# Loss models 2020: computer labs in R

A hands-on workshop

Katrien Antonio & Jonas Crevecoeur

Loss models 2020 labs | December 15 & 17, 2020

# Prologue

# Introduction

## Course

https://github.com/katrienantonio/loss-models-2020

The course repo on GitHub, where you can find the data sets, lecture sheets, R scripts and R markdown files.

## Us

https://katrienantonio.github.io/

katrien.antonio@kuleuven.be & jonas.crevecoeur@kuleuven.be

(Katrien, PhD) Professor in insurance data science at KU Leuven and University of Amsterdam

(Jonas, PhD) Post-doctoral researcher in biostatistics at KU Leuven

# Checklist

☑ Do you have a fairly recent version of R?

```
version$version.string
## [1] "R version 4.0.3 (2020-10-10)"
```

☑ Do you have a fairly recent version of RStudio?

```
RStudio.Version()$version
## Requires an interactive session but should return something like "[1] ‘1.3.1093’"
```

☑ Have you installed the R packages listed in the software requirements?

or

☑ Have you created an account on RStudio Cloud (to avoid any local installation issues)?

# Why this training?

## The goals of this training 🚀

- develop practical data handling foundations
- visualize and explore data
- cover essential actuarial modelling tasks, including fitting loss models for frequency and severity data
- learn by doing, get you started (in particular when you have limited experience in R).

> *"In short, we will cover things that we wish someone had taught us in our undergraduate programs."*
>
> This quote is from the Data science for economists course by Grant McDermott.

# Why R and RStudio?

## Data science positivism

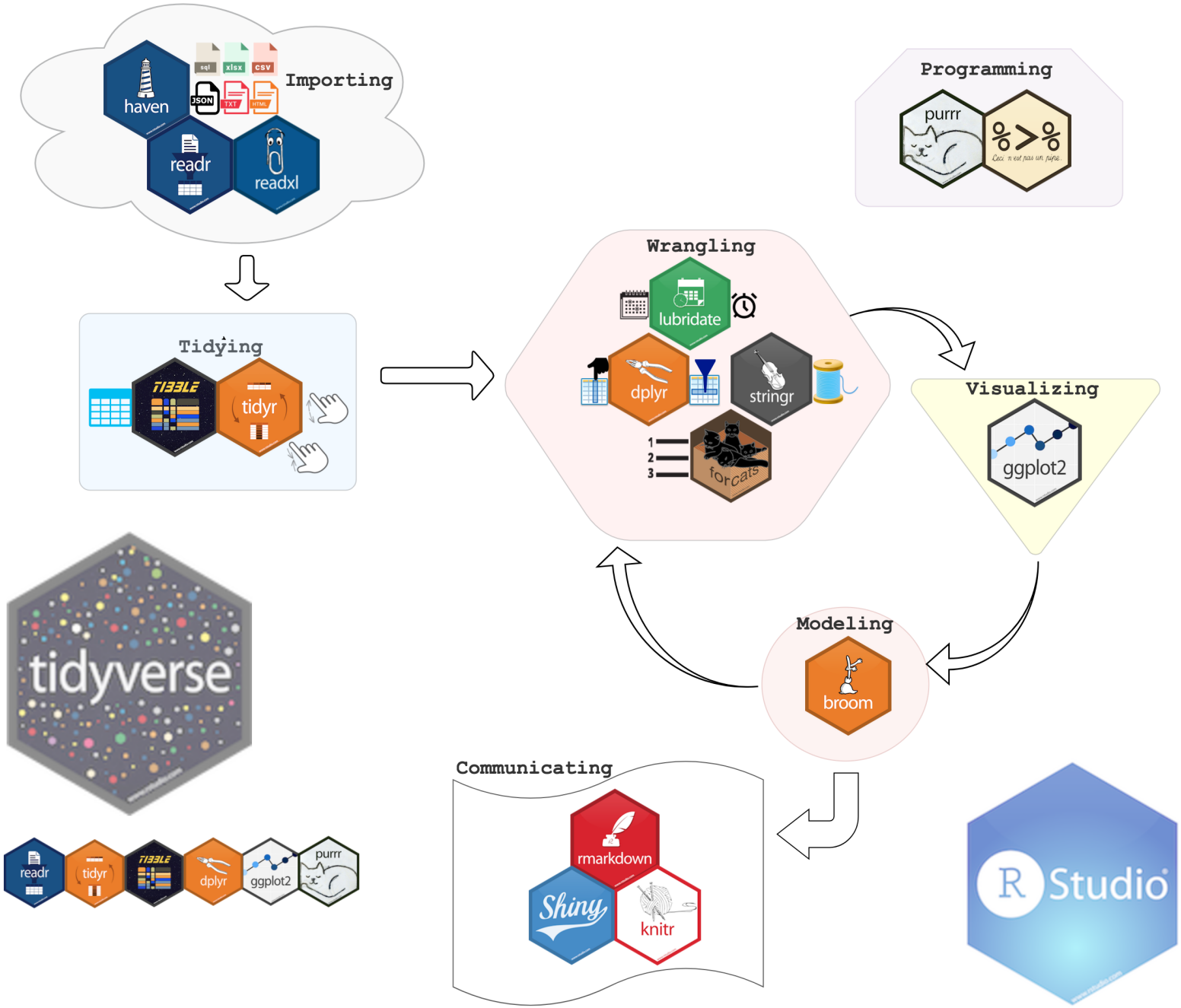
- Next to Python, R has become the *de facto* language for data science, with a cutting edge *machine learning toolbox*.
- See: The Popularity of Data Science Software
- R is open-source with a very active community of users spanning academia and industry.

## Bridge to actuarial science, econometrics and other tools

- R has all of the statistics and econometrics support, and is amazingly adaptable as a “glue” language to other programming languages and APIs.
- R does not try to be everything to everyone. The RStudio IDE and ecosystem allow for further, seemless integration (with e.g. python, keras, tensorflow or C).
- Widely used in actuarial undergraduate programs

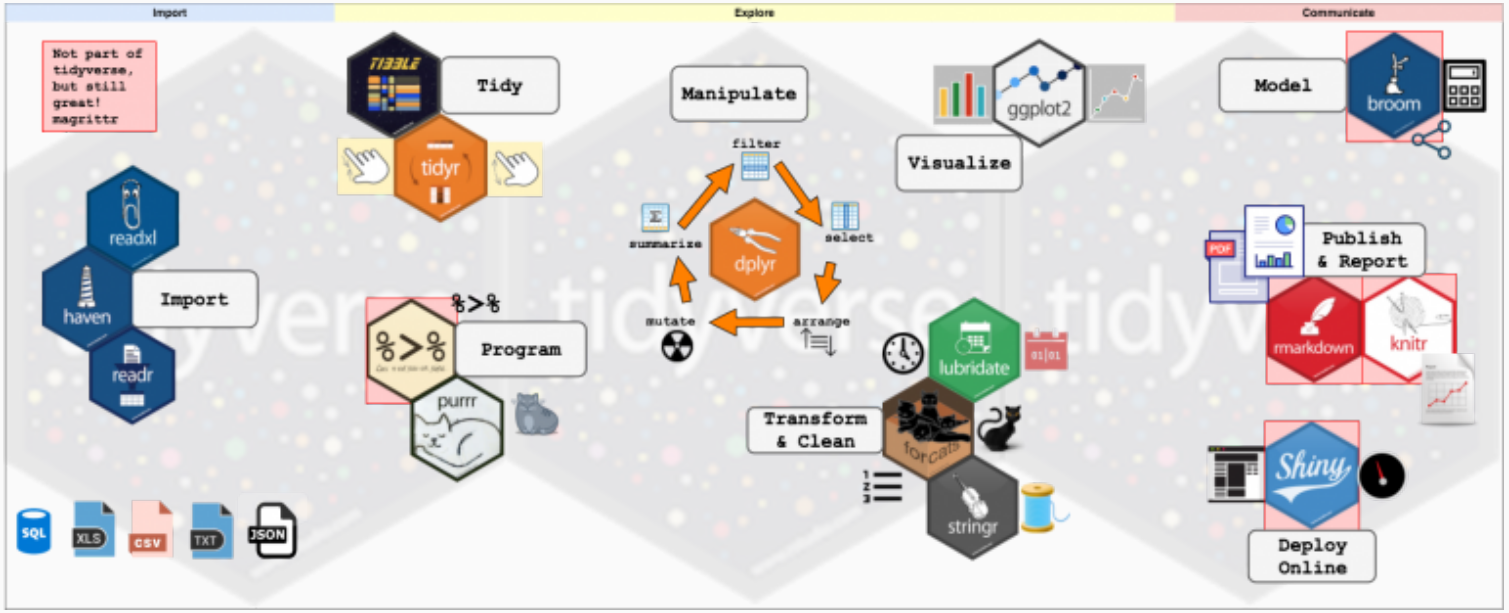
## Disclaimer + Read more

- It's also the language that we know best.
- If you want to read more: R-vs-Python, when to use Python or R or Hadley Wickham on the future of R


Importing
haven
readr
readxl
Programming
purrr
%>%
Tidying
tibble
tidyr
Wrangling
lubridate
dplyr
stringr
forcats
Visualizing
ggplot2
Modeling
broom
Communicating
rmarkdown
Shiny
knitr
RStudio
tidyverse
readr
tidyr
tibble
dplyr
ggplot2
purrr

# Welcome to the tidyverse!

> The **tidyverse** is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.




More on: tidyverse.

Install the packages with `install.packages("tidyverse")`. Then run `library(tidyverse)` to load the core tidyverse.

# Today's Outline

- Prologue
- What's out there: the R universe (see the prework)
  - why R and RStudio?
  - welcome to the tidyverse!
  - principles of tidy data
  - workflow of a data scientist
- Data wrangling and visualisation (see the prework)
  - tibbles
  - pipe operator `%>%`
  - {dplyr} instructions
  - {ggplot2} for data visualisation
  - what else is there?
- Data sets used in the session
  - MTPL data on frequency and severity of claims
  - Secura Re losses
- Fitting frequency models in R
  - frequency - severity approach
  - the $(a, b, 0)$ class
  - MLE, model selection and evaluation tools
  - excess zeroes
- Fitting severity models in R
  - from simple to complex parametric distributions
  - be aware of truncation and censoring
  - a global fit via splicing
  - case-study on the Secura Re losses
  - what else is there?

# Data sets used in the session

# Data sets used in this session - MTPL


We illustrate some first data handling steps on the Motor Third Party Liability data set. There are 163 231 policyholders in this data set.

The frequency of claiming (`nclaims`) and corresponding severity (`avg`, the amount paid on average per claim reported by a policyholder) are the **target variables** in this data set.

Predictor variables are:

- the exposure-to-risk, the duration of the insurance coverage (max. 1 year)
- factor variables, e.g. gender, coverage, fuel
- continuous, numeric variables, e.g. age of the policyholder, age of the car
- spatial information: postal code (in Belgium) of the municipality where the policyholder resides.

More details in Henckaerts et al. (2018, Scandinavian Actuarial Journal) and Henckaerts et al. (2020, North American Actuarial Journal).

# Data sets used in this session - MTPL


You can load the data from a .R script in the course material:

```
# install.packages("rstudioapi")
dir <- dirname(rstudioapi::getActiveDocumentContext()$path)
setwd(dir)
mtpl_orig <- read.table('./data/PC_data.txt',
                                  header = TRUE)
mtpl_orig <- as_tibble(mtpl_orig)
```

If you work in an R notebook or R markdown file, you can also go for:

```
# install.packages("here")
library(here)
dir <- here::here()
setwd(dir)
mtpl_orig <- read.table('./data/PC_data.txt',
                                  header = TRUE)
mtpl_orig <- as_tibble(mtpl_orig)
```

The last instruction transforms `mtpl_orig` into a `tibble`.

These instructions are recommended because they avoid referring to a specific working directory on your computer, e.g.

```
setwd("C:\\Users\\u0043788\\Dropbox\\Loss models course\\data")
mtpl_orig <- read.table('PC_data.txt', header = TRUE)
```

If you organize your data analysis or project in a folder on your computer that holds all relevant files, then the above instructions allow you (and your colleagues) to get started right away.

The only thing you need is an organized file structure.

The {rstudioapi} package is developed for RStudio. The {here} library is more general.

Do note that when working in a .Rmd, {here} will use the folder that markdown lives in as the working directory, but if you are working in a script (.R) the working directory is the top level of the project file.

First of all, we explore the structure of `mtpl_orig` with `str()`.

```
str(mtpl_orig)
## tibble [163,231 x 18] (S3: tbl_df/tbl/data.frame)
##  $ ID      : int [1:163231] 1 2 3 4 5 6 7 8 9 10 ...
##  $ NCLAIMS : int [1:163231] 1 0 0 0 1 0 1 0 0 0 ...
##  $ AMOUNT  : num [1:163231] 1618 0 0 0 156 ...
##  $ AVG     : num [1:163231] 1618 NA NA NA 156 ...
##  $ EXP     : num [1:163231] 1 1 1 1 0.0466 ...
##  $ COVERAGE: chr [1:163231] "TPL" "PO" "TPL" "TPL" ...
##  $ FUEL    : chr [1:163231] "gasoline" "gasoline" "diesel" "gasoline" ...
##  $ USE     : chr [1:163231] "private" "private" "private" "private" ...
##  $ FLEET   : chr [1:163231] "N" "N" "N" "N" ...
##  $ SEX     : chr [1:163231] "male" "female" "male" "male" ...
##  $ AGEPH   : int [1:163231] 50 64 60 77 28 26 26 58 59 34 ...
##  $ BM      : int [1:163231] 5 5 0 0 9 11 11 11 0 7 ...
##  $ AGEC    : int [1:163231] 12 3 10 15 7 12 8 14 3 6 ...
##  $ POWER   : int [1:163231] 77 66 70 57 70 70 55 47 98 74 ...
##  $ PC      : int [1:163231] 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...
##  $ TOWN    : chr [1:163231] "BRUSSEL" "BRUSSEL" "BRUSSEL" "BRUSSEL" ...
##  $ LONG    : num [1:163231] 4.36 4.36 4.36 4.36 4.36 ...
##  $ LAT     : num [1:163231] 50.8 50.8 50.8 50.8 50.8 ...
```

Or with `head()`...

```
head(mtpl_orig) %>% kable(format = 'html')
```

| ID | NCLAIMS | AMOUNT | AVG | EXP | COVERAGE | FUEL | USE | FLEET | SEX | AGEPH | BM | AGEC | POWER | PC | TOW |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1618.0010 | 1618.0010 | 1.0000000 | TPL | gasoline | private | N | male | 50 | 5 | 12 | 77 | 1000 | BRU |
| 2 | 0 | 0.0000 | NA | 1.0000000 | PO | gasoline | private | N | female | 64 | 5 | 3 | 66 | 1000 | BRU |
| 3 | 0 | 0.0000 | NA | 1.0000000 | TPL | diesel | private | N | male | 60 | 0 | 10 | 70 | 1000 | BRU |
| 4 | 0 | 0.0000 | NA | 1.0000000 | TPL | gasoline | private | N | male | 77 | 0 | 15 | 57 | 1000 | BRU |
| 5 | 1 | 155.9746 | 155.9746 | 0.0465753 | TPL | gasoline | private | N | female | 28 | 9 | 7 | 70 | 1000 | BRU |
| 6 | 0 | 0.0000 | NA | 1.0000000 | TPL | gasoline | private | N | male | 26 | 11 | 12 | 70 | 1000 | BRU |

Note that the data `mtpl_orig` uses capitals for the variable names

```
mtpl_orig %>% slice(1:3) %>% select(-LONG, -LAT) %>% kable(format = 'html')
```

| ID | NCLAIMS | AMOUNT | AVG | EXP | COVERAGE | FUEL | USE | FLEET | SEX | AGEPH | BM | AGEC | POWER | PC | TOWN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1618.001 | 1618.001 | 1 | TPL | gasoline | private | N | male | 50 | 5 | 12 | 77 | 1000 | BRUSSEL |
| 2 | 0 | 0.000 | NA | 1 | PO | gasoline | private | N | female | 64 | 5 | 3 | 66 | 1000 | BRUSSEL |
| 3 | 0 | 0.000 | NA | 1 | TPL | diesel | private | N | male | 60 | 0 | 10 | 70 | 1000 | BRUSSEL |

We change this to lower case variables, and rename `exp` to `expo`.

```
mtpl <- mtpl_orig %>%
  # rename all columns
  rename_all(function(.name) {
    .name %>%
      # replace all names with the lowercase versions
      tolower
    })
mtpl <- rename(mtpl, expo = exp)
```

Check `rename_all()` in {dplyr}.

# Your turn

To get warmed up, let's load the `mtpl` data and do some basic investigations into the variables. The idea is to get a feel for the data.

Your starting point are the instructions in the R script on the course website.

**Q**: you will work through the following exploratory steps.

1. Calculate the empirical claim frequency, per unit of exposure. Then do the same per gender. What do you conclude?
2. Visualize the distribution of `nclaims` with a bar plot, use `geom_bar` as a layer in `ggplot`. Try different versions of this bar plot.
3. Visualize the distribution of `avg` with a density plot, use `geom_density` as a layer. Restrict the range of values displayed in the plot.
4. Visualize the distribution of `avg` with a histogram, use `geom_histogram`. Play with the `binwidth`.

```
dim(mtpl)
```

```
## [1] 163231     18
```

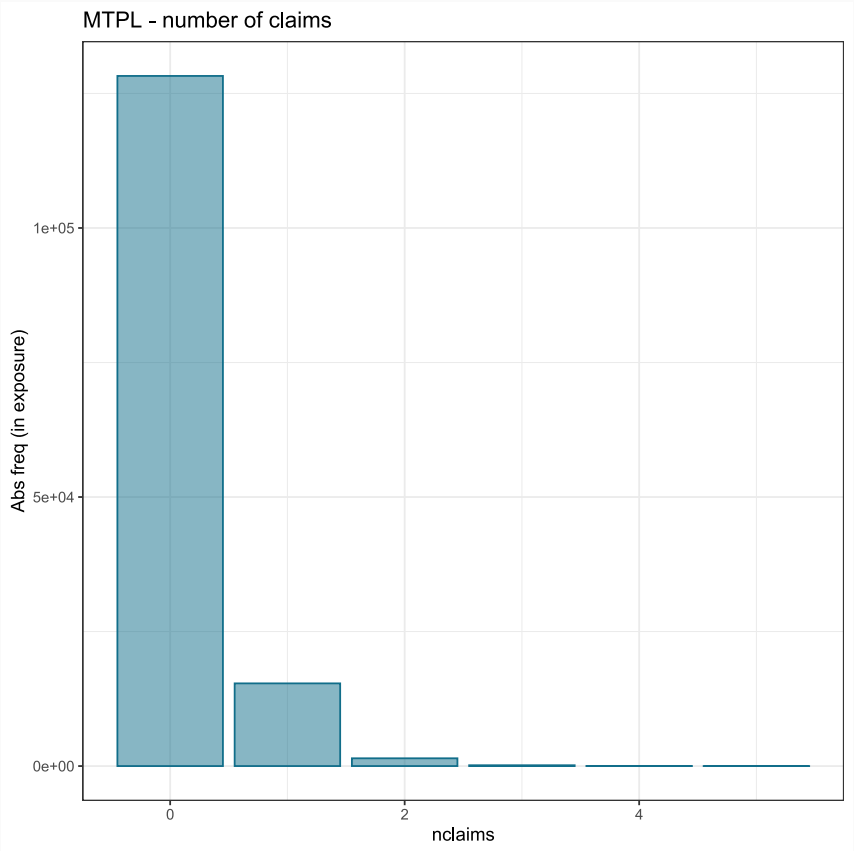
```
mtpl %>% summarize(emp_freq =
                     sum(nclaims) / sum(expo))
```

| emp_freq |
|---|
| 0.1393352 |

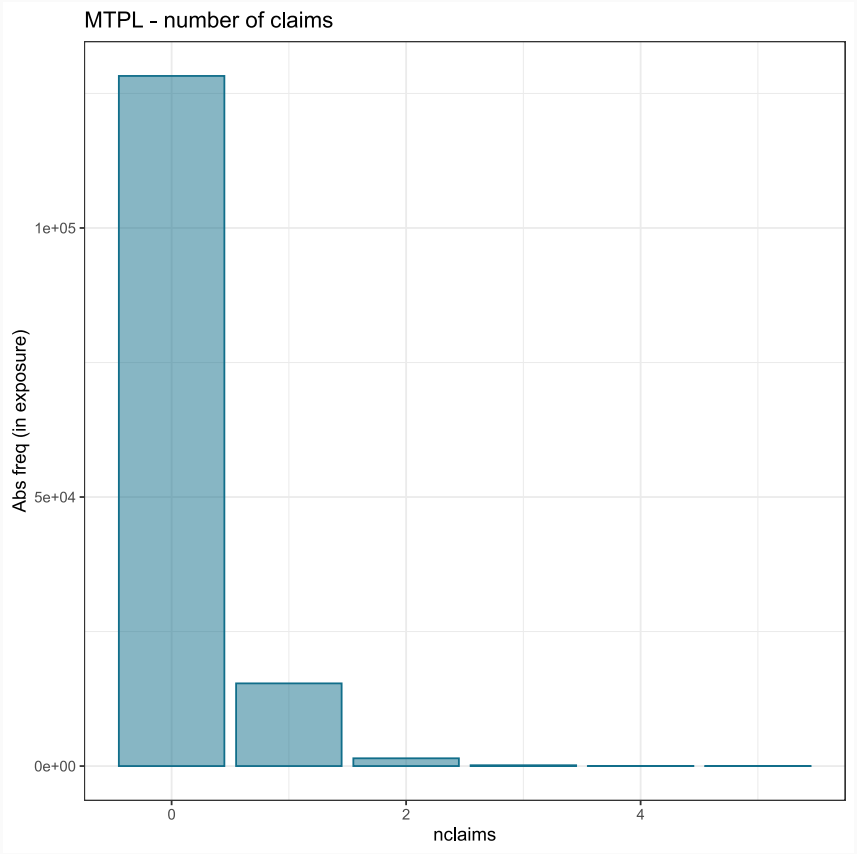
```
mtpl %>%
  group_by(sex) %>%
  summarize(emp_freq = sum(nclaims) / sum(expo))
```

| sex | emp_freq |
|---|---|
| female | 0.1484325 |
| male | 0.1361164 |

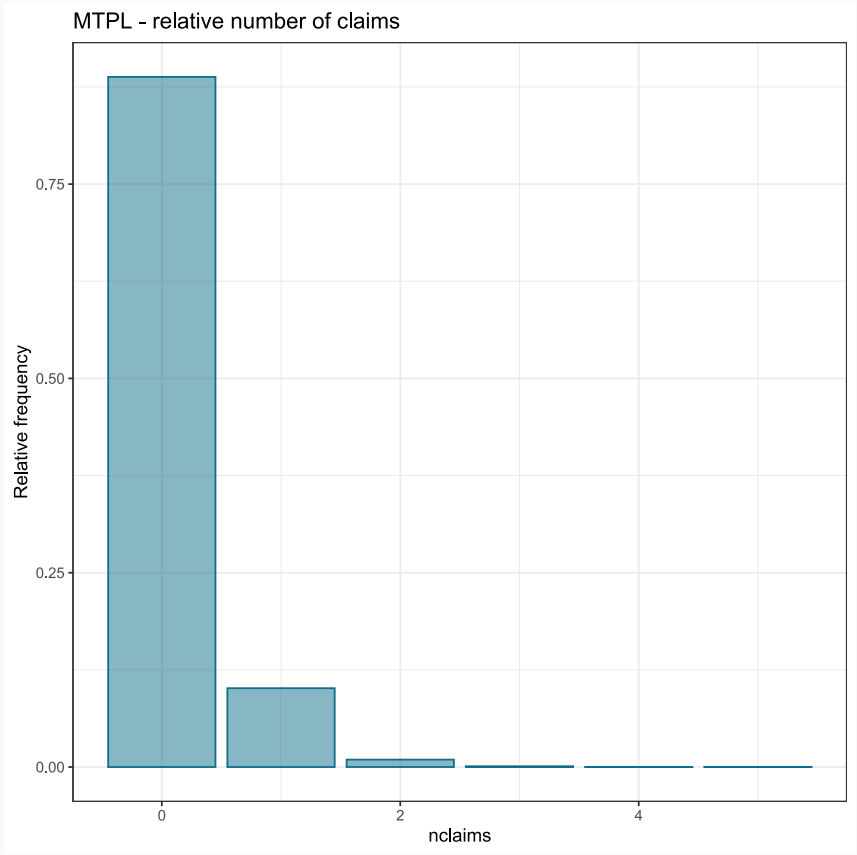
```
g <- ggplot(mtpl, aes(nclaims)) + theme_bw() +
     geom_bar(aes(weight = expo), col = KULbg,
              fill = KULbg, alpha = 0.5) +
     labs(y = "Abs freq (in exposure)") +
     ggtitle("MTPL - number of claims")
g
```



```
g <- ggplot(mtpl, aes(nclaims)) + theme_bw() +
     geom_bar(aes(weight = expo), col = KULbg,
              fill = KULbg, alpha = 0.5) +
     labs(y = "Abs freq (in exposure)") +
     ggtitle("MTPL - number of claims")
g
```

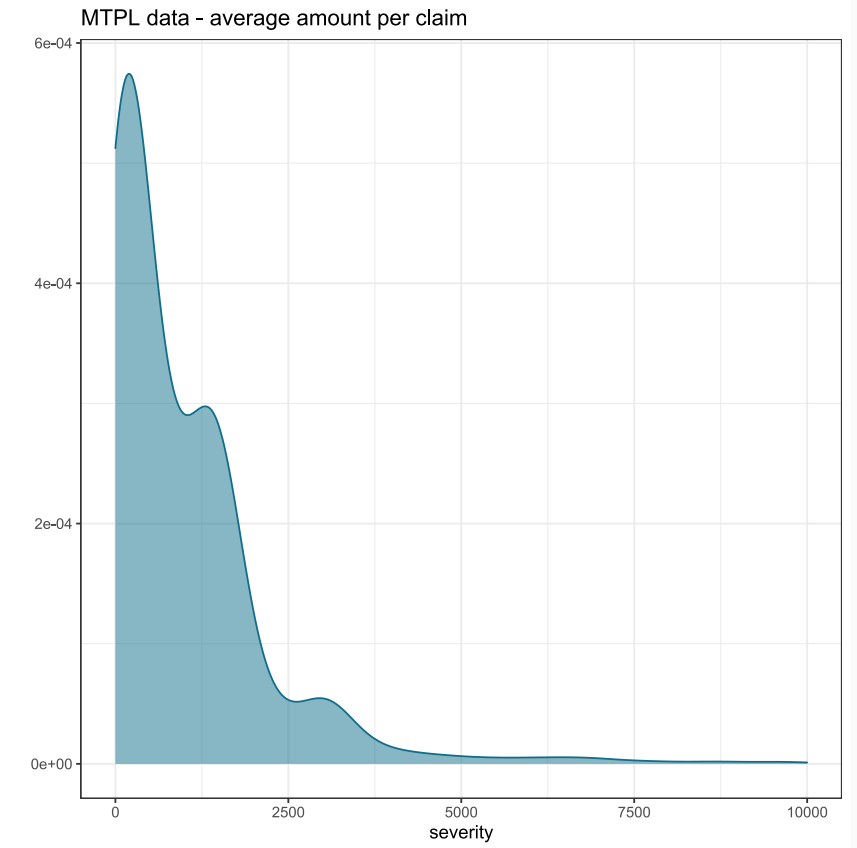



```
g <- ggplot(mtpl, aes(nclaims)) + theme_bw()
g + geom_bar(aes(y = (..count..)/sum(..count..)),
    col = KULbg, fill = KULbg, alpha = 0.5) +
  labs(y = "Relative frequency") +
  ggtitle("MTPL - relative number of claims")
```



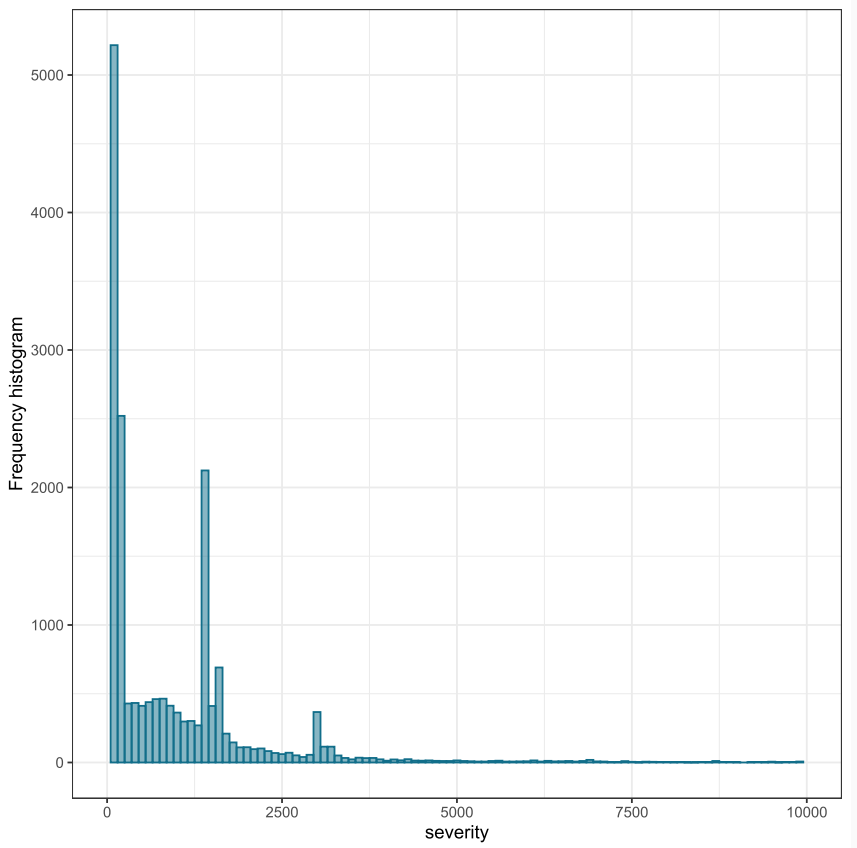
With a density plot: mind the `filter(.)` instruction and the use of `xlim(.,.)`

```
g_dens <- mtpl %>% filter(avg > 0 & avg <= 81000) %>%
          ggplot(aes(x = avg)) +
          theme_bw() +
          geom_density(adjust = 3, col = KULbg,
                       fill = KULbg, alpha = 0.5) +
          xlim(0, 1e4) +
          ylab("") + xlab("severity") +
    ggtitle("MTPL data - average amount per claim")
g_dens
```



With a histogram:

```
g <- mtpl %>% filter(avg > 0 & avg <= 81000) %>%
     ggplot(aes(x = avg)) +
      theme_bw() + xlab("severity") +
      geom_histogram(binwidth = 100, col = KULbg,
            fill = KULbg, alpha = 0.5) +
      xlim(0, 1e4) +
      labs(y = "Frequency histogram")
g
```



# Data sets used in this session - Secura Re losses

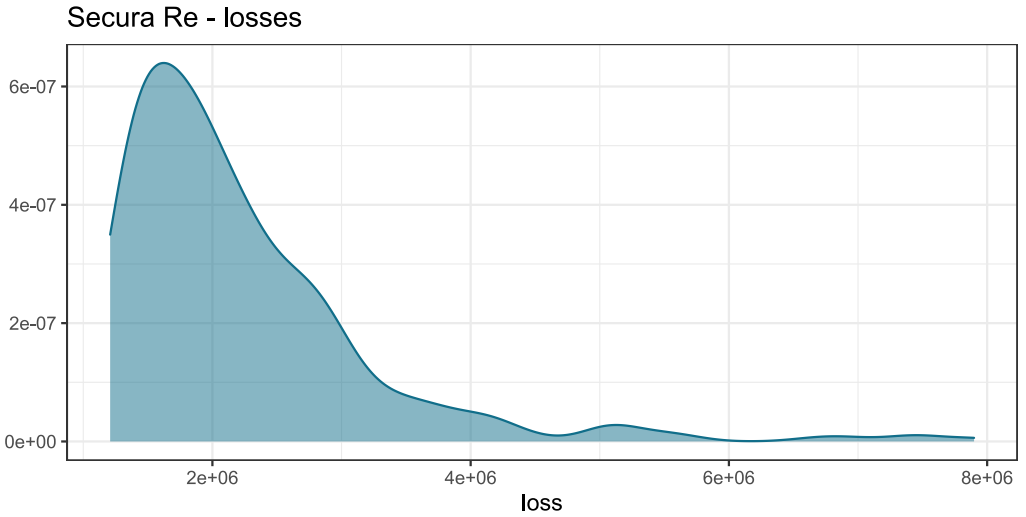
Secura Belgian Re automobile claims from 1988 to 2001 data, gathered from several European insurance companies, exceeding 1 200 000 Euro.

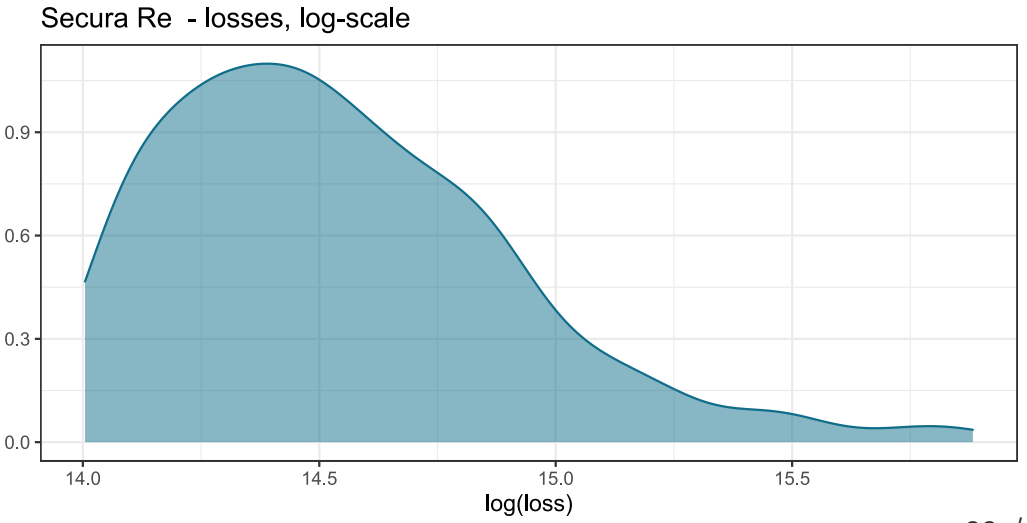
The data were, among others, corrected for inflation, see the {ReIns} library.

```
secura <- read.table(file="./data/SecuraRe.txt",
                     header = TRUE, sep = "\t")
```

The density plots on the right show typical features of insurance loss data:

- positive
- skewed to the right
- heavy right tail
- (possibly) subject to truncation and censoring.



# Your turn

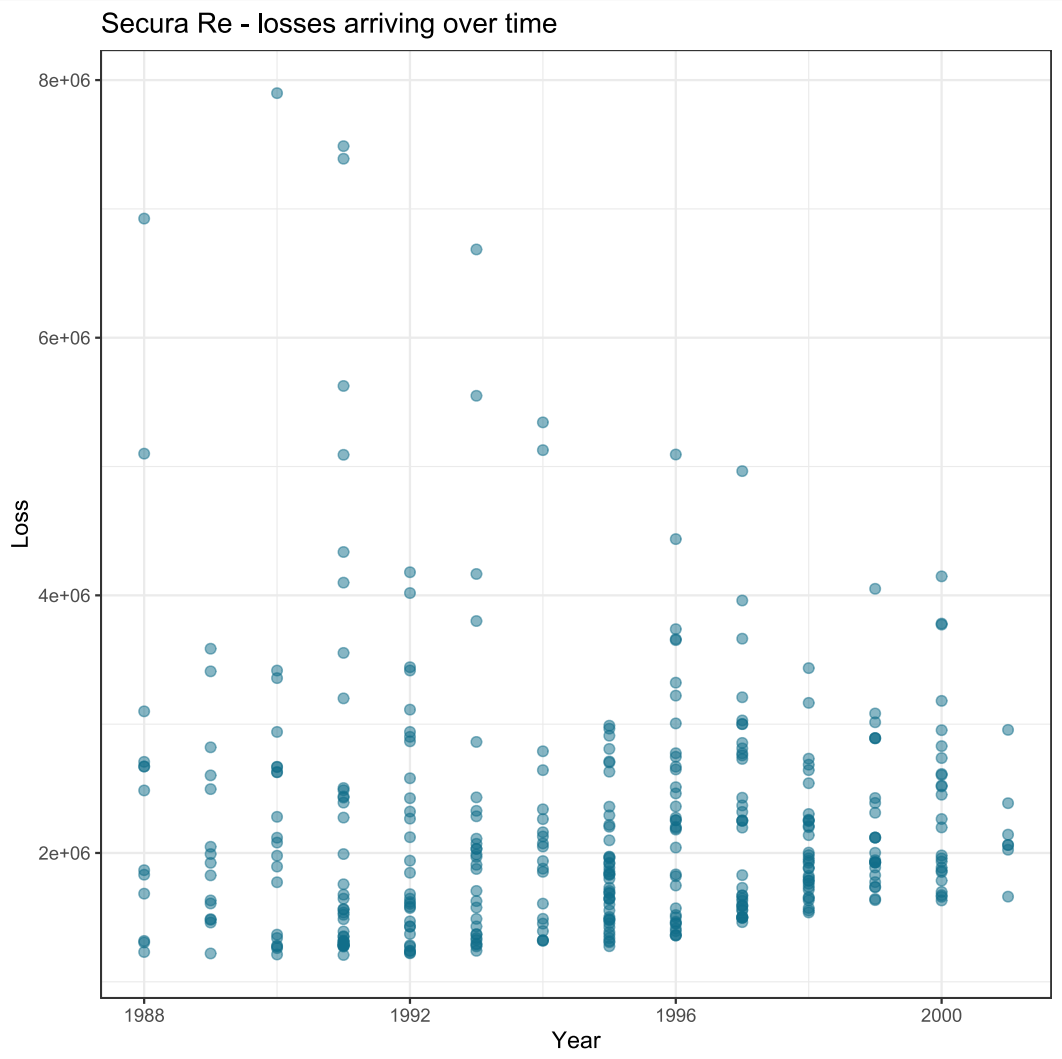
You will now use the instructions covered in the exploration of the `mtpl` data to picture the `secura` losses.

**Q**: you will work through the following exploratory steps.

1. Visualize the distribution of `Loss` in `secura` with a density plot, use `geom_density` as a layer.
2. Do the same for `log(Loss)`.
3. Plot the `Loss` versus `Year`. Use the `geom_point` layer.

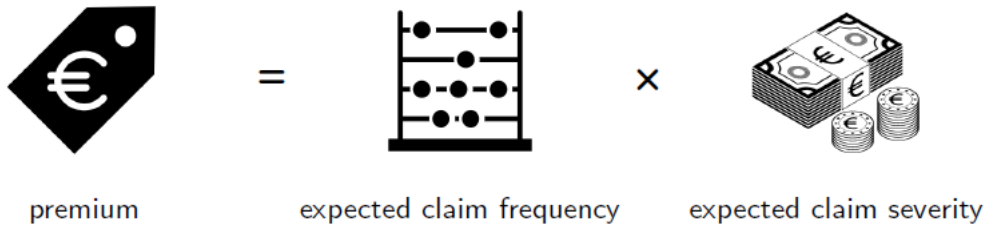
The `secura` losses over time...

```
ggplot(secura, aes(Year, Loss)) + theme_bw() +
  geom_point(colour = KULbg, size = 2, alpha = 0.5) +
  ggtitle('Secura Re - losses arriving over time')
```



# Fitting frequency models in R

# The frequency - severity approach




## Assumptions:

- claim severity and claim frequency are independent
- we ignore any additional covariates (cfr. workshop last year).

Each record $i$ in the data set registers

- exposure, $e_i$, fraction of the year in which the contract was active
- number of claims, $N_i$, observed claim count in the period of exposure
- average claim size per reported claim (or: severity), $S_i$, total losses in the period of exposure divided by the number of claims, if $N_i > 0$.

# The (a, b, 0) class

Let $N$ be a count or frequency random variable, with probability function (pf) $Pr(N = k) = p_k$.

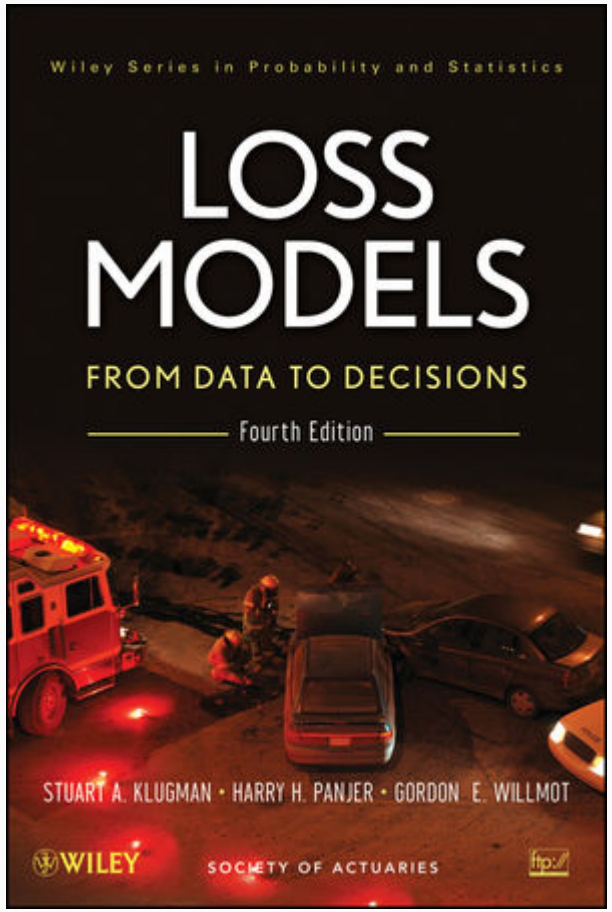
The $(a, b, 0)$-class of frequency distributions is a two-parameter class of distributions that satisfy

$$k \cdot \frac{p_k}{p_{k-1}} = a \cdot k + b \quad k \geq 1.$$

The probability $p_0$ then follows from $\sum_{k=0}^{+\infty} p_k = 1$.

Members of this class are the:

- Poisson
- Binomial
- Negative Binomial
- Geometric distribution.



Relevant claim frequency distributions in the $(a, b, 0)$ class:

| **Member** | **slope** $a$ | **Probability function** $p_k$ | **Parameter(s)** | **Moments** |
|---|---|---|---|---|
| Binomial | $a < 0$ | $\frac{n!}{k!(n-k)!} \cdot p^k \cdot (1-p)^k$ | $p \in (0, 1)$ | $E(N) = n \cdot p$ |
| | | | $n \in \mathbb{N}$ | $\mathrm{Var}(N) = n \cdot p \cdot (1-p)$ |
| Poisson | $a = 0$ | $e^{-\lambda} \cdot \frac{\lambda^k}{k!}$ | $\lambda \in (0, \infty)$ | $E(N) = \lambda$ |
| | | | | $\mathrm{Var}(N) = \lambda$ |
| Negative Binomial | $a > 0$ | $\frac{\Gamma(r+k)}{k!\Gamma(r)} \cdot \left(\frac{r}{r+\mu}\right)^r \cdot \left(\frac{\mu}{r+\mu}\right)^k$ | $\mu \in (0, \infty)$ | $E(N) = \mu$ |
| | | | $r \in (0, \infty)$ | $\mathrm{Var}(N) = \mu + \frac{\mu^2}{r}$ |

# The (a, b, 0) class: a visual check

To check the $(a, b, 0)$ relation for `mtpl$nclaims` we start from the empirical count distribution

```
mtpl$nclaims %>% table %>% prop.table
```

We store the empirical probabilities $\hat{p}_k$ in a vector

```
empirical <- mtpl$nclaims %>%
  table %>% prop.table %>% as.numeric
```

and the $k \cdot \frac{p_k}{p_{k-1}}$

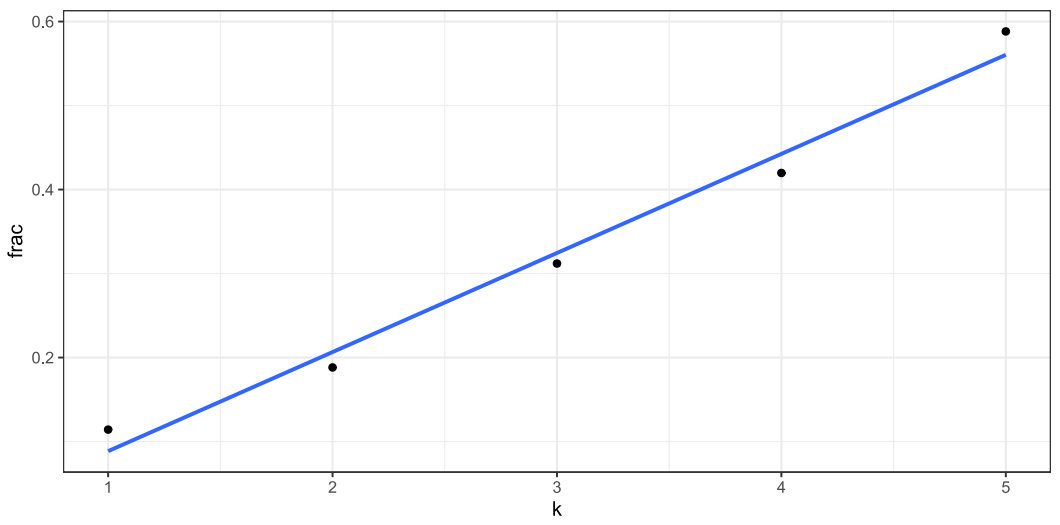
```
k <- 1:(length(empirical) - 1)
ab0_relation <- empirical[k+1] / empirical[k] * k
```

```
ab0_data <- tibble(k = k, ab0_rel = ab0_relation)
```

The positive slope when fitting $a \cdot k + b$ indicates a NegBin distribution for `mtpl$nclaims`, see

```
ab0_data %>% lm(ab0_rel ~ k, data = .)
```

and visually



# Empirical mean and variance of the claim frequency

Let's explore different ways to calculate the empirical mean claim frequency.

Inspect the following instructions and list pros and cons:

```
mean(mtpl$nclaims)
sum(mtpl$nclaims)/sum(mtpl$expo)
weighted.mean(mtpl$nclaims/mtpl$expo, mtpl$expo)
mtpl %>%
  summarize(emp_freq = sum(nclaims) / sum(expo))
```

```
## [1] 0.1239715
## [1] 0.1393352
## [1] 0.1393352
```

| emp_freq |
|---|
| 0.1393352 |

What about the empirical variance? Overdispersion!

```
m <- sum(mtpl$nclaims)/sum(mtpl$expo)
m
## [1] 0.1393352
var <- sum((mtpl$nclaims - m * mtpl$expo)^2)/
                sum(mtpl$expo)
var
## [1] 0.1517246
```

Here we use the expression for the variance:

$$\mathrm{Var}(X) = \mathrm{E}[(X - \mathrm{E}X)^2]$$

We empirically calculate the first expected value by taking exposure into account.

Moreover, we compare each realization of `nclaims` with its expected value, i.e. `m * expo` where `m` is the empirical mean claim frequency.

# Maximum likelihood estimation (MLE)

Fit a distribution to the data by **maximizing the likelihood** over the unknown parameter vector $\theta$

$$\mathcal{L}(\theta) = \prod_i Pr(N_i = n_i \mid \theta).$$

In practice, we minimize the negative log-likelihood

$$L(\theta) = -\sum_i \log(Pr(N_i = n_i \mid \theta)).$$

If exposure is available, we'll typically incorporate the exposure measure in the mean of the distribution

$$L(\theta) = -\sum_i \log(Pr(N_i = n_i \mid \theta,\ e_i)).$$

Let's make this more concrete for the Poisson distribution:

$$N_i \sim \texttt{Poi}(e_i \cdot \lambda).$$

The corresponding log-likelihood becomes:

$$\sum_i n_i \cdot \log(e_i \cdot \lambda) - e_i \cdot \lambda - n_i!$$

How about **fitting a parametric count distribution** to data in R?

# Maximum likelihood estimation (MLE) in R

A (naive) first solution uses `fitdistr(.)` from the {MASS} library

```
library(MASS)
fitdistr(mtpl$nclaims, "poisson")
##       lambda
##   0.1239715495
##  (0.0008714846)
```

Which number do you recognize here for $\hat{\lambda}$?

Alternatively, we can use the `glm(.)` function in a smart way:

```
freq_glm_poi <- glm(nclaims ~ 1,
                    family = poisson(link = "log"),
                    data = mtpl)
```

Fit a **Poisson GLM**, with **logarithmic link** function.

This implies:

$Y$ ~ Poisson, with only an intercept in the linear predictor

$$\log(E[Y]) = \beta_0,$$

or,

$$E[Y] = \lambda = \exp(\beta_0).$$

Fit this model on `data = mtpl`.

```
freq_glm_poi <- glm(nclaims ~ 1, offset = log(expo),
                    family = poisson(link = "log"),
                    data = mtpl)
freq_glm_poi %>% broom::tidy()
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | -1.970873 | 0.0070297 | -280.3632 | 0 |

What is your estimate for the expected annual claim frequency?

Use `nclaims` as $Y$.

Use only an intercept in the linear predictor `~ 1`.

Include `log(expo)` as an offset term in the linear predictor.

Then,

$$\log\left(\mathbf{expo}\right) + \beta_0.$$

Put otherwise,

$$E[Y] = \mathbf{expo} \cdot \exp\left(\beta_0\right),$$

where **`expo`** refers to `expo` the exposure variable.

# Maximum likelihood estimation (MLE) in R (continued)

**Writing a function** for the negative log-likelihood is another option.

The general framework for a one-parameter distribution then becomes

```
neg_loglik_distr <- function(par, freq, expo){
  -sum(ddistr(freq, par, log = T))
}
nlm(neg_loglik_distr, 1, hessian = TRUE,
    freq = mtpl$nclaims, expo = mtpl$expo)
```

Tweaking the general framework to the Poisson setting then becomes ...

```
neg_loglik_pois <- function(par, freq, expo){
  lambda <- expo*exp(par)
  -sum(dpois(freq, lambda, log = T))
}
sol_poi <- nlm(neg_loglik_pois, 1, hessian = TRUE,
               freq = mtpl$nclaims, expo = mtpl$expo)
```

Inspect the results

```
exp(sol_poi$estimate)
## [1] 0.1393351
sqrt(diag(solve(sol_poi$hessian)))
## [1] 0.007029369
sol_poi$minimum
## [1] 63837.82
```

Do you recognize these numbers?

We now focus on fitting the Negative Binomial distribution to the `mtpl$nclaims` frequency data, using `glm.nb()` from the {MASS} library.

```
library(MASS)
freq_glm_nb <- glm.nb(nclaims ~ 1 +
                        offset(log(expo)),
                      link = log,
                      data = mtpl)
freq_glm_nb %>% broom::tidy()
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | -1.968155 | 0.0073391 | -268.174 | 0 |

The additional parameter $r$ is estimated as (point estimate and standard error)

```
## [1] 1.435807
## [1] 0.08039722
```

Inspect the model output via

```
summary(freq_glm_nb)
##
## Call:
## glm.nb(formula = nclaims ~ 1 + offset(log(expo)), da
##     link = log, init.theta = 1.435806632)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5164  -0.5164  -0.5164  -0.4263   4.8511
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.968155   0.007339  -268.2   <2e-16 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
##
## (Dispersion parameter for Negative Binomial(1.4358)
##
##     Null deviance: 78254  on 163230  degrees of free
## Residual deviance: 78254  on 163230  degrees of free
## AIC: 127192
##
## Number of Fisher Scoring iterations: 1
##
```

# Your turn

**Q**: you will work through the following steps

1. Starting from the handcrafted Poisson likelihood and its numerical optimization, can you write similar instructions for the Negative Binomial?
2. Can you match the estimates obtained with your routine with those of the `glm.nb()` call?

Let's now focus on the handcrafted Negative Binomial likelihood optimization:

```
neg_loglik_nb <- function(par, freq, expo){
  mu <- expo*exp(par[1])
  r <- exp(par[2])
  -sum(dnbinom(freq, size = r, mu = mu, log = TRUE))
}
sol_nb <- nlm(neg_loglik_nb, c(1, 1), hessian = TRUE,
              freq = mtpl$nclaims, expo = mtpl$expo)
```

Inspect the resulting estimates

```
sol_nb$estimate
## [1] -1.9681557  0.3617181
```

We inspect the results

```
exp(sol_nb$estimate)
## [1] 0.1397143 1.4357941
sqrt(diag(solve(sol_nb$hessian)))
## [1] 0.007349782 0.056009902
sol_nb$minimum
## [1] 63593.93
```

The variance estimated by this Negative Binomial model is then

```
exp(sol_nb$estimate[1]) +
(exp(sol_nb$estimate[1])^2)/(exp(sol_nb$estimate[2]))
## [1] 0.1533096
```

which is very close to the empirical variance!

# Model selection and evaluation tools

The AIC can be used to compare models

$$AIC = 2 \cdot \#\text{param} - 2 \cdot \text{log-likelihood}.$$

```
AIC_poi <- 2*length(sol_poi$estimate) + 2*sol_poi$minimum
AIC_nb <- 2*length(sol_nb$estimate) + 2*sol_nb$minimum
c(AIC_nb = AIC_nb, AIC_poi = AIC_poi)
##   AIC_nb  AIC_poi
## 127191.9 127677.6
```

Smaller is better, thus preferred distribution is the Negative Binomial based on AIC!

Next to this, we compare the observed and fitted claim count distribution.

For example, with the fitted Negative Binomial distribution

```
observed_count = rep(0, 5)
model_based_count = rep(0, 5)

for(i in 1:5) {
  observed_count[i] = sum(mtpl$nclaims == i-1)
  model_based_count[i] = sum(dnbinom(i-1, size = freq_glm_nb$theta, mu = fitted(freq_glm_nb)))
}

data.frame(frequency = 0:4,
           observed_count = observed_count,
           model_based_count = round(model_based_count))
##   frequency observed_count model_based_count
## 1         0         144936            145014
## 2         1          16556             16347
## 3         2           1558              1685
## 4         3            162               167
## 5         4             17                16
```

How would you explain the code (in your own words)? Can you adjust the code to the Poisson distribution?

# Excess zeroes in claim count data

Standard count distributions have often difficulty to capture the large number of zeroes in claim frequency data.

We propose two strategies to model an excessive number of zeroes.

With a Zero-Inflated (ZI) distribution:

$$P(N^{ZI} = k) = \begin{cases} \pi + (1-\pi) \cdot P(N=0) & k = 0 \\ (1-\pi) \cdot P(N=k) & k > 0. \end{cases}$$

Zero-inflated count models are two-component mixture models combining a point mass at zero with a proper count distribution. Thus, there are two sources of zeroes: zeroes may come from both the point mass and from the count component.

With a Hurdle distribution:

$$P(N^{hurdle} = k) = \begin{cases} \pi & k = 0 \\ (1-\pi) \cdot \frac{P(N=k)}{1-P(N=0)} & k > 0. \end{cases}$$

How can we fit these distributions to given data with R?

The library {pscl} enables maximum likelihood estimation of zero-inflated and hurdle models for count data.

```
library(pscl)
f_ZIP <- zeroinfl(nclaims ~ 1, offset = log(expo),
                  dist = "poisson", data = mtpl)
summary(f_ZIP)
##
## Call:
## zeroinfl(formula = nclaims ~ 1, data = mtpl, offset
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.3584 -0.3584 -0.3584 -0.2976 36.1142
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.48286    0.02226  -66.63   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with log
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.46932    0.05483   -8.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
##
## Number of iterations in BFGS optimization: 11
```

The `dist = .` available in `zeroinfl(.)` are the Poisson, Negative Binomial and the geometric distribution.

The `offset = .` is an a priori known component to be included in the linear predictor of the count model (as we did before).

Usually the count model is a Poisson or negative binomial regression (with log link).

A binary model is used that captures the probability of zero inflation. In the simplest case only with an intercept but potentially containing regressors. For this zero-inflation model, a binomial model with different links can be used, typically logit or probit.

More detailed inspection of the output stored in `f_ZIP`

```
f_ZIP$coefficients
## $count
## (Intercept)
##   -1.482859
##
## $zero
## (Intercept)
##  -0.4693167
(f_ZIP_lambda <- exp(as.numeric(f_ZIP$coefficients[1])))
## [1] 0.2269878
(f_ZIP_pi <- exp(as.numeric(f_ZIP$coefficients[2]))/(1+exp(as.numeric(f_ZIP$coefficients[2]))))
## [1] 0.384778
(AIC_ZIP <- 2*length(f_ZIP$coefficients) - 2*f_ZIP$loglik)
## [1] 127241.3
```

The mean and variance (for `expo == 1`) as captured by the ZIP

```
(ZIP_mean <- (1-f_ZIP_pi)*f_ZIP_lambda)
## [1] 0.1396479
(ZIP_var <- (1-f_ZIP_pi)*f_ZIP_lambda*(1+f_ZIP_pi*f_ZIP_lambda))
## [1] 0.1518447
```

A handcrafted MLE with the ZIP distribution can be put together as

```
neg_loglik_ZIP <- function(par, freq, expo){
  lambda <- expo*exp(par[1])
  p <- exp(par[2])/(1+exp(par[2]))

  -sum((freq == 0) * (log(p + (1-p)*dpois(0, lambda, log = FALSE)))) -
    sum((freq != 0) * (log((1-p)) + dpois(freq, lambda, log = TRUE)))
}

sol_ZIP <- nlm(neg_loglik_ZIP, c(1, 1), hessian = TRUE,
               freq = mtpl$nclaims, expo = mtpl$expo)
sol_ZIP$estimate
## [1] -1.4829209 -0.4694681
```

and their corresponding standard errors

```
(sol_ZIP_se <- sqrt(diag(solve(sol_ZIP$hessian))))
## [1] 0.02224688 0.05480811
```

# Your turn

**Q**: as a final challenge

1. Adjust the code of the ZIP to fit the hurdle Poisson model, use `pscl::hurdle()`.
2. Starting from the handcrafted ZIP likelihood, write code for the hurdle Poisson model.

# Fitting severity models in R

# From simple to complex parametric distributions

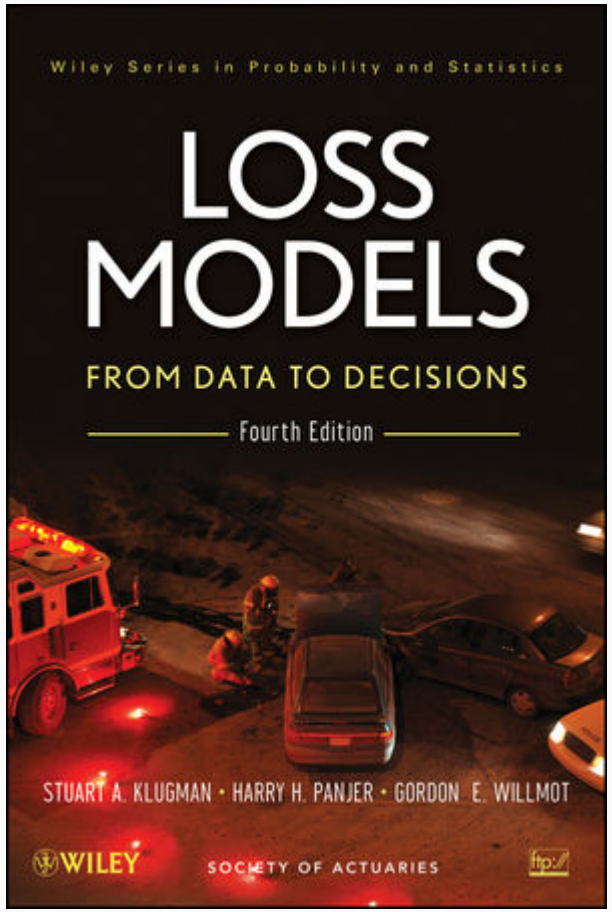
Famous examples of 1 and 2 parameter distributions used in loss modelling:

- Exponential
- Lognormal
- Gamma and Inverse Gaussian.

Some less straightforward more flexible parametric distributions:

- Burr (3 parameters)
- GB2 (4 parameters).

Typically challenging to fit, picking meaningful starting values in numerical optimization routines matters!



| Distribution | Density | Mean | R |
|---|---|---|---|
| Exponential | $f(y) = \lambda e^{-\lambda y}$ | $\mathrm{E}[Y] = 1/\lambda$ | `dexp` ... |
| Gamma | $f(y) = \frac{1}{\Gamma(\alpha)} \beta^{\alpha} y^{\alpha-1} e^{-\beta y}$ | $\mathrm{E}[Y] = \alpha/\beta$ | `dgamma` ... |
| Inverse Gaussian | $f(y) = \left( \frac{\lambda}{2\pi y^3} \right)^{1/2} \exp\left[ \frac{-\lambda(y-\mu)^2}{2\mu^2 y} \right]$ | $\mathrm{E}[Y] = \mu$ | `dinvgaus` package {statmod} |
| Lognormal | $f(y) = \frac{1}{\sqrt{2\pi}\sigma y} \exp\left[ -\frac{1}{2} \left( \frac{\log y - \mu}{\sigma} \right)^2 \right]$ | $\mathrm{E}[Y] = \exp\left( \mu + \frac{1}{2}\sigma^2 \right)$ | `dlnorm` ... |

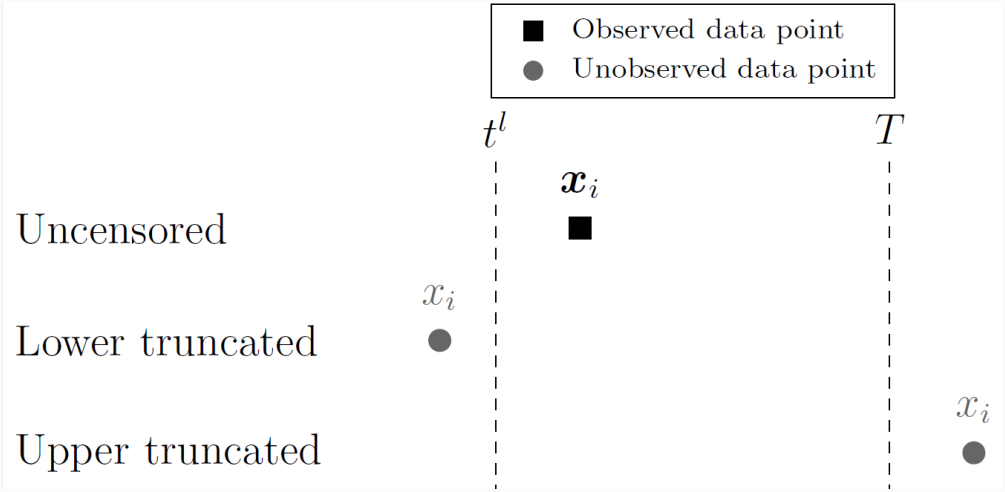
The {actuar} package has some functions related to the Burr and GB2 distributions, e.g. `dburr` and `dgenbeta`.

Mind the different parametrizations of these distributions!

# Be aware of truncation!

In `secura` only losses above 1.2M EUR are registered.

We can picture this as follows:

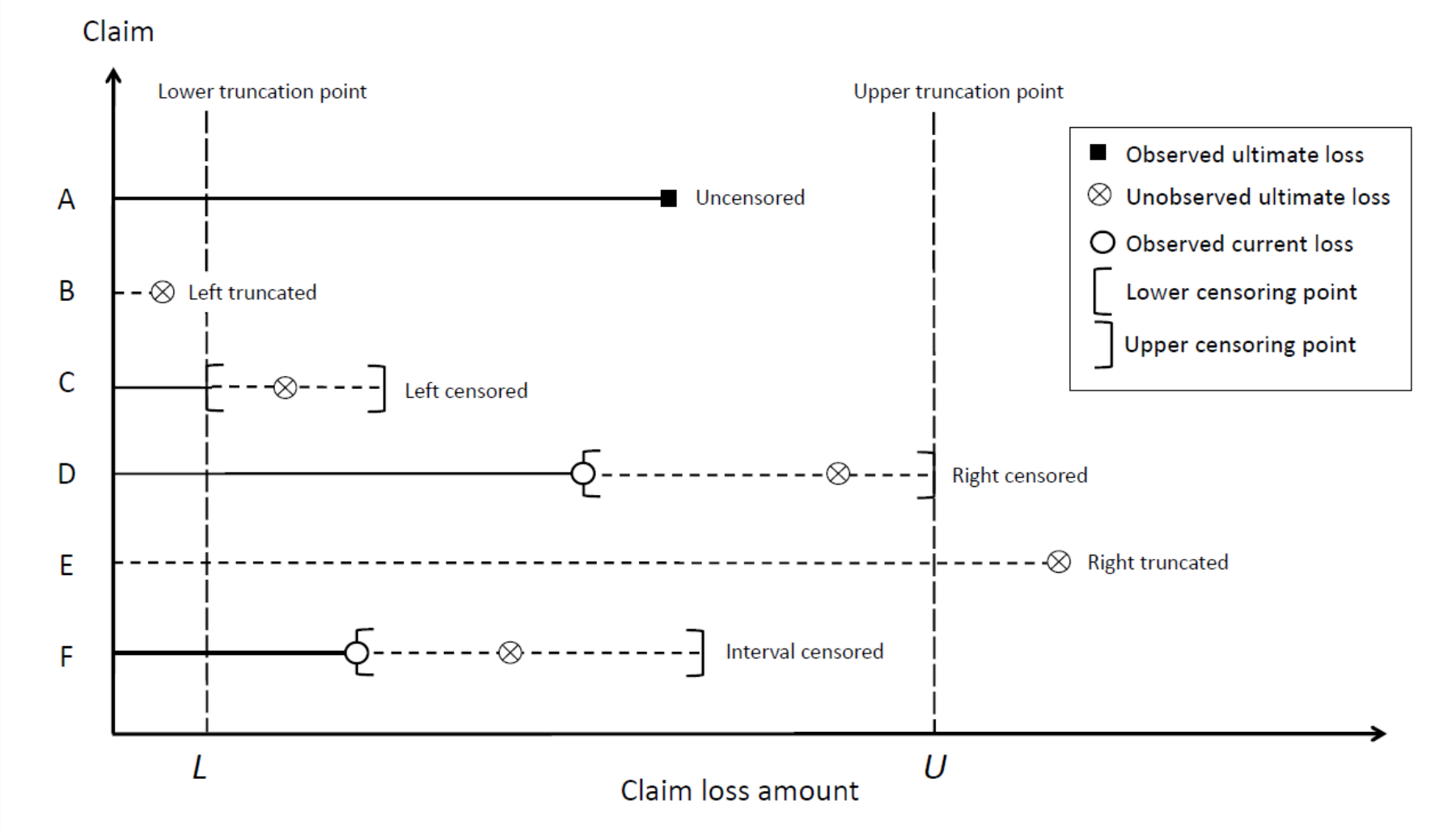



Lower/left truncation: deductible, reinsurance.

Upper/right truncation: earthquake magnitudes.

How would you tackle this left-truncation in `secura` when fitting a loss distribution to the data?

# Be aware of truncation and censoring!



# Global fitting distributions

How to find/to construct a **global fit** of insurance losses?

That's **hard**, because:

- different behaviour of attritional and large losses (body vs tail).

Typically, no standard parametric distribution provides suitable global fit:

- Lognormal, Weibull, etc. underestimate tail risk
- Pareto, Generalised Pareto Distribution (GPD), etc. can only be used for large losses
- complex distributions (e.g. GB2).

**Splicing (or composite modeling)** offers a useful strategy to construct a global fit.

Intuitively:

- different components on different intervals of losses
- glue these components together into a well-defined density.

This technique is popular e.g. in modelling operational risk data.

# Splicing: a body-tail example

Combine two distributions (one for the **body** and one for the **tail**) in a splicing model:

$$f(x;\mathbf{\Theta}) = \begin{cases} 0 & \text{if } x < t^l \\ \pi \frac{f_1^*(x;t,\mathbf{\Theta}_1)}{F_1^*(t;\mathbf{\Theta}_1) - F_1^*(t^l;\mathbf{\Theta}_1)} & \text{if } t^l \leq x \leq t \\ (1-\pi)\frac{f_2^*(x;t,\mathbf{\Theta}_2)}{1-F_2^*(t;\mathbf{\Theta}_1)} & \text{if } x > t, \end{cases}$$

where $t^l$ is the **left-truncation point** and $t$ the **split point** (body vs. tail).

Questions for you:

- which in the above should be estimated from the data?
- why the conditional densities, constructed from original pdf's $f_1^*(.)$ and $f_2^*(.)$ on the positive real line?

Some examples of body-tail fits:

| **Body** | **Tail** |
|---|---|
| Exponential | Pareto |
| Lognormal | Pareto |
| Weibull | Pareto |
| Mixture of two exponentials | Generalised Pareto |

# A body-tail fit for the Secura Re losses

Let's combine an exponential (body - below threshold $t$) and a Pareto (tail - above $t$)

$$f(x) = \begin{cases} 0 & x \leq t^l \\ \pi \cdot \frac{f_1^\star(x)}{F_1^\star(t) - F_1^\star(t^l)} & t^l \leq x \leq t \\ (1-\pi) \cdot \frac{f_2^\star(x)}{1 - F_2^\star(t)} & x > t \end{cases},$$

with

- $f_1^\star$ and $F_1^\star$ the PDF and CDF of an exponential distribution

$$\frac{f_1^\star(x)}{F_1^\star(t) - F_1^\star(t^l)} = \frac{\lambda \exp\{-\lambda x\}}{\exp\{-\lambda \cdot t^l\} - \exp\{-\lambda \cdot t\}} = \frac{\lambda \exp\{-\lambda(x - t^l)\}}{1 - \exp\{-\lambda(t - t^l)\}},$$

- $f_2^\star$ and $F_2^\star$ the PDF and CDF of a Pareto distribution with unit scale

$$\frac{f_2^\star(x)}{1 - F_2^\star(t)} = \frac{\frac{1}{\gamma} x^{-\frac{1}{\gamma}-1}}{t^{-\frac{1}{\gamma}}}.$$

# Choice of threshold

Techniques from **extreme value theory (EVT)** are necessary to estimate split point above which a Pareto distribution is plausible.

A first visual tool is the **mean excess plot**:

- the mean excess function or mean residual life function is

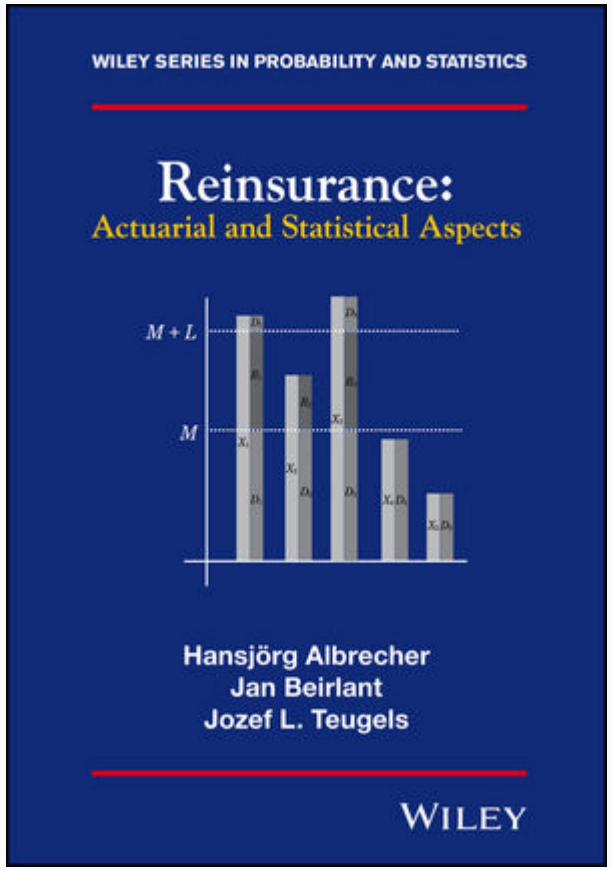
$$e(t) = E(X - t | X > t)$$

- empirically, using a sample $(x_1, \ldots, x_n)$

$$\hat{e}_n(t) = \frac{\sum_{i=1}^n x_i \mathbf{1}_{(t,\infty)}(x_i)}{\sum_{i=1}^n \mathbf{1}_{(t,\infty)}(x_i)} - t.$$

- the mean excess plot evaluates the above at values $t = x_{n-k,n}$, the $(k+1)$th largest observation.

The {ReIns} package implements many useful functions from this book:



Via {ReIns} we can easily plot the mean excess function:

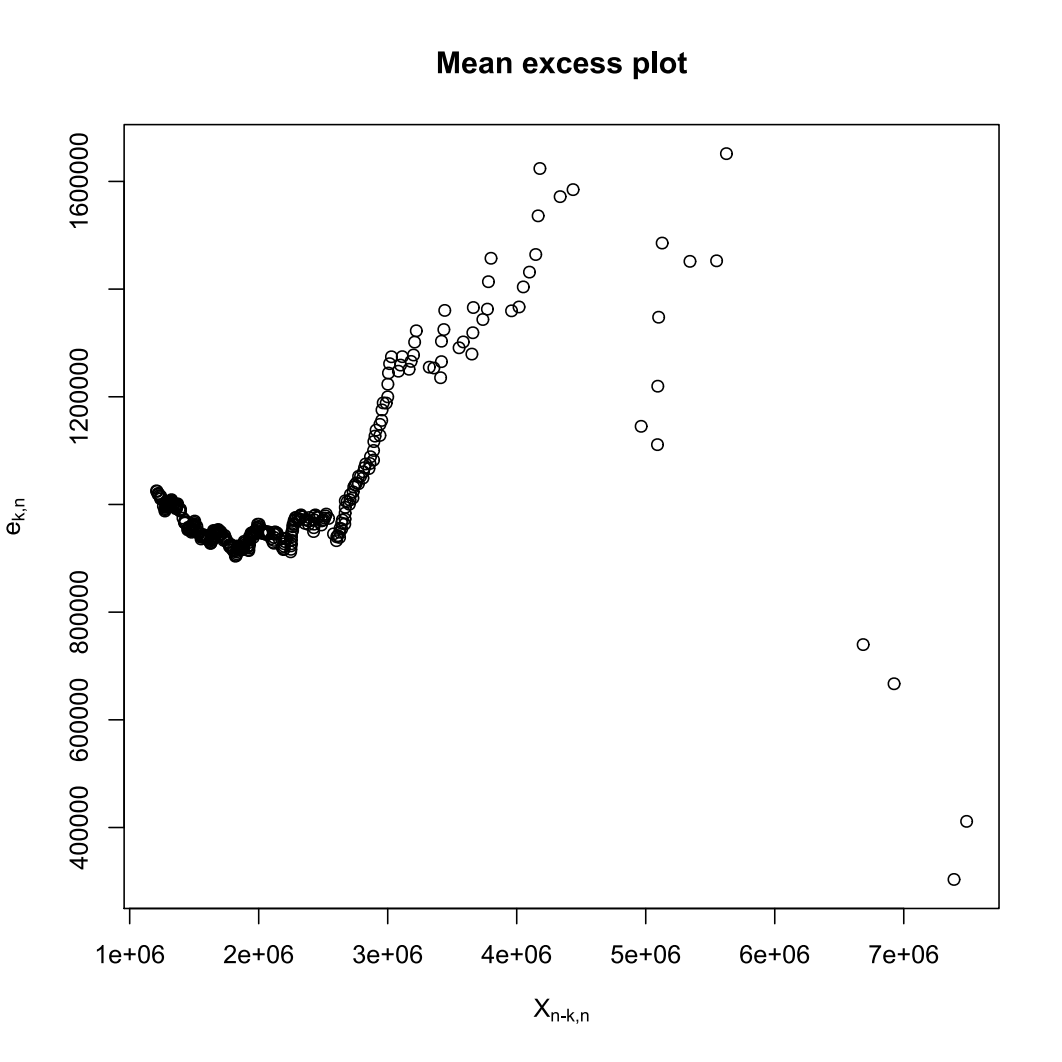
```
library(ReIns)
ReIns::MeanExcess(secura$Loss)
```

By default, the mean excess scores are plotted as a function of the data `k = FALSE`.

You can also plot these as function of the tail parameter `k = TRUE`.

The mean excess function of an EXP distribution is constant, then:

- HTE ('Heavier than exponential'), mean excess function ultimately increases
- LTE ('Lighter than exponential'), when mean excess function ultimately decreases.

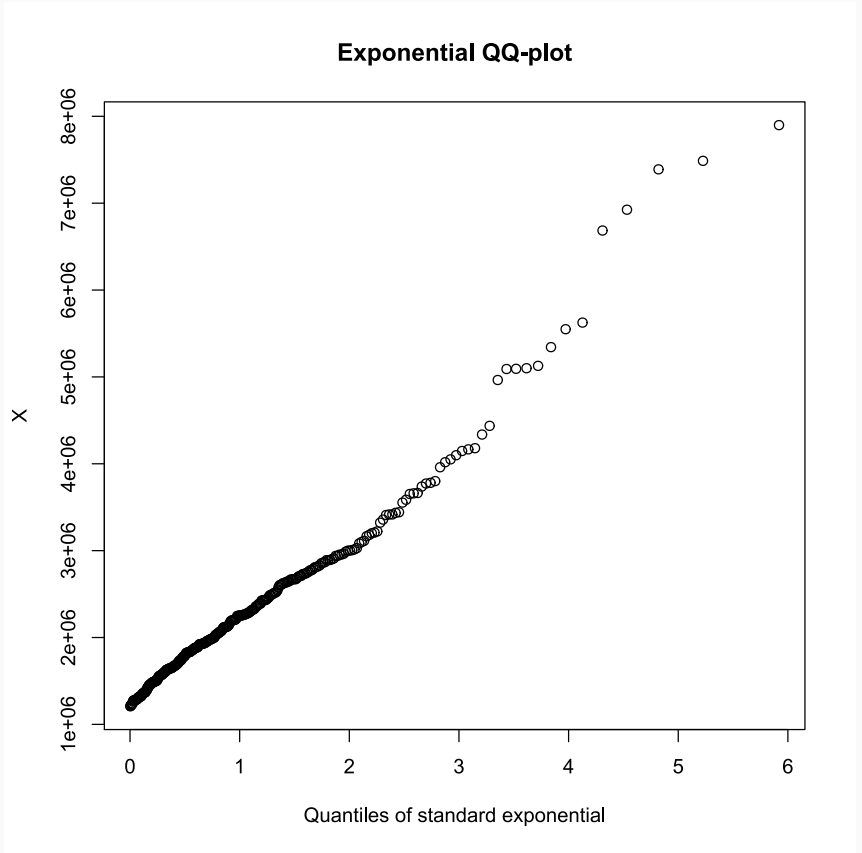


A second visual tool is a QQplot, e.g. the **Exponential QQplot**:

- recall $F_\lambda(x) = 1 - \exp(-\lambda x)$ with $x > 0$
- then $Q_\lambda(p) = -\frac{1}{\lambda}\log(1-p)$ for $p \in (0,1)$
- use $p_{i,n} := i/(n+1)$ and plot

$$(-\log(1-p_{i,n}), x_{i,n}).$$

```
ReIns::ExpQQ(secura$Loss)
```



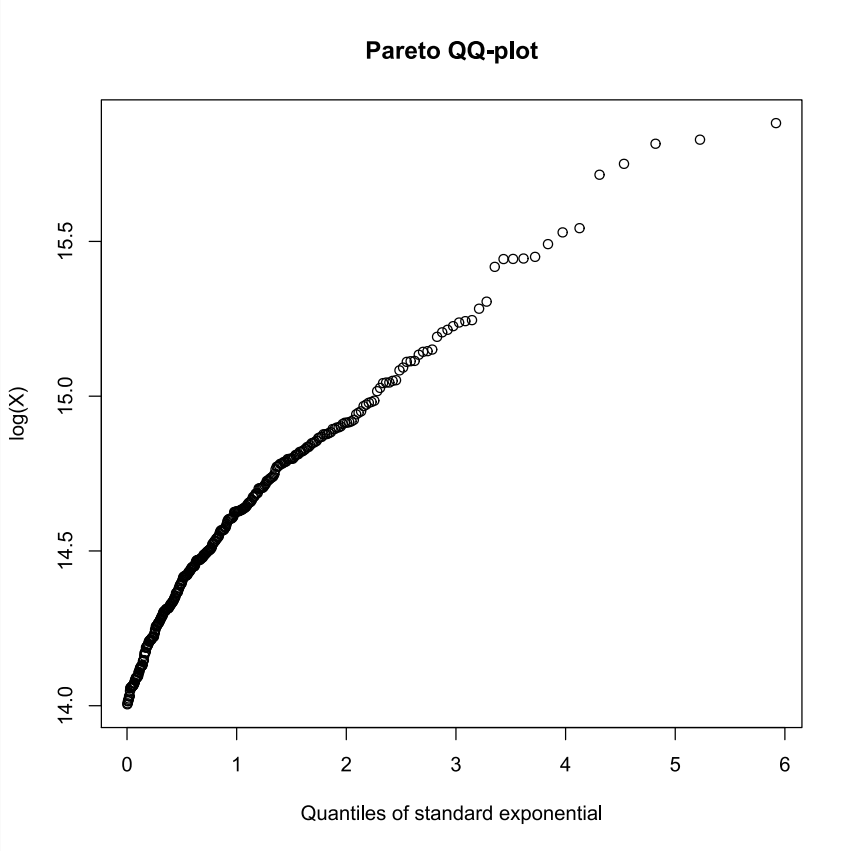
A second visual tool is a QQplot, e.g. the Pareto QQplot:

- recall $F_\gamma(x) = 1 - x^{-1/\gamma}$ with $x > 0$
- then $-\log(1-p) = \frac{1}{\gamma}\log x$ for $p \in (0,1)$
- use $p_{i,n} := i/(n+1)$ and plot

$$(-\log(1-p_{i,n}), \log x_{i,n}).$$

A distribution is of Pareto-type when the Pareto QQplot is ultimately linear near the largest observations.

```
ReIns::ParetoQQ(secura$Loss)
```



# Choice of threshold - the Hill estimator

We estimate the slope of the Pareto QQplot to the right of the reference point

$$(-\log(\frac{k+1}{n+1}), \log x_{n-k,n}).$$

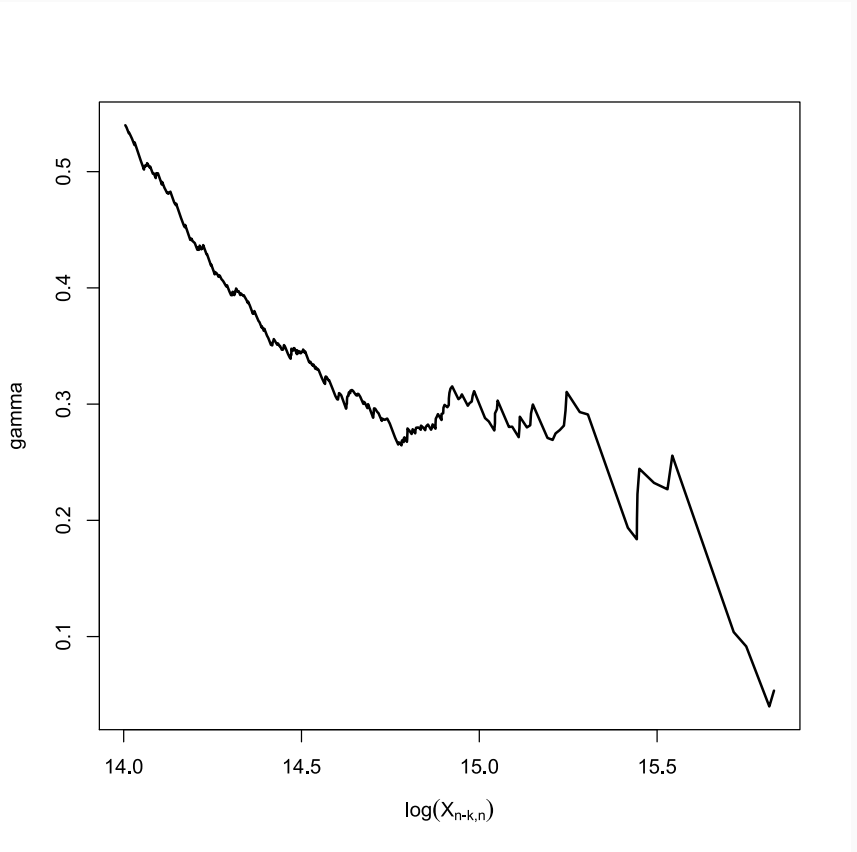
This slope can be expressed as

$$H_{k,n} = \frac{1}{k}\sum_{j=1}^{k} \log X_{n-j+1,n} - \log X_{n-k,n}.$$

When the data has a Pareto tail the Hill plot becomes linear.

But mind the bias and variance trade off!

```
H <- ReIns::Hill(secura$Loss, k = FALSE,
                 lwd = 2, plot = TRUE, main = "")
```



# Choice of threshold - the Hill estimator

We estimate the slope of the Pareto QQplot to the right of the reference point

$$(-\log(\frac{k+1}{n+1}), \log x_{n-k,n}).$$

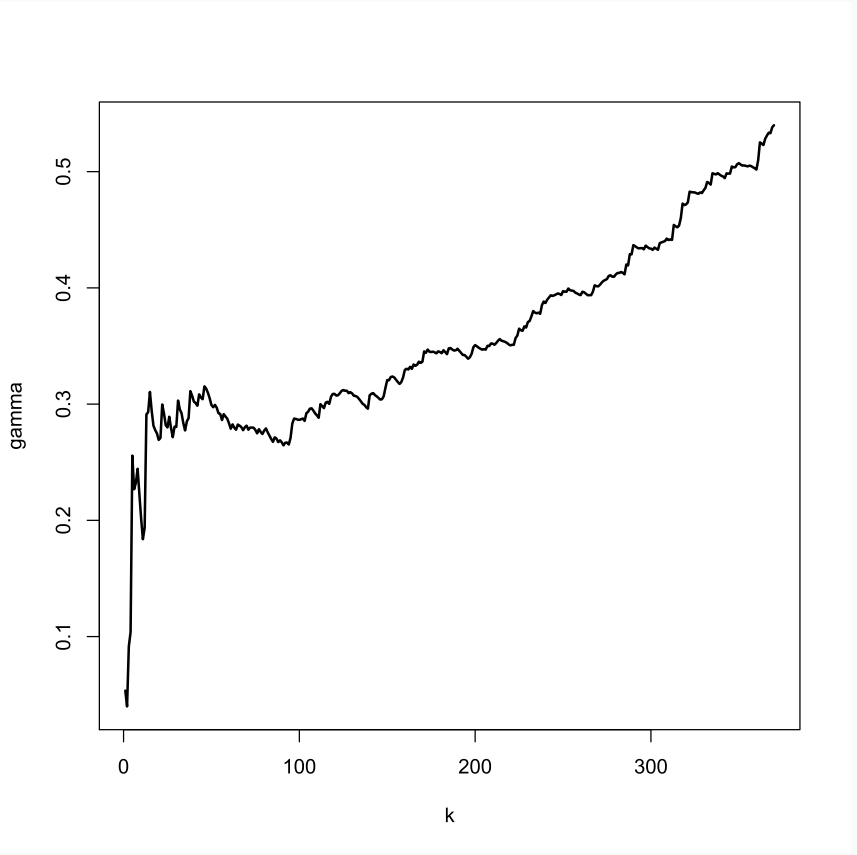
This slope can be expressed as

$$H_{k,n} = \frac{1}{k}\sum_{j=1}^{k} \log X_{n-j+1,n} - \log X_{n-k,n}.$$

When the data has a Pareto tail the Hill plot becomes linear.

But mind the bias and variance trade off!

```
H <- ReIns::Hill(secura$Loss, k = TRUE,
                 lwd = 2, plot = TRUE, main = "")
```



The Hill plot shows many possible estimates for $\gamma$ in the Pareto distribution $F(x) = 1 - x^{-1/\gamma}$ that we want to use for the tail of the distribution.

## How to pick the split point?

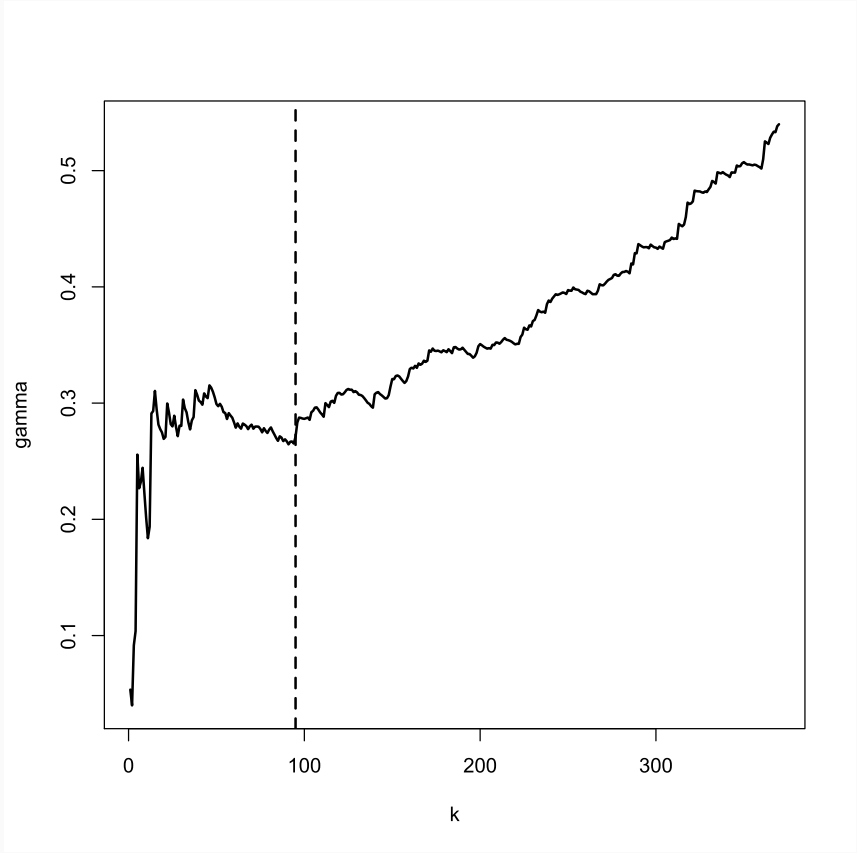
We consider the Asymptotic Mean Squared Error of $H_{k,n}$ (i.e. variance + squared bias).

We plot

$$(k, \widehat{\mathrm{AMSE}}(H_{k,n})); k = 1, \ldots, n-1\}$$

and pick the $k$ that minimizes the AMSE.

```
H <- ReIns::Hill(secura$Loss, lwd = 2, plot = TRUE,
                 main = "")
kopt <- ReIns::Hill.kopt(secura$Loss,
                         plot = FALSE)$kopt
abline(v = kopt, lwd = 2, lty = 2)
```



Using these insights from EVT we arrive at a final choice for the split point or threshold $t$.

We pick the optimal $k$ by minimizing AMSE of the Hill estimator.

Then, the corresponding split point or threshold $t := X_{n-k,n}$, the $(k+1)$th largest observation in the sample.

This reasoning also gives $\hat{\gamma}$ via the corresponding $H_{k,n}$.

```
# sample size
n <- length(secura$Loss)
n
## [1] 371

# Hill estimator
kopt <- Hill.kopt(secura$Loss, plot = FALSE)$kopt
kopt
## [1] 95

# chosen threshold
threshold <- sort(secura$Loss)[n - kopt]
threshold
## [1] 2580026

# estimate for gamma using Hill estimator
gamma <- H$gamma[kopt]
gamma
## [1] 0.2710874
```

# Exponential - Pareto splicing model

We are now ready to fit the **body-tail** model proposed earlier on:

$$f(x) = \begin{cases} 0 & x \leq t^l \\ \pi \cdot \frac{f_1^\star(x)}{F_1^\star(t) - F_1^\star(t^l)} & t^l \leq x \leq t \\ (1-\pi) \cdot \frac{f_2^\star(x)}{1 - F_2^\star(t)} & x > t \end{cases},$$

with

- $f_1^\star$ and $F_1^\star$ the PDF and CDF of an exponential distribution

$$\frac{f_1^\star(x)}{F_1^\star(t) - F_1^\star(t^l)} = \frac{\lambda \exp\{-\lambda x\}}{\exp\{-\lambda \cdot t^l\} - \exp\{-\lambda \cdot t\}} = \frac{\lambda \exp\{-\lambda(x - t^l)\}}{1 - \exp\{-\lambda(t - t^l)\}},$$

- $f_2^\star$ and $F_2^\star$ the PDF and CDF of a Pareto distribution with unit scale

$$\frac{f_2^\star(x)}{1 - F_2^\star(t)} = \frac{\frac{1}{\gamma} x^{-\frac{1}{\gamma}-1}}{t^{-\frac{1}{\gamma}}}.$$

In fact, we now have meaningful choices/estimates for:

- $t^l$ left truncation point at 1.2M EUR
- $t$ the split point estimated at `threshold`

```
threshold
## [1] 2580026
```

- the Hill estimator in `gamma`

```
gamma
## [1] 0.2710874
```

- the fraction of losses below the split point

```
(p <- sum(secura$Loss <= threshold)/n)
## [1] 0.7439353
```

or

```
(n-kopt)/n
## [1] 0.7439353
```

```r
sh <- 1200000 # shift
I <- secura$Loss <= threshold # indicator

neg.loglik <- function(par) {
    lambda <- exp(par[1]); alpha <- exp(par[2])
    # likelihood
    L <- I * (n-kopt)/n * lambda * exp(-lambda*(secura$Loss-sh))/(1 - exp(-lambda*(threshold-sh))) +
        (1-I) * kopt/n * alpha * (secura$Loss)^(-alpha-1)/(threshold)^(-alpha)
    # negative log-likelihood
    -sum(log(L))
}

m1 <- mean(secura$Loss)
par.init <- c(1/(m1-sh), 1/m1)
oo <- nlm(neg.loglik, log(par.init))
(lambda <- exp(oo$estimate[1]))
## [1] 6.710413e-07
(alpha <- exp(oo$estimate[2]))
## [1] 3.688812
(gamma <- 1/alpha)
## [1] 0.27109
```

By working out the integrals starting from the spliced density, we find:

for $t^l = 1,200,000 \leq x \leq t$:

$$F(x) = \pi \cdot \frac{1 - \exp\{-\lambda(x - 1{,}200{,}000)\}}{1 - \exp\{-\lambda(t - 1{,}200{,}000)\}}.$$

and for $x > t$ we find

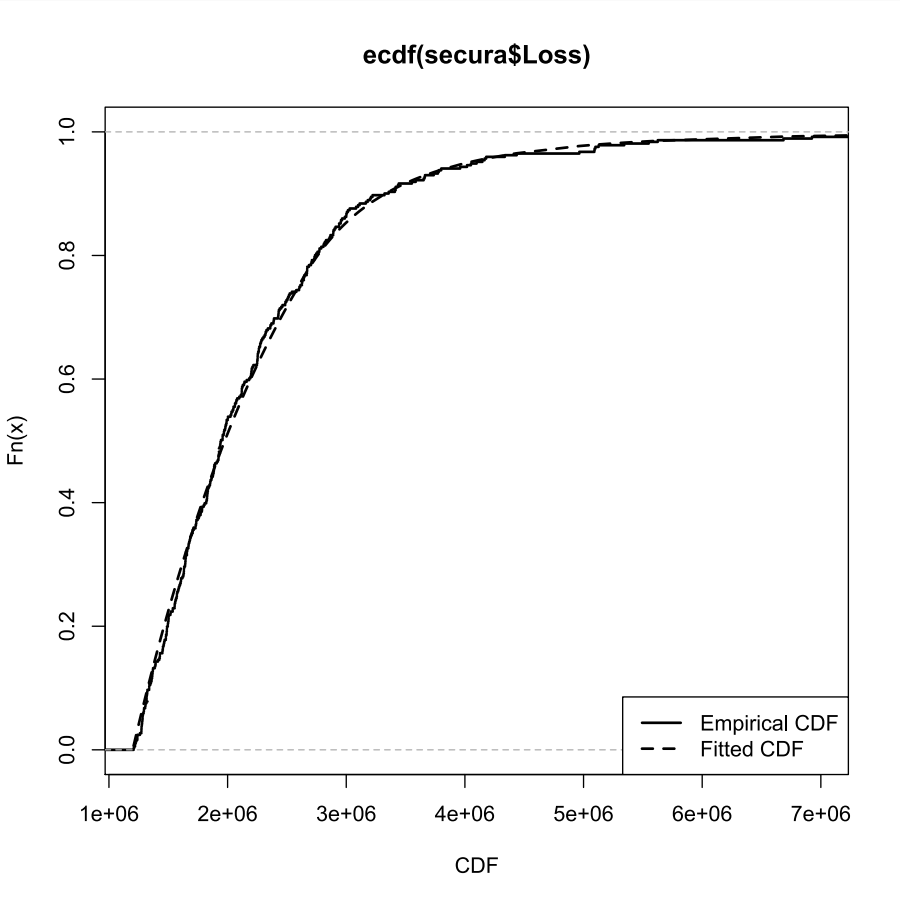
$$F(x) = 1 - (1 - \pi)\left(\frac{x}{t}\right)^{-\frac{1}{\gamma}}.$$

Note the different expressions for the CDF for the different ranges of $x$ values!

Following function can be used the compute the CDF of the spliced distribution (try the integrals!)

```
# CDF for Exp-Pa splicing model
ExpPa_cdf <- function(x, sh, threshold, lambda, gamma) {

  p <- numeric(length(x))

  p[x <= threshold] <- (n - kopt) / n * pexp(x[x <= threshold] - sh, rate = lambda) /
                                        pexp(threshold - sh, rate = lambda)

  p[x > threshold] <- 1 - kopt / n * (x[x > threshold] / threshold) ^ (-1/gamma)

  return(p)
}
```

To assess the GoF of the fitted spliced distribution, we plot the empirical CDF together with the fitted CDF.



# A mixture of Erlangs for the body

What if it is not straightforward to find a suitable distribution for the **body of the data**?

**Mixtures of Erlangs**:

- versatile class of distributions, i.e. dense in the space of positive, continuous distributions
- mathematically tractable
- fitting procedure in {ReIns} package.

But, a ME distribution has an **exponential tail**, thus no heavy tails!

More details in Verbelen et al. (2015, ASTIN Bulletin).

The pdf of a mixture of Erlangs:

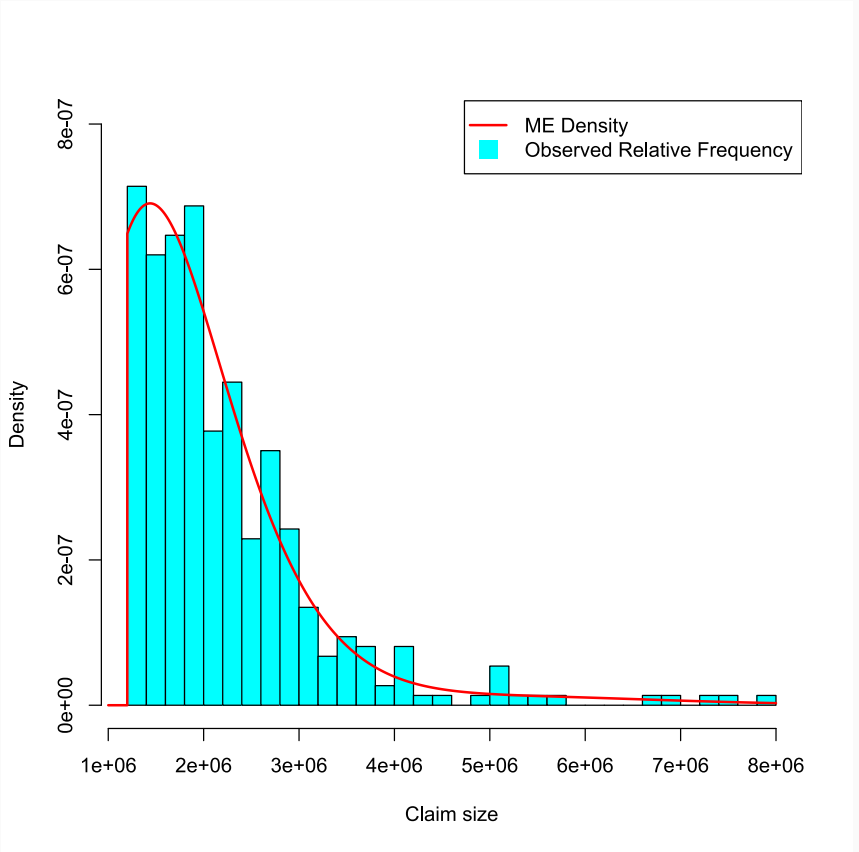
$$\begin{aligned} f_X(x;\boldsymbol{\alpha},\boldsymbol{r},\theta) &= \sum_{j=1}^{M} \alpha_j f_E(x;r_j,\theta) \\ &= \sum_{j=1}^{M} \alpha_j \frac{x^{r_j-1}e^{-x/\theta}}{\theta^{r_j}(r_j-1)!}, \end{aligned}$$

with number of Erlangs $M$, mixing weights $\boldsymbol{\alpha}$, common scale $\theta$ and positive integer shape parameters $(r_1,\ldots,r_M)$.

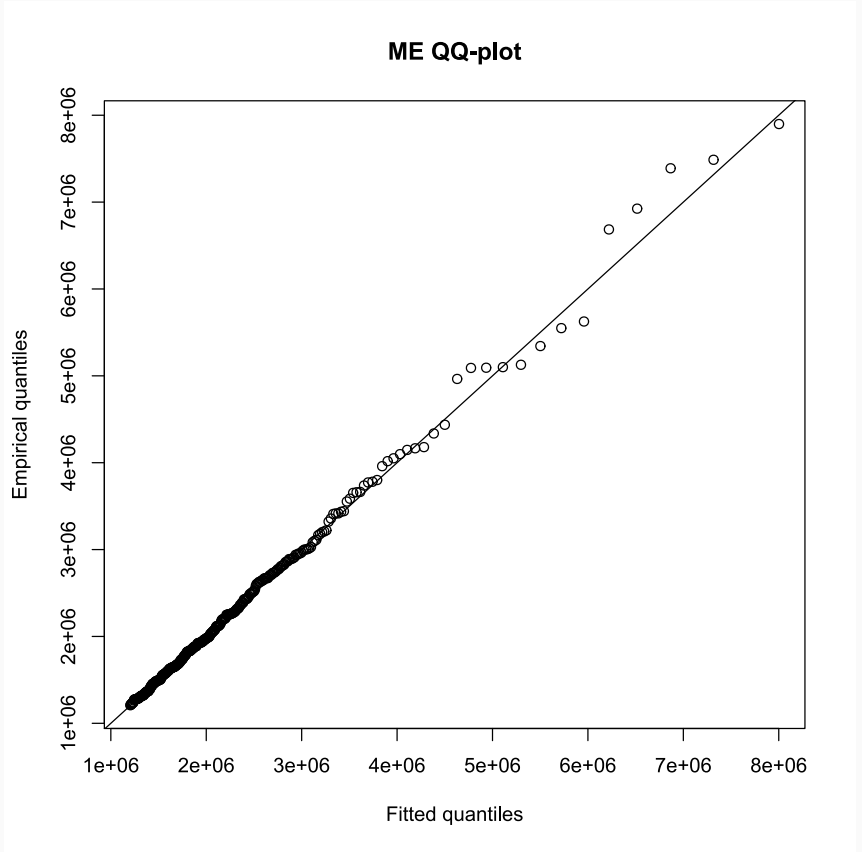
The mixing weights should satisfy $\alpha_j > 0$ and $\sum_j \alpha_j = 1$.

Question for you: the Erlang distribution may remind you of a better known distribution among actuaries. Which one?

```r
# Fit ME model using internal function from
# ReIns package
MEfit_sec <- ReIns:::.ME_tune(secura$Loss,
                              trunclower = sh,
                              criterium = "BIC",
                              M = 10,
                              s = 1:40)$best_model

# Fitted parameters
MEfit_sec$alpha
## [1] 0.9707281 0.0292719
MEfit_sec$shape
## [1]  5 16
MEfit_sec$theta
## [1] 359731.4
```



```r
ME_VaR <- Vectorize(ReIns:::.ME_VaR,
                    vectorize.args = "p")

# QQ-plot
plot(ME_VaR((1:n) / (n+1), theta = MEfit_sec$theta,
            shape = MEfit_sec$shape,
            alpha = MEfit_sec$alpha,
            trunclower = sh),
     sort(secura$Loss),
     main = "ME QQ-plot",
     xlab = "Fitted quantiles",
     ylab = "Empirical quantiles")
abline(0, 1)
```



Next, we combine what we learned on splicing with the Mixture of Erlangs distribution, to construct **a ME-Pa splicing model** with the splicing point `threshold` estimated earlier on.

Again, we start from $M = 10$ Erlangs in the mixture part for the body.

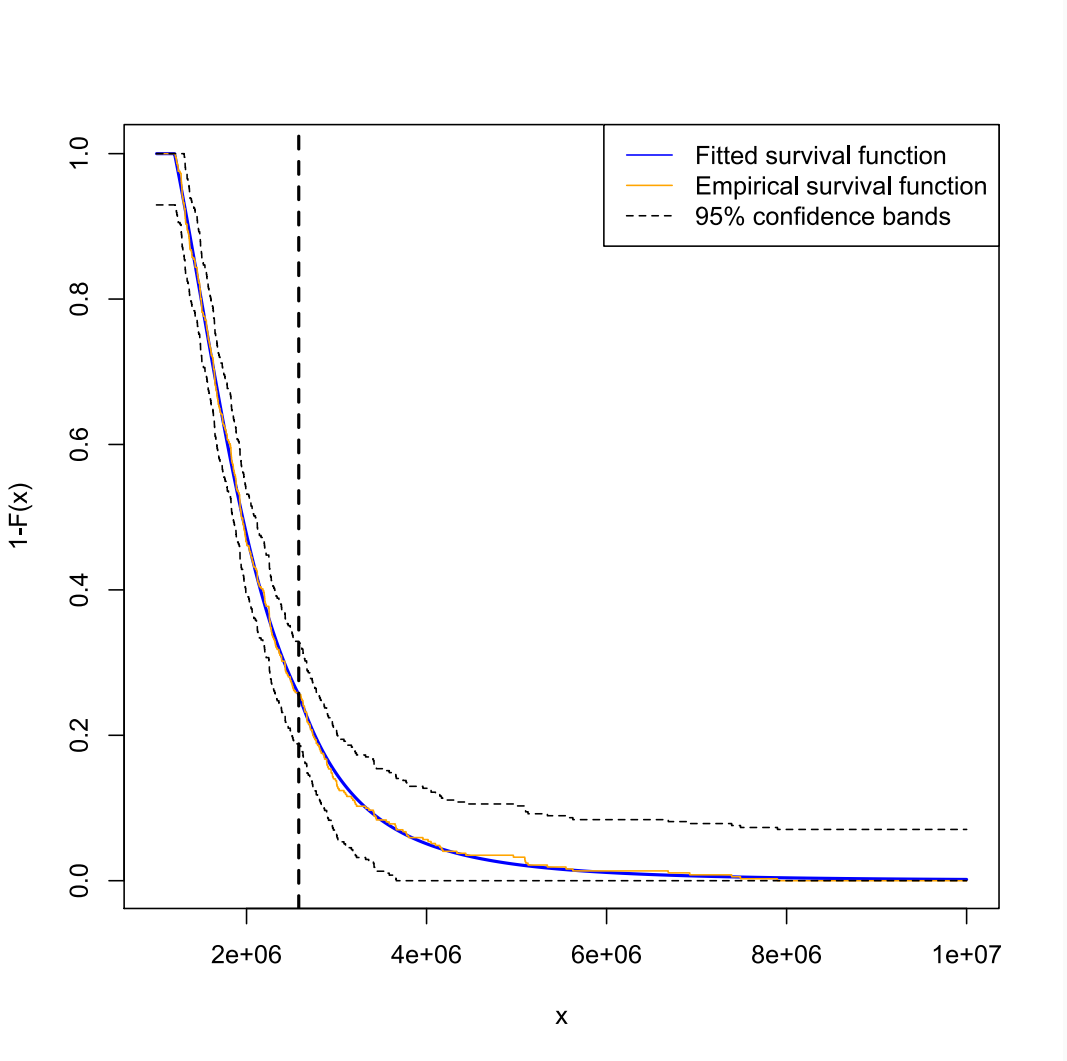
```
# Fit ME-Pa splicing model
splicefit_sec <- ReIns::SpliceFitPareto(
                    secura$Loss,
                    tsplice = threshold,
                    M = 10,
                    trunclower = sh)

# Summary
summary(splicefit_sec)
```

How would you describe the resulting model?

```
##
## ----------------------------------------
## Summary of splicing fit
## ----------------------------------------
##
## const = 0.744
##
## pi = (0.744, 0.256)
##
## t0 = 1 200 000
##
## t = 2 580 026
##
## type = (ME, Pa)
##
## * * * * * * * * * * * * * * * * * * * *
##
## p = 1
##
## r = 7
##
## theta = 249 142
##
## M = 1
##
## M_initial = 10
##
## * * * * * * * * * * * * * * * * * * * *
##
## gamma = 0.271
##
## endpoint = Inf
```

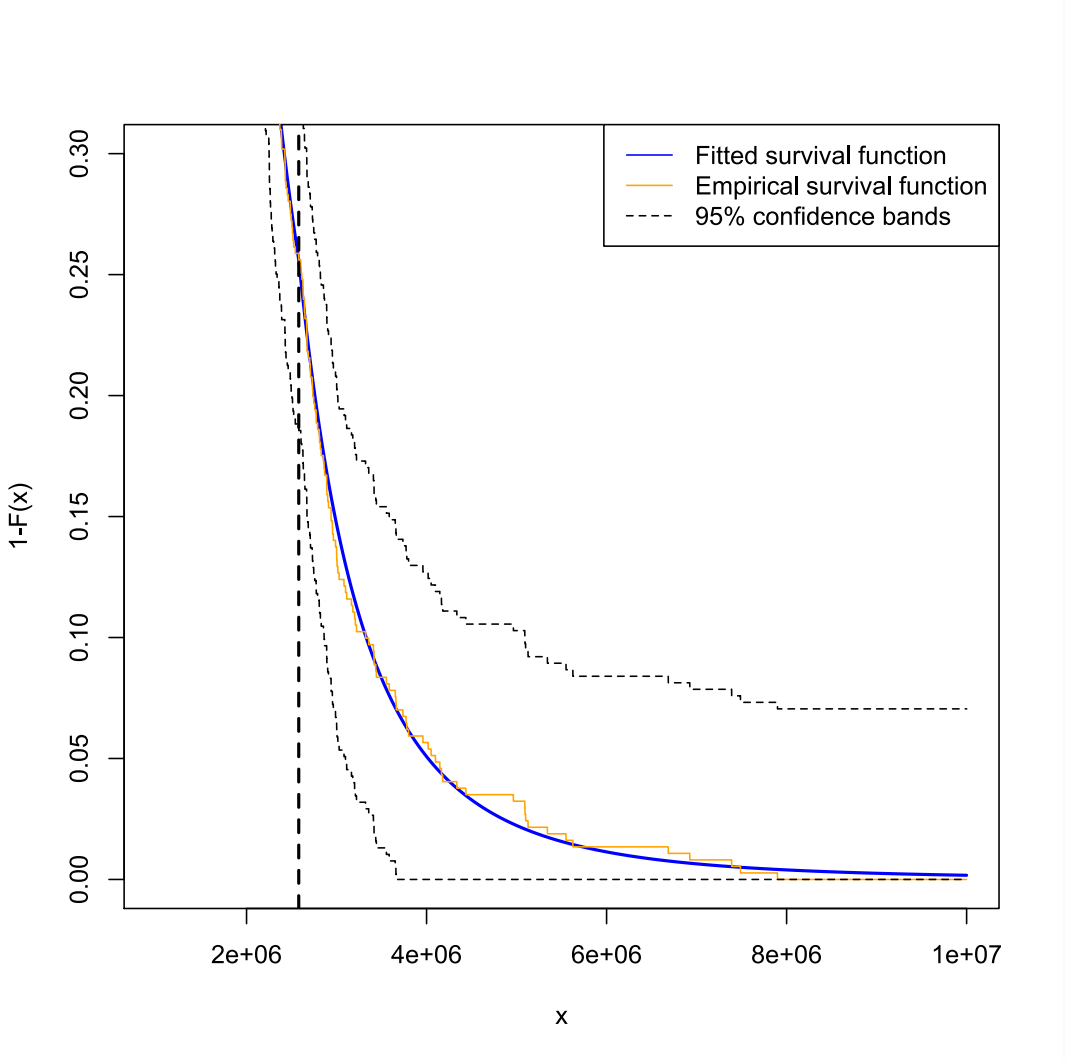
The first way to assess the GoF is a plot of the ECDF and the fitted CDF.

```
# ECDF plot
x <- seq(10^6, 10^7, 10^3)
SpliceECDF(x, secura$Loss, splicefit_sec, lwd = 2)
abline(v = splicefit_sec$t, lty = 2, lwd = 2)

# Zoomed ECDF plot
SpliceECDF(x, secura$Loss, splicefit_sec,
           lwd = 2, ylim = c(0,0.3))
abline(v = splicefit_sec$t, lty = 2, lwd = 2)
```



The first way to assess the GoF is a plot of the ECDF and the fitted CDF.

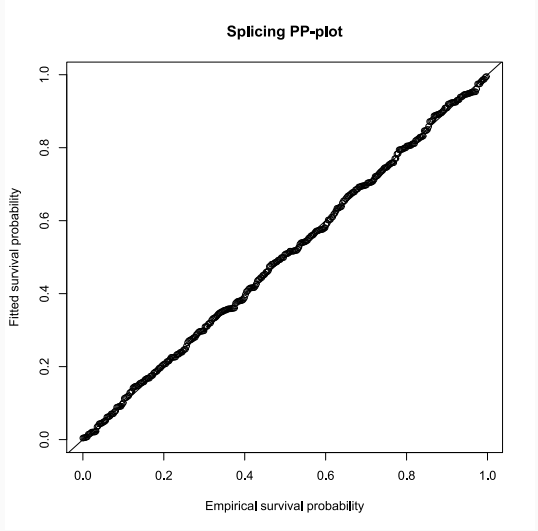
```
# ECDF plot
x <- seq(10^6, 10^7, 10^3)
SpliceECDF(x, secura$Loss, splicefit_sec, lwd = 2)
abline(v = splicefit_sec$t, lty = 2, lwd = 2)

# Zoomed ECDF plot
SpliceECDF(x, secura$Loss, splicefit_sec,
           lwd = 2, ylim = c(0,0.3))
abline(v = splicefit_sec$t, lty = 2, lwd = 2)
```

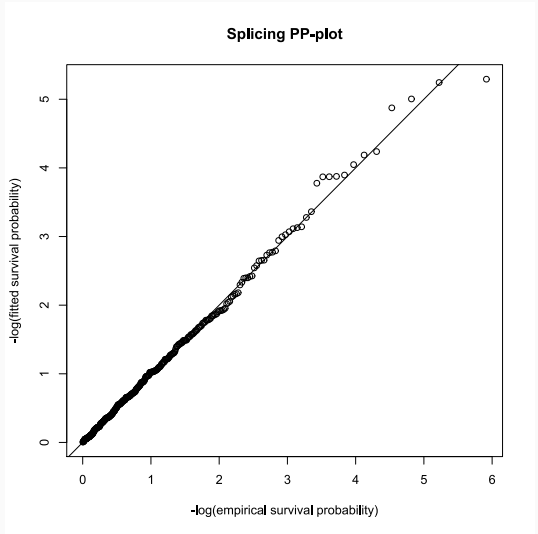


Additionally, a PP-plot can be used which plots the empirical survival function vs. the fitted survival function. To focus on the tails, a version with minus log-scale is also used, i.e.

$$-\log(1-\hat{F}(x_{i,n})) \text{ vs. } -\log(1-F(x_{i,n})).$$

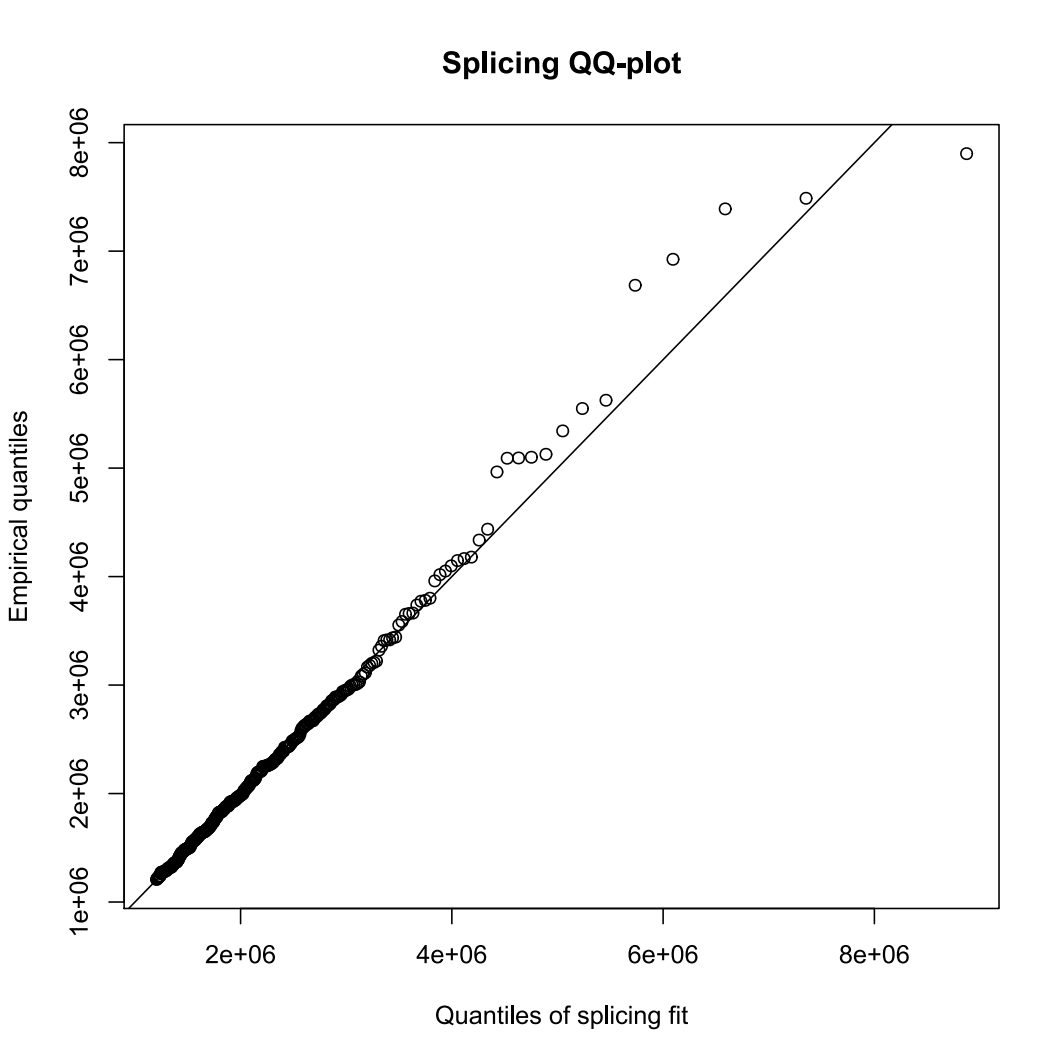
```
# PP-plot
SplicePP(secura$Loss, splicefit_sec)
# PP-plot with minus log-scale
SplicePP(secura$Loss, splicefit_sec, log = TRUE)
```



Finally, a QQ-plot can be used again.

```
# QQ-plot
SpliceQQ(secura$Loss, splicefit_sec)
```



# Risk measures

Some popular risk measures are

- Value-at-Risk

$$VaR_{1-p} = F^{-1}(1-p)$$

- Value-at-Risk quantifies tail events.

- Conditional Tail Expectation

$$\begin{aligned} CTE_{1-p} &= E(X \mid X > VaR_{1-p}) \\ &= \frac{1}{p}\int_{1-p}^{1} VaR_{\gamma}(X)d\gamma. \end{aligned}$$

- Conditional Tail Expectation quantifies the expected severity of tail events (what if?).

# Risk measures for a body-tail fit

We give a general expression for the CDF and quantile function for the body-tail model:

$$f(x) = \begin{cases} 0 & x \leq t^l \\ \pi \cdot \frac{f_1^\star(x)}{F_1^\star(t) - F_1^\star(t^l)} & t^l \leq x \leq t \\ (1-\pi) \cdot \frac{f_2^\star(x)}{1 - F_2^\star(t)} & x > t \end{cases}$$

The CDF is found by integrating the density:

$$F(x) = \begin{cases} 0 & x \leq t^l \\ \pi \cdot \frac{F_1^\star(x) - F_1^\star(t^l)}{F_1^\star(t) - F_1^\star(t^l)} & t^l \leq x \leq t \\ \pi + (1-\pi) \cdot \frac{F_2^\star(x) - F_2^\star(t)}{1 - F_2^\star(t)} & x > t \end{cases}$$

The quantile function is obtained by inverting the CDF:

$$F^{-1}(p) = \begin{cases} Q_1^* \left( \frac{p}{\pi} \cdot (F_1^\star(t) - F_1^\star(t^l)) + F_1^\star(t^l) \right) & 0 \leq p \leq \pi \\ Q_2^* \left( \frac{p-\pi}{1-\pi} \cdot (1 - F_2^\star(t)) + F_2^\star(t) \right) & \pi \leq p \leq 1 \end{cases}$$

where $Q_1^*$ and $Q_2^*$ denote the quantile functions of the body resp. tail distribution used in the splicing model.

# Risk measures for a body-tail fit

```
# Quantile function for Exp-Pa splicing model
ExpPa_quantile <- function(p, sh, threshold, lambda, gamma) {

  pi <- (n-kopt)/n

  x <- numeric(length(p))

  x[p <= pi] <- qexp(p[p <= pi] / pi * (pexp(threshold, rate = lambda) - pexp(sh, rate = lambda)) +
                       pexp(sh, rate = lambda), rate = lambda)

  x[p > pi] <- threshold * (1 - (p[p > pi] - pi) / (1-pi))^(-gamma)

  return(x)
}
```

# Risk measures for a body-tail fit

95% Value-at-Risk

```
ExpPa_quantile(0.95, sh, threshold, lambda, gamma)
## [1] 4017259
```

and check

```
ExpPa_cdf(ExpPa_quantile(0.95, sh,
                         threshold,
                         lambda, gamma),
          sh, threshold, lambda, gamma)
## [1] 0.95
```

95% Conditional Tail Expectation

```
qf <- function(p) {
  ExpPa_quantile(p, sh, threshold, lambda, gamma)
}

int <- integrate(qf, lower = 0.95, upper = 1)
int$value / 0.05
## [1] 5511323
```

The {ReIns} package has built-in functions to calculate these (and other) risk measures for the ME-Pa spliced fit, see e.g. `ReIns::VaR(p, splicefit_sec)` and `ReIns::CTE(p, splicefit_sec)`.

# Thanks!


Slides created with the R package xaringan.

Course material available via

https://github.com/katrienantonio/loss-models-2020