# Week 2 – R Data Modeling

# Last. . . but not least

- Optimization, that is
- Otherwise known as finding the distribution that best fits the data
- So we can simulate that data to help us make decisions *prospectively*
- Use `fitdistr` to help us out

## Many distributions in R: `?distributions` will tell you all

1. If `name` is the name of a distribution (e.g., `norm` for "normal"), then

   - dname = the probability *d*ensity (if continuous) or probability mass function of `name` (pdf or pmf), think "histogram"
   - pname = the cumulative *p*robability function (CDF), think "s-curve"
   - qname = the *q*uantile function (inverse to CDF), "think tolerance line"
   - rname = draw *r*andom numbers from `name` (first argument always the number of draws), think whatever you want. . . it's kind of random

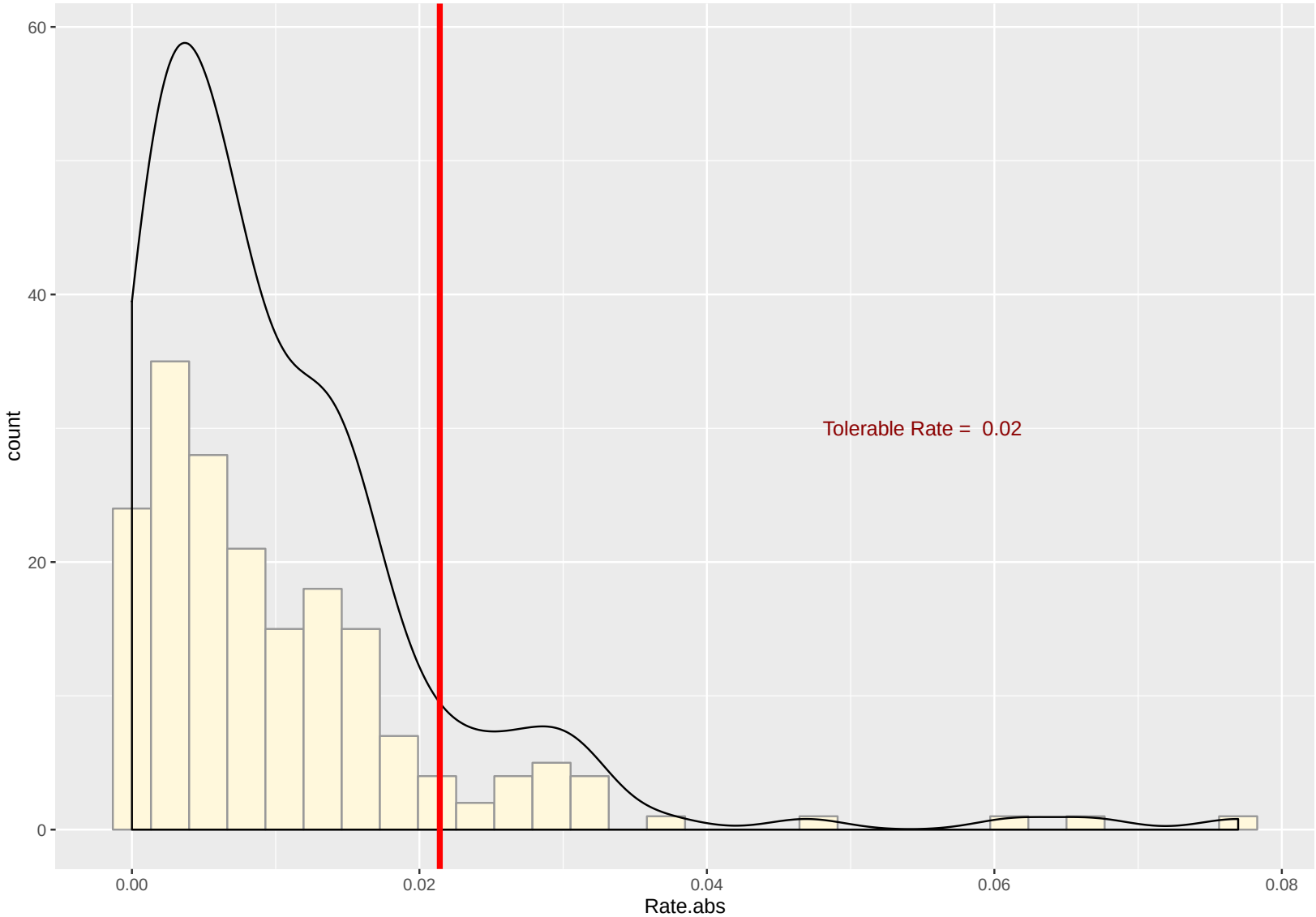
2. And ways to write your own (like the `pareto` distribution we use in finance)

- Suppose the EIUR price series is the *benchmark* in several import contracts you write as the procurement officer of your organization.
- Your concern is with volatility. Thus you think that you need to simulate the size of the price rates, whatever direction they go in.
- Draw the histogram of the absolute value of price rates.

```
require(ggplot2)
r.tol <- quantile(xmprice.r.df$Rate,
    0.95)
r.tol.label <- paste("Tolerable Rate = ",
    round(r.tol, 2))
ggplot(xmprice.r.df, aes(Rate.abs)) +
    geom_histogram(fill = "cornsilk",
        colour = "grey60") + geom_density() +
    geom_vline(xintercept = r.tol, colour = "red",
        size = 1.5) + annotate("text",
    x = 0.055, y = 30, label = r.tol.label,
    colour = "darkred")
```

Thinking. . .

# Result



- A right-skewed, thick-tailed beast for sure. . .
- Use this function to pull all of the calculations together

```
# r_moments function INPUTS: r vector
# OUTPUTS: list of scalars (mean, sd,
# median, skewness, kurtosis)
data_moments <- function(data) {
    require(moments)
    mean.r <- mean(data)
    sd.r <- sd(data)
    median.r <- median(data)
    skewness.r <- skewness(data)
    kurtosis.r <- kurtosis(data)
    result <- data.frame(mean = mean.r,
        std_dev = sd.r, median = median.r,
        skewness = skewness.r, kurtosis = kurtosis.r)
    # result <- data.frame(result, table
    # = t(result))
    return(result)
}
```

Run this

```
ans <- data_moments(xmprice.r.df$Rate.abs)
ans <- round(ans, 4)
knitr::kable(ans)
```

| mean | std_dev | median | skewness | kurtosis |
|---|---|---|---|---|
| 0.0105 | 0.0114 | 0.0072 | 2.6595 | 13.2195 |

- Right skewed
- Very thick tailed
- We will try the `gamma` and `pareto` functions
- We will make liberal use of the `fitdistr` function
- We will come back to this moments function

# Estimate until morale improves. . .

## We will try one method that works often enough in practice. . .

Method of Moments ("MM" or, more affectionately, "MOM"): Find the distribution parameters such that the moments of the data match the moments of the distribution.

## Other Methods

- `fitdistr`: Let the opaque box do the job for you; look at the package `MASS` which uses the "maximum likelihood" approach in the `fitdistr` estimating function (like `lm` for regression).
- `fitdistrplus`: For the more adventurous analyst, this package contains several methods, including MM, to get the job done.

Getting right into it all... suppose we believe that absolute price rates somehow follow a gamma distribution. You can look up this distribution easily enough in Wikipedia's good article on the subject.

## Behind managerial scenes, we can model the loss with

- A gamma severity function – Allows skew and "heavy" tails – Specified by shape, $\alpha$, and scale, $\beta$, parameters
- Especially useful for time-sensitive losses

We can specify these parameters using the mean, $\mu$, and standard deviation, $\sigma$ of the random severities, $X$. The scale parameter is

$$\beta = sigma^2/\mu,$$

and shape parameter,

$$\alpha = \mu^2/\sigma^2.$$

The distribution itself is defined as

$$f(x; alpha, \beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)},$$

where,

$$\Gamma(x) = \int_0^{\infty} x^{t-1} e^{-x} dx.$$

Enough of the math,. . . let's finally implement into R.

Load a cost sample and calculate moments and gamma parameters:

```
cost <- read.csv("cost.csv")
cost <- cost$x
cost.moments <- data_moments(cost)
cost.mean <- cost.moments$mean
cost.sd <- cost.moments$std_dev
(cost.shape <- cost.mean^2/cost.sd^2)
```

```
## [1] 19.06531
```

```
(cost.scale <- cost.sd^2/cost.mean)
```

```
## [1] 0.5575862
```

```
gamma.start <- c(cost.shape, cost.scale)
```

Using `fitdistr` from the `Mass` package we find:

```
require(MASS)
fit.gamma.cost <- fitdistr(cost, "gamma")
fit.gamma.cost
```

```
##      shape          rate
##   20.2998092    1.9095724
##  ( 2.3729249) ( 0.2259942)
```

## How good a job did we do?

- Now construct the ratio of estimates to the standard error of estimates.
- This registers the number of standard deviations away from zero the estimates are.
- If they are "far" enough away from zero, we have reason to reject the null hypothesis that the estimates are no different from zero.

```
(cost.t <- fit.gamma.cost$estimate/fit.gamma.cost$sd)
```

```
##    shape     rate
## 8.554763 8.449653
```

Nice. . . but the scale parameter is `fit.gamma.cost$estimate[2] / gamma.start[2]` times the moment estimates above.

# Try this

Use the export-input price series rates and the `t` distribution instead of the `gamma`.

Thinking. . .

# Result

Calculate the moments

```
rate <- xmprice.r.df$Rate
rate.moments <- data_moments(rate)
(rate.mean <- rate.moments$mean)
```

```
## [1] 0.001116177
```

```
(rate.sd <- rate.moments$std_dev)
```

```
## [1] 0.01545897
```

Using `fitdistr` from the `Mass` package we find:

```
fit.t.rate <- fitdistr(rate, "t", hessian = TRUE)
```

```
## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced

## Warning in log(s): NaNs produced
```

## How good a job did we do?

```
##          m          s         df
##  2.310476 10.531600  3.987537
```

- Nice. . . but that location parameter is a bit low relative to moment estimate.
- What else can we do? Simulate the estimated results and see if, at least, skewness and kurtosis lines up with the moments.

# What have we done?

- . . . Used our newly found ability to write functions
- . . . and built insightful pictures of distributions
- . . . and ran nonlinear (gamma and t-distributions are indeed very nonlinear) regressions
- All to answer critical business questions