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 at MCW and SAS Documentation

- SAS is proprietary: NOT open source software
- You don't buy SAS, you rent it: annual subscription fee
- MCW has a site-wide license for SAS
  Biostatistics buys into the license for the Cheese cluster
  But SAS is not currently available on the RCC cluster
  Students can get SAS for Windows for free from the help desk
- The SAS online documentation
  `http://support.sas.com/documentation`
- SAS was slow to move online and early web pages were terrible
- But, they have made big improvements recently
- We are running SAS v9.4: the latest version
  however, the latest documentation is more difficult to traverse
- So, I prefer the documentation for SAS v9.3 which are now included in the header above
  https://support.sas.com/en/documentation/documentation-for-SAS-93-and-earlier.html

The SAS language

- SAS is a fourth-generation language (4GL)
  4GL's are designed for special purposes
- SAS is designed for data science
  analysis, visualization, processing and management
- A strength is working with large data sets
- SAS will only use as much memory as you tell it to
  it is smart and efficient with *virtual* memory which is
  essentially hard-disk space (unlike R)
- SAS is case-insensitive and statements end in a semi-colon
- SAS is made up of multiple complementary products
- A mixture of the *DATASTEP* in Base SAS and
  PROCEDURES or *PROCS* from Base and
  other products like SAS/Stat
- The DATASTEP is vectorized
  in its own way that's different from R
- SAS' interactive capabilities are limited
- Therefore, SAS programs are typically run in batch in which
  case the SAS code is compiled and then executed: very fast

General SAS name rules

- SAS names are used for data set names, variable names, format names, informat names, etc.
- A SAS name can contain from one to 32 characters.
- The first character must be a letter or an underscore
- Subsequent characters must be letters, numbers, or underscores
- Blank spaces cannot appear in SAS names.
- Exception: format and informat names cannot end in a number and are usually far shorter but the maximum is/are 32 characters for numeric and 31 characters for character
- Special Rules for Variable Names: case preservation
  SAS remembers the combination of uppercase and lowercase letters that you use when you create the variable name. Internally, the case of letters does not matter. "CAT," "cat", and "Cat" all represent the same variable. But for presentation purposes, SAS remembers the initial case of each letter and uses it to represent the variable name when printing it.

SAS overview

- Blocks of code executed together
  start with `data` for a DATASTEP or `proc` for a PROC
  end with a `run;` statetment
- The types of SAS files related to a SAS program
  a program with file extension `.sas` (like `.R`)
  the output `.lst` (like `.Rout`)
  and the `.log` (for debugging: no R equivalent)
  the output can be in PDF, HTML, MS RTF, etc., but
  typically defaults to text for convenience (as we do here)
- BEWARE: Don't forget the `run;` statetment
  that will resulting in a confusing `.log` file
- All numeric numbers are floating point: there is no truly integer type (except for the SAS Macro facility which uses true integers and integer arithmetic)
- Character variables can be quite long nowadays, but by tradition most character variables are rarely longer than 40 or 200 characters; and typically much shorter for a large data set

SAS initialization and comments

- The SAS initialization file is `autoexec.sas`
- SAS looks first in the current directory and then in your home directory loading ONLY the first `autoexec.sas` it finds
- Copy mine: `> cp ~rsparapa/autoexec.sas ~`
- SAS has single line comments: RED is commented out

```
* THIS IS A COMMENT;
```

- Similarly, there are SAS macro single line comments

```
%* THIS IS A COMMENT;
```

  they do not appear in the `.log` when a SAS macro is run as opposed to the standard single line comments which do
- SAS has block comments

```
/*
THIS IS A COMMENT
FOR A BLOCK OF CODE
*/
```

- You can end a program at any line essentially commenting out the rest with an `endsas;` statement

Emacs/ESS and SAS function keys

- ESS[SAS] provides fairly comprehensive syntax highlighting (I wrote it) But, it is not bullet-proof and it can be fooled New SAS features are added every year: difficult to keep up
- Definitions reminiscent of the Display Manager which is rarely used today, but very influential for its time in the 1980s/1990s
- Key and Approximate Display Manager Equivalent in CAPS
- `TAB` moves margin to the next stop to the right
- `C-TAB` moves margin to the previous stop to the left
- `F1` help key, same as `C-h`
- `F2` refresh the buffer with the file contents
- `F3` SUBMIT
- `F4` PROGRAM
- `F5` LOG
- `F6` OUTPUT
- `F7` text file, if any
- `F8` go to the *shell* buffer
- `F9` open a SAS data set near point for viewing
- `F12` open a GSASFILE graphics file near point for viewing

Importing A CSV file into a SAS library

- SAS libraries are organized by directory
- For example, see `/data/shared/04224/NTDB/sas`
- Notice that there is an `autoexec.sas` file here that *includes* your ultimate version
- A good place to create a so-called LIBREF
  `libname LIBREF "DIRECTORY";`
- Data sets are accessed by one-level or two-level names
- One-level names are temporary and deleted at job completion
- The keyword `_null_` data set is not created at all
- Two-level names are permanently saved: `LIBREF.NAME`
- see `NTDB/sas/import.sas` and `NTDB/sas/dsd.sas`

```
proc import REPLACE datafile="../NTDB18.csv"
    out=ntdb.elder;
    GUESSINGROWS=MAX;
run;
proc contents VARNUM;
run;
```

SAS and vectorization: automatic looping

- See example `/data/shared/04224/NTDB/sas/vector.sas`
- The DATASTEP automatically loops over the rows/records of a data set
- A convenient form of vectorization

HW 1: Importing data with the DATASTEP

- Extra Packages for Enterprise Linux (EPEL)
  is a repository of free software packages: typically binary (already compiled if needed) for Red Hat flavored Linux distros
- You can see all of the packages installed on gouda by
  `> yum list installed \* ## escape the wild card`
  I have captured this command in the file
  `/data/shared/04224/gouda.txt`
- Chapters 4 and 5 of SbS describe how to read input data from files in arbitrary formats
- read this data with SAS into 3 variables
  the first column is `package`, the second is `version` and the third is `repo`
- how many of these packages come from the `repo` corresponding to EPEL defined by `"@epel"`?
- Hint: use the `firstobs` optional parameter

HW 2: Importing ECGs

- `https://www.physionet.org/content/ptbdb/1.0.0`
- the PTB database of ECGs has been copied to `/data/shared/04224/PTB`
- The database contain 549 conventional 12-lead resting ECGs with the corresponding 3 Frank lead ECGs
- There are 217 patients: see `CASES.txt`
- For example, 11 patients had left ventricular hypertrophy 086, 201, 210, 216, 217, 218, 219, 221, 222, 226 and 228
- And there 54 healthy controls: see `CONTROLS.txt`
- Patient 086 data is in the directory `patient086`
- The ECG data is in `s0316lre.dat`
- The data header is `s0316lre.hea`
- The first line says that there 15 leads (12+3)
- Sampled at 1000 Hz (1 per ms) for 115200 values per lead
- The next 15 lines are the leads
- The fifth number is the first value for that lead
- The sixth number is the sum of values for that lead
- However, the sixth number often doesn't seem to match

HW 2: Importing ECGs and signed integers

- We can read in the data with an `informat` like those described in Chapters 4 and 5
  see `/data/shared/04224/patient086.sas`
- The reason that the sixth number and SAS' sum don't agree is due to the way that the arithmetic was calculated by PTB
- They used 16-bit signed integers with two's complement
- A type that SAS doesn't have: see the `length` statement docs
  16-bit is old-fashioned: 32 or 64-bit are more common now
- Two's complement can ONLY represent integers from -2^{b-1} to $2^{b-1}-1$ where b is the number of bits
- And these values wrap-around
  $2^{b-1}-1+1$ is -2^{b-1} and NOT 2^{b-1}
  similarly, $-2^{b-1}-1$ is $2^{b-1}-1$
- So, if you calculate the wrap-around sums accordingly, then does it match the sixth number?