<-read_csv("C:/Users/Documents/basic_salary.csv")
```

```
# Importing a .txt file (white space delimited)
```

```
salary_data<-read_table("C:/Users/Documents/basic_salary.txt")
```

```
# Importing a .txt file (tab delimited)
```

```
salary_data<-read_tsv("C:/Users/Documents/basic_salary.txt")
```

Package data.table

- The package **data.table** provides many functions for data manipulation tasks. Its **`fread()`** function is meant to import data from regular(i.e. every row of your data needs to have the same number of columns) delimited files directly into R and is one of the most efficient means of dealing with very large data.
- It is much faster and convenient than other methods and one of the great things about this function is that all controls, expressed in arguments such as `sep` are automatically detected.

```
# Install and load data.table
# Import file using fread()
```

```r
install.packages("data.table")
library(data.table)
salary_data<-fread("C:/Users/Documents/basic_salary.csv")
```

Package readxl

- readxl is an R package that provides function to read Excel worksheets in both `.xls` and `.xlsx` formats.

```
# Install and load readxl
# Import file using read_excel
```

```r
install.packages("readxl")
library(readxl)
salary_data<-read_excel("C:/Users/Documents/basic_salary.xlsx")
```

Importing Files Interactively

- To manually select the directory and file where your dataset is located use **file.choose()** function .
- **file.choose()** can be used as an independent function or can be put inside a function with or without other arguments.

```
salary_data <- read.csv(file.choose())
```

This opens a dialog box that allows you to choose the file interactively.

Importing SAS , STATA , SPSS using Package foreign

- The package **foreign** is used to import data from SAS, STATA and SPSS.

```
# Install and load package foreign
```

```
install.packages("foreign")
library(foreign)
```

```
# For SAS
# Save SAS dataset in transport format. Requires SAS on your system.
```

```
read.xport(“dataset.xpt”)
```

```
# For SPSS
```

```
read.spss(“dataset.sav”,use.value.labels=TRUE)
```

```
# For Stata binary
```

```
read.dta(“dataset.dta”)
```

* **`read.dta()`** function reads a file in Stata version 5–12 binary format into a data frame

Importing MySQL Database Package RMySQL

- To work with MySQL database in R, use package **RMySQL**:

```
install.packages("RMySQL")

library(RMySQL)
```

```
# Create a database connection object.
```

```
mydb <- dbConnect(MySQL(), user='user', password='password',
dbname='database_name', host='host')
```

```
# Save a results set object to retrieve data from the database
```

```
rs = dbSendQuery(mydb, "select * from some_table")
```

Queries can be run using the **dbSendQuery().**

```
# Access the result in R
```

```
data = fetch(rs, n=-1)
```

To access the results in R, **fetch()** is used as it saves the results of the query as a data frame object.
n= specifies no. of records to be retrieved.
n=-1 retrieves all pending records.

Importing PostgreSQL Database Using Package RPostgreSQL

- To work with PostgreSQL database in R, use package **RPostgreSQL**:

```
install.packages("RPostgreSQL")

library(RPostgreSQL)
```

```
# Create a database connection object.
drv <- dbDriver("PostgreSQL")
```

This command establishes connection to PoststgreSQL.

```
con <- dbConnect(drv, host='host', port=‘port',dbname='database_name',
user='user', password='password')
```

```
# Obtain a table into R data frame

myTable <- dbReadTable(con, "tablename")
```

Package RODBC

- There are several packages in R which allow the user to connect to an Oracle database. The most commonly used packages are: RODBC, RJDBC and ROracle.

RODBC – implements ODBC database connectivity.

```
install.packages("RODBC")
library(RODBC)
```

```
# Create a database connection object.
```

```
con <-odbcConnect("data", uid="user", pwd="password")
```

```
# Query the database and put the results into a data frame
```

```
mydata <- sqlQuery(con, "SELECT * FROM TABLENAME.DATATABLE")
```

sqlQuery(channel,query,errors)

- ❑ **channel =** database connection object
- ❑ **errors =** logical : if TRUE halt's and display a character vector of error message(s), else return **-1**. Default is FALSE

Importing Oracle Database Using Package RODBC, RJDBC and ROracle

Additionally, RJDBC package is based on the database interface (DBI) uses JDBC as the back-end connection to the database.

ROracle is an open source R package supporting a DBI-compliant Oracle driver based on the high performance OCI library. It requires Oracle Instant Client or Oracle Database Client to be installed on the client machine.

Exporting CSV, TXT and XLSX Files

Sometimes you may want to export data saved as object from R workspace to different file formats. Methods for Exporting R objects into CSV, TXT and XLSX formats are given below:

```
# To a CSV File
write.csv(mydata, file = "MyData.csv")
```

mydata is object name which is saved in csv format with filename **MyData** in default working directory.

```
# To a Tab Delimited Text File
write.table(mydata, "c:/mydata.txt", sep="\t")
```

mydata object will be saved as tab delimited txt file as we have specified the **sep= "\t"**. It will be saved in C drive of the system.

```
# To a Excel Spreadsheet
install.packages("xlsx")
library(xlsx)
write.xlsx(mydata, "c:/mydata.xlsx")
```

package **xlsx** is used to import **.xlsx** files

Exporting SPSS, SAS and STATA Files

Methods for Exporting R objects into SPSS, SAS and STATA formats using package foreign:

```
install.packages("foreign")

library(foreign)
```

write.foreign(df, datafile, codefile, package = c("SPSS", "Stata", "SAS"), ...)

```
# To SPSS
write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sps",package="SPSS")

# To SAS

write.foreign(mydata, "c:/mydata.txt", "c:/mydata.sas", package="SAS")


# To STATA
write.dta(mydata, "c:/mydata.dta")
```

Quick Recap

In this session, how to import and export data using different functions and packages.

Here is a quick recap:

Import Files Using Base Functions	• **`read.table()`** is used for importing csv and txt files • **`read.csv()`** is used specifically for importing csv files and is more efficient than **`read.table()`** • **`file.choose()`** is used to manually select the file of any format from wherever it is located. This function can be put inside another function.
Handle Missing Values	• Missing Values are filled with NA while importing files using **`read.csv()`** and **`read.table()`**.
Import Files Using Packages	• **`readr`**, **`data.table`** and **`readxl`** provides functions importing data in different file formats. These packages are very helpful in case of large data.
Export Files Using Base Functions and Packages	• **`write.csv()`** exports csv files • **`write .table()`** exports txt files • package **`xlsx`** provides function **`write.xlsx()`** to export excel files

Quick Recap

In this session, we learnt the ways of importing SAS, SPSS, STATA files and MySQL, PostgreSQL and Oracle database Here is a quick recap:

Importing SAS, SPSS and STATA files	• Packages: **`foreign`** and **`Hmisc`**. • For SPSS and SAS, **`Hmisc`** is recommended for ease and functionality
Importing MySQL database	• Package: **`RMySQL`**
Importing PostgreSQL database	• Package: **`RPostgreSQL`**
Importing Oracle database	• Packages: **`RODBS`**, **`RJDBC`**, **`ROracle`**
Exporting SAS, SPSS and STATA files	• Package: **`foreign`**