**Welcome to Software Carpentry Etherpad for the workshop at UCLA!**

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of the Software Carpentry and Data Carpentry community; this is not for general purpose use (for that, try etherpad.wikimedia.org).

**Users are expected to follow our code of conduct: http://software-carpentry.org/conduct.html**



**Workshop Links**

This etherpad: http://pad.software-carpentry.org/ucla-2017
Website: https://ucla-data-archive.github.io/2017-10-16-ucla/
Setup: https://ucla-data-archive.github.io/2017-10-16-ucla/#setup
Pre-workshop suvey: https://www.surveymonkey.com/r/swc_pre_workshop_v1?workshop_id=2017-10-16-ucla
Twitter handle for workshop: #uclaswc2017

**Sign-in**

Name/Affiliation (Dept. or Unit)/Role/email/twitter handle

Tim Dennis/UCLA Data Archive/Director/timdennis@ucla.edu/jt14den
Dawn Childress/UCLA Digital Library/@kirschbombe
Mike Peters/UCLA Geography/mike.peters@ucla.edu
Deidre Whitmore/ UCLA Cotsen Institute of Archaeology @whitmost
Cesar Sul / USC HPC/ Recearch Computing Facilitator/ csul@usc.edu
Genevieve Aquino / UPLB Phil. Genome Center/ gbaquino@up.edu.ph
Niqui O'Neill / UCLA Digital Library
Camille Sultana/UCSD, Chemistry/PostDoc/csultana@ucsd.edu
Meredith Conroy/CSUSB Political Science/professor/@sidney_b
Jessica Mentesoglu, DIITOS, UCLA Library, jmenteosglu@library.ucla.edu
Isaac Williams/UCLA Digital Library
Sharon Shafer/UCLA Digital Information & Information Technology

**The Unix Shell - Links**

- Setup - download the .zip: http://dawnchildress.com/shell-exercises/setup/

Exercises:

- Navigating Files and Directories Exercises: http://dawnchildress.com/shell-exercises/02-filedir/

- Working with Files and Directories Exercises: http://dawnchildress.com/shell-exercises/03-create/
- Pipes and Filters Exercises: http://dawnchildress.com/shell-exercises/04-pipefilter/
- Loops Exercises: http://dawnchildress.com/shell-exercises/05-loop/

**Helpful resources to investigate later**

- TLDR: https://tldr.sh/
- Explain Shell: https://explainshell.com/

TAB completion is your best friend! If you begin to type a path then press TAB it will autocomplete
Up arrow to get the last command (and on)
history to show all of the commands with numbers (allows you to redo a command by calling that number - !605)
clear and enter to clear off your screen

pwd - print working directory
whoami - tells you your user account
ls - lists files and directories in your current working dir
ls -F -- -F is called a flag and makes unix distinguish b/t files and directories adding a / to directories
cd - changes directories, so cd foo will change directories into foo/
cd .. - goes up one directory to parent dir
see the tree structure of unix file system: https://swcarpentry.github.io/shell-novice/fig/home-directories.svg
cd - will take you to the previous directory

cd ~ will take you to your "home" directory
Or just cd will take you to home directory
man pages are short for "manual", to get info about a command: man somecommand or man ls

clear to clear the screen
mkdir <directory name> make a new directory
cd thesis - go into the dir you created
nano draft.txt - use nano editor to create a file
rm deletes forever (doesn't store in the trash folder)
rm -i for interactive (asks if you want to remove) <- respond with 'y' or yes, enter does not work. Need to explicitly agree to deleting
mv oldfilename.txt newfilename.txt to rename a file. Be sure to check if there is alrea
dy a file with that name or not, this will overwrite if it exists alreadyra
mv somedirectory/filename.txt . the dot is movint to current directory (rather than an explicit path)
* is a wildcard (any characters, zero or more, can be used to find any filename with a specific extension by using *.ext)
text*[AB] will return any string with "text", zero or more characters, ending with either A or B
wc word count

wc > newfile.txt will print results out to a new text file

cp *.dat original-*.dat doesn't work because it is expecting the second term to be the target, instead you need to loop through each:

```
for filename in basilisk.dat unicorn.dat
do
head -n 3 $filename
done
```

In this for loop: for each filename (a variable which will take the place of each of the filenames, can be anything as long as what you define is what you call later, helpful to use descriptive names) in each file print the first three lines
$variable to call the variable

echo prints, handy way to check your work

```
for filename in *.dat
do
head -n 100 $filename | tail -n 20
done
```

note: put $filename in " quotes if there are blanks in the filename

```
for $filename in *.dat; do echo cp $filename original-$filename; done
```

make a backup/original copy of files

```
for datafile in NENE*[AB}
do
echo $datafile       actual copy line: echo cp $datafile stats-$datafile
done
```

echo confirms the correct files are being pulled

Shell Scripts
bash script file extension: .sh

run script either:  bash script.sh  or
chmod +x  script.sh
./script.sh

$1 first parameter, $2 second parameter, etc
# to "comment" or describe what is happening, how to use, any information that might be useful to you or others in the future when running the script

## Afternoon - R for Reproducible Scientific Analysis

1. Intro to R & RStudio - 55m
2. Project mgmt in Rstudio - 15m
3. Data Structures 55m
4. GGPLOT2 - 60

**R Resources**

- Task Views on https://cran.r-project.org/
- rOpenSci (especially published packages) https://ropensci.org/
- Meetups: https://www.meetup.com/rladies-la/

**R for Reproducible Scientific Analysis Links**

- Introduction to R and RStudio Lesson and Exercises: https://swcarpentry.github.io/r-novice-gapminder/01-rstudio-intro/
- Project Management with RStudio Lesson and Exercises: https://swcarpentry.github.io/r-novice-gapminder/02-project-intro/
- Data Structures Lesson and Exercises: https://swcarpentry.github.io/r-novice-gapminder/04-data-structures-part1/
- Creating Publication Quality Graphics Lesson and Exercises: https://swcarpentry.github.io/r-novice-gapminder/08-plot-ggplot2/

Shortcuts can be found under Tools/Keyboard Shortcuts Help
Run will run everything up to the line your cursor is on (shortcut: Ctrl + enter)
Will follow order of operations, display in scientific notation, to prevent scientific notation: You can do:
format(functionResult, scientific=FALSE);
or:
as.integer(functionResult);

1 == 1 is asking if 1 equals 1 (results in TRUE/FALSE)
1 != 2 is asking if 1 does not equal 2
1 < 2 less than
1 <= 1 less than or equal to
Caution: 0.1+0.05 == 0.15 results in FALSE - unable to evaluate due to different levels of precision

- Use all.equal function: all.equal(0.1+0.05, 0.15)

# is used to "comment" or insert notes that will not be ran but provide information to future users, including yourself; use liberally!

<- or = Assignment operators (set equal to); conventially the R community uses <- more often; can use either but keep it consistent

***Tip***: you can put parentheses around to get it to print

Variable names can contain letters, numbers, underscores and periods. They cannot start with a number nor underscore nor contain spaces at all (can start with a period but will be set as

hidden). Different people use different conventions for long variable names, these include

- periods.between.words
- underscores_between_words
- camelCaseToSeparateWords

R is *vectorized*, meaning that variables and functions can have vectors as values.
1:5 will produce 1 2 3 4 5
Can be incorporated: 2^(1:5) produces 2 4 8 16 32
You can think about applying this by running a function across columns in spreadsheets
Dataframes are composed of vectors, vectors must all be the same data type (hence the column = vector)

ls will list all of the variables and functions stored in the global environment (aka your working R session)
ls(all.names=TRUE) to show hidden variables also
rm will remove objects

**Packages!**
"bags of functions" -Tim Dennis

to see what has been installed: installed.packages()

install.packages("packagename1", "packagename2")
update.packages("packagename1")
remove.packages("packagename1")

Creating a project in RStudio
We're going to create a new project in RStudio:

1. Click the "File" menu button, then "New Project".
2. Click "New Directory".
3. Click "Empty Project".
4. Type in the name of the directory to store your project, e.g. "swc_ucla".
5. If available, select the checkbox for "Create a git repository." (We'll come back to this tomorrow)
6. Click the "Create Project" button.

Challenge 1 - downloading data

- Begin by downloading the little dataset about cats from here https://raw.githubusercontent.com/ucla-data-archive/ucla-swc17/master/data/feline-data.csv
- Right click on anywhere on the data and "Save as"to you data/ folder you created in last episode.
-

TAB completion works in RStudio also!
Case sensitive i.e. View(cats) not view(cats)
Explore data by calling table then column with $: cats$weight

Check column data type with typof(cats$weight)

Factors have levels (categories) but they are stored as integers; like an index (Edson)

cats <- read.csv(file="https://raw.githubusercontent.com/ucla-data-archive/ucla-swc17/master/data/feline-data_v2.csv")

- 

***Tip***: ? before a function will open the help interface i.e. ?str

**Data Types**
R is particular about data types, must be the same type in a column
Atomic vector, default logical (TRUE/FALSE)

- double
- integer
- complex
- logical
- character

concatenate with the c function

- concat_vector <- c(2,6,3)
- 

Type coercion: forcing a data type when mixed types
coercion rules go: logical -> integer -> numeric -> complex
common to convert types in order to categorize (label) or plot data

can overwrite a column by:
object$vector <- as.type(object$vector)

Reproducible research note: your code should do all of the cleaning, transforming, analyzing etc so you can rerun the code and get the same results; not recommended to rely on session in RStudio

Factors usually look like character data, but are typically used to represent categorical information.

**Publication-Quality Graphics**

*gapminder <- read.csv("https://goo.gl/BtBnPg", header = T)*

ggplot - main plotting function

https://swcarpentry.github.io/r-novice-gapminder/08-plot-ggplot2/#challenge-1

***Tip:*** sometimes you have to restart your session  (Session/Restart R)

When adding multiple plots it is important to note that they are layered: geom_line() + geom_point() places the points over the lines
Aesthetics can be global or applied to specific plots

Found this cool chart summarizing ggplot functionality
https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

######################################################################
## Day 2: Version Control with Git

**Version Control with Git Lessons and Exercises**

- Automated Version Control: https://swcarpentry.github.io/git-novice/01-basics/
- Setting up Git: https://swcarpentry.github.io/git-novice/02-setup/
- Creating a Repository: https://swcarpentry.github.io/git-novice/03-create/
- Tracking Changes: https://swcarpentry.github.io/git-novice/04-changes/
- Exploring History: https://swcarpentry.github.io/git-novice/05-history/
- Ignoring Things: https://swcarpentry.github.io/git-novice/06-ignore/
- Remotes in Git: https://swcarpentry.github.io/git-novice/07-github/
- Collaborating: https://swcarpentry.github.io/git-novice/08-collab/
- Conflicts: https://swcarpentry.github.io/git-novice/09-conflict/
- Git with RStudio: https://swcarpentry.github.io/git-novice/14-supplemental-rstudio/

**Helpful Git Resources**

- official Git documentation: https://git-scm.com/doc
- https://ndpsoftware.com/git-cheatsheet.html

Version control tracks changes to files in repositories
"unlimited undo"

To set up Babun/Terminal with your github username, email:
```
git config --global user.name "yourusername"
git config --global user.email "yourgithubemail"
```

To set your favorite text editor refer to the list on this page: https://swcarpentry.github.io/git-novice/02-setup/

To check your settings:
```
git config --listpwpwd
```

To turn a directory (folder) into a git repository, navigate to the location:
```
git init
```

To check, to see what has gone on:
```
git status
```

*Tip:* don't put a repository inside of a repository!

To remove a git repository delete the .git folder

To add a file:
    git add filename.ext

Once added you need to commit WITH a message describing the change:
    git commit -m "explain what you did"

To see a record of all commits:
    git log
To get out of the git log: q

To see the differences between the current state of the file and the most recently saved version:

- git diff

If you have made changes or added a file and already ran git add, git diff won't show the changes unless you include the staging area:
    git diff --staged

-

Summary of the "areas"
Workspace = untracked changes
Staging area = added files (but not committed to the repo yet)
Repository = committed, official version

To skip the staging area you can add and commit at the same time with the -a flag

- git commit -a -m "edited file..."

In a series of commits (referred to as a chain) the most recent is called the HEAD
The second to last would be HEAD~1

Git doesn't track folder- ONLY the files within them
if you want to track a folder you can create a dummy file within that folder (empty txt file)
However if you have multiple changes to files within a directory you can add them all at once by:
    git add directory

https://ndpsoftware.com/git-cheatsheet.html

Reminder: you can print out file contents in the terminal/Babun window with cat filename.ext

To restore (when files are in the staged area but not the repo yet)

- git checkout eaf3cb mars.txt
- use the start of the commit identifier that corresponds to which version you want to use and which file
-

Note: It's important to remember that we must use the commit number that identifies the state of the repository ***before*** the change we're trying to undo. A common mistake is to use the number of the commit in which we made the change we're trying to get rid of.

To create file without using nano

- touch filename1.ext filename2.ext folder/filename3.ext
- this will create 3 files and 1 folder
-

To ignore (not track) files

- create a .gitignore file and enter the filenames that you don't want tracked (can use * as a wildcard) into the .gitignore file
-

the order within the .gitignore file matters:
entries below other entries will override the previous entries

-

##GitHub

Students can get a free account with the ability to create private repositories: https://education.github.com/pack

GIt with RStudio: https://swcarpentry.github.io/git-novice/14-supplemental-rstudio/

***Tip:*** When collaborating with others Pull before you Push! Pull before you start working, pull before you push. Commit often - makes it easier when attempting to resolve conflicts later!

####################################
##Day 2: R for Reproducible Scientific Analysis

**R for Reproducible Scientific Analysis Lessons and Exercises:**

- Subsetting data: https://swcarpentry.github.io/r-novice-gapminder/06-data-subsetting/
- Functions: https://swcarpentry.github.io/r-novice-gapminder/10-functions/
- Dataframe Manipulation with dplyr: https://swcarpentry.github.io/r-novice-gapminder/13-dplyr/

- Dataframe Manipulation with tidyr: https://swcarpentry.github.io/r-novice-gapminder/14-tidyr/

##In RStudio  get the data for this afternoon from GitHub
1. File>New Project choose Version Control and then Git
2. Fill out the form:

- Repository URL: https://github.com/jt14den/ucla-swc-d2.git
- It will fill in directory name with the repo name by default (leave it)
- Create project as a subdirectory of: ~/Desktop
- after you create project you'll see a data subdirectory, a README and license file.
-

**Helpful Resources:**
https://swcarpentry.github.io/r-novice-gapminder/06-data-subsetting/##challenge-1

Subsetting data

- six different ways to subset any kind of object, and three different subsetting operators for the different data structures

Corresponding index (the element's number, starting with 1)  x[1]
Multiple elements at once  x[c(1, 2)]
Slices of elements  x[1:3]
Skip element (return all except)  x[-1]

- Same ability to do multiple and slices with the -

Order of operations applies per usual
Can also use name  x["a"]

Can similarly subset with other logical operations  x[TRUE] and x[x>3] and x[names(x) ~= "a"]

**%in%**  Remember you can search for help on operators by wrapping them in quotes: help ("%in%") or ?"%in%".

https://swcarpentry.github.io/r-novice-gapminder/06-data-subsetting/#challenge-2

```
seAsia <- c("Myanmar","Thailand","Cambodia","Vietnam","Laos")
## read in the gapminder data that we downloaded in episode 2
gapminder <- read.csv("data/gapminder.csv", header=TRUE)
## extract the `country` column from a data frame (we'll see this later);
## convert from a factor to a character;
## and get just the non-repeated elements
countries <- unique(as.
year_country_gdp_euro <- gapminder %>%
  filter(continent == "Europe") %>%
  select(year,country, gdpPercap)
"Europe") %>%
```

```
  select(year,country, gdpPercap)
"Europe") %>%
  select(year,country, gdpPercap)
```