

Graduate School Class Reminders

- ▶ Maintain six feet of distancing
- ▶ Please sit in the same chair each class time
- ▶ Observe entry/exit doors as marked
- ▶ Use hand sanitizer when you enter/exit the classroom
- ▶ Use a disinfectant wipe/spray to wipe down your learning space before and after class
- ▶ Media Services: 414 955-4357 option 2

Documentation on the web

- ▶ CRAN: <http://cran.r-project.org>
- ▶ R manuals: <https://cran.r-project.org/manuals.html>
- ▶ SAS: <http://support.sas.com/documentation>
- ▶ Step-by-Step Programming with Base SAS 9.4 (SbS):
<https://documentation.sas.com/api/docsets/basess/9.4/content/basess.pdf>
- ▶ SAS 9.4 Programmer's Guide: Essentials (PGE):
<https://documentation.sas.com/api/docsets/lepg/9.4/content/lepg.pdf>
- ▶ Wiki: <https://wiki.biostat.mcw.edu>

HW 0: Homework problem for today

- ▶ RTFM and increase the font size of your screen
- ▶ I need to be able to look over your shoulder
- ▶ And I can't do that if the font size is small
- ▶ I'm optically challenged
- ▶ You don't have to submit this: just show me
- ▶ If not already addressed: who are the teams?
- ▶ Take turns submitting the HW problems to me
- ▶ Make them organized so I can easily review

HW 1: Homework problem for today

- ▶ Create a shell script to find the largest file per directory
- ▶ Starting in a directory find the largest file among this directory's file and its sub-directories for final results: `/data/shared/04224`
- ▶ Use recursion: in each directory, return largest file's name
- ▶ We can sort these files to find the largest file
- ▶ Hints: use pipes, `ls`, `head/tail` and (`echo` for debugging)
- ▶ `ls -lS `biggest` | head -n 1`
- ▶ add a directory for scripts to `PATH`:
`gouda$ PATH="$PATH:$PWD"`
- ▶ white space in shell scripts is very important/inflexible check docs and pay attention to Emacs' syntax highlighting
- ▶ spaces in filenames/directories is tricky
`eval FILE="\ "$PWD/$i\ "`
- ▶ Ignore files with special characters in their names like `$`
- ▶ You have to be fault-tolerant/defensive-programming
`gouda$ cd "$NEW" 2> /dev/null && biggest`
- ▶ Emacs Signals menu useful for debugging: `BREAK` and `KILL`

Outline for today

- ▶ Introduction to bash: Bourne-Again shell
- ▶ Bash shell initialization
- ▶ Bash shell scripting
- ▶ Hands-on with Emacs/ESS and R

Bash shell initialization

- ▶ `/etc/profile.d/custom.sh`
- ▶ `~/.bash_profile`
- ▶ `~/.bashrc`

Bash shell scripting

- ▶ Bash shell scripts allow you to create your own commands
- ▶ Powerful feature that promotes code re-use
- ▶ We have already seen the example `emacs-26.3`
- ▶ A very simple example: `/usr/local/bin/path`
- ▶ The environment variable `PATH` is a list of directories containing commands to type at the command line: `gouda$`
- ▶ Let's look at another example: `/usr/local/bin/hidden`
- ▶ The first line (if present) is a comment `#` followed by `!` called "shebang" for `#!` and the name of the shell like `#!/bin/bash`
- ▶ Shell choices (worst to best): `/bin/sh` for the Bourne shell, `/bin/ksh` for the Korn shell, `/bin/zsh` for the Z shell and `/bin/bash` for the Bourne-Again shell

Never use the C shell `/bin/csh`

- ▶ Return code 0 is a success and 1 is a failure: `$?`
- ▶ `gouda$ hidden ~/.Rprofile; echo $?`
- ▶ `gouda$ hidden emacs-26.3; echo $?`
- ▶ `gouda$ hidden file-does-not-exist; echo $?`
- ▶ And a more complex example: `/usr/local/bin/pf`

Bash shell scripting

- ▶ Simple syntax documented in the man page: `M-x man bash`
- ▶ Let's look at the following entries of **built-ins**
- ▶ `[[expression]]`
- ▶ **CONDITIONAL EXPRESSIONS**
- ▶ Other commands often found in scripts: **NOT built-ins**
- ▶ the `find` command: `M-x man find`
- ▶ the `sort` command: `M-x man sort`
- ▶ white space is very important/inflexible
- ▶ pay attention to Emacs' syntax highlighting

if command syntax: RED and BLUE lines optional

```
if IF-CONDITIONAL-EXPRESSION
then IF-THEN-LINE
  ADDITIONAL-LINES-AS-NEEDED
elif ELIF-CONDITIONAL-EXPRESSION-1
then ELIF-THEN-LINE-1
  ADDITIONAL-LINES-AS-NEEDED
:
elif ELIF-CONDITIONAL-EXPRESSION-M
then ELIF-THEN-LINE-M
  ADDITIONAL-LINES-AS-NEEDED
else ELSE-LINE
  ADDITIONAL-LINES-AS-NEEDED
fi
```

for command syntax: RED and BLUE clauses are optional

```
for VARIABLE in VALUE1 VALUE2 ...  
do LINE  
  ADDITIONAL-LINES-AS-NEEDED  
done
```

- ▶ Suppose that VARIABLE is i
- ▶ Typically, LINE and/or ADDITIONAL-LINES-AS-NEEDED will have references to the variable, \$i, which loops over VALUES
- ▶ for example, see
/data/shared/TorqueCluster/synch/yum-installed

```
for i in x y emacs-* {1..3}  
do echo $i  
done
```

Emacs/ESS and R

- ▶ Launch the latest version of emacs with your script:
`emacs-26.3`
- ▶ From the command line, you can get brief documentation
- ▶ In the `*shell*` buffer: `gouda$ emacs --help`
- ▶ Similarly: `gouda$ R --help` or `gouda$ R -h`
- ▶ Consult the man page for more: `M-x man emacs`
- ▶ Or `M-x man R`
- ▶ For the complete emacs manual: `F1 r`
- ▶ Or for the other manuals available: `M-x info` or `F1 i`

Emacs/ESS and R

- ▶ Copy my R profile to your home directory: press F8
gouda\$ cp ~rsparapa/.Rprofile ~
- ▶ Open it with C-x C-f .Rprofile
- ▶ Notice the file's title at the top, the menus and the toolbar
- ▶ Check out the mode-line at the bottom
- ▶ Create a directory for your personal library: press F8
gouda\$ mkdir -p ~/R/4.0.4/lib64/R/library
- ▶ Let's create an R program: C-x C-f lecture2.R
- ▶ Launch the default version, 4.0.4, of R: M-x R
- ▶ If you wanted the previous version 3.6.2: M-x R-3.6.2

Emacs/ESS and R

- ▶ There used to be <- keys on old keyboards
- ▶ Even before my time and I go back to the mid-80s
- ▶ But, what key should we use for assignment today?
- ▶ Now, we should just use =
- ▶ Since == is used to test for equivalence (as we'll see)
- ▶ R snobs still use <- but it is an anachronism
- ▶ Type in a simple R program (don't type the comments #)
- ▶ Or copy it from the shared directory: lecture2.R

```
a = installed.packages() # first line
# = creates an object to the left
table(a[ , "LibPath"])
b <- a
# <- creates an object to the left like =
b -> c
# -> creates an object to the right
ls()
```

Emacs/ESS and R

- ▶ In order of likely use and importance
- ▶ To submit a paragraph (code block bounded by blank lines):
C-c C-p
- ▶ To submit the whole buffer: C-c C-b
- ▶ To submit a single line: C-Enter
- ▶ To submit a region (a highlighted area):
C-c C-r