M4 Approximations for Compound Distributions

General Insurance Modelling : Actuarial Modelling III [1]

Professor Benjamin Avanzi

06 May 2023

[1]References: Chapter 4 of Wuthrich (2022) | → latest slides

1. Algorithms for compound distributions
2. Approximations
3. ✠ Bringing it all together: SUVA case study

1 Algorithms for compound distributions

2 Approximations

3 ✠ Bringing it all together: SUVA case study

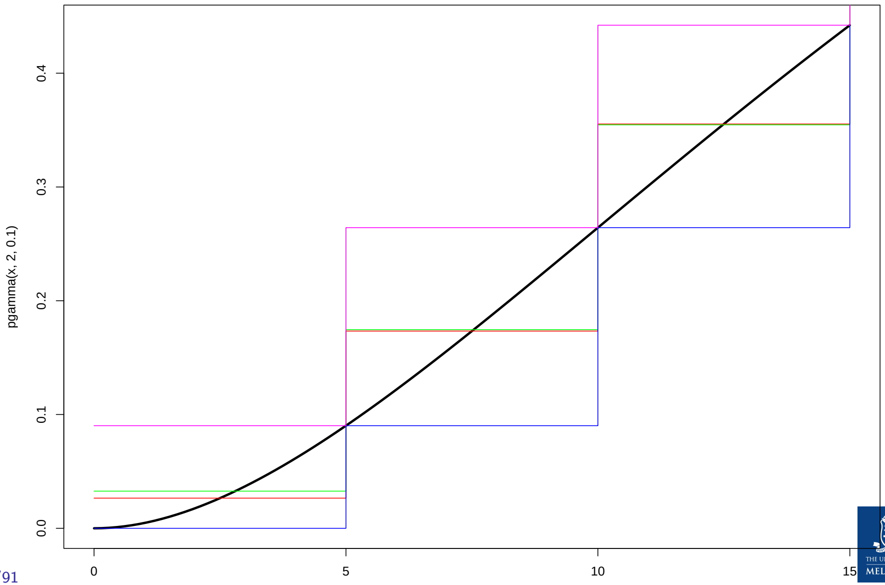
1. Algorithms for compound distributions
   - Preliminary: Discretisation
   - Panjer's recursion algorithm (MW 4.2.1)
   - Panjer's recursion in R
   - Panjer's recursion for compound Poisson
   - ✠ de Pril's recursion algorithm
   - ✠ Fast Fourier Transform

Preliminary: Discretisation

- In the algorithms considered in this module (and in real life..!), severity distributions are discrete.
- In Module 3 we discussed how to fit parametric distributions to real data, but these were continuous (which often allows to define a whole df easily with just a few parameters).
- This is not a restriction as the distribution of Y can easily be discretised to an arbirary level of precision.
- Different methods have different properties and are used for different purposes.
- A number of those methods are easily implemented, thanks to the function `discretise` of `actuar`.

There are three (four) methods, all covered by the R function `actuar::discretise`:

- Mass dispersal (`method="rounding"`): a first try to allocate weights in a rather 'unbiased' way. In this case, the true cdf passes through the midpoints of the discretisation intervals.
- Unbiased (`method="unbiased"`): this method matches the first moment of the discretised and the true distributions.
- Lower (`method="lower"`) and Upper (`method="upper"`) bounds, which sandwich the correct distribution between two arbitrarily close bounds. Here the idea is to intentionally underestimate and overestimate the distribution. However, the gap in-between can be chosen to be arbitrarily small.


pgamma(x, 2, 0.1)
0.0
0.1
0.2
0.3
0.4
0
5
10
15

in `discretise`, we will also need to specify:

- `step` $= h$: a currency span (say, \$10 or \$100)
- `from` $= 0$: the left limit of discretisation (we set this to 0 here, a natural bound for positive claims distributions)
- `to` $= mh$: the right limit of discretisation
- `cdf`: the distribution to discretise (in cumulative, cdf form—the `pfoo` function)

Method of rounding (mass dispersal)

- apportion the probability mass to a finite set of points $0, h, 2h, \ldots, mh$
- if the range of X is not finite, m has to be chosen in order to have a good representation of the right tail
- the span h should be carefully chosen as to be relatively small compared to the mean (beware of units!)
- Let g_j be the probability mass placed at jh, $j = 0, 1, \ldots, m$. We have

$$\begin{aligned} g_0 &= G(h/2) \\ g_j &= G(jh + h/2) - G(jh - h/2), \quad g = 1, 2, \ldots, m-1 \\ g_m &= 1 - G(mh - h/2) \end{aligned}$$

such that $\sum_{j=0}^{m} g_j = 1$.

(Note that if G is mixed and a mass happens to be at an interval threshold, it is customary—for conservativeness—to allocate it to the next interval, so that G would be understood as càglàd here.)

Unbiased method

- Let g_j be the probability mass placed at jh, $j = 0, 1, \ldots, m$. We have

$$\begin{aligned} g_0 &= 1 - E[\min(X, h)]/h \\ g_j &= (2E[\min(X, j)] - E[\min(X, j-h)] - E[\min(X, j+h)])/h, \\ &\quad j = 1, 2, \ldots, m-1 \\ g_m &= (E[\min(X, m)] - E[\min(X, m-h)])/h - (1 - G(m)) \end{aligned}$$

such that $\sum_{j=0}^{m} g_j = G(m)$. - Note that in this case the parameter `lev` must be specified. This is an expression that computes the limited expected value of the distribution corresponding to `cdf`. This is readily available in `actuar` for many distributions.

Method of lower and upper bounds

Define:

$$G((k+1)d) - G(kd) = \begin{cases} \Pr[Y_1^+ = (k+1)d] \equiv g_{k+1}^+ & \text{(shifted to the right)} \\ \Pr[Y_1^- = kd] \equiv g_k^- & \text{(shifted to the left)} \end{cases}$$

Note that from a stochastic dominance point of view

- Y_1^+ (`method="lower"`) overestimates (is larger than) Y_1, whereas
- Y_1^- (`method="upper"`) underestimates (is smaller than) Y_1.

This means that we can sandwich the **true** distribution:

$$\Pr[Y^- > y] \leq \Pr[Y > y] \leq \Pr[Y^+ > y], \text{ or}$$

$$\Pr[Y^- \leq y] \geq \Pr[Y \leq y] \geq \Pr[Y^+ \leq y] \Longleftrightarrow G^+(y) \leq G(y) \leq G^-(y).$$

This can be the best option, especially for patchwork distributions; see Example 4.12 in Wuthrich (2022).

Examples

Let us discretise

```
stp <- 5
final <- 80
# these are pmf's
gamma.discr.round <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "rounding")
gamma.discr.unbia <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "unbiased", lev = levgamma(x,
    2, 0.1))
gamma.discr.lower <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "lower")
gamma.discr.upper <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "upper")
```

```
length(gamma.discr.round)
```

```
## [1] 16
```

```
length(gamma.discr.unbia)
```

```
## [1] 17
```

```
length(gamma.discr.lower)
```

```
## [1] 17
```

```
length(gamma.discr.upper)
```

```
## [1] 16
```

Note the different lengths (either m or $m+1$), although they all start with g_0.

Furthermore, the masses don't add up to 1 necessarily:

```
sum(gamma.discr.round)
```

```
## [1] 0.996231
```

```
pgamma(final - stp/2, 2, 0.1)
```

```
## [1] 0.996231
```

```
sum(gamma.discr.unbia)
```

```
## [1] 0.9969808
```

```
sum(gamma.discr.lower)
```

```
## [1] 0.9969808
```

```
sum(gamma.discr.upper)
```

```
## [1] 0.9969808
```

```
pgamma(final, 2, 0.1)
```

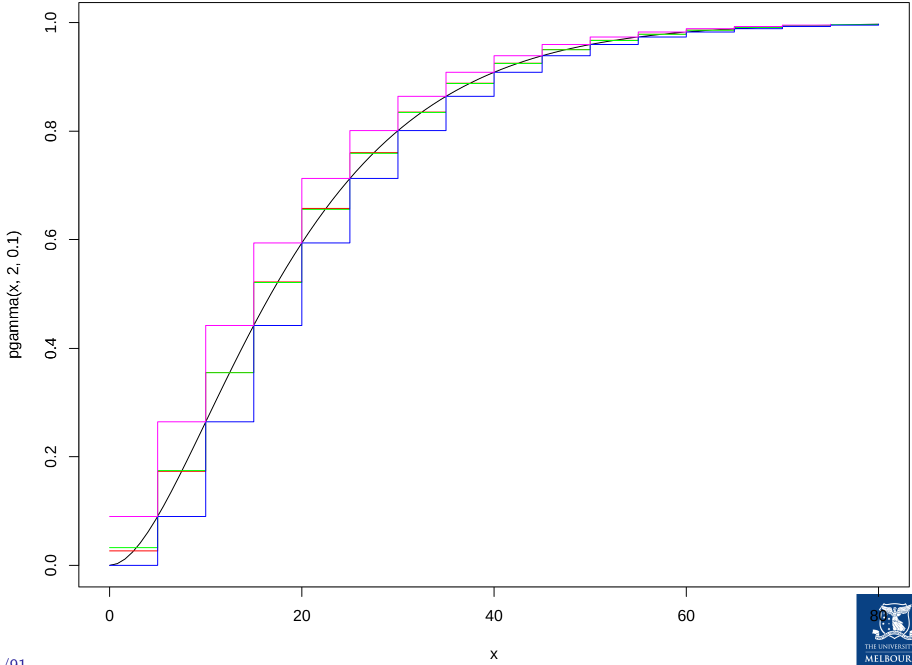
```
## [1] 0.9969808
```

This is because the algorithm does not adjust for the tail:

- `rounding` has $g_m = G(m + h/2) - G(m - h/2)$ rather than $g_m = 1 - G(m - h/2)$ so that $1 - G(m + h/2)$ is missing.
- `unbiased` has that correction $-(1 - G(m))$ (due to the fact that this mass does not correspond to this interval) which is then missing.
- `lower` has $g_0 = 0$ and then the mass $G(m)$ is apportioned to the end of the m intervals. $1 - G(m)$ remains unallocated.
- `upper` has its first allocation from $G(m)$ at 0, so has one less mass in the vector. $1 - G(m)$ remains unallocated.

In practice one would choose m sufficiently large so that these are issues are not present. If they are, then, an adjustment might be necessary, and `rounding` becomes the most natural one to use (just add the missing mass at g_m as per the definition above).

```r
# getting df's
gamma.discr.round.df <- cumsum(gamma.discr.round)
gamma.discr.unbia.df <- cumsum(gamma.discr.unbia)
gamma.discr.lower.df <- cumsum(gamma.discr.lower)
gamma.discr.upper.df <- cumsum(gamma.discr.upper)
# comparing discretisation techniques
curve(pgamma(x, 2, 0.1), from = 0, to = final)
lines((0:(final/stp - 1)) * stp, gamma.discr.round.df, type = "s",
  pch = 20, col = "red")
lines((0:(final/stp)) * stp, gamma.discr.unbia.df, type = "s",
  pch = 20, col = "green")
lines((0:(final/stp)) * stp, gamma.discr.lower.df, type = "s",
  pch = 20, col = "blue")
lines((0:(final/stp - 1)) * stp, gamma.discr.upper.df, type = "s",
  pch = 20, col = "magenta")
```


pgamma(x, 2, 0.1)
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
x

1 Algorithms for compound distributions
- Preliminary: Discretisation
- Panjer's recursion algorithm (MW 4.2.1)
- Panjer's recursion in R
- Panjer's recursion for compound Poisson
- ✠ de Pril's recursion algorithm
- ✠ Fast Fourier Transform

Panjer's recursion algorithm (MW 4.2.1)

- The remarkable property of the (a, b) class of (frequency) distributions allows us to develop a recursive method to get the distribution of S for discrete Y's.
- When Y is continuous, simply discretise its cdf first.
- The algorithm is very stable when N is Poisson and Negative Binomial, but less stable when N is Binomial.

Let S have a compound distribution on Y, where the following are mutually independent:

- N belongs to the (a, b) class of distributions;
- Y are identically distributed, non-negative and discrete.

We have then

$$f_S(s) = \frac{1}{1 - ag_0} \sum_{j=1}^{s} \left(a + b\frac{j}{s} \right) g_j f_S(s - j), \quad s = 1, 2, \ldots,$$

with starting value

$$f_S(0) = \begin{cases} \Pr[N = 0], & \text{if } g_0 = 0 \\ M_N[\ln g_0], & \text{if } g_0 > 0. \end{cases}$$

Note that if $g_y = 0$ for $y > y_{\max}$ then the upper bound of the sum can be reduced to $\min(s, y_{\max})$.

1 Algorithms for compound distributions
- Preliminary: Discretisation
- Panjer's recursion algorithm (MW 4.2.1)
- Panjer's recursion in R
- Panjer's recursion for compound Poisson
- ✠ de Pril's recursion algorithm
- ✠ Fast Fourier Transform

Panjer's recursion in R

Panjer's recursion can be performed using the `aggregateDist` function using the `method="recursive"`.

- The frequency distribution can be any of the $(a, b, 0)$ or $(a, b, 1)$ class of distributions (that is, with arbitrary masses at 0).
- The severity distribution must be discrete on $0, 1, \ldots, m$ for some monetary unit.

Important parameters include:

- `model.freq`: name of the distribution (e.g. `="poisson"`)
- `model.sev`: df of the discrete (-ised) distribution of Y
- `x.scale`: value of an amount 1 in the severity model (monetary unit)
- `maxit`: maximum number of iterations, which often needs to be increased (if it is too small then the df of S does not reach 1, leading to an error message)

Example

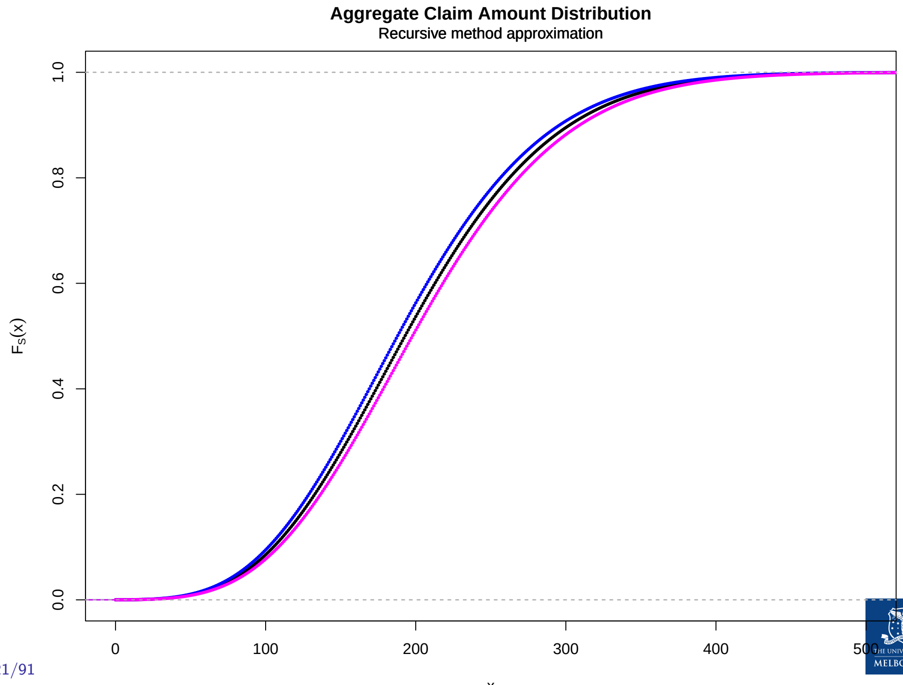
We want to calculate the distribution of

$$S = \sum_{i=1}^{N} Y_i,$$

where $N \sim \text{Poi}(10)$ and $Y_1 \sim \text{gamma}(2, 0.1)$.

```r
stp <- 1
final <- 200
# these are pmf's
gamma.discr.unbia <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "unbiased", lev = levgamma(x,
    2, 0.1))
gamma.discr.lower <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "lower")
gamma.discr.upper <- discretise(pgamma(x, 2, 0.1), from = 0,
  to = final, step = stp, method = "upper")
```

```
S.unbia.cdf <- aggregateDist(method = "recursive", model.freq = "poisson",
  lambda = 10, model.sev = gamma.discr.unbia, x.scale = stp,
  maxit = 1000)
S.lower.cdf <- aggregateDist(method = "recursive", model.freq = "poisson",
  lambda = 10, model.sev = gamma.discr.lower, x.scale = stp,
  maxit = 1000)
S.upper.cdf <- aggregateDist(method = "recursive", model.freq = "poisson",
  lambda = 10, model.sev = gamma.discr.upper, x.scale = stp,
  maxit = 1000)
plot(S.unbia.cdf, pch = 20, xlim = c(0, 500), col = "black",
  cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Aggregate Claim Amount Distribution
Recursive method approximation
$F_S(x)$
x
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

1. Algorithms for compound distributions
    - Preliminary: Discretisation
    - Panjer's recursion algorithm (MW 4.2.1)
    - Panjer's recursion in R
    - Panjer's recursion for compound Poisson
    - ✠ de Pril's recursion algorithm
    - ✠ Fast Fourier Transform

Panjer's recursion for compound Poisson

If $S \sim$ compound Poisson(λ, g_x) the algorithm reduces to

$$f_S(s) = \frac{\lambda}{s} \sum_{j=1}^{s} j\, g_j f_S(s-j)\,.$$

with starting value

$$f_S(0) = e^{\lambda(g_0 - 1)}$$

(whether g_0 is positive or not).

Example A 12.4.2 (recomputed)

Effectively, the recursion formula boils down to

$$f_S(s) = \frac{1}{s}\left[0.2 f_S(s-1) + 0.6 f_S(s-2) + 0.9 f_S(s-3)\right], \quad (\text{for } s > 2)$$

with starting value

$$f_S(0) = \Pr[N=0] = e^{-0.8} = 0.44933.$$

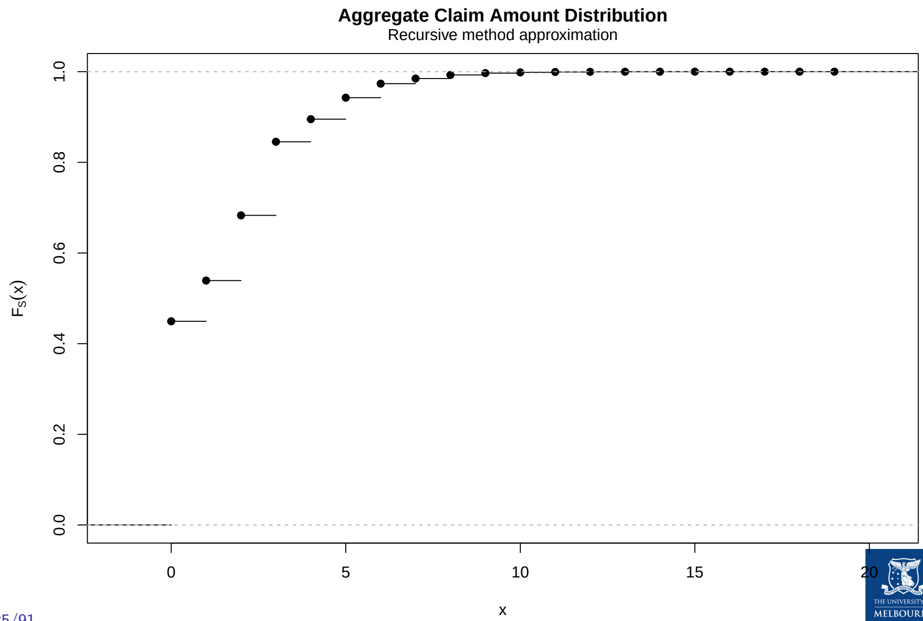
We have then

$$\begin{aligned} f_S(1) &= 0.2 f_S(0) = 0.2 e^{-0.8} = 0.089866 \\ f_S(2) &= \tfrac{1}{2}\left[0.2 f_S(1) + 0.6 f_S(0)\right] = 0.32 e^{-0.8} = 0.14379 \\ f_S(3) &= \tfrac{1}{3}\left[0.2 f_S(2) + 0.6 f_S(1) + 0.9 f_S(0)\right] = 0.3613 e^{-0.8} = 0.16236 \\ & \textit{etc} \ldots \end{aligned}$$

```r
fs <- aggregateDist(method = "recursive", model.freq = "poisson",
  lambda = 0.8, model.sev = c(0, 0.25, 0.375, 0.375), x.scale = 1)
diff(fs)[1:4]
```

```
## [1] 0.44932896 0.08986579 0.14378527 0.16235753
```

```
plot(fs)
```


Aggregate Claim Amount Distribution
Recursive method approximation
$F_S(x)$
x
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20

1 Algorithms for compound distributions

- Preliminary: Discretisation
- Panjer's recursion algorithm (MW 4.2.1)
- Panjer's recursion in R
- Panjer's recursion for compound Poisson
- ✠ de Pril's recursion algorithm
- ✠ Fast Fourier Transform

✠ de Pril's recursion algorithm

As a corollary, a recursion algorithm can be developed for the *n*-th convolution of a non-negative and discrete random variable Y with positive mass of probability at 0. Let g_k be its pmf.

First let us rewrite

$$\begin{aligned} M_Y(t) &= \sum_{k=0}^{\infty} g_k e^{tk} = g_0 + \sum_{k=1}^{\infty} g_k e^{tk} \\ &= g_0 + [1-g_0] \sum_{k=1}^{\infty} \frac{g_k}{1-g_0} e^{tk} \\ &= q + p m_{\widetilde{Y}}(t). \end{aligned}$$

where

- $q = g_0$,
- $p = 1 - g_0$, and
- the pmf of $\widetilde{Y}$ is: $\widetilde{g}_0 = 0$, $\widetilde{g}_k = \frac{g_k}{1-g_0}$, $k = 1, 2, \cdots$.

Thus

$$E\left[e^{t(Y_1+\ldots+Y_n)}\right] = (M_Y(t))^n = \left(q + pm_{\widetilde{Y}}(t)\right)^n,$$

which means that the n-th convolution of Y is compound Binomial with parameters $(m = n, p = 1 - g_0, \widetilde{G}(y))$, a member of the (a, b) class with

$$\begin{aligned} a &= -\tfrac{p}{1-p} \\ &= -\tfrac{1-g_0}{g_0}, \\ b &= \tfrac{(m+1)p}{1-p} \\ &= \tfrac{(n+1)(1-g_0)}{g_0}. \end{aligned}$$

Applying Panjer's algorithm yields

$$\begin{aligned} g_k^{*n} &= \frac{1}{1 - a\widetilde{g}_0} \sum_{j=1}^{k} \left(a + \frac{bj}{k} \right) \widetilde{g}_j g_{k-j}^{*n} \\ &= \frac{1}{g_0} \sum_{j=1}^{\min(k, y_{\max})} \left[(n+1)\frac{j}{k} - 1 \right] g_j g_{k-j}^{*n} \end{aligned}$$

with

$$g_0^{*n} = g_0^n.$$

Note that the g_j's are indeed the original ones.

Numerical example

Let $g_k = (1+k)/10$, $k = 0, 1, 2$ and 3. We have

$$g_k^{*2} = \sum_{j=1}^{\min(k,3)} \left[3\frac{j}{k} - 1\right](1+j)g_{k-j}^{*2} \text{ with } g_0^{*2} = [g_0]^2$$

$$\begin{aligned}
g_0^{*2} &= (0.1)^2 = 0.01 \\
g_1^{*2} &= [3/1-1]\cdot 2\cdot 0.01 = 0.04 \\
g_2^{*2} &= [3/2-1]\cdot 2\cdot 0.04 + [6/2-1]\cdot 3\cdot 0.01 = 0.10 \\
g_3^{*2} &= [3/3-1]\cdot 2\cdot 0.10 + [6/3-1]\cdot 3\cdot 0.04 + [9/3-1]\cdot 4\cdot 0.01 = 0.20 \\
g_4^{*2} &= [3/4-1]\cdot 2\cdot 0.20 + [6/4-1]\cdot 3\cdot 0.10 + [9/4-1]\cdot 4\cdot 0.04 = 0.25 \\
g_5^{*2} &= [3/5-1]\cdot 2\cdot 0.25 + [6/5-1]\cdot 3\cdot 0.20 + [9/5-1]\cdot 4\cdot 0.10 = 0.24 \\
g_6^{*2} &= [3/6-1]\cdot 2\cdot 0.24 + [6/6-1]\cdot 3\cdot 0.25 + [9/6-1]\cdot 4\cdot 0.20 = 0.16
\end{aligned}$$

brute force convolution:

```
pmf <- c(1/10, 2/10, 3/10, 4/10)
range <- length(pmf)
range2 <- (range - 1) * 2 + 1
pmf <- c(pmf, rep(0, (range2 - range)))
ps2.conv <- c()
for (i in 1:range2) {
  ps2.conv <- c(ps2.conv, sum(pmf[1:i] * pmf[i:1]))
}
ps2.conv
```

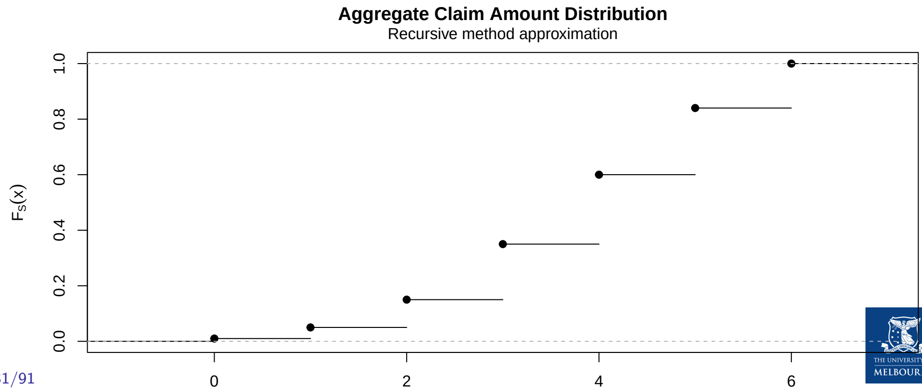
```
## [1] 0.01 0.04 0.10 0.20 0.25 0.24 0.16
```

Using de Pril's algorithm

```r
ps2 <- aggregateDist(method = "recursive", model.freq = "binomial",
  size = 2, prob = 1 - 1/10, model.sev = c(0, 2/10, 3/10, 4/10)/(1 -
    1/10), x.scale = 1, xlim = c(0, 6))
diff(ps2)
```

```
## [1] 0.01 0.04 0.10 0.20 0.25 0.24 0.16
```

```r
plot(ps2)
```



1 Algorithms for compound distributions
- Preliminary: Discretisation
- Panjer's recursion algorithm (MW 4.2.1)
- Panjer's recursion in R
- Panjer's recursion for compound Poisson
- ✠ de Pril's recursion algorithm
- ✠ Fast Fourier Transform

✠ Fast Fourier Transform

Not covered this year.

1 Algorithms for compound distributions

2 Approximations

3 ✠ Bringing it all together: SUVA case study

Approximations

Possible motivations:

- It is not possible to compute the distribution of S:
  - no detailed data is available except for the moments of S
  - technical issues (impossible to fit a tractable model to data)
- The risk of having a sophisticated—but wrong—model is too high
  - Only limited data is available to fit the model
  - It may be argued that the approximation is very accurate
- A quick approximation is needed.
- A higher level of accuracy is not required (does not justify the resources necessary to calculate an exact probability)

Note: let $\varsigma_S \equiv \gamma_1$ and $\gamma_2(S) \equiv \gamma_2$ in this section.

2 Approximations

- Approximations assuming a symmetrical distribution
- Approximations using a skewed distribution
- ✠ Edgeworth's approximation (MW 4.1.3)
- ✠ Approximating the Individual Model with the Collective Model

The Normal approximation (MW 4.1.1)

- The Central Limit Theorem suggests that

$$\begin{aligned}\Pr\left[S \leq s\right] &= = \Pr\left[\frac{S - E[S]}{\sqrt{Var(S)}} \leq \frac{s - E[S]}{\sqrt{Var(S)}}\right] \\ &\approx \Pr\left[Z \leq \frac{s - E[S]}{\sqrt{Var(S)}}\right] = \Phi\left(\frac{s - E[S]}{\sqrt{Var(S)}}\right),\end{aligned}$$

where Z is a standard Normal random variable and where $\Phi\left(\cdot\right)$ denotes its cdf

- The classical CLT approximation holds for a fixed number of claims. Here, we typically have a random number of claims.

- Assuming $S \sim \text{CompPoi}(\lambda v, G)$, Theorem 4.1 of Wuthrich (2022) states that
$$\frac{S - \lambda v E[Y]}{\sqrt{\lambda v E[Y^2]}} \Longrightarrow \mathcal{N}(0,1) \quad \text{as } v \to \infty.$$
- This leads to the approximation
$$\Pr[S \le s] = \Pr\left[\frac{S - \lambda v E[Y]}{\sqrt{\lambda v E[Y^2]}} \le \frac{s - \lambda v E[Y]}{\sqrt{\lambda v E[Y^2]}}\right] \approx \Phi\left(\frac{s - \lambda v E[Y]}{\sqrt{\lambda v E[Y^2]}}\right).$$
- Note that this holds only when G has a finite second moment, which suggests that this should be used only for S_{sc}.

Note that this approximation performs poorly

- individual model: for small n (CLT not effective)
- collective model: for small λv (compound Poisson) and small r (compound negative binomial)
- for highly skewed distributions

Normal Power

Two levels:

- **NP1**: this is the CLT approximation
- **NP2**: the same idea, but with a correction taking the skewness into account. We have

  $$\Pr\left[\frac{S-E[S]}{\sqrt{Var(S)}} \leq s\right] \approx \Phi\left(z\right)$$

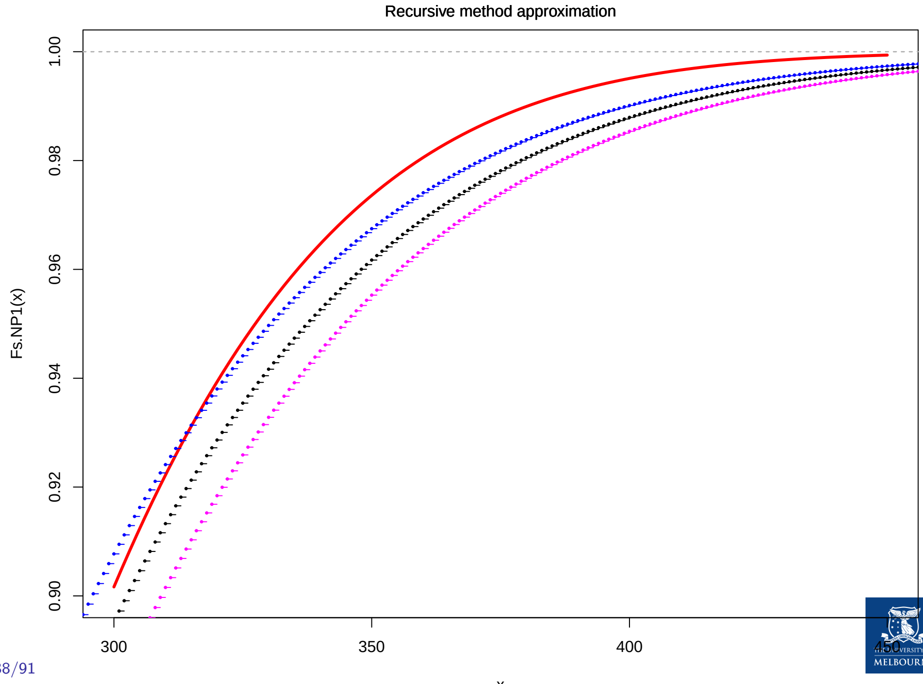
  with

  $$s(z) = z + \frac{\gamma_1}{6}\left(z^2 - 1\right) \quad \text{or} \quad z(s) = \sqrt{\frac{9}{\gamma_1^2} + \frac{6s}{\gamma_1} + 1} - \frac{3}{\gamma_1}$$

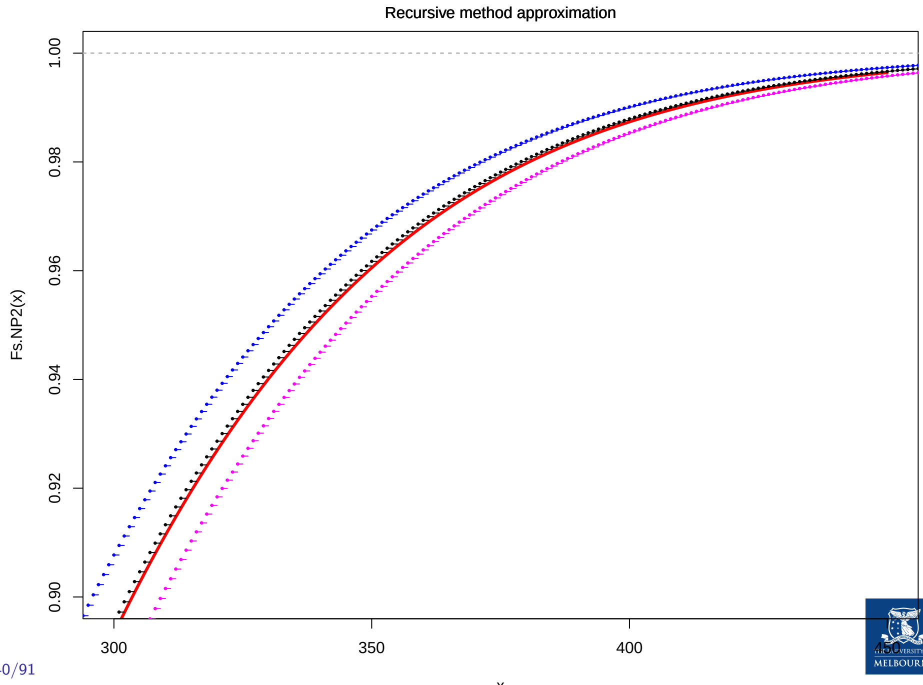
  NP2 is effective for $S > E[S] + \sqrt{Var(S)}$.

- NP1 and NP2 approximations are available with the aggregateDist function
- For instance here use on a compound Poisson random variable with gamma distributed claims:

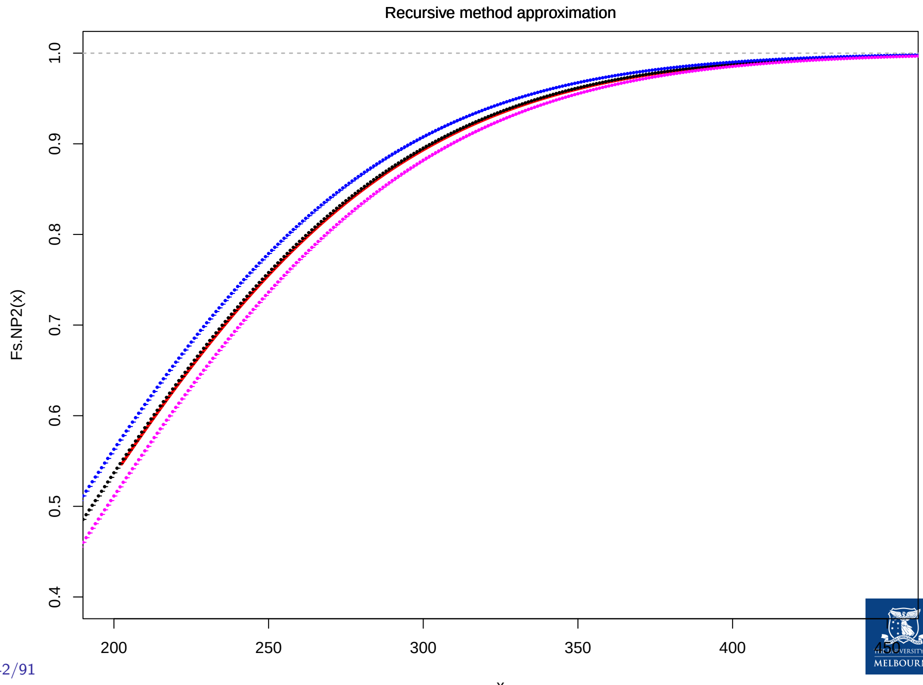
```r
gamma <- 2
c <- 0.1
lambda <- 10
moments <- c(lambda * gamma/c, lambda * (gamma/c^2 + (gamma/c)^2),
  gamma * (gamma + 1) * (gamma + 2)/c^3/lambda^(1/2)/(gamma/c^2 +
    (gamma/c)^2)^(3/2))
Fs.NP1 <- aggregateDist("normal", moments = moments)
curve(Fs.NP1, xlim = c(300, 450), ylim = c(0.9, 1), col = "red",
  lwd = 3)
lines(S.unbia.cdf, pch = 20, cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Recursive method approximation
Fs.NP1(x)
1.00
0.98
0.96
0.94
0.92
0.90
300
350
400
450
x

```r
Fs.NP2 <- aggregateDist("npower", moments = moments)
curve(Fs.NP2, xlim = c(300, 450), ylim = c(0.9, 1), col = "red",
  lwd = 3)
lines(S.unbia.cdf, pch = 20, cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Recursive method approximation
Fs.NP2(x)
1.00
0.98
0.96
0.94
0.92
0.90
300
350
400
450

```r
curve(Fs.NP2, xlim = c(200, 450), ylim = c(0.4, 1), col = "red",
  lwd = 3)
lines(S.unbia.cdf, pch = 20, cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Recursive method approximation
Fs.NP2(x)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
200
250
300
350
400
450

2 Approximations

- Approximations assuming a symmetrical distribution
- Approximations using a skewed distribution
- ✠ Edgeworth's approximation (MW 4.1.3)
- ✠ Approximating the Individual Model with the Collective Model

The translated gamma and LN approximations (MW 4.1.2)

Idea: Rather than correcting the CLT approximation, we use here a distribution that is naturally (positively) skewed, and use the third moment (skewness), which will be fitted thanks to a translation of the distribution (a third parameter to match the third moment).

We have then

$$X = k + Z, \text{ where } Z \sim \begin{cases} \Gamma(\gamma, c), \text{ or} \\ \text{LN}(\mu, \sigma^2) \end{cases}.$$

Then match expected value, variance and skewness.

Note: k appears only in the expected value. It allows to shift the distribution without affecting its scale (and central moments).

As an example, fitting a translated gamma (k, γ, c) to a compPois$(\lambda v, G(y))$ $S \equiv X$ boils down to solving

$$\begin{aligned} E[S] = \lambda v E[Y] &= k + \gamma / c; \\ Var(S) = \lambda v E[Y^2] &= \gamma / c^2; \\ \varsigma_S = \frac{E[Y^3]}{(\lambda v)^{1/2} E[Y^2]^{3/2}} &= 2\gamma^{-1/2}; \end{aligned}$$

such that

$$\begin{aligned} \gamma &= \left(\frac{2(\lambda v)^{1/2} E[Y^2]^{3/2}}{E[Y^3]} \right)^2; \\ c &= \left(\frac{\gamma}{\lambda v E[Y^2]} \right)^{1/2}; \\ k &= \lambda v E[Y] - \frac{\gamma}{c}. \end{aligned}$$

Properties of the translated gamma approximation:

- If $\gamma \to \infty$, $c \to \infty$ and $k \to -\infty$ such that

$$E[S] = k + \frac{\gamma}{c} = \mu \text{ (constant)} \quad \text{and} \quad \frac{\gamma}{c^2} = \sigma^2 \text{ (constant)}$$

  then the translated gamma converges to the Normal (μ, σ^2) distribution. In this sense, this approximation is a **generalisation of the CLT approximation**.
- The distribution of a **compound binomial random variable approaches the gamma distribution** if the expected number of claims is large and the claim amount distribution has relatively small dispersion (see Theorem 12.A.1 in the Appendix of Chapter 12 of Bowers et al. (1997)).
- This approximation should be used for distributions with a positive skewness only.

```
# matching moments:
gamma <- (2/moments[3])^2
c <- (gamma/moments[2])^(1/2)
k <- moments[1] - gamma/c
# check:
c(gamma/c + k, moments[1])
```

```
## [1] 200 200
```

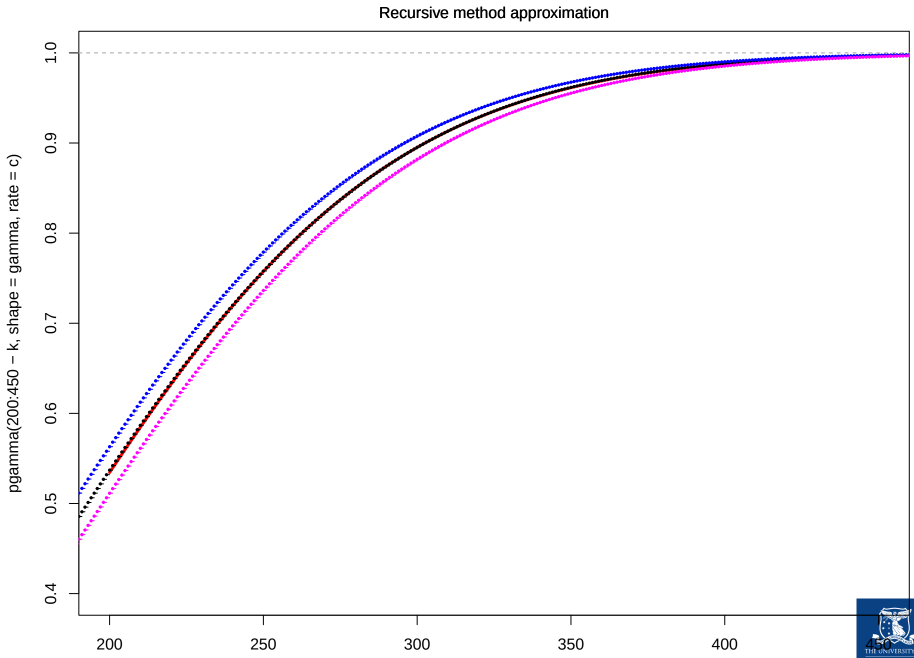
```
c((gamma/c^2), moments[2])
```

```
## [1] 6000 6000
```

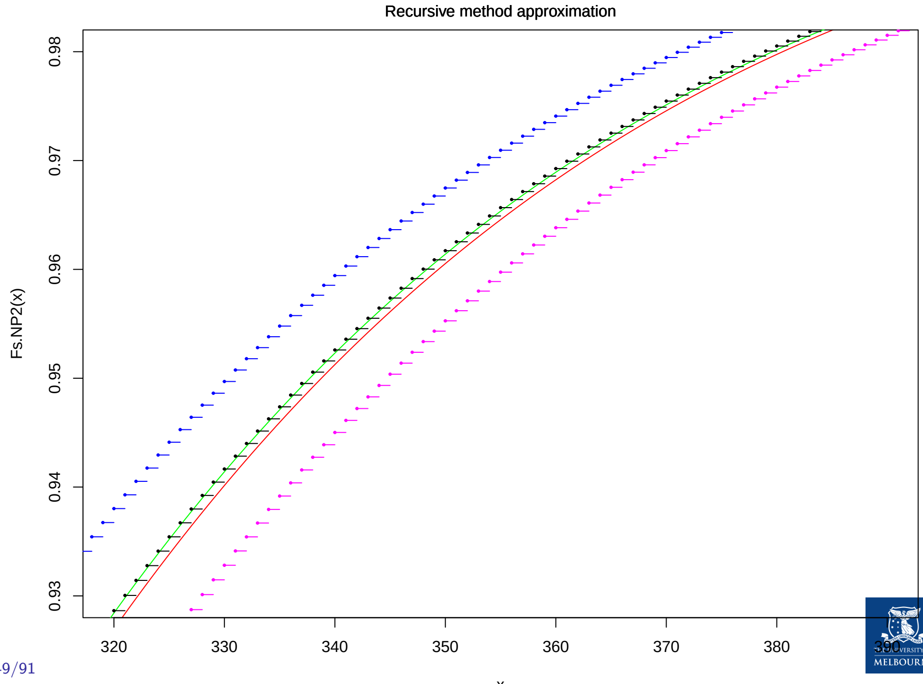
```
c(2 * gamma^(-1/2), moments[3])
```

```
## [1] 0.5163978 0.5163978
```

```
plot(200:450, pgamma(200:450 - k, shape = gamma, rate = c), xlim = c(200,
  450), ylim = c(0.4, 1), col = "red", lwd = 3, type = "l",
  xlab = c(""))
lines(S.unbia.cdf, pch = 20, cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Recursive method approximation
pgamma(200:450 − k, shape = gamma, rate = c)
0.4
0.5
0.6
0.7
0.8
0.9
1.0
200
250
300
350
400
450

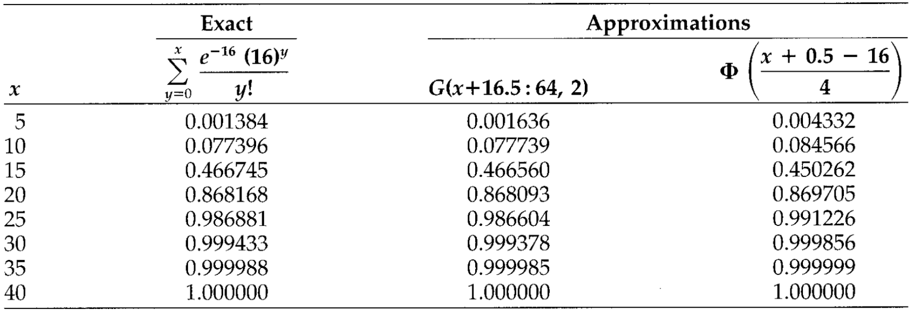
```
curve(Fs.NP2, xlim = c(320, 390), ylim = c(0.93, 0.98), col = "red",
  lwd = 1)
lines(300:450, pgamma(300:450 - k, shape = gamma, rate = c),
  col = "green", lwd = 1, type = "l")
lines(S.unbia.cdf, pch = 20, cex = 0.5)
lines(S.upper.cdf, pch = 20, col = "blue", cex = 0.5)
lines(S.lower.cdf, pch = 20, col = "magenta", cex = 0.5)
```


Recursive method approximation
Fs.NP2(x)
0.93
0.94
0.95
0.96
0.97
0.98
320
330
340
350
360
370
380
390
x

When approximating a discrete distribution with a continuous one, a half-integer discontinuity correction needs to be applied. Let $S \sim \text{Poisson}(\lambda = 16)$. Possible approximations are

- Translated gamma with $\gamma = 64$, $c = 2$ and $k = -16$.
- CLT with $E[S] = Var(S) = 16$

Results are below:



	Exact	**Approximations**	
x	$\sum_{y=0}^{x} \frac{e^{-16}(16)^y}{y!}$	$G(x+16.5:64,\ 2)$	$\Phi\left(\frac{x+0.5-16}{4}\right)$
5	0.001384	0.001636	0.004332
10	0.077396	0.077739	0.084566
15	0.466745	0.466560	0.450262
20	0.868168	0.868093	0.869705
25	0.986881	0.986604	0.991226
30	0.999433	0.999378	0.999856
35	0.999988	0.999985	0.999999
40	1.000000	1.000000	1.000000

Note that the discretisation techniques take this into account.

2 Approximations

- Approximations assuming a symmetrical distribution
- Approximations using a skewed distribution
- ✠ Edgeworth's approximation (MW 4.1.3)
- ✠ Approximating the Individual Model with the Collective Model

✠ Edgeworth's approximation (MW 4.1.3)

Let

$$\Phi^{(k)}(x) = \frac{d^k}{dx^k}\Phi(x).$$

We have then (each term improves the accuracy)

$$\Pr\left[\frac{S - E[S]}{\sqrt{Var(S)}} \leq z\right] \approx \Phi(z) - \frac{\gamma_1}{6}\Phi^{(3)}(z) + \frac{\gamma_2}{24}\Phi^{(4)}(z) + \frac{\gamma_1^2}{72}\Phi^{(6)}(z).$$

Note

$$\begin{aligned}
\Phi^{(1)}(x) &= \varphi(x) = \tfrac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2} \\
\Phi^{(2)}(x) &= -x\varphi(x) \\
\Phi^{(3)}(x) &= (x^2-1)\varphi(x) \\
&\text{etc}\ldots
\end{aligned}$$

Problem is: it sometimes leads to negative probabilities. . .

2 Approximations

- Approximations assuming a symmetrical distribution
- Approximations using a skewed distribution
- ✠ Edgeworth's approximation (MW 4.1.3)
- ✠ Approximating the Individual Model with the Collective Model

✠ Approximating the Individual Model with the Collective Model

Mainly motivated by the properties of compound distributions, in particular the compound Poisson:

- aggregation
- recursive methods

We want to approximate $\widetilde{S}$ (IRM) by S (CRM):

$$\widetilde{S} = \sum_{i=1}^{n} I_i b_i \approx S = \sum_{i=1}^{n} N_i b_i,$$

where $N_i \sim \text{Poisson}(\lambda_i)$ represents the number of claims of amount b_i. S is then compound Poisson with parameters

$$\lambda = \sum_{i=1}^{n} \lambda_i \quad \text{and} \quad P(x) = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda} I_{[b_i,\infty)}(x).$$

Two possible assumptions for λ_i

- $\lambda_i = E[I_i] = q_i$: the canonical collective model
  - expected number of claims of size b_i will be the same between the two models
  - also implies equal expected claim numbers in both $\widetilde{S}$ and S
  - finally

$$\begin{aligned} E\left(\widetilde{S}\right) = \textstyle\sum_{i=1}^{n} q_i b_i &= \lambda \textstyle\sum_{i=1}^{n} \frac{q_i}{\lambda} b_i = E\left(S\right), \text{ but} \\ Var\left(\widetilde{S}\right) = \textstyle\sum_{i=1}^{n} q_i \left(1 - q_i\right) b_i^2 &< \textstyle\sum_{i=1}^{n} q_i b_i^2 = Var\left(S\right). \end{aligned}$$

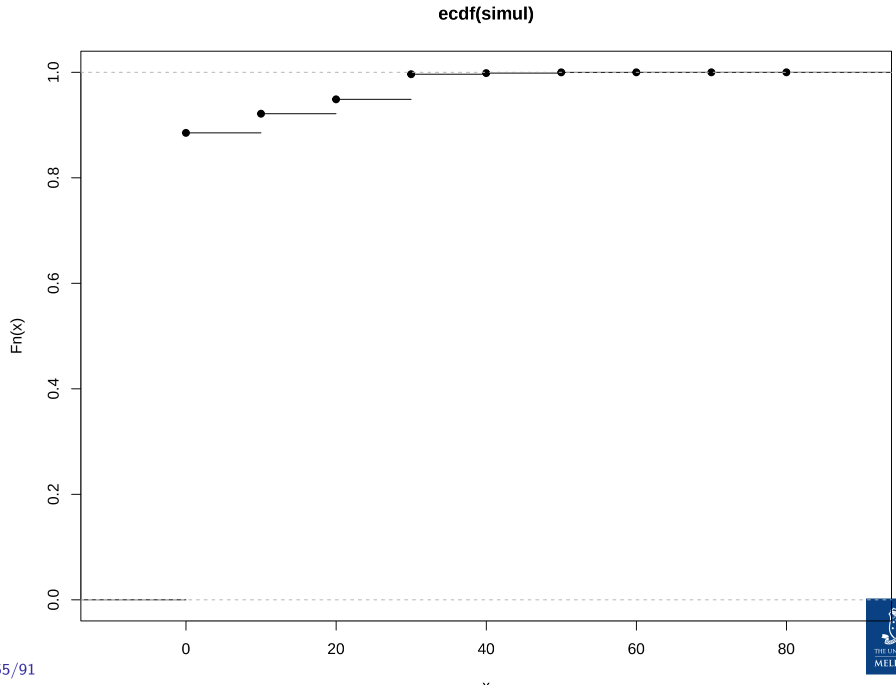
- $\lambda_i = -\ln(1 - q_i) > q_i$
  - $\Pr[S = 0] = \Pr[\widetilde{S} = 0]$

Example

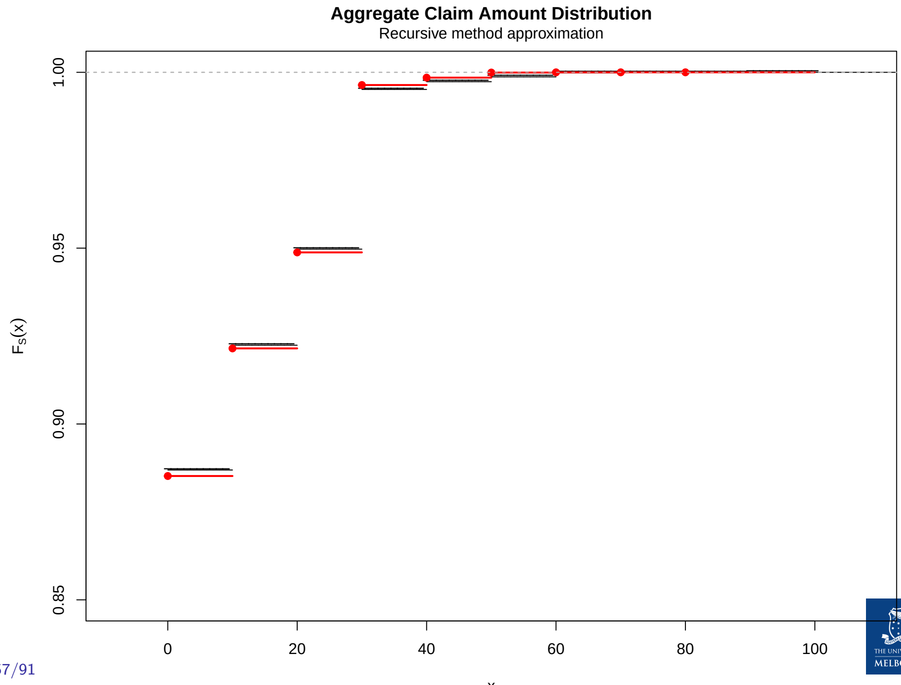
```
qi <- c(0.01, 0.02, 0.03, 0.01, 0.05)
bi <- c(10, 20, 10, 20, 30)
set.seed(10052021)
nsim <- 1e+07
simulated <- cbind(c(rbinom(nsim, 1, qi[1]) * bi[1]), c(rbinom(nsim,
  1, qi[2]) * bi[2]), c(rbinom(nsim, 1, qi[3]) * bi[3]), c(rbinom(nsim,
  1, qi[4]) * bi[4]), c(rbinom(nsim, 1, qi[5]) * bi[5]))
simul <- rowSums(simulated)
mean(simul)
```

```
## [1] 2.497127
```

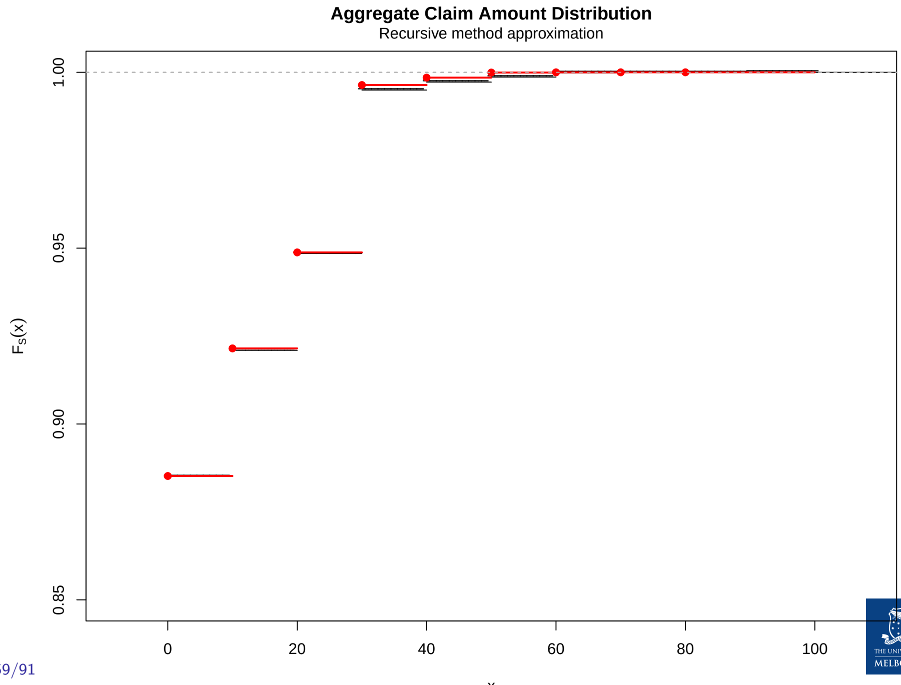
```
plot(ecdf(simul))
```


ecdf(simul)
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80

```r
lambdai.1 <- qi
lambda.1 <- sum(lambdai.1)
pmf.1 <- c(rep(0, 10), (lambdai.1[1] + lambdai.1[3])/lambda.1,
  rep(0, 9), (lambdai.1[2] + lambdai.1[4])/lambda.1, rep(0,
    9), lambdai.1[5]/lambda.1)
fs.1 <- aggregateDist("recursive", model.freq = "poisson", model.sev = pmf.
  lambda = lambda.1)
plot(fs.1, ylim = c(0.85, 1), pch = "-")
lines(ecdf(simul), col = "red", lwd = 2)
```


Aggregate Claim Amount Distribution
Recursive method approximation
$F_S(x)$
0.85
0.90
0.95
1.00
0
20
40
60
80
100

```
lambdai.2 <- -log(1 - qi)
lambda.2 <- sum(lambdai.2)
pmf.2 <- c(rep(0, 10), (lambdai.2[1] + lambdai.2[3])/lambda.2,
  rep(0, 9), (lambdai.2[2] + lambdai.2[4])/lambda.2, rep(0,
    9), lambdai.2[5]/lambda.2)
fs.2 <- aggregateDist("recursive", model.freq = "poisson", model.sev = pmf.
  lambda = lambda.2)
plot(fs.2, ylim = c(0.85, 1), pch = "-")
lines(ecdf(simul), col = "red", lwd = 2)
```


Aggregate Claim Amount Distribution
Recursive method approximation
$F_S(x)$
1.00
0.95
0.90
0.85
0
20
40
60
80
100
x

1 Algorithms for compound distributions

2 Approximations

3 ✠ Bringing it all together: SUVA case study

✠ Bringing it all together: SUVA case study

- We come back to the SUVA data; see Avanzi, Cassar, and Wong (2011).
- The right tail of the medical costs will be best modelled using an Extreme Value distribution (see Module 6), but this will result in a poor fit in the left tail.
- As a consequence we will fit the SUVA medical costs using two layers, S_{sc} and S_{lc}. If S is one medical cost claim (which can be scaled up easily), then

$$S = S_{\text{sc}} + S_{\text{lc}} = \sum_{i=1}^{N_{\text{sc}}} X_{i,\text{sc}} + \sum_{j=1}^{N_{\text{lc}}} X_{j,\text{lc}},$$

  where $N_{\text{sc}} \sim \text{Poi}(\lambda_{\text{sc}})$ and $N_{\text{lc}} \sim \text{Poi}(\lambda_{\text{lc}})$, all mutually independent with the X_{sc} and X_{lc}.

- The small claims layer can be tackled in different ways:
  - use the empirical distribution for X_{sc} and get the distribution of S_{sc} with Panjer. This is OK if the data is smooth enough and we don't necessarily want the smoothing effect of a parametric distribution.
  - use an approximation such as translated gamma or normal power on S_{sc}. This has the potential of smoothing out a coarse empirical distribution.
- The large claims layer fits X_{lc} using EVT and then uses Panjer to get the distribution of S_{lc}.
  - A threshold of $10,000 is chosen; see Module 6 for a justification.

3 ✠ Bringing it all together: SUVA case study
- Preliminaries
- Split claims
- Modelling the small claims
- Modelling the large claims
- Getting the overall distribution

Preliminaries

Basic parameters:

```
M <- 10000
data.scale <- 100
```

Loading data:

```
SUVA <- read_excel("SUVA.xls")
as_tibble(SUVA)
```

medcosts	dailyallow
407	0
12591	13742
269	0
142	0
175	0
298	839
47	0
59	0
191	7446
159	0

3 ✠ Bringing it all together: SUVA case study
- Preliminaries
- Split claims
- Modelling the small claims
- Modelling the large claims
- Getting the overall distribution

Split claims

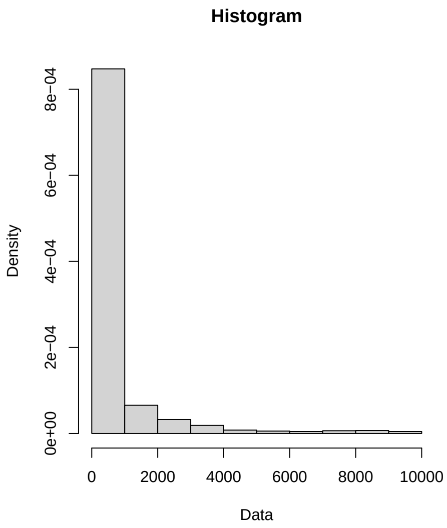
```
Xsc <- SUVA$medcosts[SUVA$medcosts <= M]
Xsc <- Xsc[Xsc > 0]
Xlc <- SUVA$medcosts[SUVA$medcosts > M]
lambda.sc <- length(Xsc)/(length(Xsc) + length(Xlc))
lambda.lc <- length(Xlc)/(length(Xsc) + length(Xlc))
c(lambda.sc, lambda.lc)
```

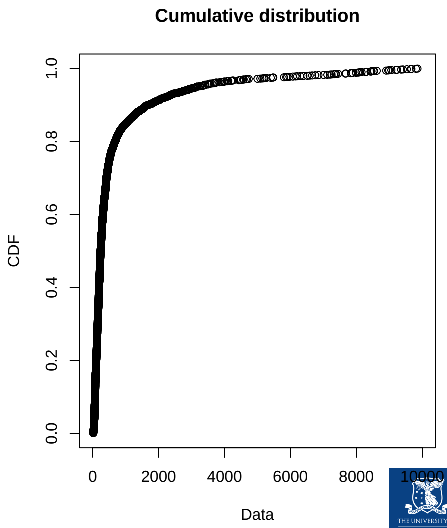
```
## [1] 0.97109827 0.02890173
```

3 ✠ Bringing it all together: SUVA case study
- Preliminaries
- Split claims
- Modelling the small claims
- Modelling the large claims
- Getting the overall distribution

Modelling the small claims

```
plotdist(Xsc)
```

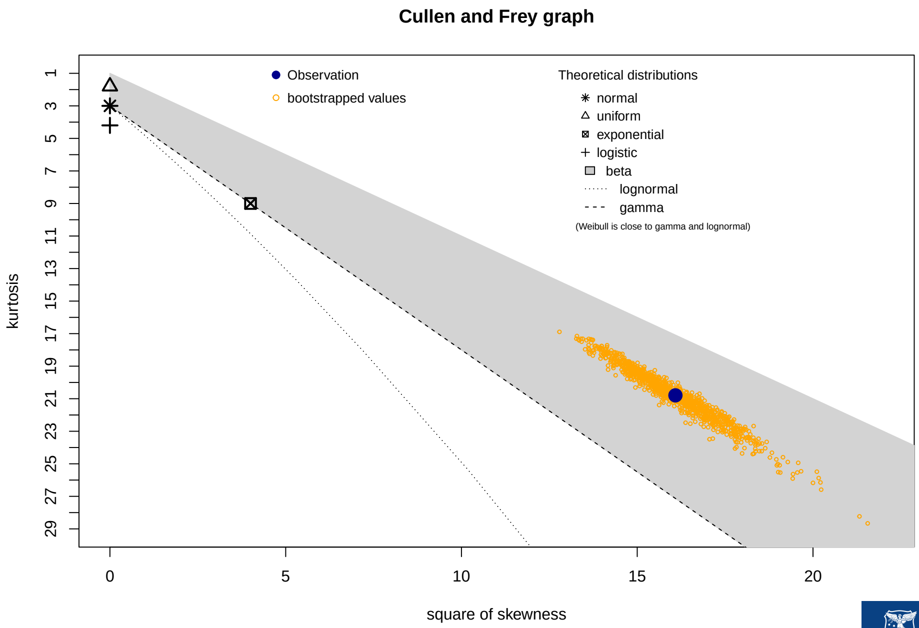




```
Xsc.cullen <- descdist(Xsc, boot = 1000)
```

```
Xsc.cullen
```

```
## summary statistics
## ------
## min:  15   max:  9854
## median:  240
## mean:  699.1978
## estimated sd:  1390.096
## estimated skewness:  4.010796
## estimated kurtosis:  20.79387
```

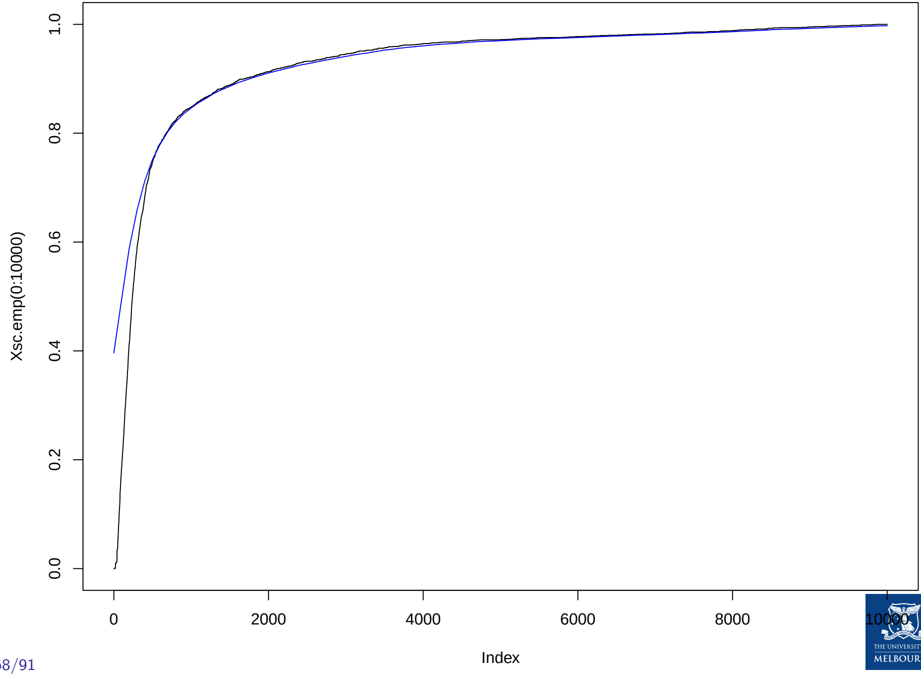

Cullen and Frey graph
● Observation
○ bootstrapped values
Theoretical distributions
✳ normal
△ uniform
⊠ exponential
+ logistic
▨ beta
······ lognormal
- - - gamma
(Weibull is close to gamma and lognormal)
kurtosis
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
square of skewness
0
5
10
15
20

S_{sc} using the empirical distribution of X_{sc} and Panjer

```r
Xsc.emp <- ecdf(Xsc)
Xsc.emp.pmf <- discretize(Xsc.emp, from = 0, to = M, step = data.scale,
  method = "rounding")
sum(Xsc.emp.pmf)
```

```
## [1] 1
```

```r
# and this the cdf
Xsc.emp.df <- cumsum(Xsc.emp.pmf)
# Panjer
Ssc.emp.df <- aggregateDist(method = "recursive", model.freq = "poisson",
  model.sev = Xsc.emp.pmf, lambda = lambda.sc, x.scale = data.scale,
  maxit = 1000)
# Ssc pmf
Ssc.emp.pmf <- diff(Ssc.emp.df)
# plotting the fit
plot(Xsc.emp(0:10000), type = "l", col = "black")
lines((0:(M/data.scale) * data.scale), Ssc.emp.df((0:(M/data.scale) *
  data.scale)), pch = "-", col = "blue")
```



Xsc.emp(0:10000)
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
Index

S_{sc} using an translated gamma approximation

```
# Moments of Xsc:
moments <- c(Xsc.cullen$mean, Xsc.cullen$sd^2, Xsc.cullen$skewness)
# getting parameters of translated gamma:
gamma <- (2/moments[3])^2
c <- (gamma/moments[2])^(1/2)
k <- moments[1] - gamma/c
# checking moments match:
c(gamma/c + k, moments[1])
```

```
## [1] 699.1978 699.1978
```

```
c((gamma/c^2), moments[2])
```

```
## [1] 1932366 1932366
```

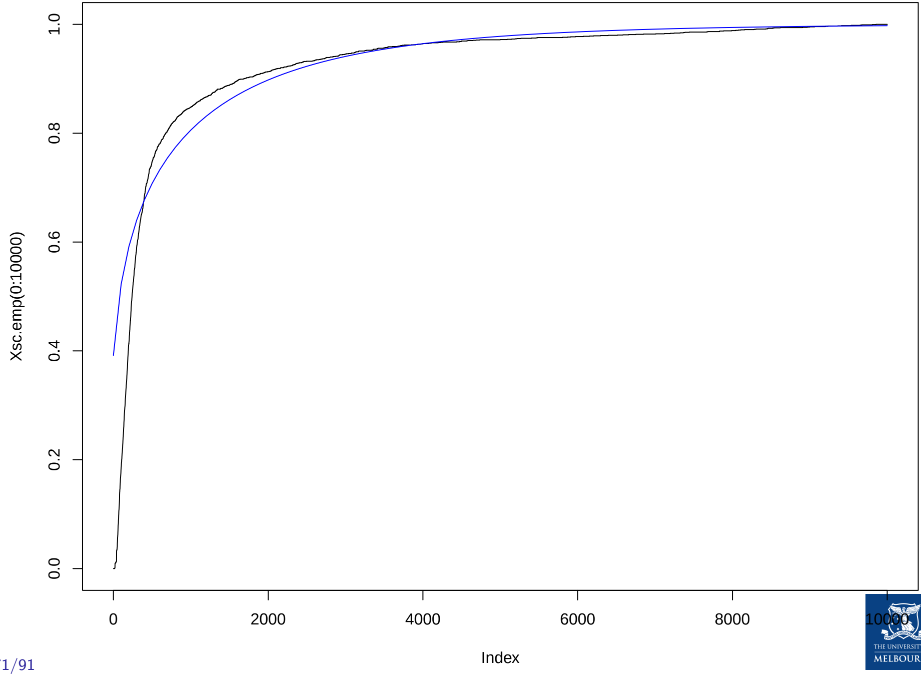
```
c(2 * gamma^(-1/2), moments[3])
```

```
## [1] 4.010796 4.010796
```

No Panjer here - just discretisation of the gamma:

```
Ssc.tlg.pmf <- discretize(pgamma(x - k, shape = gamma, rate = c),
  from = 0, to = 3 * M, step = data.scale, method = "rounding")
Ssc.tlg.df <- cumsum(Ssc.tlg.pmf)
plot(Xsc.emp(0:10000), type = "l", col = "black")
lines((0:(M/data.scale) * data.scale), Ssc.tlg.df[1:(M/data.scale +
  1)], pch = "-", col = "blue")
```

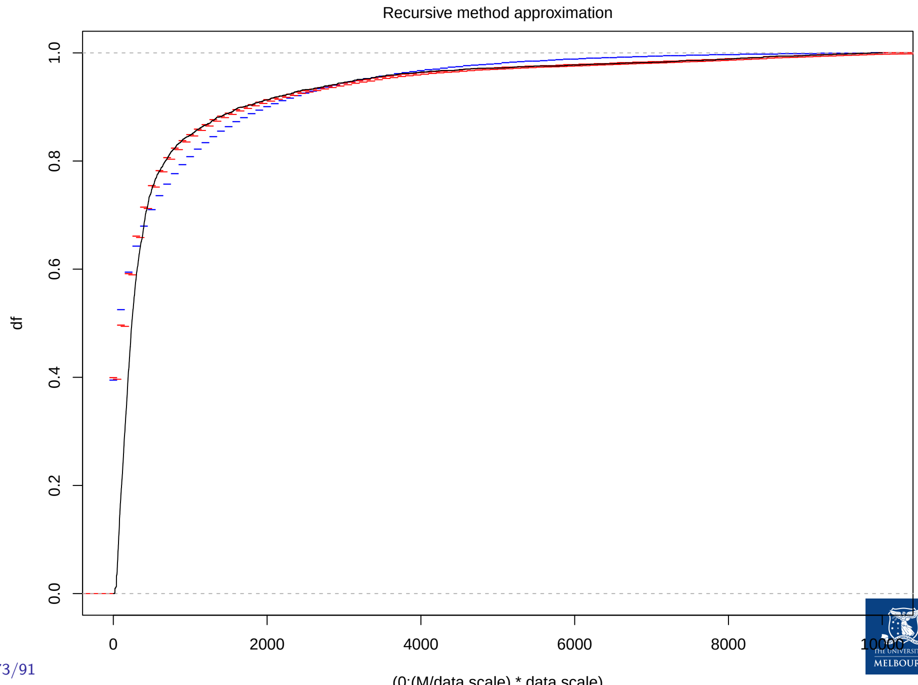
One can see with this graph that the right tail is good (as expected), but the left tail not so much. It will be interesting to compare final results with the ecdf version (which would have a better match on this graph obviously)



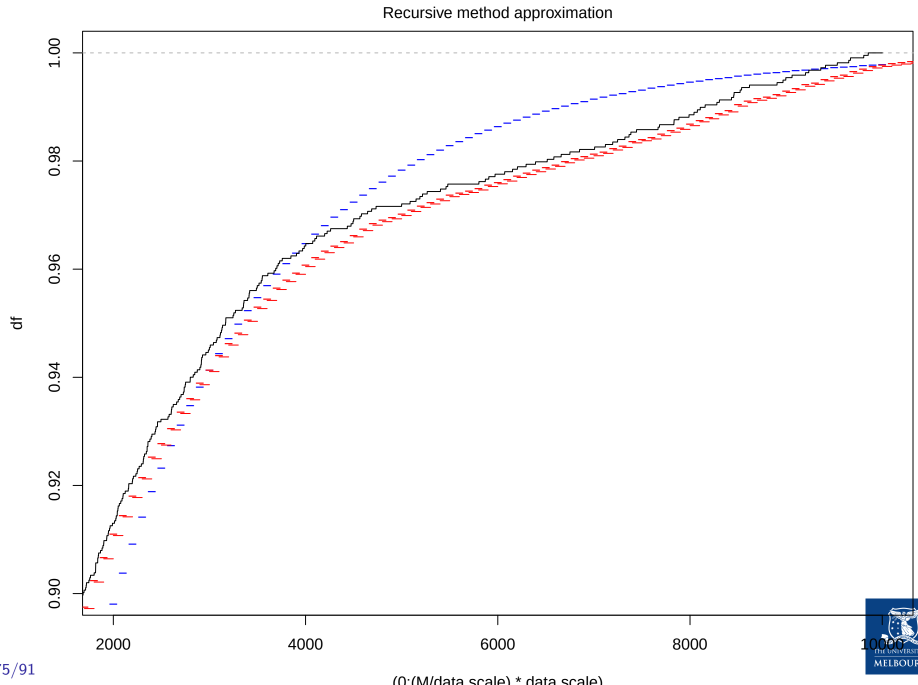
Xsc.emp(0:10000)
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
Index

Comparison

```
plot((0:(M/data.scale) * data.scale), Ssc.tlg.df[1:(M/data.scale +
  1)], pch = "-", col = "blue", xlim = c(0, 10000), ylim = c(0,
  1), ylab = "df")
lines(Ssc.emp.df, pch = "-", col = "red")
lines(Xsc.emp(0:10000), type = "l", col = "black")
```


Recursive method approximation
df
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
(0:(M/data scale) * data scale)

```r
plot((0:(M/data.scale) * data.scale), Ssc.tlg.df[1:(M/data.scale +
  1)], pch = "-", col = "blue", xlim = c(2000, 10000), ylim = c(0.9,
  1), ylab = "df")
lines(Ssc.emp.df, pch = "-", col = "red")
lines(Xsc.emp(0:10000), type = "l", col = "black")
```


Recursive method approximation
df
1.00
0.98
0.96
0.94
0.92
0.90
2000
4000
6000
8000
10000
(0:(M/data scale) * data scale)

3 ✠ Bringing it all together: SUVA case study

- Preliminaries
- Split claims
- Modelling the small claims
- Modelling the large claims
- Getting the overall distribution

Modelling the large claims

```
fit.SUVA <- fevd(SUVA$medcosts, threshold = M, type = "GP", time.units = "1
# do we have mass below M?
pextRemes(fit.SUVA, M, lower.tail = TRUE)
```

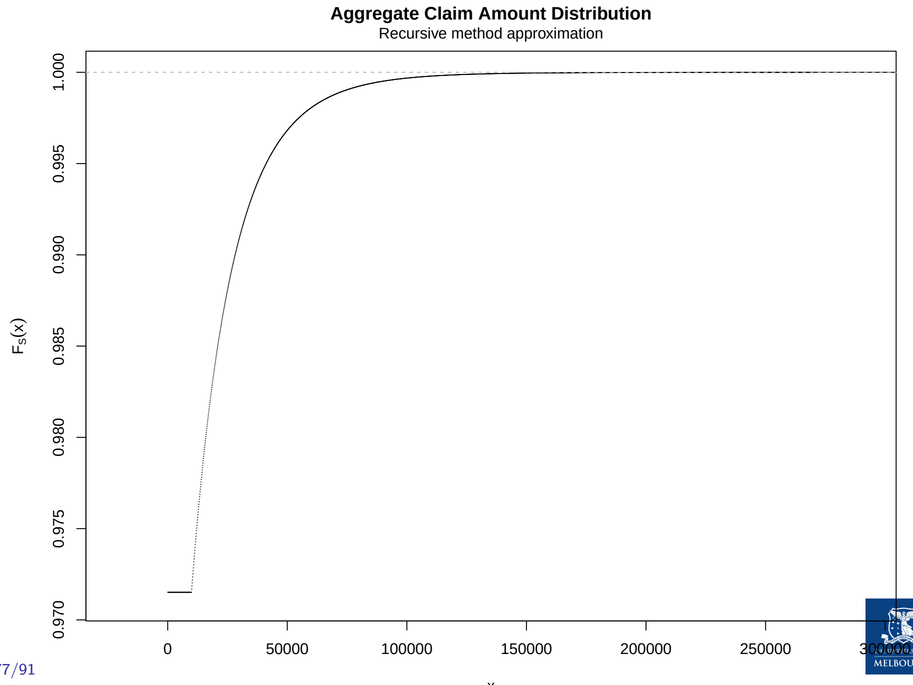
```
## [1] 0
```

```
# getting the pmf:
Xlc.pmf <- discretize(pextRemes(fit.SUVA, x, lower.tail = TRUE),
  from = 0, to = 1e+06, step = data.scale)
sum(Xlc.pmf)
```

```
## [1] 1
```

```
# distribution of Slc
Slc.df <- aggregateDist(method = "recursive", model.freq = "poisson",
  model.sev = Xlc.pmf, lambda = lambda.lc, x.scale = data.scale,
  maxit = 50000)
plot(Slc.df, ylim = c(lambda.sc, 1))
```

```
# pmf of Slc
Slc.pmf <- diff(Slc.df)
```


Aggregate Claim Amount Distribution
Recursive method approximation
$F_S(x)$
0.970
0.975
0.980
0.985
0.990
0.995
1.000
0
50000
100000
150000
200000
250000
300000
x

3 ✠ Bringing it all together: SUVA case study

- Preliminaries
- Split claims
- Modelling the small claims
- Modelling the large claims
- Getting the overall distribution

Getting ready

```
# we'll put the masses in here
Stotal.emp.pmf <- c()
Stotal.tlg.pmf <- c()
# working out vector sizes
lastpoint <- 1e+06
masses <- lastpoint/data.scale + 1
# adding 0's as required for the convolutions to work
Ssc.emp.pmf <- c(Ssc.emp.pmf, rep(0, masses - length(Ssc.emp.pmf)))
Ssc.tlg.pmf <- c(Ssc.tlg.pmf, rep(0, masses - length(Ssc.tlg.pmf)))
Slc.pmf <- c(Slc.pmf, rep(0, masses - length(Slc.pmf)))
```

Performing the convolutions

```
# and the convolutions
for (i in 1:masses) {
  Stotal.emp.pmf[i] <- sum(Ssc.emp.pmf[1:i] * Slc.pmf[i:1])
  Stotal.tlg.pmf[i] <- sum(Ssc.tlg.pmf[1:i] * Slc.pmf[i:1])
}
sum(Stotal.emp.pmf)
```

```
## [1] 0.999998
```

```
sum(Stotal.tlg.pmf)
```

```
## [1] 0.9999981
```

```
Stotal.emp.cdf <- cumsum(Stotal.emp.pmf)
Stotal.tlg.cdf <- cumsum(Stotal.tlg.pmf)
```

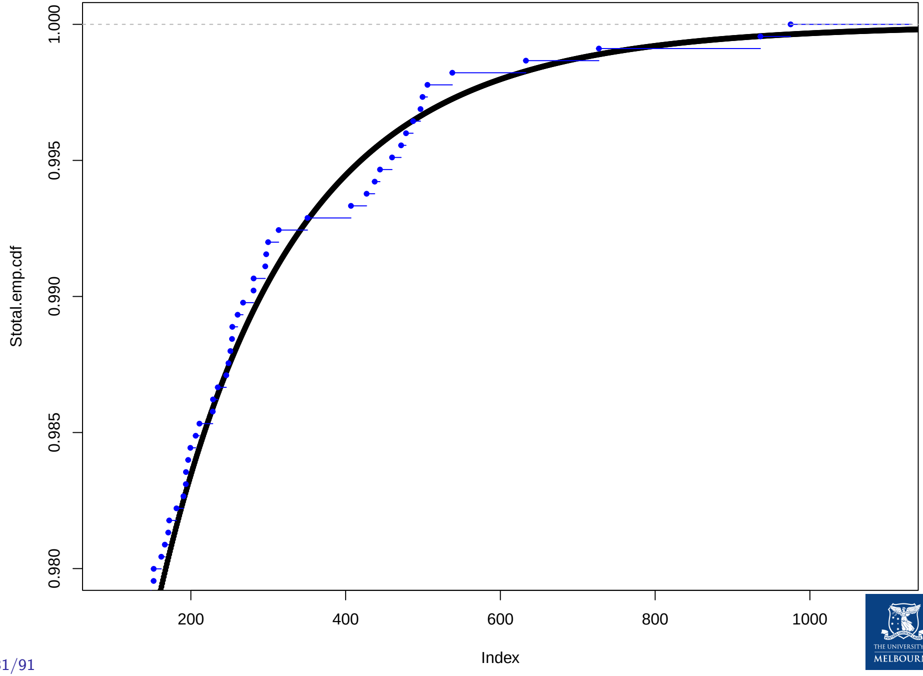
Examining the ECDF approach fit

```r
plot(Stotal.emp.cdf, pch = 20, xlim = c(100, 1100), ylim = c(0.98,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

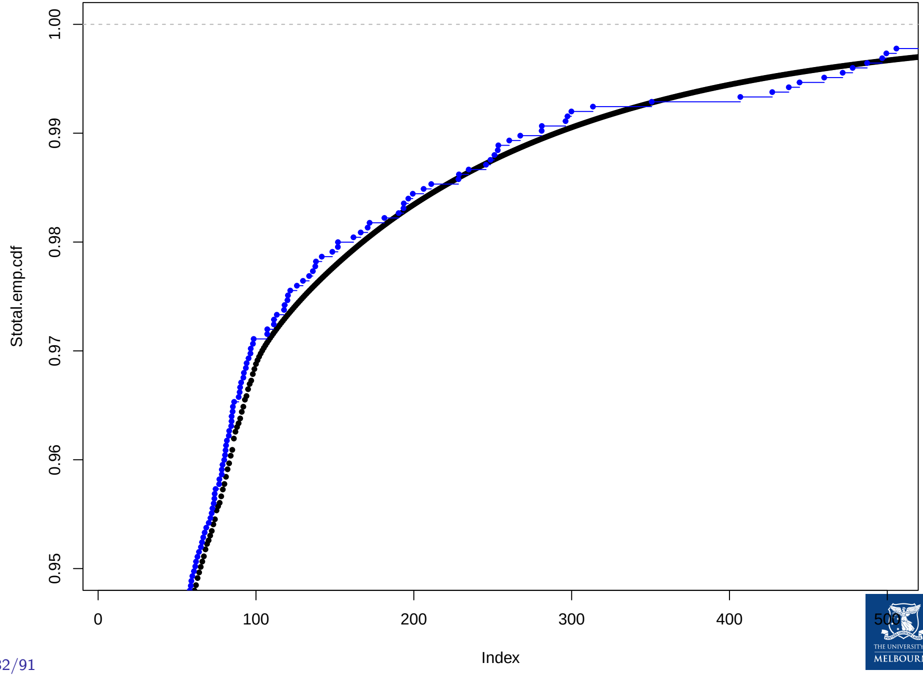
```r
plot(Stotal.emp.cdf, pch = 20, xlim = c(10, 500), ylim = c(0.95,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

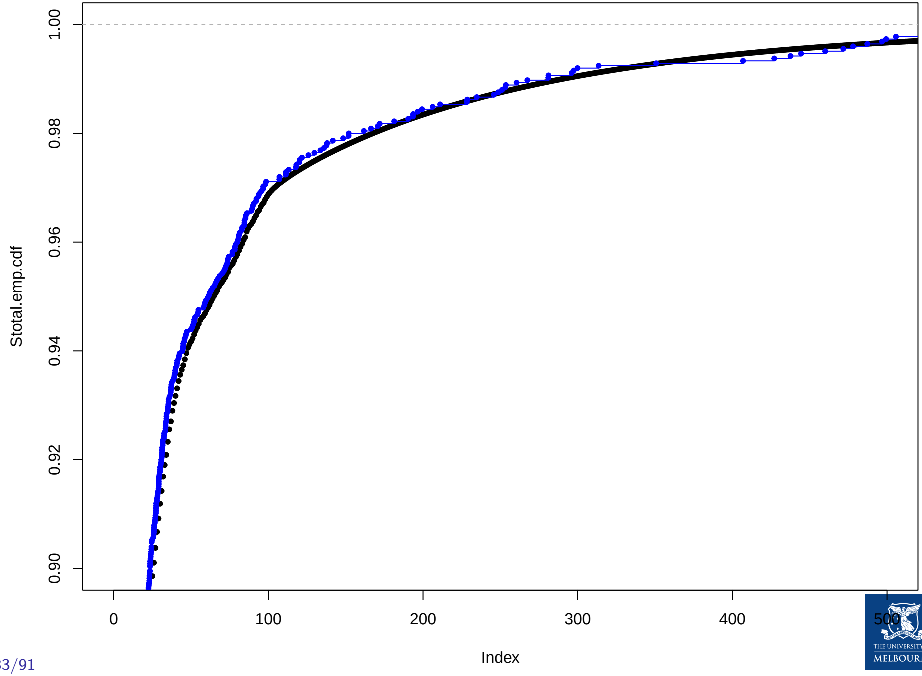
```r
plot(Stotal.emp.cdf, pch = 20, xlim = c(0, 500), ylim = c(0.9,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

```r
plot(Stotal.emp.cdf, pch = 20, xlim = c(0, 20), ylim = c(0, 1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

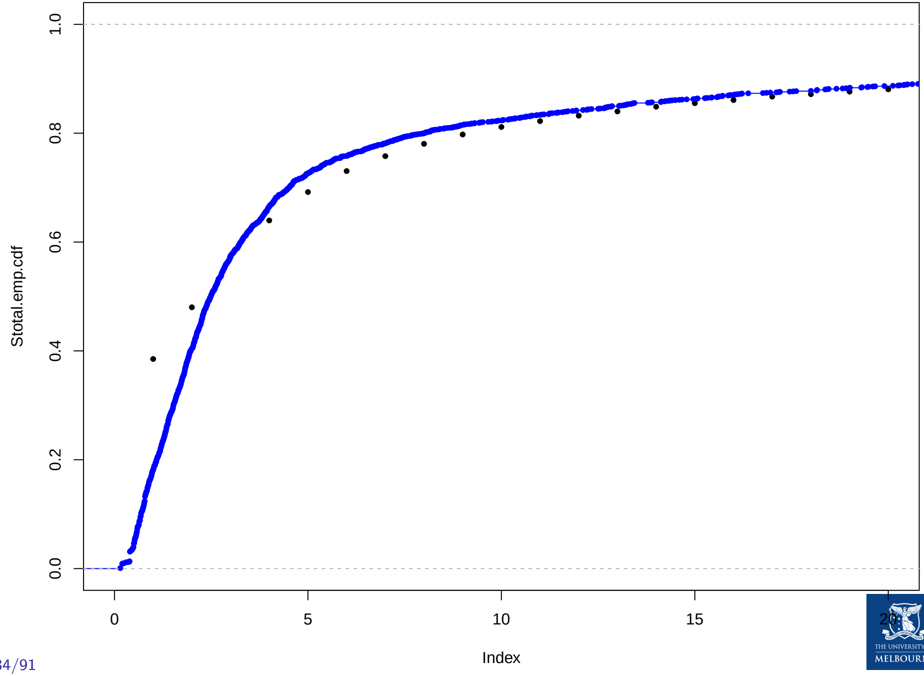


Stotal.emp.cdf
1.000
0.995
0.990
0.985
0.980
200
400
600
800
1000
Index


Stotal.emp.cdf
Index
0.95
0.96
0.97
0.98
0.99
1.00
0
100
200
300
400
500



Stotal.emp.cdf
0.90
0.92
0.94
0.96
0.98
1.00
0
100
200
300
400
500
Index


Stotal.emp.cdf
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Index

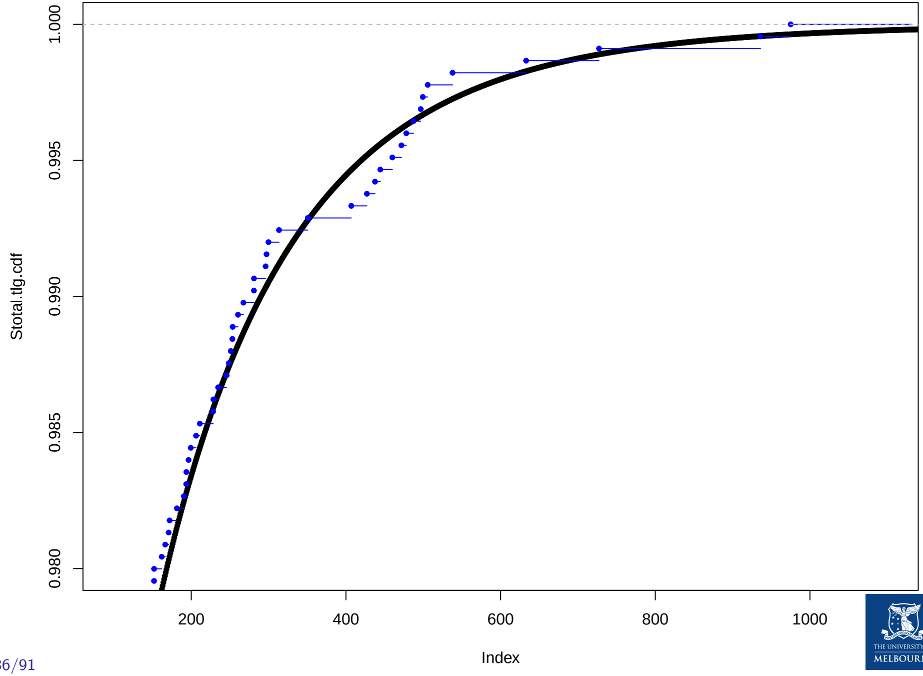
Examining the translated gamma approach fit

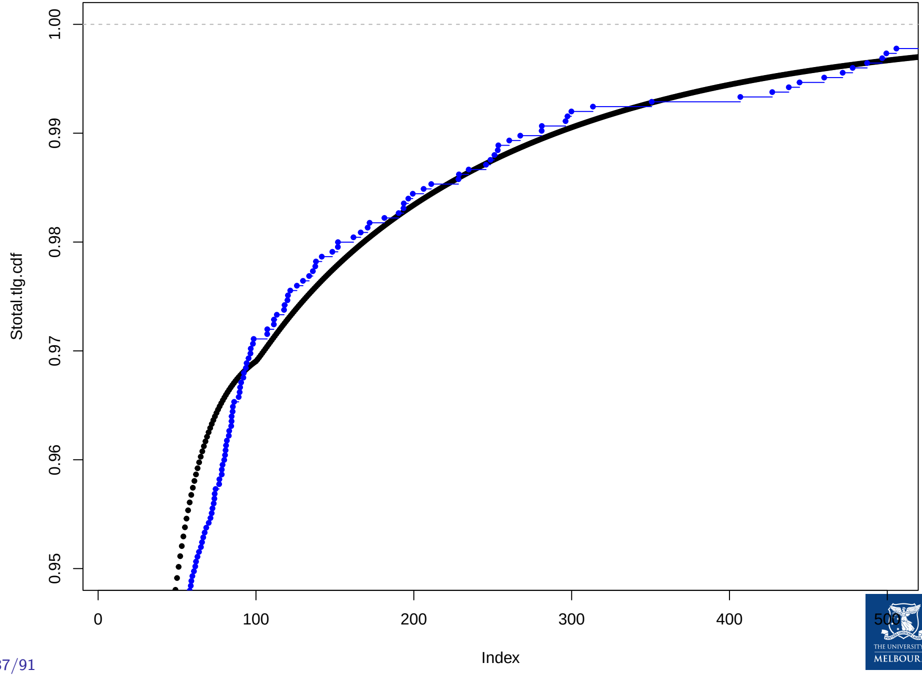
```
plot(Stotal.tlg.cdf, pch = 20, xlim = c(100, 1100), ylim = c(0.98,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

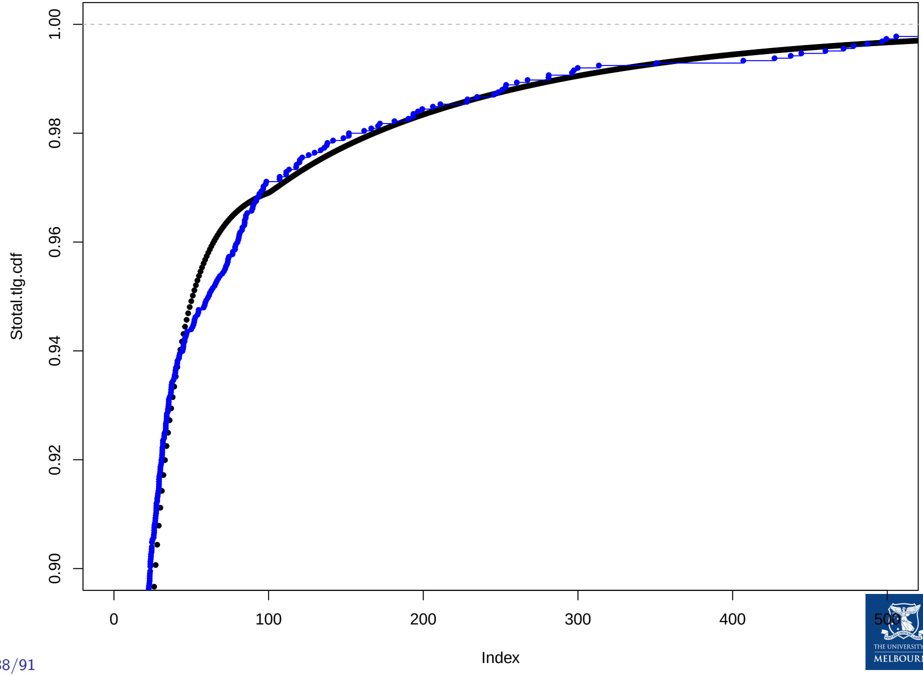
```
plot(Stotal.tlg.cdf, pch = 20, xlim = c(10, 500), ylim = c(0.95,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

```
plot(Stotal.tlg.cdf, pch = 20, xlim = c(0, 500), ylim = c(0.9,
  1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```

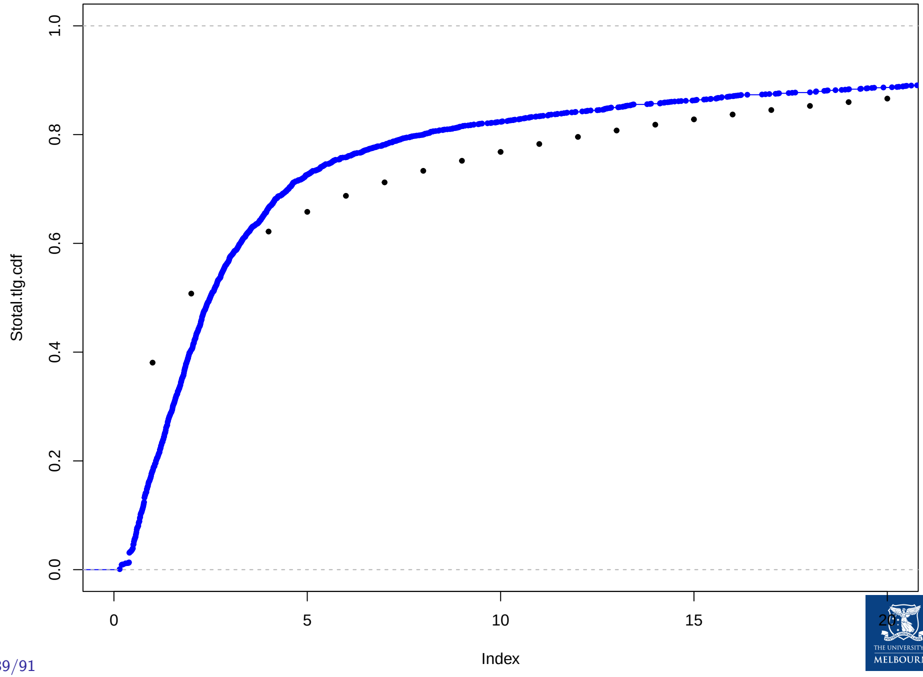
```
plot(Stotal.tlg.cdf, pch = 20, xlim = c(0, 20), ylim = c(0, 1))
lines(ecdf(SUVA$medcosts[SUVA$medcosts > 0]/data.scale), pch = 20,
  col = "blue")
```


Stotal.tlg.cdf
1.000
0.995
0.990
0.985
0.980
200
400
600
800
1000
Index


Stotal.tlg.cdf
1.00
0.99
0.98
0.97
0.96
0.95
0
100
200
300
400
500
Index



Stotal.tlg.cdf
1.00
0.98
0.96
0.94
0.92
0.90
0
100
200
300
400
500
Index



Stotal.tlg.cdf
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Index

Moments

```
em1 <- c(emm(SUVA$medcosts[SUVA$medcosts > 0], 1), sum((((1:masses) -
  1) * data.scale) * (Stotal.emp.pmf)), sum((((1:masses) -
  1) * data.scale) * (Stotal.tlg.pmf)))
em2 <- c(emm(SUVA$medcosts[SUVA$medcosts > 0], 2), sum((((1:masses) -
  1) * data.scale)^2 * (Stotal.emp.pmf)), sum((((1:masses) -
  1) * data.scale)^2 * (Stotal.tlg.pmf)))
em3 <- c(emm(SUVA$medcosts[SUVA$medcosts > 0], 3), sum((((1:masses) -
  1) * data.scale)^3 * (Stotal.emp.pmf)), sum((((1:masses) -
  1) * data.scale)^3 * (Stotal.tlg.pmf)))
sd <- (em2 - em1^2)^0.5
skew <- (em3 - 3 * em1 * sd^2 - em1^3)/sd^3
Smoments <- data.frame(rbind(em1, sd, skew))
names(Smoments) <- c("SUVA", "Emp", "Tlg")
Smoments
```

	SUVA	Emp	Tlg
em1	1492.765229	1478.788445	1494.851875
sd	5763.108871	5932.145105	5897.220298
skew	8.885693	9.739098	9.893872

References I

Avanzi, Benjamin, Luke C. Cassar, and Bernard Wong. 2011. "Modelling Dependence in Insurance Claims Processes with Lévy Copulas." *ASTIN Bulletin* 41 (2): 575–609.

Bowers, Newton L. Jr, Hans U. Gerber, James C. Hickman, Donald A. Jones, and Cecil J. Nesbitt. 1997. *Actuarial Mathematics*. Second. Schaumburg, Illinois: The Society of Actuaries.

Wuthrich, Mario V. 2022. "Non-Life Insurance: Mathematics & Statistics." Lecture notes. RiskLab, ETH Zurich; Swiss Finance Institute.