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 3: Probability distributions

0. Normal `norm` Example:
   $\boldsymbol{Z \sim \mathrm{N}(\textbf{mean} = 10,\ \textbf{sd} = 25), \mathrm{P}[Z < c] = 0.025}$: what is $\boldsymbol{c}$?
1. Beta `beta` (the incomplete Beta function):
   $\boldsymbol{Z \sim \mathrm{Beta}\,(\textbf{shape1} = 10,\ \textbf{shape2} = 25), \mathrm{P}[Z < c] = 0.025, \mathrm{P}[Z < d] = 0.975}$: what is $\boldsymbol{c}$ and $\boldsymbol{d}$?
2. Gamma `gamma` (the incomplete Gamma function):
   $\boldsymbol{Z \sim \mathrm{Gamma}\,(\textbf{shape} = 10,\ \textbf{rate} = 25), \mathrm{P}[Z < c] = 0.025, \mathrm{P}[Z < d] = 0.975}$: what is $\boldsymbol{c}$ and $\boldsymbol{d}$?
3. F `f`: $\boldsymbol{Z \sim \mathrm{F}\,(\textbf{df1} = 10,\ \textbf{df2} = 25), \mathrm{P}[Z < c] = 0.025, \mathrm{P}[Z < d] = 0.975}$: what is $\boldsymbol{c}$ and $\boldsymbol{d}$?
4. Poisson `pois`: $\boldsymbol{Z \sim \mathrm{Poisson}\,(\textbf{lambda} = 25), \mathrm{P}[Z \le 15] = p}$: what is $\boldsymbol{p}$?
5. Normal `norm`: $\boldsymbol{Z \sim \mathrm{N}(\textbf{mean} = 10,\ \textbf{sd} = 25)}$
   Generate 10000 random $\boldsymbol{Z}$ and graphically compare the empirical CDF, `ecdf()`, with the true CDF. And calculate $\boldsymbol{\sigma^2}$ by Monte Carlo integration to compare with the true value.

Matrices

Mathematically, a matrix is represented as follows.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

R is a column-major language, i.e., matrices are laid out in consecutive memory locations by traversing the columns: $[a_{11}, a_{21}, \ldots, a_{12}, a_{22}, \ldots]$. R is written in C and Fortran where Fortran is a column-major language as well. However, C and C++ are row-major languages, i.e., matrices are laid out in consecutive memory locations by traversing the rows: $[a_{11}, a_{12}, \ldots, a_{21}, a_{22}, \ldots]$. So, if you have written an R function in C/C++, then pass it A transpose rather than A itself, i.e., `t(A)` as `t()` is the R transpose function.

Section 5.7: Matrices (two-dimensional arrays)

- `A=matrix(DATA, nrow=n, ncol=p)` creates A$_{n \times p}$
  `DATA` goes down the columns filling in column-major order
- `A=matrix(DATA, nrow=n, ncol=p, byrow=TRUE)`
  `DATA` goes across the rows filling in row-major order
  but A$_{n \times p}$ is still a column-major matrix!
- `diag(n)` creates an identity matrix `n` by `n`
- `A %*% B` is matrix multiplication
- `crossprod(A, B)` is the cross-product
  like `t(A) %*% B` but more efficient
- `crossprod(A)` is like `t(A) %*% A`
- `solve(A, b)` solves the equation `b = A %*% x` for `x`
- `solve(A)` creates A^{-1}, but matrix inversion can be unstable
- For example, when calculating `b %*%` A^{-1} `%*% b`, it is more numerically stable as `b %*% solve(A, b)`

Section 5.4: The recycling rule for mixed vector and array (or matrix) arithmetic

The precise rule affecting element by element mixed calculations with vectors and arrays is somewhat quirky and hard to find in the references.

- The expression is scanned from left to right.
- Any short vector operands are extended by recycling their values until they match the size of any other operands.
- As long as short vectors and arrays only are encountered, the arrays must all have the same dim attribute or an error results.
- Any vector operand longer than a matrix or array operand generates an error.
- If array structures are present and no error or coercion to vector has been precipitated, the result is an array structure with the common dim attribute of its array operands.

Section 5.4: The recycling rule for mixed vector and array (or matrix) arithmetic

Examples

$$A_{n\times p} - b_n = A - B \text{ where } B = \left[b_n^1, \dots, b_n^p\right]_{n\times p}$$

Recycling is what you likely intended

$A_{n\times p} - b_p = A - C$ difficult to demonstrate in general

so assume $n = 4,\ p = 3$ and $b = [b_1, b_2, b_3]$

which results in $C = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_3 & b_1 \\ b_3 & b_1 & b_2 \\ b_1 & b_2 & b_3 \end{bmatrix}$

Recycling is NOT what you likely intended

$$A^t_{p\times n} - b_p = A^t - D \text{ where } D = \left[b_p^1, \dots, b_p^n\right]_{p\times n}$$

Recycling is what you likely intended

Section 6: Lists

```
case = list(type="breast", ICDO="C50.3", stage="IIA",
    sex="F", side="L", birth=as.Date("1970/04/05"),
    diag=as.Date("2019/12/05"),
    surg=as.Date("2020/03/02"),
    T=list(P="T1a", size=0.45),
    N=list(P="N1", C="N0", surg="SLNB",
    positive=3, removed=7, max=0.25), M="M0")
```

- A *list* is an object consisting of an ordered collection of objects of singular or diverse types known as *components*
- Components are *numbered* and can always be referred to as such: `case[[1]]`, `case[[2]]`, `case[[3]]`, `...`
- Components of lists may also be *named*, and in this case the component may be referred to either by giving the component name or as a character string
- So, the first component can be referenced by `case[[1]]` or `case$type` or `case$"type"`
- Or, rarely, as a list itself called a *sublist*: `case[1]`

Comma separated values (CSV) files

- CSV files are a standard that arose from Fortran data files evolving late-60s to mid-70s: roughly what we have today
- Sadly, subject matter experts (SME) LOVE Excel and CSVs (we tell them to stop, but they don't listen)
- So CSV files are the de facto standard for data exchange
- But CSVs from spreadsheets are not type safe
  e.g., a numeric column can have text entries, etc.
- In the column for weights, they can type `deceased` in a cell
- So the `weight` column will be character instead of numeric
- Although, theoretically, they are trivial to import,
  the practice of SMEs will make your life difficult
- There does not appear to be a complete solution to this problem with R (other than manual editing)
- SAS has a few tricks that might help as we will see later

Section 8: Probability distributions

- Consider a discrete random variable (RV): $Y \in \{a, \dots, b\}$
- Probability density function (PDF):
  $f(c) = \mathrm{P}[Y = c]\,\mathrm{I}(c \in \{a, \dots, b\}) \in [0, 1]$
- Cumulative density function (CDF):
  $F(c) = \mathrm{P}[Y \leq c] = \sum_{y \in \{a,\dots,b\} \cap y \leq c} f(y) \in [0, 1]$
- $\mathrm{P}[c < Y \leq d] = \mathrm{P}[Y \leq d] - \mathrm{P}[Y \leq c] = F(d) - F(c)$
- Consider a continuous RV: $Y \in (a, b)$
- PDF: $f(c)\mathrm{I}(c \in (a, b)) \geq 0$
- CDF: $F(c) = \mathrm{P}[Y < c] = \mathrm{I}(a < c) \int_a^{\min(b,c)} f(y)\mathrm{d}y \in [0, 1]$
- $\mathrm{P}[c < Y < d] = \mathrm{P}[Y < d] - \mathrm{P}[Y < c] = F(d) - F(c)$
- For either a discrete or a continuous RV Y
  Inverse CDF: $p = F(c) \Rightarrow c = F^{-1}(p) = F^{-1}(F(c))$

Pseudo-random numbers and Monte Carlo integration

- CDFs can be used for many integral/summation calculations
- The Expectation of a function of a random variable, $g(Y)$
  Discrete RV: $\mathrm{E}[g(Y)] = \sum_{y \in \{a,\ldots,b\}} g(y) f(y)$
  Continuous RV: $\mathrm{E}[g(Y)] = \int_a^b g(y) f(y) \mathrm{d}y$
- Mean $= \mathrm{E}[Y] = \mu$, i.e., $g(Y) = Y$, the identity function
- Variance $= \mathrm{E}\left[Y^2\right] - \mathrm{E}[Y]^2 = \sigma^2$
- For some $g(Y)$ functions, there is no closed form solution
- Monte Carlo Integration: generate a *large* sample of pseudo-random numbers $Y_1, \ldots, Y_N$ and approximate $\mathrm{E}[g(Y)] \approx N^{-1} \sum_{i=1}^N g(Y_i)$

Section 8: Probability distributions

- The PDF R function is `dxxx` for density
- The CDF R function is `pxxx` for probability
- The Inverse CDF R function is `qxxx` for quantiles
- The Random Number R function is `rxxx` for random (paired with `set.seed` for reproducibility)
- `xxx` is one of the following suffixes:
  `beta, binom, cauchy, chisq, exp, f, gamma, geom, hyper, lnorm, logis, nbinom, norm, pois, signrank, t, unif, weibull, wilcox`
- These functions mean that we don't need all of those tables at the end of our textbooks
- We can calculate probabilities, summations/integrals, etc. for these distributions with R

Probability and discrete RVs: sports examples

- Coach K and the Duke Blue Devils college basketball team have been to 13 NCAA Final Fours
- During these appearances, they have won 5 NCAA Championships
- What is the probability that they would have won less than 5?
- Aaron Rodgers and the Green Bay Packers football team are 1 for 4 in the NFC Championship Playoff Game?
- What is the probability that they would have won more than 1 game?

Section 8.3: Hypothesis testing

- This course is a co-requisite with Statistical Models and Methods I
- The intent is not to teach you statistics, but to be prepared
- For example, what hypothesis tests are available?
- Here's a list of those in the `stats` package
- `> ls("package:stats", pattern="test$")`