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`
- SAS 9.3: `https://support.sas.com/en/documentation/documentation-for-SAS-93-and-earlier.html`
- 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` (MCW/VPN)

SAS settings

- We have already discussed `autoexec.sas`
- The following settings in `autoexec.sas` were chosen as ideal for debugging errors without being overly burdensome
  `options nofmterr mprint errors=max noovp dkrocond=error;`
- `nofmterr` turns off errors when a user-defined format is not found since it happens far too often
- `mprint` shows code generated by SAS macros in the `.log`
- There are global config files which you can find on our system `/usr/local/sas/SAS18w47/SASHome/SASFoundation/9.4`
- There is `setinit.sas` that gives you the annual expiration date and a list of SAS products installed
- There is `sasv9.cfg` that is provided by SAS which should not be changed
- There is `sasv9_local.cfg` where local settings can be found like `-sasautos`

Debugging SAS programs

- A few tips
- Use the `.log`: check it after every run
- `F5` will take you to the first error or potential error
  Potential errors are `NOTE`s that are likely suspicious
  as defined by me in the ESS source code
- However, don't be complacent and overly rely on `F5`
  review the `.log` before even considering the `.lst`
- Likely, the most common error is forgetting a semi-colon
- Or, accidentally using a colon instead of a semi-colon
  technically, not a syntax error since SAS uses colons for addresses (see the `link` statement)
  so the error might appear to be on the next line

Debugging SAS programs: Potential Errors

https://github.com/emacs-ess/ESS/blob/a694b2627992bda5489c1b4b5bb750c590aa8d85/lisp/ess-sas-a.el#L732

```
NOTE: MERGE statement has more than one data set with repea
NOTE: Variable .* is uninitialized.
NOTE: SAS went to a new line when INPUT statement reached p
NOTE 485-185: Informat .* was not found
NOTE: Estimated G matrix is not positive definite.
NOTE: Compressing data set .* increased size by
NOTE: ERROR DETECTED IN ANNOTATE=
WARNING: Apparent symbolic reference .* not resolved.
WARNING: Length of character variable has already been set
WARNING: Not all variables in the list
WARNING: RUN statement ignored due to previous errors.
WARNING: Values exist outside the axis range
WARNING: Truncated record.
```

Debugging SAS programs: the put statement

- Useful for adding debugging info to the `.log`
- Automatic variable *PDV* (program data vector) lists that might be useful: `_ALL_`, `_NUMERIC_` and `_CHARACTER_`
- Create your own PDV lists: `VAR1--VARn` which is all variables from `VAR1` to `VARn` in order like `proc contents VARNUM`
- `put VALUE1 ... VALUEn;` can be variable/PDV lists
- `VALUEi` is either a character literal, a variable or a variable array reference
  but not a numeric literal nor a date literal, etc.
- `VALUEi` can be followed by a format
  (unless it is a literal)
- `VALUEi=` puts the variable name before the value
- Similarly, there is the `%put ...;` statement
- And, there is the `list;` statement for debugging the `input` statement which is like `put _all_;`

Debugging SAS programs: suppressing the `.log`/`.lst`

- For the `.log`, this is probably NOT a good idea
- However, if it is necessary, you can suppress it with the `%_printto` macro (which relies on `PROC PRINTTO`)
- For UNIX/Linux: `%_printto(log=/dev/null);`
- For Windows: `%_printto(log=nul:);`
- For both: `%_printto(log=%_null);`
- To turn the `.log` back on before the end of the program `%_printto();`
- Similarly, you can turn off the `.lst`
  more often desired than turning off the `.log`
- For UNIX/Linux: `%_printto(/dev/null);`
- For Windows: `%_printto(nul:);`
- For both: `%_printto(%_null);`
- To turn the `.lst` back on before the end of the program `%_printto();`

Redirecting the `.log`/`.lst`

- For the DATASTEP only: see the `file` statement
- Redirection is also a feature of the `%_printto` macro
- `%_printto(log=NAME.log);`
- To stop redirection of the `.log` before the end of the program `%_printto();`
- Similarly, you can redirect the `.lst`
- `%_printto(NAME.lst);`
- To stop redirection of the `.lst` before the end of the program `%_printto();`
- Notice that I'm keeping the extensions `.log` and `.lst` so that emacs recognizes the files via their extensions
- As we have seen, ESS[LOG] uses colors for syntax highlighting which allows you to get a visual inspection of the source code
- Currently, ESS[LST] doesn't have much functionality since it is just text, but that could change