Precept One: Review of Probability, Simulations, and Data Manipulation/Merging

Soc 504

02.09.17

Outline

- **Logistics**: problem set, office hours, replication project
- **Simulation**...steps towards:
  1. Begin with a random variable
  2. *Fully characterize* the random variable via a PMF/PDF
  3. *Summarize* some aspect of the random variable via expectation
  4. For step three, two general ways to find this summary:
     - 4.1 Solve for using formulae + tools from calculus and probability (aka analytically)
     - 4.2 Approximate using random sampling
  5. Today's precept is focused on: approximate using random sampling
- **Review of data manipulation and merging**: to help you with replication/extension and future work

Logistics

- Problem set 1: due **11:59 PM on Wednesday February 15th**
  - Challenge problem (zombies!): if you do this, don't need to do Problem 1
  - Optional problem (school dropout map!)
- Office hours (rotate each week; each in Wallace 190 usually, Friday's is in Wallace 165 this week only)
  - **Tuesdays**: 7:30-9:30 PM
  - **Fridays**: 1:30-3:30 PM
- Replication paper: **5:00 PM on Friday February 19th** need to:

  Choose a partner (talk to the preceptors if you're having difficulty with this) and e-mail Brandon, cc-ing the two preceptors, with 3 potential papers to replicate, including a 2-3 sentence explanation of why you are considering each paper. Papers can be from any literature but should be recent (last five years) and high impact. The reason for 3 options is in case you have issues obtaining data from your first choices. We will approve these choices (or tell you to look for a new one).

If you want our help in finding a partner...

- Email myself and Ian (can also cc Brandon but not necessary) with general topical interests by *this Friday (02.10) at midnight*- if we don't hear from you, we'll assume you're set on this front
- We'll help work out possible matches so you can meet next week to choose 3 papers

Alvin Roth- won Nobel Prize for work in stable matching applied to school choice, medical residency, and kidneys. Our stable matching will be much more ad-hoc but the stakes are also significantly lower than in those markets.

Context for simulation

Should be review from Soc 500 Precept 2

1. Begin with a random variable
2. *Fully characterize* the random variable via a PMF/PDF
3. *Summarize* some aspect of the random variable via expectation
4. For step three, two general ways to find this summary:
   - 4.1 Solve for using formulae + tools from calculus and probability (aka analytically)
   - 4.2 Approximate using random sampling

Random variables refresher: definition, notation, and example

- *Definition:* a function from the sample space (S or Ω), to the real line, in other words a numerical value calculated from the outcome of a random experiment.
- *Notation* (using example of outcome/dependent variable as RV; can also index using i):
  - Y: random variable (before we know/observe the outcome of the experiment)
  - y: realization of random variable (after we know/observe the outcome)
- *Main types*: discrete v. continuous

(Hopefully hypothetical) example

Event: How many minutes late to precept is Rebecca (Y_{RJ})?

- Re-run the experiment of the Thursday precept 5 times and might get the following realizations y_{rj}:
  1. $y_{rj} = 0$
  2. $y_{rj} = 10$
  3. $y_{rj} = 0$
  4. $y_{rj} = 0$
  5. $y_{rj} = 120$
     - Bounded since we're not counting early as negative late so $Pr(Y_{rj} < 0) = 0$
- If Y_{RJ} is **discrete**, above suggests that there might be a higher probability of drawing a Rebecca lateness value of 0 than other values
- If Y_{RJ} is **continuous**, above suggests that there might be a higher probability of drawing a Rebecca lateness value *within the range* of $0 - 0.01$ than within the range of $50 - 50.01$
  - *Lecture check*: why do we use the language "within the range" when discussing the probability that a continuous r.v. takes some outcome?

Helpful for checking that intuition...fully characterize the lateness random variable with its PMF/PDF

If we're measuring time as discrete (whole minutes), the *probability mass function* (PMF) characterizes the probability of drawing different discrete lateness values (e.g., 0 minutes; 10 minutes; 110 minutes)

If we're measuring time as continuous, the *probability density function* (PDF) characterizes the probability of Rebecca's lateness falling within a particular range (e.g., 0-5 minutes v. 20-40 minutes)

Once we have *fully characterized* the random variable via a PMF/PDF, we can *summarize* the random variable using expectation

- Expectation (mean) for discrete r.v.:

$$E(Y) = \sum_i y_i P(Y = y_i)$$

where $P(X = x)$ is the probability mass function (PMF).

  - *Present example*: $E(Y) = 0.7 \times 0 + 0.2 * 10 + 0.1 * 120 = 14$ minutes late

- Expectation (mean) for continuous r.v.:

$$E(Y) = \int_{-\infty}^{\infty} y f(y) dy$$

where $f(y)$ is the probability density function (PDF).

  - *Present example*: depends on $f(y)$!

- Can also use expectation to find variance and other QOI (Lecture 2)

Preview of GLM: how do we choose f(y) that characterizes lateness?

Each distribution was originally derived from a very specific set of theoretical assumptions. These assumptions may be stated in abstract mathematical form, but they may also be interpreted as political assumptions about the underlying process generating the data...the first principles from which they were originally derived represent interesting political science situations and are much closer to both data and theory (King, p. 41)

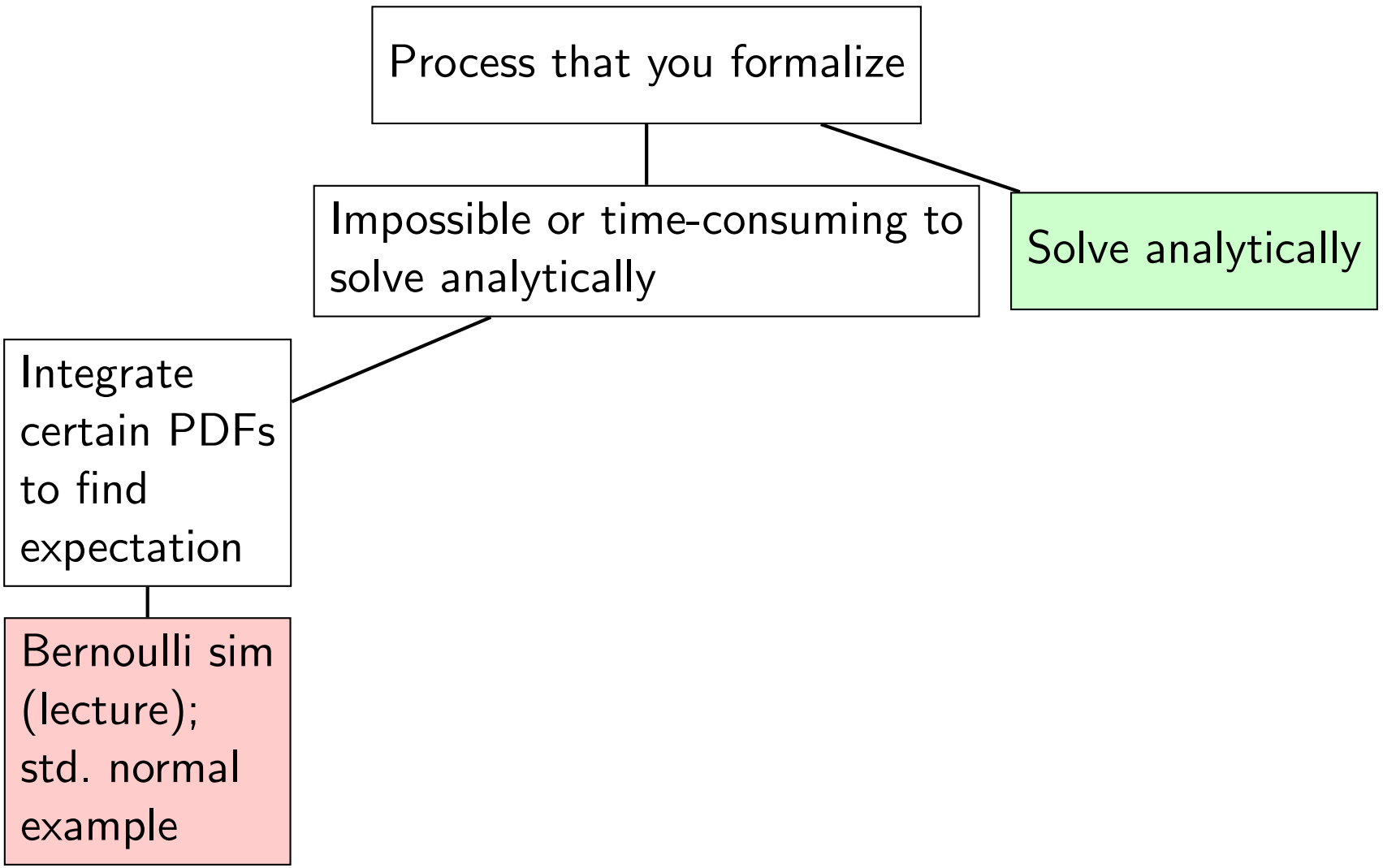
Preview of GLM: how do we choose f(y) that characterizes lateness?

- Thinking about data-generating process for lateness outcome (DGP)
- *DGP*: most of the time not late at all, some of the time miss shuttle (a little bit late), or very rarely sleep through alarm (miss entirely and very late)
- With that DGP in mind, begin to formalize characteristics we're looking for in candidate distributions:
  - *Continuous v. discrete*: do we measure lateness as discrete minutes or as an underlying continuous variable?
  - *Symmetric v. skewed*: DGP suggests that lateness *is not* distributed according to a uniform density or normal symmetric density; process above suggests higher density around 0 so positive/right skew
  - *Boundedness*: can't be negative late, so $Pr(Y < 0) = 0$
  - Next few weeks: will review many distributions, some of which describe this DGP well but many of which describe this DGP poorly

Summing up thus far and where next...

1. Begin with a random variable
2. *Fully characterize* the random variable via a PMF/PDF
3. *Summarize* some aspect of the random variable via expectation
4. For step three, two general ways to find this summary:
   - 4.1 Solve for using formulae + tools from calculus and probability (aka analytically)
   - 4.2 Approximate using random sampling

You gave us a formula to calculate the expectation using integration, and we learned how to integrate in calculus. When or why would we use simulation in this context?



Example: find prob. of falling within region of density

1. Say we transform the lateness variable (or more realistically, are working with a different r.v.) where the DGP is best characterized using a standard normal distribution ($\mu = 0$; $\sigma^2 = 1$)
2. We want to find the probability that an observation's lateness is between the mean and one unit above the mean, translating #2 into math:
   - Begin with general form for finding area under region of PDF:

$$I = \int_a^b f(y)dy$$

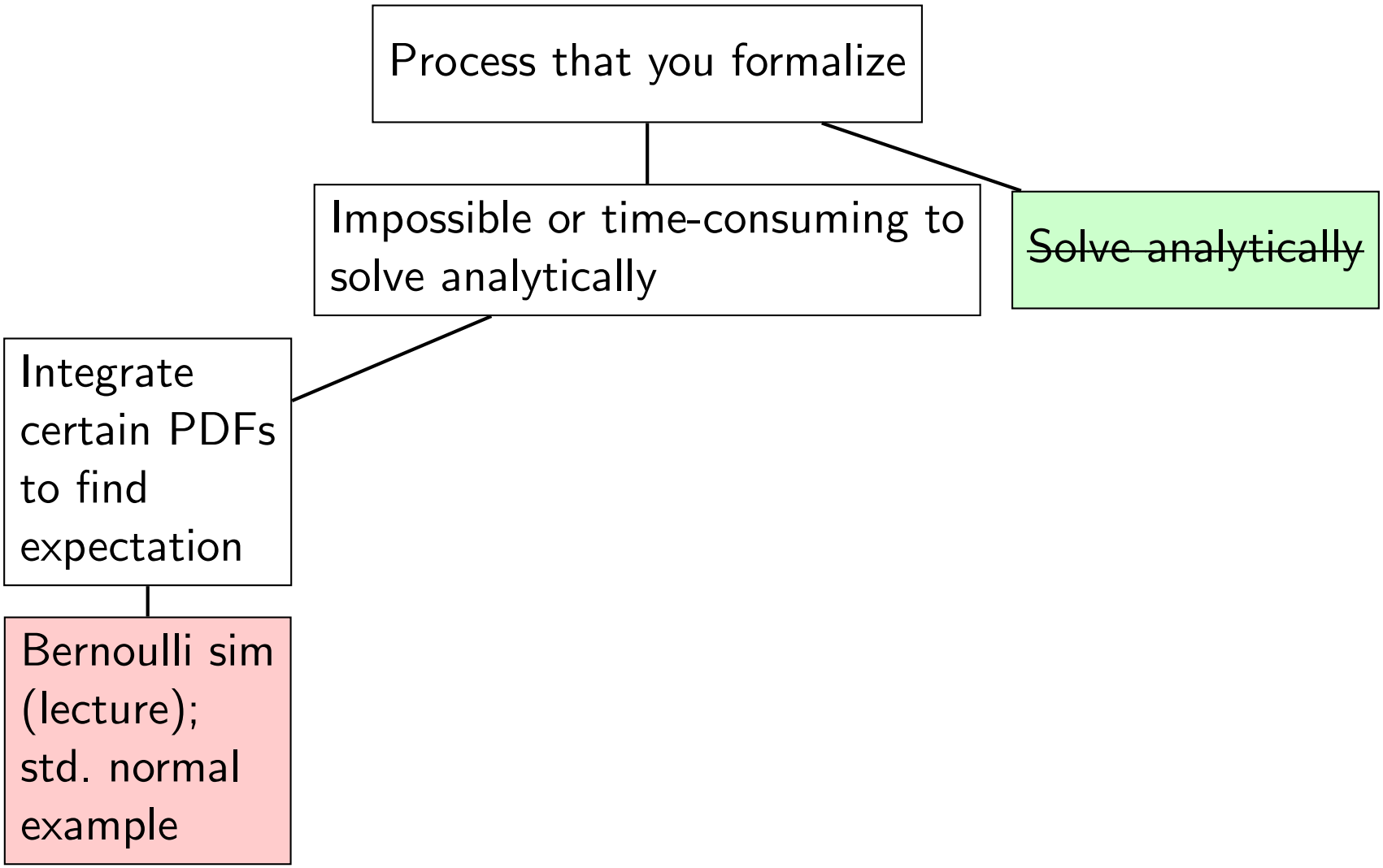
   - For $f(y)$, substitute in standard normal density (find in Brandon's slides or on Wikipedia...); for a and b, substitute in the range of the distribution we're interested in:

$$\phi(y) = \int_0^1 \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}} dy = \frac{1}{\sqrt{2\pi}} \int_0^1 e^{\frac{-y^2}{2}} dy$$

   - Anyone want to come to the board and integrate...?[1]

[1] You should actually know the answer due to standard normal's properties

Blocked path for analytic solution



Since analytic solution is a blocked path, need to approximate the integral using tool from set of methods for numerical integration

- *Focus in precept*: integration using Monte Carlo method
- *Discussed in lecture (in slightly different context):*
  - Trapezoid (trapezoidal; trapezium) rule: break up area of interest into sub-intervals of equal length, map trapezoid onto each sub-interval, find area of each interval's trapezoid and then sum
  - Many other methods

Zooming in: specific Monte Carlo steps for evaluating a PDF

- Integral:

$$I(f) = \int_a^b f(y)dy$$

- Approximate with:

$$I \approx \hat{I}_M = \frac{1}{M}\sum_{i=1}^{M} f(y)$$

Approach:

1. Randomly sample M points, $Y_1 .... Y_M$, from a uniform distribution over $[a, b]$
   - *Why sample from uniform?* Want every point within range we're integrating over $[a, b]$ to have an equal probability of being chosen; if sampled from normal, then points towards middle of $[a, b]$ would have higher probability of being evaluated than points towards end
2. Evaluate $f(y)$ at each of those points
3. Take mean of step 2 and multiply by $b - a$ (1 in this case)

Example with standard normal as M increases from $5 \rightarrow 500 \rightarrow 50000$ (also in your .rmd)

$$I(f) = \int_0^1 \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}} dy \tag{1}$$

Set your seed to 1848 and evaluate the above integral using the following steps:

1. Randomly sample $M = 5$, $M = 500$, and $M = 50,000$ points, $Y_1 .... Y_M$, from a uniform distribution over $[a, b]$
2. Evaluate equation 1 at each of those points
3. Create a density plot of step #2 that contrasts $M = 5$ to $M = 500$ to $M = 50000$; how does the density change as $M \rightarrow \infty$?
4. Take mean of step 2 and compare results for $M = 5$ to $M = 500$ to $M = 50,000$
5. Compare means from step #4 with solution via R's *pnorm*

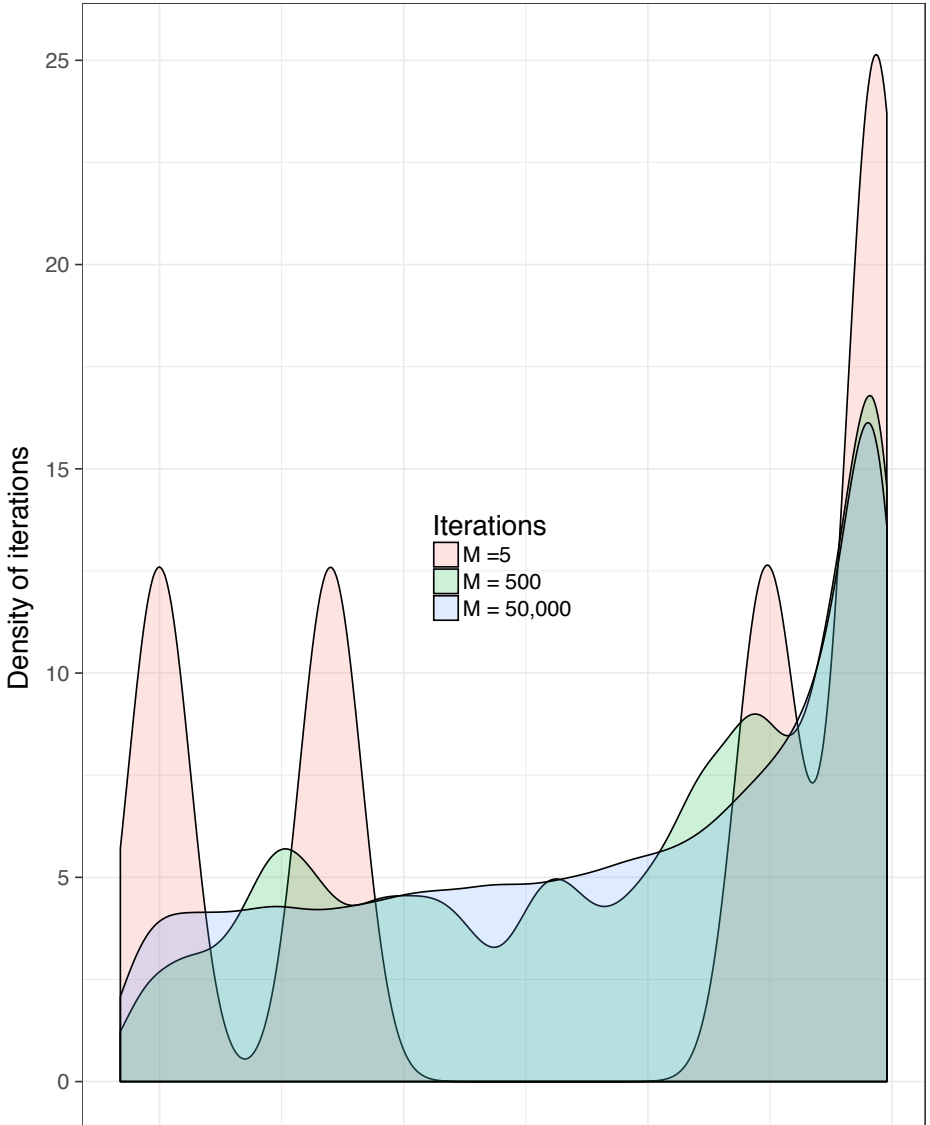
Code for steps 1 and 2

$$I(f) = \int_0^1 \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}} dy$$

```
set.seed(1848)
normal.MC.int <-function(M){
 points_from_ab<-runif(n = M, min = 0, max = 1)
#write out density function
#can also use dnorm in this case but we want
#to practice writing out functions for distributions
#that don't have a built-in dnorm equivalent in R
 exp((-points_from_ab^2)/2/sqrt(2*pi)
}

#apply function to M = 5, 500, 50000 iter.
m_results<-sapply(c(5, 500, 50000), normal.MC.int)
```

Comparing density of results under different numbers of iterations



Comparing probability found by mean across M iterations with probability from pnorm

pnorm result: 0.3413447

Iterations	$\hat{I}_M = \frac{1}{M}\sum_{i=1}^{M} f(y)$
M =5	0.3405771501
M = 500	0.3442309409
M = 50,000	0.3410664949

(See .rmd solutions for data transformation of m_results from slide 19 for plot and to generate these means)

Soc 500 refresher: why can we use pnorm to check our simulation?

- *pnorm:* gives the cumulative distribution function for a normal distribution
- How does this relate to the probability we were seeking to find?

$$\int_0^1 \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}} dy$$

- Answer:
  - In math: $F(y) = \int_a^b f(y)dy$
  - In words: we integrate PDF to get CDF and then evaluate in the appropriate range ($F(b) - F(a)$); so in the previous code, we used R's CDF function for a normal distribution (*pnorm*) to check our results

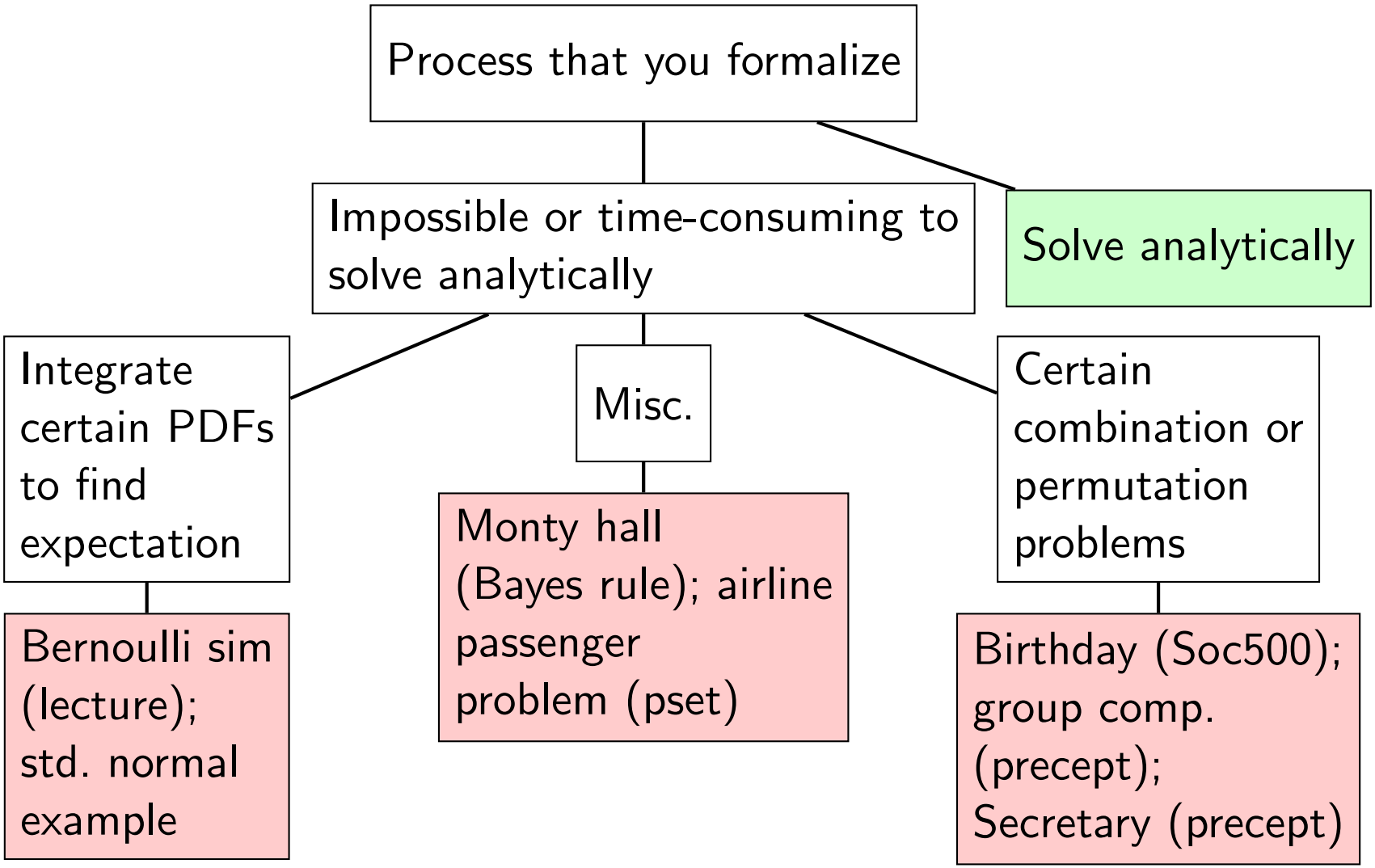
The fact that $M = 500$ and $M = 50,000$ provided better approximations of the integral than $M = 5$ should give a strong hint about the conceptual foundation for Monte Carlo integration and other uses of repeated iterations to simulate quantities that we'll review next...

Foundation of Monte Carlo: Law of Large Numbers

- Key principle: **The Law of Large Numbers**
  - Let $Y_1, \ldots, Y_n$ be n *independent* and *identically* distributed random variables. Let $E[Y_i] = \mu$ and $Var(Y_i) = \sigma^2$.
  - As n goes to ∞, the sample average $\frac{\sum_{i=1}^{N} Y_i}{n}$ converges almost surely to the true value $E[Y_i] = \mu$.
  - The same thing holds for deterministic functions of Y_i.
- What does this mean? It means that if I repeatedly take a lot of independent samples from a process and average the results, I get close to the true mean output of that process!

And because of that general foundation, we can generalize the process behind Monte Carlo *integration* to use Monte Carlo methods for a variety of other problems

Zooming out: typology of uses of Monte Carlo simulation beyond specific case of integration of PDFs



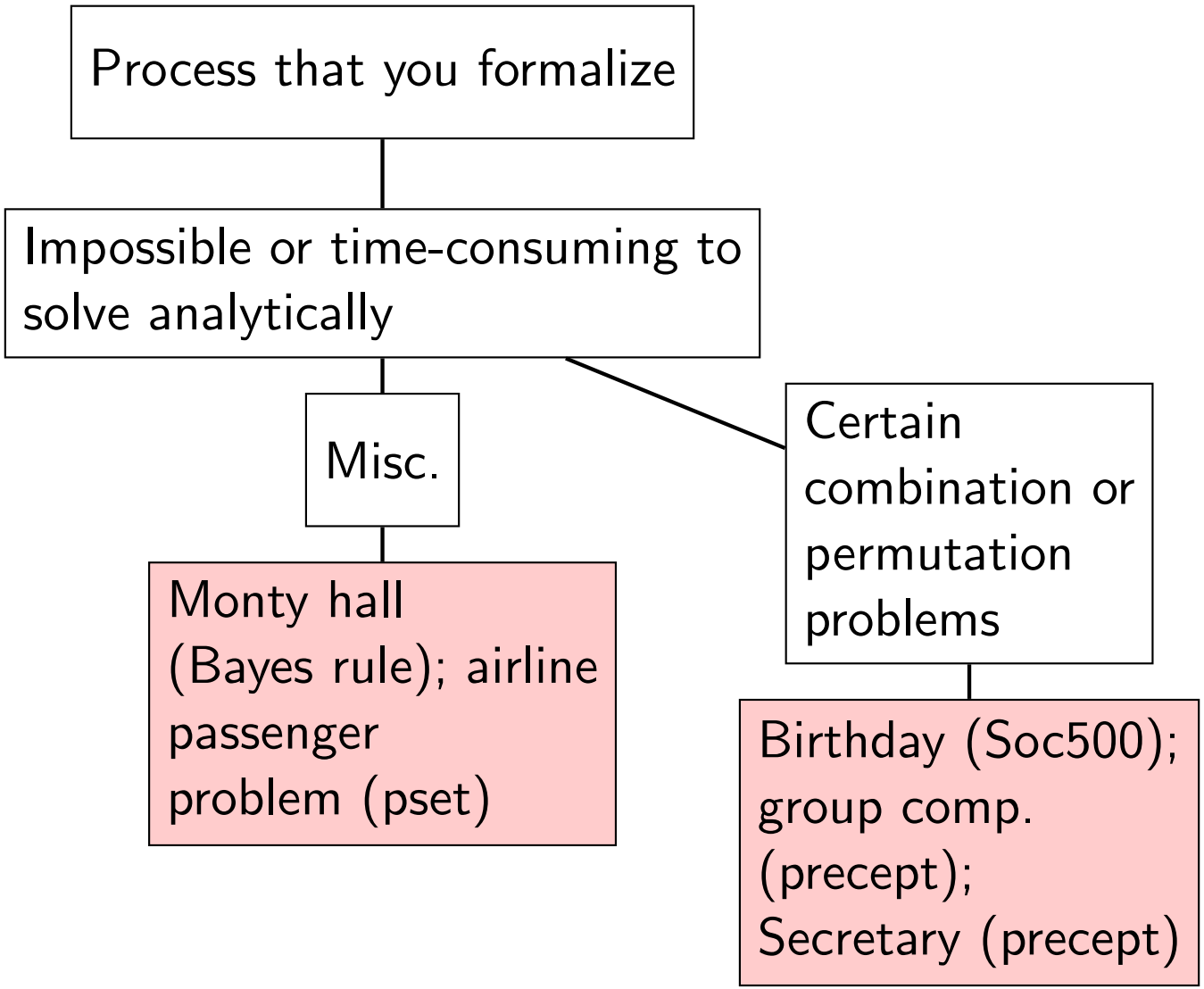
Zooming out: general Monte Carlo steps for a range of problems/applications beyond integration

1. Write down a *probabilistic model* of the process you're interested in
   - *Standard normal*: wrote down density function for normal distribution
   - *Monty Hall*: coded two strategies (switching to unopened door; not switching to unopened door) and enumerated which strategy would mean door choice $=$ winning door
   - *Birthday problem*: sampled N people from 365 days of year and enumerated count of those with same birthday
2. Repeatedly sample from the random components of the model to obtain realizations of the outcome you care about
   - *Standard normal example*: M random draws from uniform
   - *Monty Hall*: for loop from $1 \rightarrow M$ simulations
   - *Birthday problem*: for loop from $1 \rightarrow M$ simulations
3. Compute the summary of interest using the realizations (e.g. mean; 5th and 95th percentiles)
   - *Standard normal example*: mean
   - *Monty Hall and birthday problem*: enumeration of interest divided by number of simulations

Terminology: Monte Carlo v. simulation

Simulation means producing random variables with a certain distribution just to look at them. For example, we might have a model of a random process that produces clouds. We could simulate the model to generate cloud pictures, either out of scientific interest or for computer graphics. As soon as we start asking quantitative questions about, say, the average size of a cloud or the probability that it will rain, we move from pure simulation to Monte Carlo (NYU CS G22.2112-001)

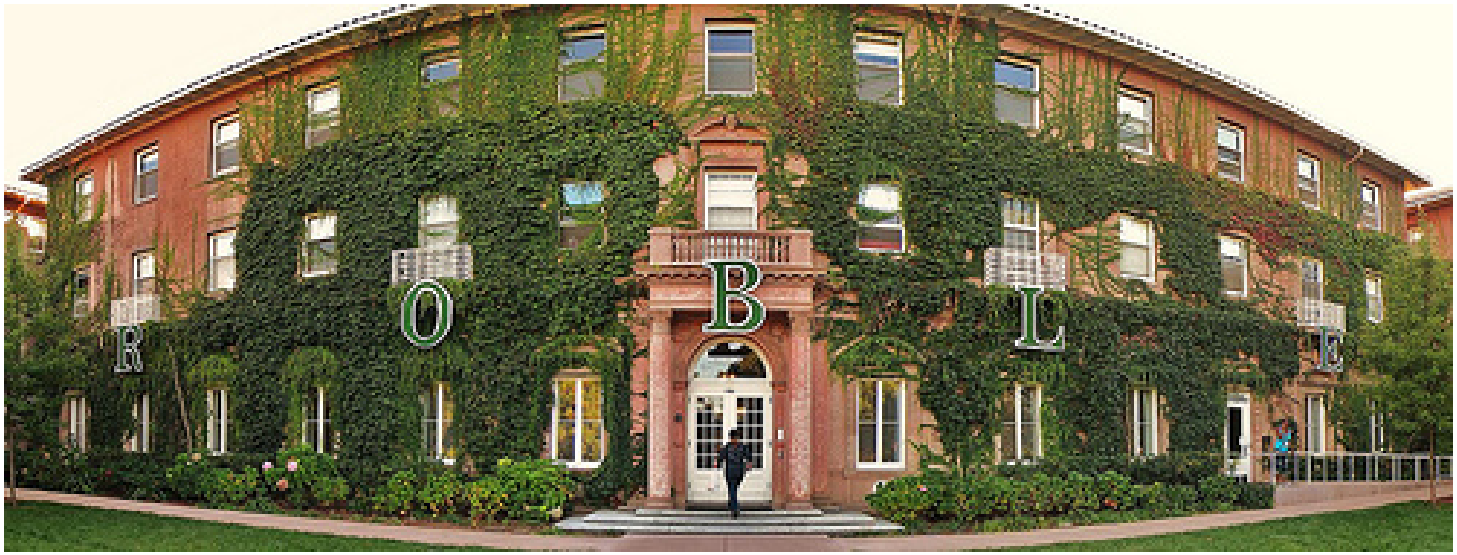
More practice with two examples from other branches of zoomed out typology



Two examples on this branch

1. Drawing friend groups with certain composition from dorms
2. Adopting the puppy with optimal cuteness from a queue of puppies you get to meet

Example one: drawing friendship groups from dorm



Example one: drawing friendship groups from dorm

Suppose you are a social scientist invited to observe a college dormitory with **55 students** from **5 different states** (so 11 students from each state). You have data on the composition of the students' observed friendship groups. You want to eventually compare those observed values to the probability of those groups forming by random chance. The first step in that comparison is to find the probabilities of certain groups forming by chance.

Set your seed to 01530 and use simulation with $M = 50,000$ to calculate the probability of two outcomes. *Hint*: you could either use 1) a for loop or 2) a function + apply or replicate

1. A **seven person** friendship group randomly drawn from the dorm has **three** students from one state and **four** students from a different state
2. A **seven person** friendship group randomly drawn from the dorm has **six** students from one state and **one** from a different state

We'll then as a group compare to analytic solution

Overview of how we approached running simulation

1. Created the population to sample from (*dorm* in the code) composed of 5 states $\times$ 11 students $=$ 55 students
2. Wrote a function (could also use loop) that takes in a randomly drawn group of 7 friends from the dorm pop. created in step 1 and enumerates two conditions we're looking to quantify (in our case, used *ifelse* for enumeration):
   - 2.1 3 students one state; 4 students other state
   - 2.2 6 students one state; 1 student other state
3. Applied or replicated function $M = 50,000$ times
4. Stored results and took mean to find probability of forming that group type

Creating the population and a function that tabulates randomly sampled friend groups displaying certain char.

```
set.seed(01530)
#create dorm to sample from
dorm <-rep(seq(1:5), 11)

#write a function takes in a sample of friends
and adds 1 to a vector if certain composition
compare<-function(friendgroup){
 three_four <-ifelse(all(c(3, 4) %in% table(friendgroup),1,
 0))
 six_one <-ifelse(any(table(friendgroup)==6, 1, 0))
 return(c(three_four, six_one))
}
```

If use function rather than loop, reminder of two options for how to iterate M times before finding mean

1. replicate

```
sim_results_rep <- replicate(50000, compare(sample(dorm,
7)))

rowMeans(sim_results_rep)
```

2. apply family

```
sim_results_app <- apply(as.matrix(1:50000), 1,
function(x) {compare(friendgroup = sample(dorm, 7))})

rowMeans(sim_results_app)
```

Analytic solution

1. 3 students one state; 4 students other state

$$\frac{5 * \binom{11}{3} * 4 * \binom{11}{4}}{\binom{55}{7}} = 0.0054$$

2. 6 students one state; 1 student other state

$$\frac{5 * \binom{11}{6} * 4 * \binom{11}{1}}{\binom{55}{7}} = 0.0005$$

Generalizable takeaways

Might be useful to think of problem in two separate, sequential steps

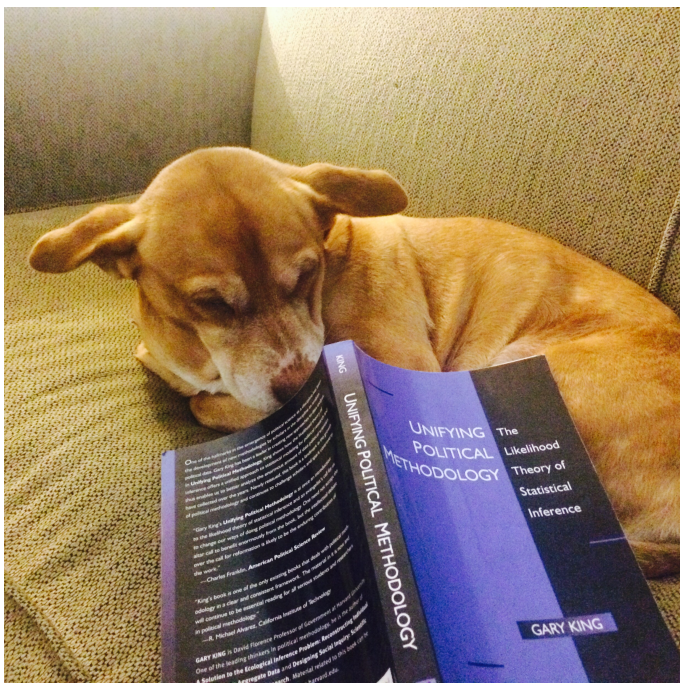
1. What quantity am I try to produce and how can I use code to characterize that quantity? (step where we created the dorm population, code to sample a 7-person friend group from that pop, and code to tabulate whether a friend group has a certain composition)
2. How can I iterate step 1 M times to draw upon LLN (mean of M simulations approximates true probability) (step where we embedded that code in a function and used apply or replicate to repeat M times)

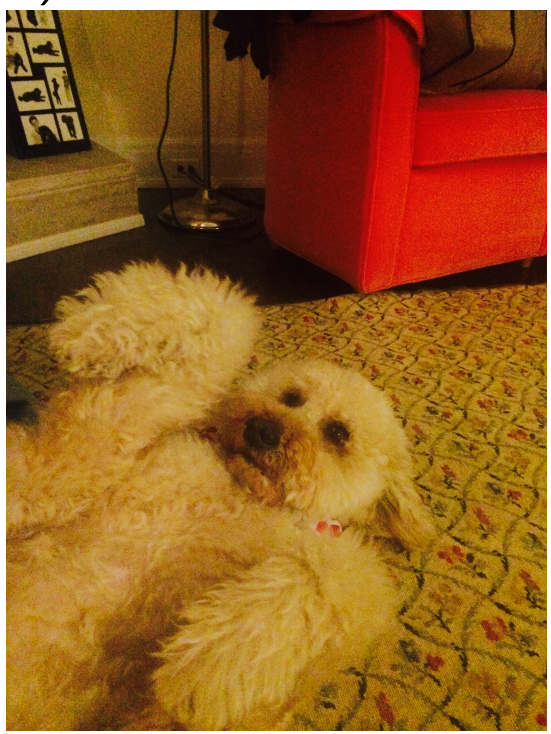
Example two: the Secretary Problem

Old version: choosing the optimal secretary

Example two: the Secretary problem

New version: choosing the optimal puppy (*nb*: consent for photos was obtained from each puppy's legal guardian)





Rules of problem

Goal: choose the puppy with optimal cuteness

1. You have n puppies to choose from that are waiting in the next room
2. You meet the puppies, one-by-one, in a random order
3. After you play with each puppy, you can evaluate that puppy's cuteness relative only to puppies that came before him/her, and then make a decision:
   - 3.1 Adopt- you immediately adopt the puppy and the process stops
   - 3.2 Pass - you can't return to the puppy if you decide later that you want to adopt it/want a second chance (they're cute...someone else will take it)

   Throughout, you have no idea about the distribution of cuteness in the pool of puppies but you do know the total number of puppies; assume that the puppies can be ranked (after meeting) in order of cuteness and that there are no ties

Question: when should you stop meeting puppies and commit to adopt? If you stop too soon, could miss optimal puppy that comes later; if you stop too late, could pass on optimal puppy that comes sooner

Remembering the typology, see if we can solve analytically by enumeration

- General strategy: meet some sample of puppies, rank their cuteness, and then choose the next puppy that exceeds the highest cuteness in the first sample
  - Example (with a larger sample): meet the first 4 puppies, rank them, and then adopt the next puppy you meet who exceeds the highest ranking of those first four
- Raises question: when in the puppy queue should we stop, take stock of the cuteness, and then choose the next puppy that exceeds existing highest cuteness?

Source for general strategy: datagenetics.com

Could try to solve analytically via enumeration, but even at $N = 3$ puppies, gets complicated

- Possible order of randomly meeting puppies, with higher number = higher cuteness
  1. 1 2 3
  2. 1 3 2
  3. 2 1 3 2 1 3
  4. 2 3 1
  5. 3 1 2
  6. 3 2 1
- Stop choice and fraction of times you choose optimal puppy:
  - Sample 0 and take first puppy in queue: $\frac{2}{6}$
  - Sample 1 and take highest subsequent puppy: $\frac{3}{6}$
  - Sample 2 and take highest subsequent puppy: $\frac{2}{6}$
- In case of $N = 3$, best to stop and choose next highest at $N = 1$ $(\frac{1}{3})$ puppies... Fine if we're meeting 3 puppies, but what if we have 50 puppies we get the chance to meet and select the cutest?! Could find analytic solution but challenging

Turn to simulation...

With 50 puppies and setting seed to 0134, use simulation to find which stopping point in the queue has highest likelihood of yielding puppy with optimal cuteness

(One) way to simulate

```
#set seed
set.seed(0134)

#generate cuteness ratings
cuteness <- 1:50

#create function that for each iteration,
#its applied, returns which stops out of
#1:50 yielded optimal puppy; fed a
#certain order of cuteness
optim_puppy <- function(puppy_order){

#iterate through stopping at different points
for(stop in 1:50){
```

(One) way to simulate continued...

```
#find the highest cuteness of the puppies
#seen thus far at that stopping point
  cutoff_cuteness <- max(puppy_order[1:stop])

  #select first subsequent puppy with higher cuteness
  puppy_adopt_index <- stop +
   which(puppy_order[stop+1:length(puppy_order)]
                              > cutoff_cuteness)[1]

 #if we already passed the cutest puppy, above will be NA
 #and we're stuck with last puppy
  if(is.na(puppy_adopt_index)){
      puppy_adopt_index <- length(puppy_order)
    }

  #find the cuteness value for the puppy we adopted
  puppy_adopted_cuteness <- puppy_order[puppy_adopt_index]
```

(One) way to simulate continued...

```
  #if that puppy has optimal cuteness, assign 1 to counter
  optimal_counter[stop] <- ifelse(puppy_adopted_cuteness == 50,
  1, 0)

  #for each iteration, identify which stop(s) allowed
  #us to select optimal puppy
  opt_stops <- which(optimal_counter == 1)
  }
  return(opt_stops)
}
#regenerate puppy order 50,000 times and feed function
optimal_stops <- unlist(replicate(50000,
                  optim_puppy(sample(cuteness, 50, replace = FALSE))))

#out of 50,000 iterations, see which stop had the highest count
table(optimal_stops)[table(optimal_stops)
== max(table(optimal_stops))]
```

Top 5 stop choices for choosing the optimal puppy

Stop	Count optimal out of 50,000
18	18915
17	18865
19	18864
20	18809
16	18711

Accords with analytic solution which is that as N (number of puppies in this case) increases, optimal stop point converges to:

$$\frac{N}{e} = \frac{50}{e} \approx 18.4$$

These examples–PDF's, friend groups, puppies–may seem scattered, but remember general approach to simulation

1. Write down a *probabilistic model* of the process you're interested in
   - *Friend groups*: coded way to enumerate when a friend group had certain composition
   - *Optimal puppy*: coded way to see which stops out of stopping at puppy 1 to stopping at puppy 50 yield optimal solution
2. Repeatedly sample from the random components of the model to obtain realizations of the outcome you care about
   - *Friend group*: randomly sampled groups of friends of size 7 from dorm
   - *Optimal puppy*: randomly sampled order in which you meet puppies
3. Compute the summary of interest using the realizations (e.g. mean; 5th and 95th percentiles)
   - *Friend group*: success at creating group over M iterations
   - *Optimal puppy*: which stopping point is optimal in highest number of iterations

Data manipulation and merging

Structure of review

1. Refresher on dplyr vocabulary (from methods camp slides)
2. Work together through example that illustrates three concepts:
   - 2.1 Aggregating to higher units of analysis
   - 2.2 Reading in data formats beyond .csv: focus on reading in .txt files as structured data versus unstructured strings
   - 2.3 Cleaning column names and subsetting data to columns of interest based on naming patterns
   - 2.4 Merging
     - 2.4.1 Preparing data for merge
     - 2.4.2 Types of merges
     - 2.4.3 Post-merge troubleshooting and diagnostics

In each case, we'll work through an example together that highlights each of the techniques rather than providing an exhaustive list of commands

Review of dplyr vocabulary

Goal	base R	dplyr
Extract columns	data[, c("col1", "col2"...)]	select(data, col1, col2...)
Extract rows	data[variable == condition,]	filter(data, variable == condition)
Arrange by column value (default = ascending)	data[order(variable),]	arrange(data, variable)
Add new variables to data.frame	data$newvar ¡- log(data$oldvar)	mutate(data, newvar = log(oldvar))
Grouped summary statistics	tapply(data$outcomevar, list(data$groupvar1, data$groupvar2...), function to perform)	summarise(group_by(data, groupvar1, groupvar2), stat1 = function 1 to perform, stat2 = function 2 to perform..)

Combining multiple verbs via piping

Example from methods camp slides of grouping AddHealth data by gender and debt status (e.g., males with no debt), finding the mean ratings of no cheating vs. love versus money's importance for a relationship, and arranging results by rating of money's importance

- Without piping:
```
arrange(summarise(group_by(addh, gender, debt),
nocheatavg = mean(nocheating), loveavg = mean(love), moneyavg =
mean(money)), desc(moneyavg))
```
- With piping:
```
addh %>%
group_by(gender, debt) %>%
summarise(nocheatavg = mean(nocheating), loveavg = mean(love),
moneyavg = mean(money)) %>%
arrange(desc(moneyavg))
```

Moving to example: variation in reporting of college crime by institutional characteristics

Three data sources:

1. Department of Education (DOE) data on report of sexual assaults on college campuses (SA)
2. IPEDS data on contextual features of colleges (e.g., selectivity; demographics)
3. OCR data on Title IX investigations

Step one: aggregate each college's SA count across campuses using dplyr verbs

1. Load reported_offense.csv data and save as sa_all
2. Print the number of unique institutions versus total number of observations in data– are they the same? If not, why not?
3. Create a data.frame that aggregates *reported_sa* (reported sexual assaults (SA)) so that each row is one institution and you have the following columns. When doing this aggregation, group by the institution's ID rather than its name:
   - Institution's name: some ID's are associated with multiple "names" but just choose the first name listed
   - Institution's ID
   - Count of reported SA across campuses
   - Rate of reported SA across campuses (count/sum of InstSize across campuses)
4. Save this data as sa_inst

Code for aggregating counts of reports across each unique college's campuses

```
#summarizing rates
sa_inst <-  sa_all %>% group_by(ID) %>%
            summarize(Institution = Institution[1],
            total_sa = sum(reported_sa),
            total_sa_rate = total_sa/sum(InstSize))
```

Generalizing: when is this form of aggregation useful?

Non-exhaustive examples:

1. Computing descriptive statistics for defined groups
   - *Example:* mean income across each gender $\times$ marital status cell, could group by gender and marital status
2. Combining more granular units of analysis into higher units of analysis
   - *Example*: previous example (didn't care about differences across campus, only wanted one count and rate per college). Others: have counties nested in states and only care about state outcomes, have students nested in schools and only care about school-level outcomes, etc
3. Efficiently summarizing trends
   - *Example*: Data is long format with each row as a county-year and the country's debt-GDP ratio; want to summarize the mean debt-GDP ratio for all countries and plot by year

Step two: read in data formats beyond .csv: focus on .txt files

A research assistant started to put together a list of colleges in a .txt file with current Title IX investigations by the DOE's Office for Civil Rights (OCR). You asked the RA to note the Title IX coordinator's name at each of these schools, but they got busy with thesis work and so only looked up this information for two of the schools. So some of the colleges have three columns of data (State, College name, OCR coordinator name) while most have two columns.

Important function: read.table and different options

- *When to use:* .txt file that you want to read in as structured data
  - *Sidenote- when not to use:* when you want to read in a file as a single string–for instance, you want to create a data.frame where each row is a document and one column is *all of* the document's text. read.table causes issues here because it wants to separate the text into separate columns based on some delimiter so should use read_file in readr or readLines
- Arguments also present in read.csv but may be esp. important for reading in .txt as structured data:
  - **sep**: instead of variables/columns being separated by commas as in a .csv file, .txt files often separate by tab so you may want

    ```
    sep = "\t"
    ```
  - **fill**: if set to TRUE, R fills in rows with unequal length with blank spaces so that you don't get error about unequal row length in data.frames (remember that unlike lists, df and matrices need equal length)

Step two: code for reading in .txt file with rows of unequal length

```
#read in file
ocr_results <- read.table("OCR_invest.txt", header = TRUE,
                          sep = "\t",
                          fill = TRUE)
```

Step three: cleaning column names and selecting columns

1. Load the IPEDS data on college contextual characteristics which is in the more standard .csv format (*IPEDS.csv*)
2. You should notice the column names have dots where spaces were in the .csv file names. Use gsub to remove dots by referencing this guide: `http://www.endmemo.com/program/R/gsub.php`
3. Then, restrict the data to the following columns: UnitID, InstitutionName, and any column containing the pattern SAT, then filter to those with positive values on SATCriticalReading25thpercentilescoreIC2013_RV

Step three: code

```
#remove periods from column names
#option one: use \\ to tell gsub to escape special character
clean_cols <- gsub("\\.", "", colnames(ipeds))

#option two: set argument fixed = TRUE to
#tell gsub to interpret pattern literally
clean_cols_2 <- gsub(".", "", colnames(ipeds), fixed = TRUE)

#assign to column names
colnames(ipeds) <- clean_cols

#restrict to cols of interest
ipeds_2 <- ipeds %>% select(UnitID,
            InstitutionName,
            contains("SAT")) %>%
            filter(SATCriticalReading25thpercentilescoreIC2013_RV >
```

Overview of merging

1. Preparing data for merge (will focus on when reviewing troubleshooting, since this can involve going back to prepare data for merge better)
2. **Types of merges**
3. Post-merge troubleshooting and diagnostics

Can think of types of merges through framework of data as sets (circles on venn diagram)

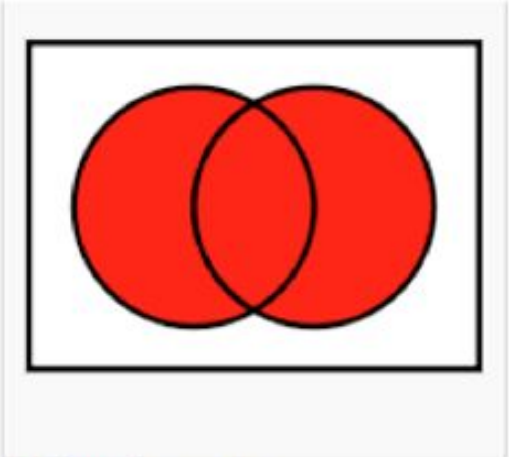
We have three datasets:

1. SA reports data (Crim): 2163 colleges, has ID's and names
2. OCR investigations data (OCR): 55 colleges, only has names
3. IPEDS data on selectivity (IPEDS): 1303 colleges, has ID's and names

We'll see what happens when we practice three common types of join on different pairs of those datasets. A helpful resource for these is: http://stat545.com/bit001_dplyr-cheatsheet.html#full_joinsuperheroes-publishers

Merge option one: retain all observations present in dataset A *OR* dataset B

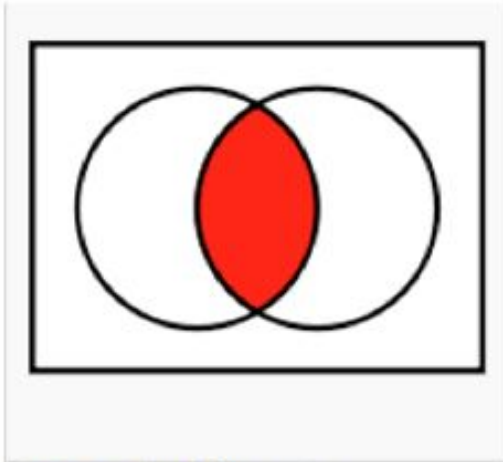
- Concept:



- Implementation:
  - Base R: merge with
    `all.x = TRUE and all.y = TRUE`
  - dplyr:
    `full_join`
- Example to practice with: Merge IPEDS and Crim based on numeric ID, retaining all colleges in either IPEDS *or* Crim

Merge option two: retain all observations present in dataset A *AND* dataset B

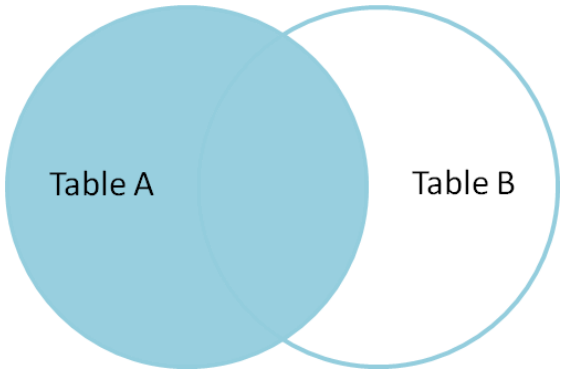
- Concept:



- Implementation:
  - Base R: merge with no flags specified
  - dplyr:
    `inner_join`
- Example to practice with: Merge IPEDS and Crim based on numeric ID, retaining only colleges in IPEDS *and* Crim

Merge option three: retain all of dataset A's observations but only observations in dataset B present in dataset A

- Concept:



- Implementation:
  - Base R: specify A first and merge with `all.x = TRUE`
  - dplyr: specify A first and merge with `left_join`
- Example to practice with: treating the Crim data as dataset A, merge in the OCR investigations data, retaining OCR data on colleges only if they are in the Crim data. Since the OCR data lacks ID's, merge based on the college's name

Last example was probably more challenging, which leads to a brief focus on troubleshooting two challenges

1. Identifying observations lost in merge
2. Merging based on string (much more we could cover but here, focus on common issue of difficult to detect leading and trailing whitespace in string)

Troubleshooting one: identifying observations lost in merge

- Thinking of merges as unions, intersections, and other ways of combining two sets (e.g., left join) leads us to base R's family of *set* operations that can help us explore which observations will remain in a merge
- For each of the three merges you performed based on the last three slides, use the appropriate set-related command to identify which observations will be in the merged dataset
  - `union(A, B)`
  - `intersect(A, B)`
  - Observations in A not in B:

    `setdiff(A, B)`

Troubleshooting two: when merging on character strings, checking for sources of string incompatibility

- If previous step returns a different number of rows remaining than you would expect given the data, and the merge is based on a string rather than an ID, want to troubleshoot ways that two strings that look alike when you view them might not merge properly in R
- Vast topic, but today, will focus on how to address one particularly annoying error: leading and trailing whitespace in the string
- Your .rmd contains an example we'll practice with
  - Try an inner_join of df_leading and df_noleading based on the sibnames column
  - That should only return Adam's favorite dog
  - Use one of the following commands on the names column in df_leading to remove leading whitespace from Becky and trailing whitespace from Bruce
    - `trimws(varname)`
    - `gsub("^\\s+|\\s+$", "", varname)`

Recap

- **Logistics**: problem set, office hours, replication project
- **Simulation**...steps towards:
  1. Begin with a random variable
  2. *Fully characterize* the random variable via a PMF/PDF
  3. *Summarize* some aspect of the random variable via expectation
  4. For step three, two general ways to find this summary:
     - 4.1 Solve for using formulae + tools from calculus and probability (aka analytically)
     - 4.2 Approximate using random sampling
  5. Today's precept is focused on: approximate using random sampling
- **Review of data manipulation and merging**: to help you with replication/extension and future work