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` (MCW/VPN)

HW part 1: NTDB annual hospital case volume

- The number of patients seen by a given hospital is thought to be prognostically beneficial for many types of treatment like cancer, cardiovascular, trauma, etc.
  i.e., the *practice makes perfect* theory
- The NTDB case reporting is required for those hospitals which are designated trauma centers
- And their total case volume is fairly trivial to calculate
- However, the trauma designation is appeearently an annual designation and some hospitals go in and out of case reporting by year of hospital visit making their total volume a poor representation of their actual practice
- Therefore, average annual case volume is needed: calculate it
- Copy the data and my short program as a starting point
- `> cp /data/shared/04224/NTDB/NTDB18.csv ~`
- `> cp /data/shared/04224/NTDB/R/NTDB18.R ~`

HW part 2: NTDB cold-decking missing imputation

- For many a US Census, they were conducted on paper
- Often items were either mistakenly or intentionally left blank
- If the collected data is (NOT) Missing Completely at Random, you can (NOT) drop missing records from statistical analysis
- Hot-decking is the substitution of a nearby neighbor's value to replace a missing value on the resident's form
- Cold-decking is the substitution of any neighbor's value to replace a missing value on the resident's form
- For one or more missing covariates, record-level cold-decking imputation can be employed that is biased towards the null, i.e., non-missing values from another record are randomly selected regardless of the outcome
- This simple missing data imputation method is sufficient for data sets with relatively few missing values
- Perform cold-decking for the NTDB variables: `white`, `black`, `other`, `hispanic`, `male`, `sbp`, `pulse` and `rr` (respiratory rate)
- Hints: use nested `for`/`while` loops and `set.seed`/`sample.int` for random integers

HW part 2: NTDB cold-decking missing imputation

- More details
- Suppose that we have the following 5 variables:
  household income, owned home vs. renting,
  age of home, number of rooms and number of occupants
- It is reasonable to assume that these variables have a relationship between them
- Suppose record i has the observed/missingness pattern
  $\mathtt{A}_i$ $\mathtt{B}_i$ NA NA NA
- And we randomly draw record j to replace its values
  $\mathtt{C}_j$ $\mathtt{D}_j$ NA $\mathtt{E}_j$ $\mathtt{F}_j$
- Now, record i looks like this
  $\mathtt{A}_i$ $\mathtt{B}_i$ NA $\mathtt{E}_j$ $\mathtt{F}_j$
- So, we randomly draw again: record k
  $\mathtt{G}_k$ NA $\mathtt{H}_k$ $\mathtt{I}_k$ NA
- Now, record i looks like this
  $\mathtt{A}_i$ $\mathtt{B}_i$ $\mathtt{H}_k$ $\mathtt{E}_j$ $\mathtt{F}_j$

HW part 3 (mid-term): NTDB mortality data analysis

- A proper statistical analysis of the NTDB is FAR beyond the scope of this course
- However, it is a very convenient data set to practice a few of the tools described in Sections 11 and 12
- We will perform an analysis for the outcome `dead` based on annual average volume, `white`, `black`, `other`, `hispanic`, `male` and `sbp`
- N.B. the data is clustered by the hospital: `traumactr`
  for such a large data set, typical clustering techniques are too consuming so we will include `traumactr` as a variable above
  there are only 18 of them so this is a reasonable short-cut
- We will compute common statistics and make graphical figures
- For example, we will compute Receiver Operating Characteristic (ROC) curves

HW part 4: NTDB stratified random sampling

- the `gcstot` variable is the Glasgow Coma Scale (GCS) prognostically: 3 is the worst and 15 is the best
- as we will see, this is the most powerful predictor of mortality
- suppose that we want to divide the patient cohort into, say M=5, random samples of roughly equal size
- imbalances in a strong predictor like GCS will make the random samples substantially differ in mortality relationships due to these unwanted imbalances alone
- create a function to perform stratified random sampling (SRS) that accepts the data, *strata* and a setting for M
- SRS *stratifies* on one (or more) important variables called strata variable(s)
- sort the data by the strata variable(s) and then use random permutations of 1:M to divide the data into M samples
- unsort the data and return it in the original order, i.e., generally, you want to return data in the same order as that provided by the user
- Hint: use the `order` function to sort/unsort

R expressions: `expr`

- Expressions can end in a semi-colon or end-of-line
- Expressions can be spread over multiple lines IF each line ends in an operator like + so that the R interpreter knows that there is more to come
- Each expression returns a value
- You can group several expressions in curly brackets:
  { $\texttt{expr}_1$; ...; $\texttt{expr}_m$ }
  where the return value comes from the last: $\texttt{expr}_m$

```
library(parallel) ## an example of multi-threading
library(tools)
for(i in 1:mc.cores) mcparallel({psnice(value=19); expr})
```

We will return to multi-threading later

R expressions and logical conditions

- A condition is an R expression that can be evaluated as a TRUE or FALSE logical value either directly or indirectly
- A direct evaluation of a condition will yield TRUE or FALSE
- For example, `i %in% 2:3`
- An indirect evaluation occurs by type conversion or coercion
- A character value will yield an error
- A numeric or logical value of `NA` will yield an error
  often this will force you to use && or || instead of & or |
  such as in HW part 2
- A numeric value of `NaN` will yield an error
- A numeric value of zero is FALSE and all others are TRUE (except `NA` and `NaN`)

R `if` command syntax: RED and BLUE lines optional

$\texttt{if(cond}_1\texttt{) \{ expr}_1\texttt{ \}}$

$\texttt{else if(cond}_2\texttt{) \{ expr}_2\texttt{ \}}$

$\vdots$

$\texttt{else if(cond}_m\texttt{) \{ expr}_m\texttt{ \}}$

$\texttt{else \{ expr}_{m+1}\texttt{ \}}$

```
obj.list = mccollect() ## continuing multi-threading ex
obj = obj.list[[1]]
if(mc.cores==1 | class(obj)[1]!=type) {
    return(obj)
} else {
    m = length(obj.list)
    if(mc.cores!=m)
        warning(paste0("The number of items is only ", m))
    ...
}
```

R `for` command syntax

`for( VARIABLE in` $\texttt{expr}_1$ `)` $\texttt{expr}_2$

- If the `VARIABLE` is `i`, typically, $\texttt{expr}_2$ will involve `i`
- $\texttt{expr}_2$ can include `break` which ends the loop
- $\texttt{expr}_2$ can include `next` which indexes `i` and starts $\texttt{expr}_2$ over
- BEWARE: loops are rather slow for the R interpreter
- But hardware is constantly getting faster: its all relative
- 100,000,000 loops of this example takes 11s on my laptop

```
system.time({
M = 12
A = matrix(nrow=M, ncol=M)
dimnames(A)[[1]]=paste0(1:M)
dimnames(A)[[2]]=paste0(1:M)
for(i in 1:M)
    for(j in 1:M)
        A[i, j] = i*j ## times tables
if(M==12) print(A)
})
```

R `while` command syntax

```
while(cond) expr
```

- `expr` can include the `break` command which ceases

Writing your own R functions

`NAME1 = function(` $\texttt{name}_1$`, ...,` $\texttt{name}_m$ `) expr`

- the value of `expr` is returned
- but you have more control than in a typical R expression you can return a value at any place within `expr` via the `return` function
- Call this function via `NAME1 (`$\texttt{expr}_1$`, ...,` $\texttt{expr}_m$ `)`
- Or: `NAME1 (`$\texttt{name}_m$`=`$\texttt{expr}_1$`, ...,` $\texttt{name}_1$`=`$\texttt{expr}_m$ `)`
- Some, or all, arguments can have default values so that you do not have to supply every single argument

`NAME2 = function(` $\texttt{name}_1$`=`$\texttt{expr}_1$`, ...,` $\texttt{name}_m$`=`$\texttt{expr}_m$ `) ...`

To call: `NAME2 (` $\texttt{expr}_1$`, ...,` $\texttt{expr}_m$ `)`

Or: `NAME2 ( )`

Or: `NAME2 (` $\texttt{name}_n$`=`$\texttt{expr}_1$ `)`

Also, you can abbreviate the `names`, but it can't be ambiguous

The ... argument

- Often from within your new function, `NAME1`, you are calling another function (or functions): for example, let's say `NAME2`
- But `NAME2` might have a large number of arguments
- So it would be painful to include all of them in `NAME1`
- The `...` argument is a catch-all for named arguments that are unknown to `NAME1`
- `dots = function(...)  c(...)`

Lexical scope and nested environments

- S and S-Plus have *static* scope
- R has lexical scope

```
cube = function(n) {
  sq = function() n*n ## nested function definition
  n*sq()
}
## first evaluation in S
S> cube(2)
Error in sq(): Object "n" not found
S> n = 3
S> cube(2)
[1] 18
## then the same function evaluated in R
R> cube(2)
[1] 8
```

Lexical scope

- There are three types of variables in a function
  named arguments, local variables and free variables
- A free variable becomes a local variable upon assignment
- Otherwise, a free variable is sought by recursively widening the scope to the nested environments all the way to the global environment if necessary
  if not found, then an error is generated when the function runs
  it is not an error when the function is defined because the variable could be created at a later time
- EXCEPTION: a free variable (in its wider environment) can be altered from within a function by the *super-assignment* operator `<<-` or via the `assign` function
  in my experience, this is rarely necessary and probably should be avoided whenever possible

Object-orientation

- *Generic* functions can be specialized to have different actions for objects of different types
- They are named as `NAME.CLASS` where `NAME` is the function name to be called, i.e., `NAME(object)`
- And `CLASS` is the object type as determined by the `class` attribute, i.e., `class(object)`
- Generic functions have a `NAME.default` action that is used when either there is no `class` attribute for an object or the specific `NAME.CLASS` does not exist
- For *visible* functions (and some *invisible*), you can just type their name and see their definition
- For others, you need the `getAnywhere` function
- See `lecture4.R` and the generic `print` function
- There is typically help for the default method
  ```
  > ?print.default
  ```
- But others may, or may not, be documented

Section 11: Statistical models in R

- This reading is assigned due to the co-requisite requirement of this course, i.e., Statistical Models and Methods I
- However, in the interest of time, I'm not going to lecture on this material since there are many other things that need to be discussed
- I will show some examples based on the NTDB, etc.
- And assign homework/exam problems based on it
- So, if you have any questions, please feel free to ask
- But a firm understanding of statistical concepts is not an expected outcome of this course
- Rather, it is imperative that you learn effective R programming skills
- Learning R intimately will help you perform statistical analyses and complete related statistical programming tasks, but we will only discuss statistical practice in a cursory manner

American College of Surgeons National Trauma Data Bank (NTDB)

In the US Code of Federal Regulations (CFR), 45 CFR 46.101 (title 45, Public Welfare, section 46.101), research using certain publicly available data sets does not involve "human subjects". The data contained within these specific data sets are neither identifiable nor private and thus do not meet the federal definition of "human subject" as defined in 45 CFR 46.102. These research projects do not need to be reviewed and approved by the Institutional Review Board, (IRB). MCW's Human Research Protection Program (HRPP) has a list of their recognized "publicly available data sets". https://www.mcw.edu/departments/human-research-protection-program/researchers/smartform-instructions/public-data-sets

NTDB and data frames

- the NTDB has all de-identified trauma admissions for those aged 65 and older (restricted to 89 or less here)
  at all designated trauma center hospitals in the US
  about 1,000,000 admissions in the 2015 edition
  including the years 2007 to 2015
- the NTDB does NOT come as a CSV
- but I created a CSV for the first 18 trauma centers
  `/data/shared/04224/NTDB/NTDB18.csv`
  about 50,000 admissions: there is an (incomplete) data dictionary in the same directory
- this is for educational purposes ONLY (HIPAA exception)
- the `read.csv` function creates a `data.frame`
- a `data.frame` is the typical R data set object
  essentially a list of vectors of the same length
- like a matrix where the rows are lists
  and the columns are vectors of potentially diverse types
- the `subset` function creates a subset `data.frame`