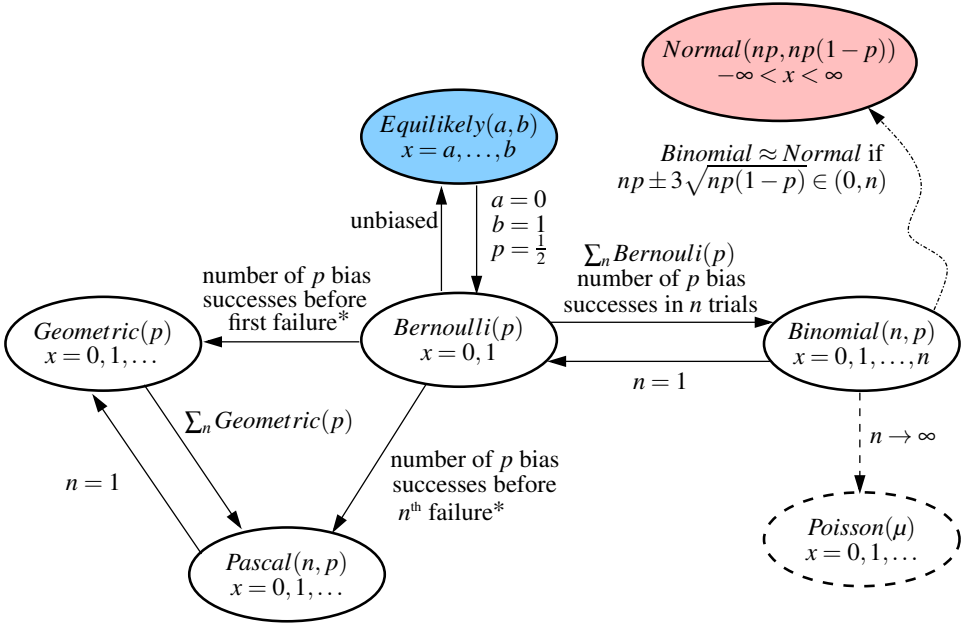
Algorithm Design Criteria

- The design of a correct, exact and efficient algorithm to generate corresponding random *variates* is often complex
  - Portability - implementable in high-level languages
  - Exactness - histogram of variates should converge to pdf
  - Robustness - performance should be insensitive to small changes in parameters and should work properly for all reasonable parameter values
  - Efficiency - it should be time efficient (*set-up* time and *marginal* execution time) and memory efficient
  - Clarity - it is easy to understand and implement
  - Synchronization - exactly one call to `Random()` is required
  - Monotonicity - it is synchronized and the transformation from u to x is monotone increasing (or decreasing)
- Inversion satisfies some criteria, but not necessarily *all*



*other sources think of *Geometric* and *Pascal* as number of failures before the first or nth success.
Pascal(n, p) also called the *negative binomial* distribution.

(I wish I had seen this 30+ years ago. . .)

Write pseudo code for generating a Binomial random variate, given `Bernoulli(p)`.

How does it fair in the context of our **Random Variate Criteria**?

Write pseudo code for generating a Binomial random variate, given `Bernoulli(p)`.

```
function Binomial(n, p)

s ← 0
while( n > 0 ) do (
  if( Bernoulli(p) ) increment s
  decrement n
)
return s
```

How does it fair in the context of our **Random Variate Criteria**?

It is **portable**, **exact**, **robust**, and **clear**

It is not **efficient** or **syncronized** (and syncronized precludes **monotone**)

…assuming *Bernoulli*(p) is portable, exact and robust; which it is :)

Techniques for Variate Generation:

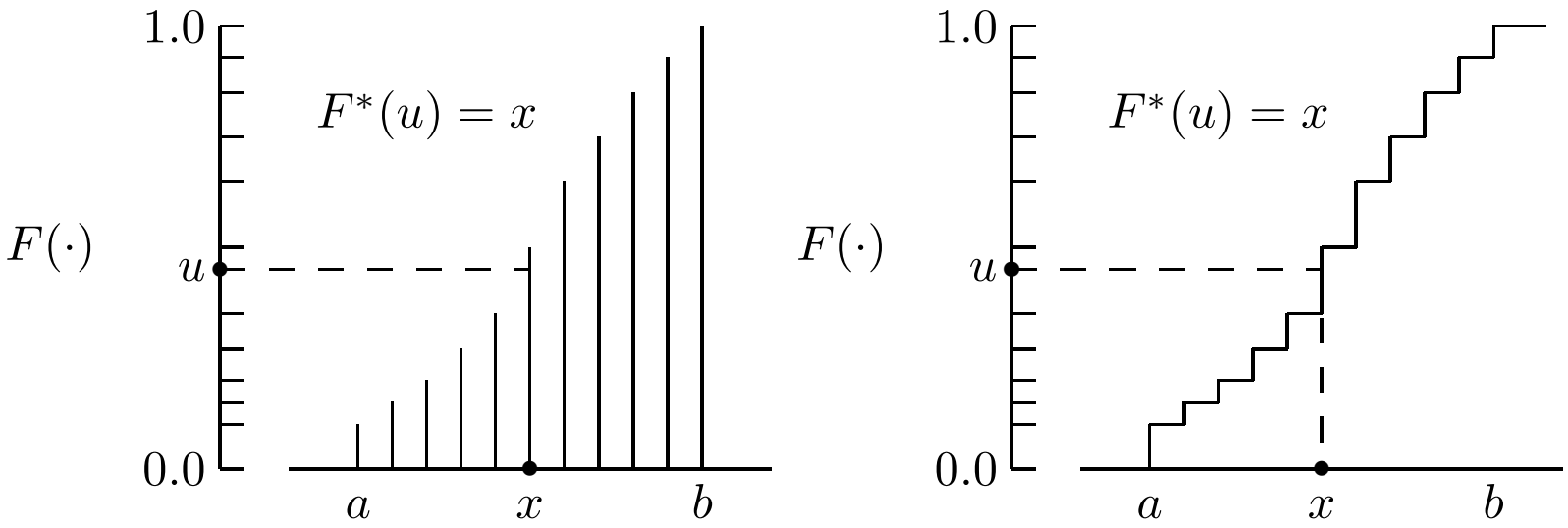
i Generation by CDF inversion (using $F^{-1}(u)$)
Already discussed in previous lectures!
e.g. $Uniform(a,b)$**,** $Equilikely(a,b)$**, distribution of radi in a circle**

ii Generation by construction
Already demonstrated in this lecture, $Binomial(n,p)$

iii Generation by CDF search...

Example 6.2.1

Two common ways of plotting a cdf with $\mathcal{X} = \{a, a+1, \ldots, b\}$:



Theorem (6.2.1)

Let $\mathcal{X} = \{a, a+1, \ldots, b\}$ where b may be ∞ and $F(\cdot)$ be the cdf of X For any $u \in (0, 1)$,

- *if $u < F(a)$, $F^*(u) = a$*
- *else $F^*(u) = x$ where $x \in \mathcal{X}$ is the unique possible value of X for which $F(x-1) \leq u < F(x)$*

Algorithm 6.2.1

- For $\mathcal{X} = \{a, a+1, \ldots, b\}$, the following linear search algorithm defines $F^*(u)$

Algorithm 6.2.1

```
x = a;
while (F(x) <= u)
    x++;
return x; /*x is F*(u)*/
```

- Average case analysis:
  - Let Y be the number of `while` loop passes
  - $Y = X - a$
  - $E[Y] = E[X - a] = E[X] - a = \mu - a$

The Need for Truncation

Too often novice simulation writers forget that many probability distributions have ∞-tails.
For instance they determine an interarrival time for coffee shop customers with ($\bar{r} = 8.127$ minutes for the system):

$$r_i = Exponential(8.127)$$

Forgetting this may (albeit rarely) return a value of r_i on the order of **hours**, **days**, or **years** — something inconceivable in the real-world system.
What we want to do is **truncate** the distribution, assuring the maximum of r_i is some reasonable value, hopefully determined by empirical evidence.

The Need for Truncation

What we want to do is **truncate** the distribution, assuring the maximum of r_i is some reasonable value, hopefully determined by empirical evidence.

```
function truncatedRi( r̄, max_{r_i} )

  r_i ← Exponential(r̄)
  if ( r_i > max_{r_i} ) then (
    return max_{r_i}
  ) else (
    return r_i
  )
```

The Need for Truncation

What we want to do is **truncate** the distribution, assuring the maximum of r_i is some reasonable value, hopefully determined by empirical evidence.

```
function truncatedRi( r̄, max_{r_i} )

  r_i ← Exponential(r̄)
  if ( r_i > max_{r_i} ) then (
    return max_{r_i}
  ) else (
    return r_i
  )
```

Histogram of Wrong Truncation Technique for $Exponential(8.127)$

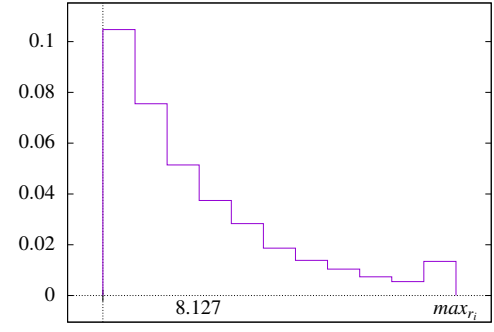
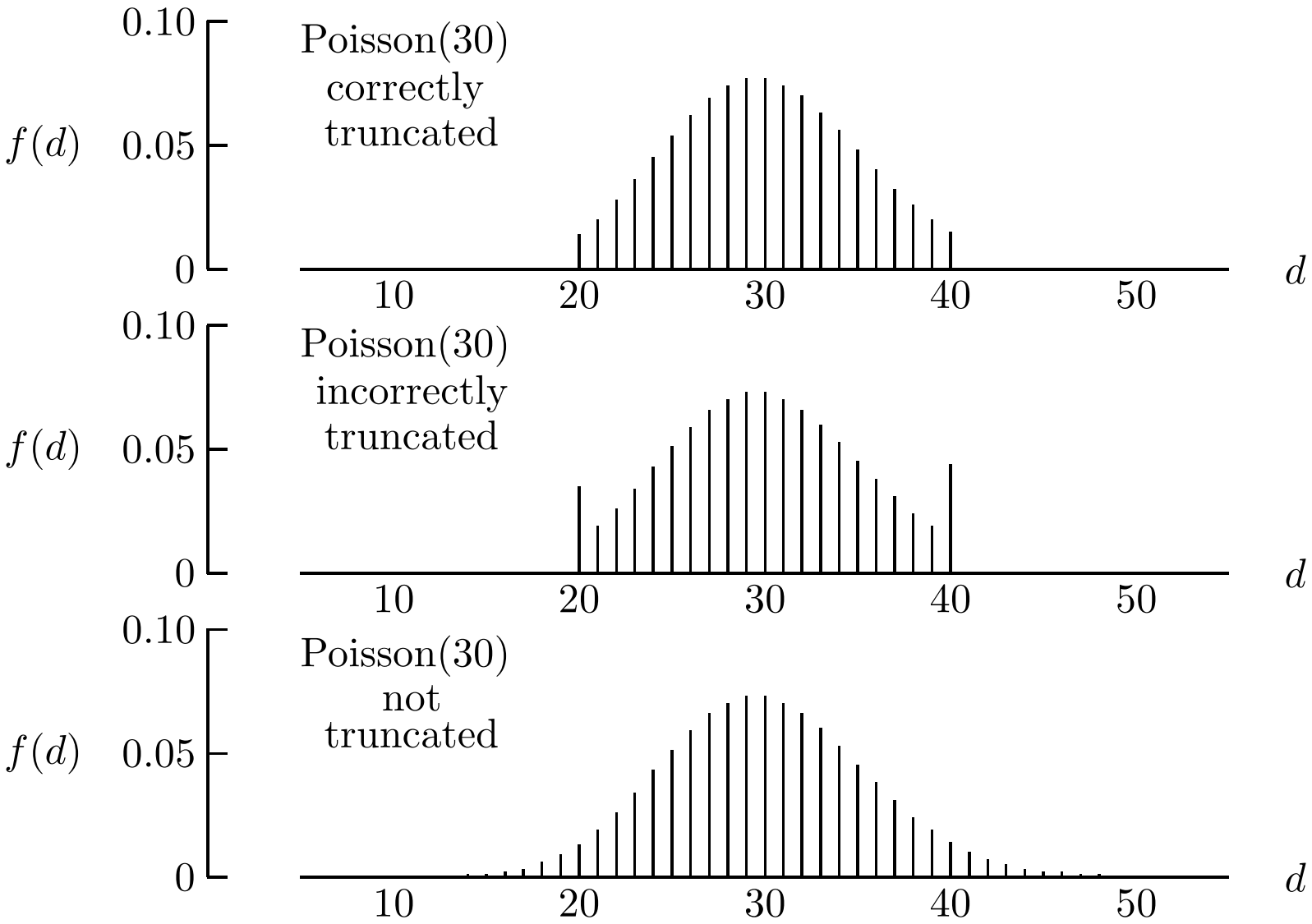
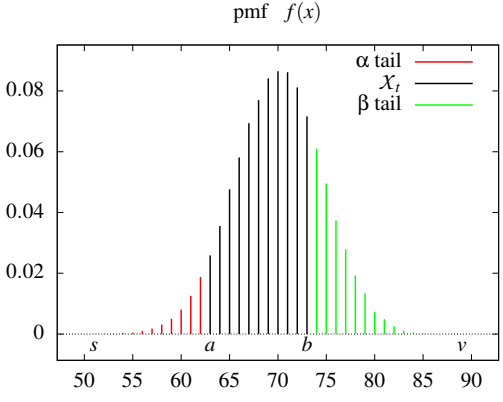


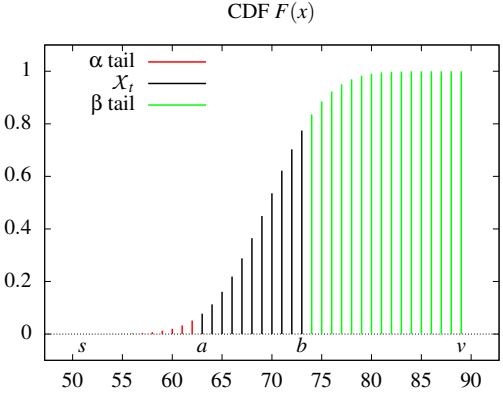
Illustration of pdfs



Truncation Geometry

From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ to $\mathcal{X}_t = \{a, \ldots, b\}$





Two cases:

i. Known $[a, b]$ sub-interval

ii. Known α and β tail probabilities

Two techniques:

a. **CDF modification**

b. **Constrained inversion**

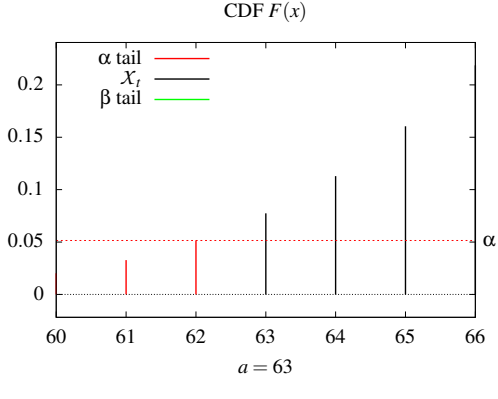
Truncation of Random Variables (variates)

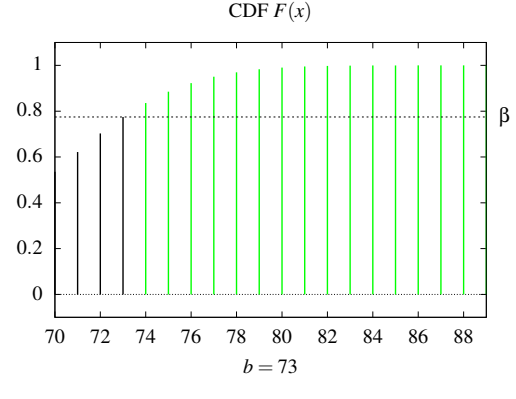
From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ **to** $\mathcal{X}_t = \{a, \ldots, b\}$

Two cases:

i. Known $[a,b]$ sub-interval, you need α and β

$$\alpha = Pr(X < a) = F(a-1) \qquad \beta = Pr(X > b) = 1 - F(b)$$





Truncation of Random Variables (variates)

From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ **to** $\mathcal{X}_t = \{a, \ldots, b\}$

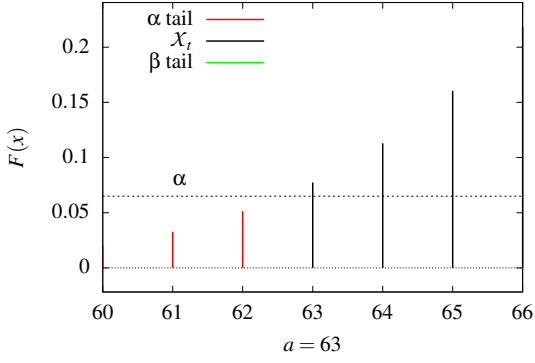
Two cases:

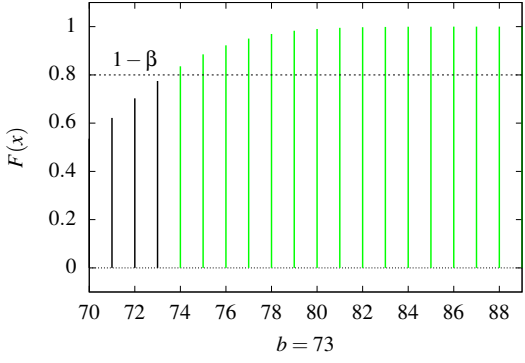
ii. Known α and β tail probabilities, you need to find a and b

$$a = F^*(\alpha) \qquad b = F^*(1-\beta)$$

Note these are approximations:

$$F(a-1) \leq \alpha \qquad 1-\beta < F(b)$$



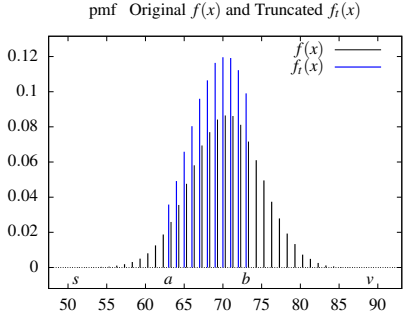


Truncation of Random Variables (variates)

From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ **to** $\mathcal{X}_t = \{a, \ldots, b\}$

Two techniques:

a. Truncation by **CDF modification** (needs a and b)



The total probability contained within $\mathcal{X}_t$ is

$$\sum_{x=a}^{x=b} f(x) = F(b) - F(a-1)$$

So define a new $f_t(x)$ from a to b that adds to 1:

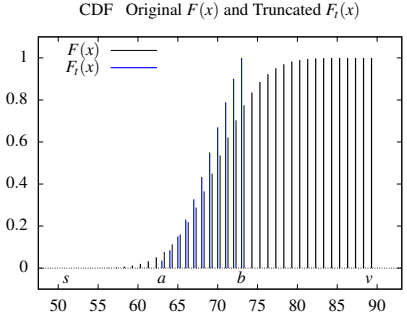
$$f_t(x) = \frac{f(x)}{F(b) - F(a-1)} \qquad x \in \mathcal{X}_t$$

Truncation of Random Variables (variates)

From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ **to** $\mathcal{X}_t = \{a, \ldots, b\}$

Two techniques:

a. Truncation by **CDF modification** (needs a and b)



The total probability contained within $\mathcal{X}_t$ is

$$\sum_{x=a}^{x=b} f(x) = F(b) - F(a-1)$$

So define a new $f_t(x)$ from a to b that adds to 1:

$$f_t(x) = \frac{f(x)}{F(b) - F(a-1)} \qquad x \in \mathcal{X}_t$$

$$F_t(x) = \sum_{d=a}^{d=x} f_t(d) = \frac{F(x) - F(a-1)}{F(b) - F(a-1)} \qquad x \in \mathcal{X}_t$$

eg $F_t(a+2) = \dfrac{f(a)}{F(b)-F(a-1)} + \dfrac{f(a+1)}{F(b)-F(a-1)} + \dfrac{f(a+2)}{F(b)-F(a-1)} = \dfrac{F(a+2)-F(a-1)}{F(b)-F(a-1)}$

Truncation of Random Variables (variates)

From $\mathcal{X} = \{s, s+1, \ldots, a, \ldots, b, b+1, \ldots, v\}$ **to** $\mathcal{X}_t = \{a, \ldots, b\}$

Two techniques:

b. Truncation by **constrained inversion** (needs α and β)

Not as complicated as CDF modification, you'll work through this with **an LGA question**.