COMP828 Additional Plot and Working with Data

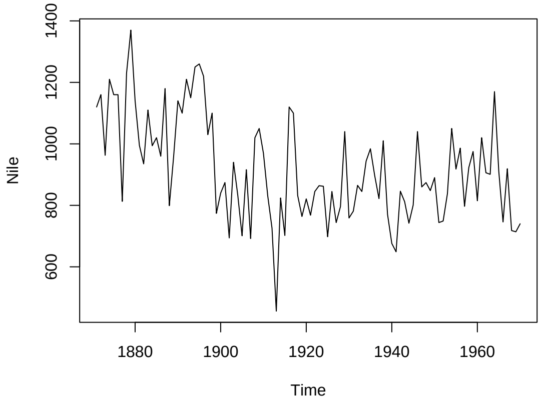
Nuttanan Wichitaksorn

Department of Mathematical Sciences
Auckland University of Technology

Plots

Line plot is useful to visualize and summarize the key characteristics of the time series data.

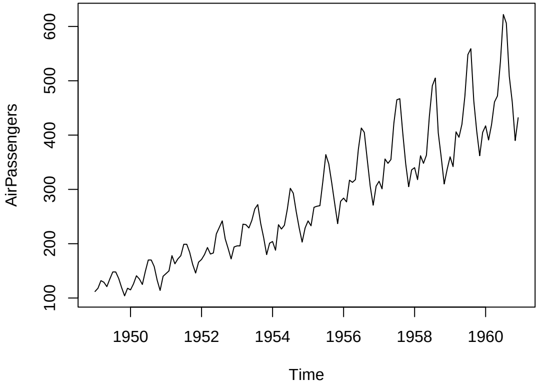
```
plot(Nile)
```



Plots (cont.)

Line plot

```r
plot(AirPassengers)
```



Lists[1]

- `list` is an object consisting of an ordered collection of objects known as its *components*.
- A list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, and a function.

```
Lst <- list(name="Fred", wife="Mary", no.children=3,
child.ages=c(4,7,9))
```

- Components are always *numbered* and may always be referred to.
- If `Lst` is the name of a list with four components, these may be individually referred to as `Lst[[1]]`, `Lst[[2]]`, `Lst[[3]]`, and `Lst[[4]]`.
- If `Lst[[4]]` is a vector subscripted array then `Lst[[4]][1]` is its first entry.

[1]See Section 6 in https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

Lists (cont.)

- If `Lst` is a list, then the function `length(Lst)` gives the number of (top level) components it has.
- Components of lists may also be *named*, and in this case the component may be referred to either by giving the component name as a character string in place of the number in double square brackets, or, more conveniently, by giving an expression of the form for the same thing.

```
name$component_name
```

- Additionally, one can also use the names of the list components in double square brackets, i.e., `Lst[["name"]]` is the same as `Lst$name`.
- ‘`[[...]]`’ is the operator used to select a single element, whereas ‘`[...]`’ is a general subscripting operator.

Lists (cont.)

Constructing and modifying lists

- New lists may be formed from existing objects by the function `list()`.

```
Lst <- list(name_1=object_1, ..., name_m=object_m)
```

- An assignment of the form sets up a list `Lst` of *m* components using *object_1,...,object_m* for the components and giving them names as specified by the argument names.
- If these names are omitted, the components are numbered only.
- Lists can be extended by specifying additional components. For example

```
Lst[5] <- list(matrix=Mat)
```

Lists (cont.)

Concatenating lists

- When the concatenation function `c()` is given list arguments whose components are those of the argument lists joined together in sequence.

```
list.ABC <- c(list.A, list.B, list.C)
```

- Recall that with vector objects as arguments the concatenation function similarly joined together all arguments into a single vector structure.

- In this case, all other attributes, such as `dim` attributes, are discarded.

Data Frames

- A *data frame* is a list with class `data.frame`. However, there are restrictions on lists that may be made into data frames.
- The components must be vectors (numeric, character, or logical), factors, numeric matrices, lists, or other data frames.
- Matrices, lists, and data frames provide as many variables to the new data frame as they have columns, elements, or variables, respectively.
- Vector structures appearing as variables of the data frame must all have the same length, and matrix structures must all have the same number of rows.
- A data frame may, for many purposes, be regarded as a matrix with columns possibly of differing modes and attributes. It may be displayed in matrix form, and its rows and columns extracted using matrix indexing conventions.
- Examples: `mtcars` and `iris` datasets

Data Frames (cont.)

Making data frames

- Objects satisfying the restrictions placed on the columns (components) of a data frame may be used to form one using the function `data.frame`:

```
accountants <- data.frame(home=statef, loot=incomes, shot=incomef)
```

- A list whose components conform to the restrictions of a data frame may be *coerced* into a data frame using the function `as.data.frame()`.
- The simplest way to construct a data frame from scratch is to use the `read.table()` function to read an entire data frame from an external file.
- For more info on working with data frames, see https://www.tutorialspoint.com/r/r_data_frames.htm#

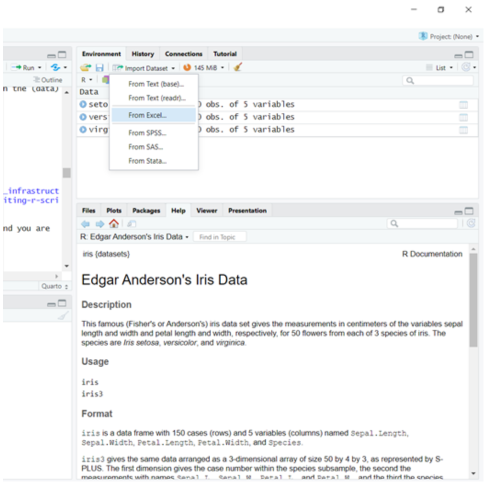
Reading/Importing Data

- One of the most convenient ways to read or import data from a file into `R` is to use `read.table()` function.

- However, for the new `R` users, doing this task through the “Import Dataset” in the (data) “Environment” might easier.

Reading/Importing Data (cont.)

Excel file



```
n the (data)
Data
seto        ) obs. of 5 variables
vers        ) obs. of 5 variables
virg        ) obs. of 5 variables
_infrastruct
iting-r-scri
nd you are
```

R: Edgar Anderson's Iris Data

iris {datasets} R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

```
iris
iris3
```

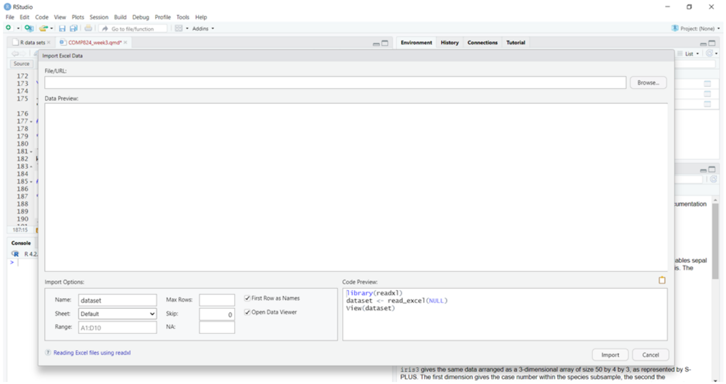
Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the measurements with names `Sepal L.`, `Sepal W.`, `Petal L.`, and `Petal W.`, and the third the species

Reading/Importing Data (cont.)

Excel file



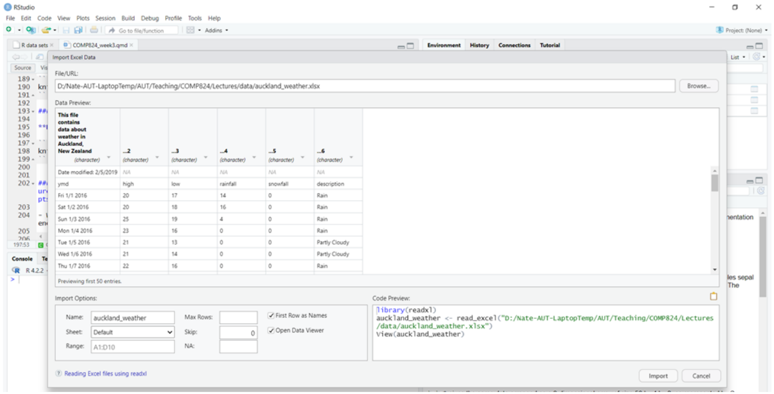
Name: dataset
Sheet: Default
Range: A1:D10
Max Rows:
Skip: 0
NA:
✓ First Row as Names
✓ Open Data Viewer

```
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
```

iris3 gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the

Reading/Importing Data (cont.)

Excel file


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Import Excel Data
Browse...
Import
Cancel

File/URL: D:/Nate-AUT-LaptopTemp/AUT/Teaching/COMP824/Lectures/data/auckland_weather.xlsx

Data Preview:

This file contains data about weather in Auckland, New Zealand (character)	...2 (character)	...3 (character)	...4 (character)	...5 (character)	...6 (character)
Date modified: 2/5/2019	NA	NA	NA	NA	NA
ymd	high	low	rainfall	snowfall	description
Fri 1/1 2016	20	17	14	0	Rain
Sat 1/2 2016	20	18	16	0	Rain
Sun 1/3 2016	25	19	4	0	Rain
Mon 1/4 2016	23	16	0	0	Rain
Tue 1/5 2016	21	13	0	0	Partly Cloudy
Wed 1/6 2016	21	14	0	0	Partly Cloudy
Thu 1/7 2016	22	16	0	0	Rain

Previewing first 50 entries.

Import Options:

Name:	auckland_weather	Max Rows:		✓ First Row as Names
Sheet:	Default	Skip:	0	✓ Open Data Viewer
Range:	A1:D10	NA:		

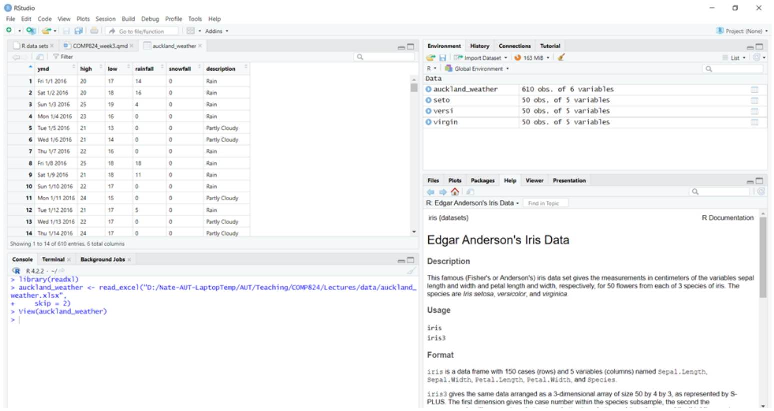
Code Preview:

```
library(readxl)
auckland_weather <- read_excel("D:/Nate-AUT-LaptopTemp/AUT/Teaching/COMP824/Lectures
/data/auckland_weather.xlsx")
View(auckland_weather)
```

Reading Excel files using readxl

Reading/Importing Data (cont.)

Excel file


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
R data sets
COMP824_week3.qmd
auckland_weather
Filter
Environment History Connections Tutorial
Import Dataset
163 MiB
List
Global Environment
Files Plots Packages Help Viewer Presentation
R: Edgar Anderson's Iris Data
Find in Topic
Console Terminal Background Jobs
R 4.2.2 · ~/

	ymd	high	low	rainfall	snowfall	description
1	Fri 1/1 2016	20	17	14	0	Rain
2	Sat 1/2 2016	20	18	16	0	Rain
3	Sun 1/3 2016	25	19	4	0	Rain
4	Mon 1/4 2016	23	16	0	0	Rain
5	Tue 1/5 2016	21	13	0	0	Partly Cloudy
6	Wed 1/6 2016	21	14	0	0	Partly Cloudy
7	Thu 1/7 2016	22	16	0	0	Rain
8	Fri 1/8 2016	25	18	18	0	Rain
9	Sat 1/9 2016	21	18	11	0	Rain
10	Sun 1/10 2016	22	17	0	0	Rain
11	Mon 1/11 2016	24	15	0	0	Partly Cloudy
12	Tue 1/12 2016	21	17	5	0	Rain
13	Wed 1/13 2016	22	17	0	0	Partly Cloudy
14	Thu 1/14 2016	24	17	0	0	Partly Cloudy

Showing 1 to 14 of 610 entries, 6 total columns

```
> library(readxl)
> auckland_weather <- read_excel("D:/Nate-AUT-LaptopTemp/AUT/Teaching/COMP824/Lectures/data/auckland_
weather.xlsx",
+     skip = 2)
> View(auckland_weather)
>
```

```
R ▾   Global Environment ▾
Data
auckland_weather   610 obs. of 6 variables
seto               50 obs. of 5 variables
versi              50 obs. of 5 variables
virgin             50 obs. of 5 variables
```

iris (datasets) R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

```
iris
iris3
```

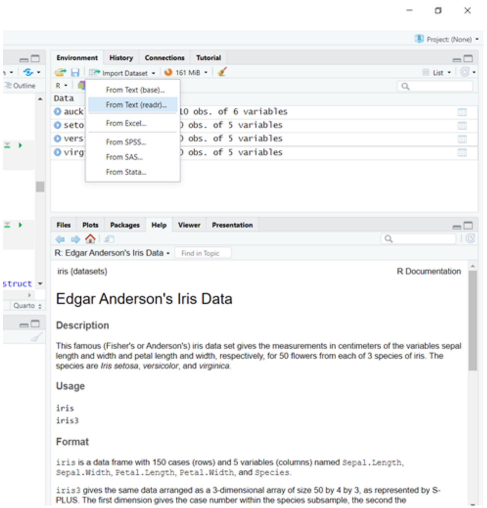
Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the

Reading/Importing Data (cont.)

CSV file



```
Data
auck   10 obs. of 6 variables
seto    obs. of 5 variables
vers    obs. of 5 variables
virg    obs. of 5 variables
```

R: Edgar Anderson's Iris Data

iris {datasets}   R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

```
iris
iris3
```

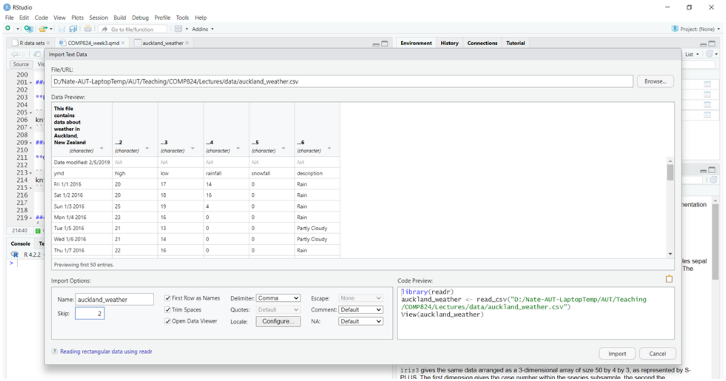
Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the

Reading/Importing Data (cont.)

CSV file



File/URL: D:/Nate-AUT-LaptopTemp/AUT/Teaching/COMP824/Lectures/data/auckland_weather.csv

Data Preview:

This file contains data about weather in Auckland, New Zealand (character)	...2 (character)	...3 (character)	...4 (character)	...5 (character)	...6 (character)
Date modified: 2/5/2019	NA	NA	NA	NA	NA
ymd	high	low	rainfall	snowfall	description
Fri 1/1 2016	20	17	14	0	Rain
Sat 1/2 2016	20	18	16	0	Rain
Sun 1/3 2016	25	19	4	0	Rain
Mon 1/4 2016	23	16	0	0	Rain
Tue 1/5 2016	21	13	0	0	Partly Cloudy
Wed 1/6 2016	21	14	0	0	Partly Cloudy
Thu 1/7 2016	22	16	0	0	Rain

Previewing first 50 entries.

Import Options:

Name: auckland_weather
Skip: 2
✓ First Row as Names
✓ Trim Spaces
✓ Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default

Code Preview:

```r
library(readr)
auckland_weather <- read_csv("D:/Nate-AUT-LaptopTemp/AUT/Teaching
/COMP824/Lectures/data/auckland_weather.csv")
View(auckland_weather)
```

iris3 gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the

Reading/Importing Data (cont.)

- Either importing data from an Excel or CSV file, can you work with the dataset?
- Try this:

```r
summary(auckland_weather$rainfall)
```

- What we need to do is to convert the “character” variables to the “numeric” ones as:

```r
auckland_weather <- as.data.frame(sapply(auckland_weather, as.numeric))
```

- However, we need to be careful that not all variables are numeric.