

# File System Tree

STAT 133

Gaston Sanchez

Department of Statistics, UC–Berkeley

`gastonsanchez.com`

`github.com/gastonstat`

Course web: [gastonsanchez.com/teaching/stat133](https://gastonsanchez.com/teaching/stat133)

# Managing Files

# File Management

File management is crucial for any data analysis project  
Common types of files:

- ▶ Data files
- ▶ Code files (e.g. functions)
- ▶ Analysis files
- ▶ Presentation and Report files

Also, many tools such as R, LaTeX, markdown, etc require knowing where files are located in your computer

# File Management

Good file management allows you to:

- ▶ find things more easily
- ▶ make changes more easily
- ▶ benefit from work you've already done
- ▶ be understood by others
- ▶ collaborate with others

# Why File Management?

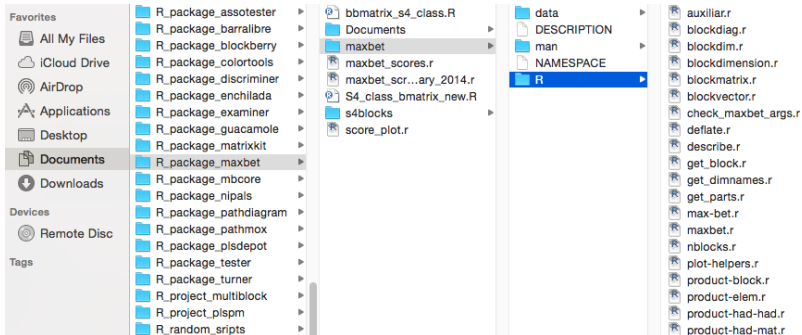
## Common tasks

- ▶ Access and organize your files
- ▶ Control creation of files
- ▶ Control deletion of files
- ▶ Control relocation of files
- ▶ Control modification of files

# Organization of Files

How does our computer organize files?

# Files and Directories

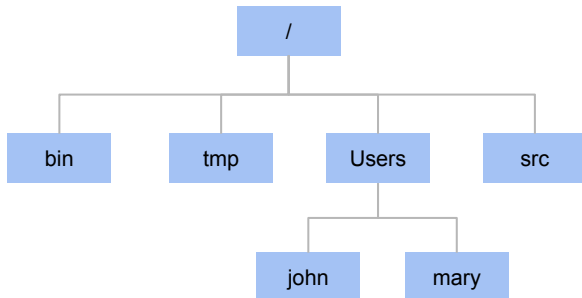


# Organization of Files

- ▶ The computer organizes files within directories
- ▶ Directories and files follow a tree structure
- ▶ A tree structure is a hierarchical structure
- ▶ Hierarchical means that directories are located inside other directories



# Files and Directories



# Filesystem

- ▶ The nested hierarchy of folders and files on your computer is called the **filesystem**
- ▶ The filesystem follows a tree-like structure

# Directories

# Files and Directories

There are two special directories in UNIX-like OS:

- ▶ The top level directory, named `"/`, called the **root directory**
- ▶ The **home directory**, named `~`, which contains all your files

# Root Directories

- ▶ A root directory is the first level in a disk (such as a hard drive)
- ▶ It is the root out of which the file tree “grows”
- ▶ All other directories are subdirectories of the root directory
- ▶ On Unix-like system, including Macs, the root directory is denoted by a forward slash: /

# Root Directory

- ▶ The root directory is the most inclusive folder on the system
- ▶ The root directory serves as the container of all other files and folders
- ▶ A Unix-based system (e.g. OS X) has a single root directory
- ▶ Windows users usually have multiple roots (C: , D: , etc)

# Root Directories on Windows

- ▶ On Windows computers you can have multiple root directories (one for each storage device)
- ▶ On Windows, root directories are given a drive letter assignment
- ▶ On Windows, the most common root directory is `C:\` (denoting the C partition of the hard drive)

# Home Directory

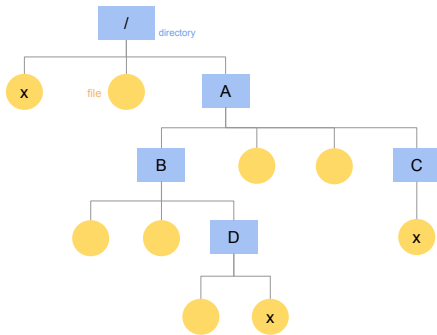
- ▶ User's personal files are found in the `/Users` directory
- ▶ e.g. mine is `/Users/Gaston`
- ▶ A user directory is the **home** directory



# Subdirectories and Parent Directories

- ▶ We store files in subdirectories of the root directory
- ▶ Inside these subdirectories may be further subdirectories and so on
- ▶ A directory containing other directories is referred to as the **parent directory**
- ▶ Directories inside other directories are referred to as **child directories**

# Directories and Subdirectories

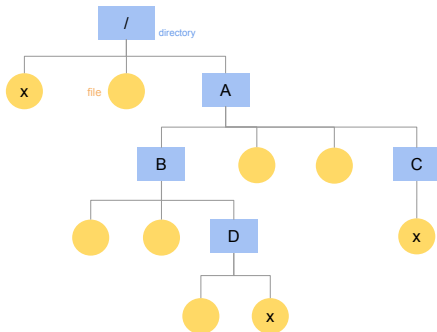


- ▶ A is a child directory of the root directory
- ▶ A is the parent directory of B and C

# Working Directory

- ▶ Another special type of directory is the so-called **working directory**
- ▶ The working directory is the current directory where you perform any task
- ▶ If you go to your Desktop, then the Desktop is your current directory
- ▶ When you use R, the working directory is the directory where the program automatically looks for files

# Working Directory



If you are standing in **B**, then this is your working directory

# Paths

# Path

- ▶ Each file and directory has a unique name in the filesystem
- ▶ Such unique name is called a **path**
- ▶ In UNIX-like systems the root directory's path is a slash /

# Filesystem

- ▶ The written description of where something is located in the filesystem is called the **path**
- ▶ The path is a list of directory names separated by slashes
- ▶ If the path is for a file, then the last element of the path is the file's name
- ▶ e.g. `/Users/Gaston/Documents/data.txt`
- ▶ A path can be **absolute** or **relative**

# Paths

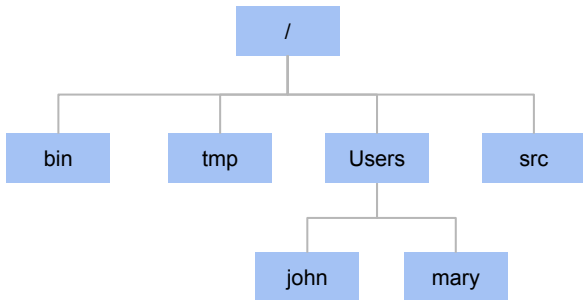
- ▶ An **absolute path** is a complete and unambiguous description of where something is in relation to the root
- ▶ If a path begins with a slash, then it's called an absolute path
- ▶ A **relative** describes where a folder or file is in relation to another folder
- ▶ If a path does not begin with a slash, then it is a relative path



# Paths

- ▶ There are two special relative paths: `.` and `..`
- ▶ The single period `.` refers to the current directory
- ▶ The two periods means the parent directory, one level above
- ▶ For instance, if the current directory is `/Users/XYZ/abc`, then `.` refers to this directory, and `..` refers to `/Users/XYZ`

# Files and Directories

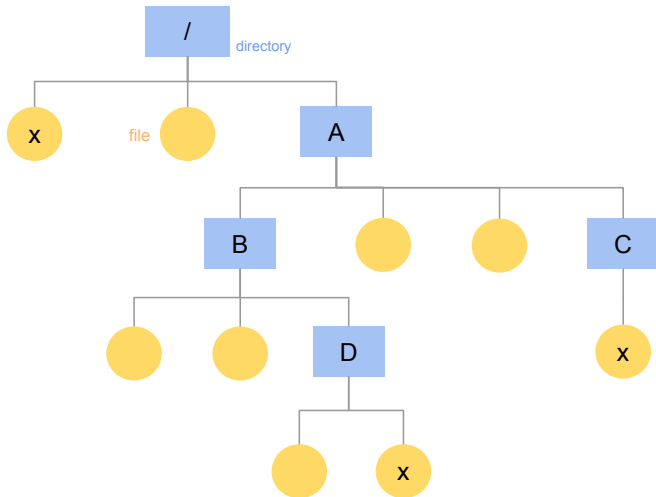


# Path Names

## Full path name

- ▶ path from the top level directory, /, to the file or directory of interest
- ▶ For mary the full pathname is: `/Users/mary`

# Files and Directories

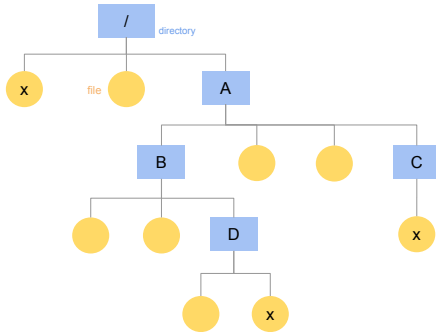


# Path Names

## Relative path name

- ▶ path from the current directory to the file or directory of interest
- ▶ Relative path to D from A: B/D
- ▶ Equivalently: ./B/D (. refers to current directory)

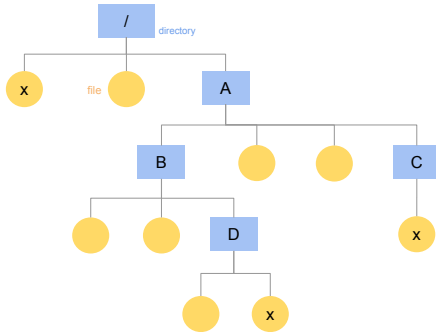
# Relative Path Names



Relative path to D from A: **B/D**

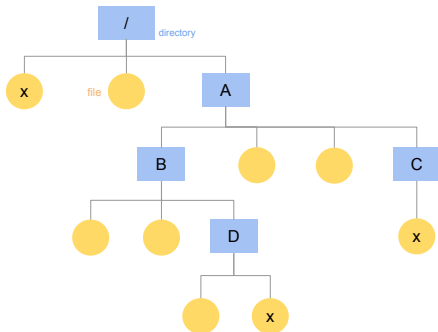
Equivalently: **./B/D** ( . refers to current directory)

# Relative Path Names



Relative path to D from C: `../B/D`  
(`..` refers to parent directory)

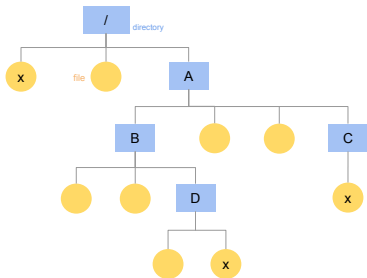
# Relative Path Names



Relative path to x at the top from within C?



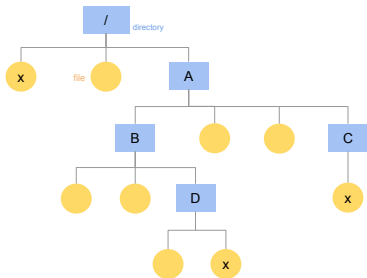
# Relative Path Names



Relative path to x at the top from within C?

- a) ../A/x
- b) ../../x
- c) ../x
- d) /x

# Relative Path Names



Relative path to x in D from within C?

- a) ../D/x
- b) ../B/D/x
- c) ../../A/B/D/x
- d) /A/B/D/x

# Filesystem

- ▶ Root Directory
- ▶ Home Directory
- ▶ Working Directory
- ▶ Directory Tree
- ▶ Absolute path names
- ▶ Relative path names

# File Manipulation Commands in R

# R File Management Functions

function	description
<code>getwd()</code>	shows the current working directory
<code>list.files()</code>	see all the files and subdirectories in the current working directory
<code>setwd()</code>	sets the current working directory
<code>dir.create()</code>	create a new directory
<code>file.create()</code>	create a new blank file
<code>cat()</code>	create a new file and put text into it, or append text to an existing file
<code>file.append()</code>	attempts to append two files
<code>unlink()</code>	delete files and directories
<code>file.rename()</code>	rename a file or move a file
<code>file.copy()</code>	copy a file to another directory
<code>file.exists()</code>	check whether a file exists

# getwd()

`getwd()` allows you to find your current working directory

```
# working directory (for these slides)  
getwd()
```

```
## [1] "/Users/Gaston/Documents/stat133/lectures/22-file-system"
```

# list.files()

`list.files()` displays the files and subdirectories of the working directory

```
# files and directories in my working directory  
wd <- list.files()  
head(wd)
```

```
## [1] "22-file-system.nav" "22-file-system.pdf" "22-file-system."  
## [4] "22-file-system.snm" "22-file-system.tex" "22-file-system."
```

# list.files()

You can also specify a different path

```
# contents in the stat133 github repo
```

```
list.files(path = '~/Documents/stat133/stat133')
```

```
## [1] "datasets"           "HW1"                 "HW2"
## [4] "HW3"                 "HW4"                 "HW5"
## [7] "HW6"                 "HW7"                 "HW8"
## [10] "lab01"               "lab02"               "lab03"
## [13] "lab04"               "lab05"               "lab06"
## [16] "lab07"               "lab08"               "lab09"
## [19] "lab10"               "lecture-jul-07.Rmd"  "lecture-regex.Rmd"
## [22] "more-regex.Rmd"      "README.md"
```



# setwd()

`setwd()` allows you to set a working directory (this is where R will look for files and subdirectories)

```
# setting a working directory  
setwd('~/Documents/Consulting')
```

## setwd()

Assuming that there is a subdirectory Data inside Consulting, we could read a file like so:

```
# setting a working directory  
df <- read.csv('Data/dataset.csv')
```

# dir.create()

`dir.create()` allows you to create a new directory

```
# new directory  
dir.create('/Users/john/Documents/stat133/HW6')  
  
# new directory (Windows)  
dir.create('C:\\Documents\\stat133\\HW6')
```

# file.create()

`file.create()` allows you to create a new blank file

```
# new file 'functions.R'  
file.create('/Users/john/Documents/stat133/HW6/functions.R')  
  
# new file (on Windows)  
file.create('C:\\Documents\\stat133\\HW6\\functions.R')
```

# cat()

cat() can be used to create a new file and put text into it

```
# new file 'functions.R'  
cat('# Homework 6',  
    '\n# Your name',  
    '\n# Description',  
    file = '/Users/john/Documents/stat133/HW6/myscript.R')
```

# file.append()

file.append() attempts to append two files

```
# append two files  
file.append('data1.csv', 'data2.csv')
```

# unlink()

`unlink()` deletes files and directories (warning: deletion is permanently)

```
# delete a file  
unlink('/Users/john/Documents/stat133/HW6/myscript.R')
```

# file.rename()

file.rename() renames a file

```
# rename a file  
file.rename(from = 'script.R', to = 'analysis.R')
```



## file.rename()

`file.rename()` can also be used to fully move a file from one directory to another

```
# move a file  
file.rename(from = 'old-project/analysis.R',  
            to = 'new-project/analysis.R')
```

# file.copy()

`file.copy()` copies a file to another directory

```
# move a file  
file.copy(from = 'old-project/analysis.R',  
          to = 'new-project/analysis.R')
```

# file.exists()

`file.exists()` checks whether a

```
# checking existence of a file  
file.exists('homework05_instructions.pdf')
```

## Related functions

- ▶ `file.info()`
- ▶ `file.mode()`
- ▶ `file.mtime()`
- ▶ `file.size()`
- ▶ `file.access()`
- ▶ `system()`