⊞— Plot Utilities

$t0 := time(1)$

1. Define two vectors.

$$x := \begin{bmatrix} 0.132 \\ 0.322 \\ 0.511 \\ 0.701 \\ 0.891 \\ 1.082 \\ 1.27 \\ 1.46 \\ 1.65 \\ 1.839 \\ 2.029 \\ 2.219 \end{bmatrix} \quad y := \begin{bmatrix} 0.1 \\ 0.258 \\ 0.543 \\ 0.506 \\ 0.606 \\ 0.622 \\ 0.569 \\ 0.453 \\ 0.438 \\ 0.316 \\ 0.29 \\ 0.195 \end{bmatrix} \quad \beta := \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

$n := length(\beta)$ $\quad m := length(x)$

2. Define a fitting function (Weibull density with unknown parameters).

FUNCTION

$$ft(x) := \beta_1 \cdot \beta_2 \cdot x^{\beta_2 - 1} \cdot \exp\left(-\beta_1 \cdot x^{\beta_2}\right)$$

Jacobian matrix for finite differences

```
derfh(B) := | for i ∈ [1..m]
            |   for j ∈ [1..n]
            |     β_j := β_j + h
            |     jjh1 := fx(β)_i
            |     β_j := β_j − 2·h
            |     jjh2 := fx(β)_i
            |     jjh0_ij := (jjh1 − jjh2) / (2·h)
            |     β_j := β_j + h
            | jjh0
```

3. Define initial guess values for the two parameters.

Initial Assumption Values Vector

$\beta := [0.8\ \ 1]^T$

4. Use an equation to minimize inside a solve block.

Function adjusted to the data

```
fx(β) := | for i ∈ [1..m]
         |   f_i := ft(x_i) − eval(y_i)
         | f
```

Example: Using minerr for Nonlinear

The **minerr** function is similar to the **find** function, except that it returns an a

1. Define two vectors.

$$x := \begin{bmatrix} 0.132 \\ 0.322 \\ 0.511 \\ 0.701 \\ 0.891 \\ 1.081 \\ 1.27 \\ 1.46 \\ 1.65 \\ 1.839 \\ 2.029 \\ 2.219 \end{bmatrix}$$

$n := length(y) - 1$

2. Define a fitting function (Weibull density with unknown parameters).

$$Wb(x,\alpha,\beta) := \alpha \cdot \beta \cdot x^{\beta-1} \cdot \exp(-\alpha \cdot x^{\beta})$$

3. Define initial guess values for the two parameters.

$\alpha := 0.8$

$\beta := 1$

4. Use an equation to minimize inside a solve block.

$$resid(\alpha,\beta) := y - \overrightarrow{Wb(x,\alpha,\beta)}$$

5. Add a solve block and use **minerr** to solve the problem. The **minerr** functi
summing and squaring of the residuals.

$0 = resid(\alpha,\beta)$

$$\begin{bmatrix} \alpha 1 \\ \beta 1 \end{bmatrix} := minerr(\alpha,\beta)$$

5. Use the Levenberg-Marquardt method to minimize this problem.

LEVENBERG MARQUARDT METHOD

$Eps := 1.110 \cdot 10^{-16}$ $eta1 := \sqrt{Eps}$ $eta2 := eta1$ $h := eta1$ $tol := \sqrt{Eps} \cdot normi(\beta)$

$f := fx(\beta)$ $J := derfh(\beta)$

$A := J^T \cdot J$ $g := J^T \cdot f$ $ng := normi(g)$

$F := \frac{f^T \cdot f}{2}$ $mu := eta1 \cdot max(Diag(A))$ $nu := 2$ $stop := 0$

```
while ¬stop
  if ng ≤ eta2
    stop := 1
  else
    try
      p := (A + mu·identity(n))^(-1)·(-g)
    on error
      break
    np := normi(β)
    nx := eta2 + normi(β)
    if np ≤ eta2·nx
      stop := 2
    else
      ""
  if ¬stop
    xnew := β + p
    fn := fx(xnew)
    Jn := derfh(xnew)
    Fn := (fn^T·fn)/2
    dL := (p^T·(mu·p − g))/2
    dF := F − Fn
    if (dL_1 > 0) ∧ (dF_1 > 0)
      β := xnew
      F := Fn
      J := Jn
      f := fn
      A := J^T·J
      g := J^T·f
      ng := normi(g)
      mu := mu·Max(1/3, 1 − (2·dF_1/dL_1 − 1)^3)
      nu := 2
    else
      mu := mu·nu
      nu := 2·nu
  else
    ""
```

$$mu := mu \cdot Max\left(\frac{1}{3}, 1 - \left(2 \cdot \frac{dF_1}{dL_1} - 1\right)^3\right)$$

The parameters for best fit are the calculated values:

SOLUTION

$$\beta = \begin{bmatrix} 0.502 \\ 2.00 \end{bmatrix}$$

6. Calculate the sum of squares implicitly minimized by this method (By JEAN).

$$f(x,\beta) := \beta_1 \cdot \beta_2 \cdot x^{\beta_2 - 1} \cdot \exp\left(-\beta_1 \cdot x^{\beta_2}\right) \qquad t := x$$

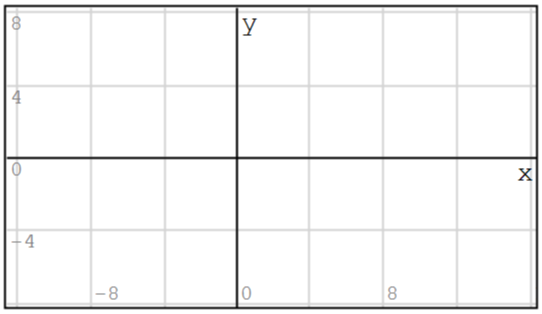
Relative residuals

```
Residuals := | for i ∈ [1..rows(t)]
             |    Δ_i := eval(1 − y_i / f(t_i, β))
             | augment(t, Δ)
```

SSD = "Sum Square Differences"

$$SSD := \sum_{i=1}^{m} \left[\left(y_i - f(t_i,\beta)\right)^2\right] = 0.0186$$

Relative residuals



{BarSize(Residuals, 0.0394)

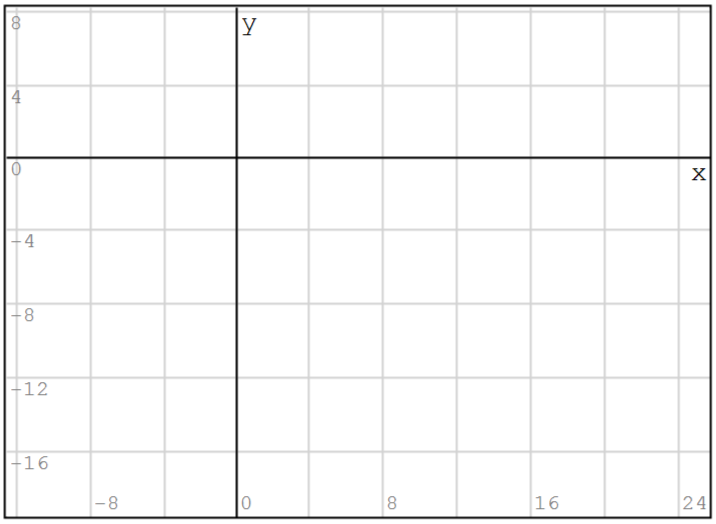
RMS difference

$$\sqrt{\frac{SSD}{m}} = 0.0394$$

7. Plot the best Weibull fit versus the x-y data.

```
data := plotG(x, y, ".", 12, "blue")

FIT := | "range, populate"
       | U := [0, 0.01..4]
       | for i ∈ [1..rows(U)]
       |    vy_i := eval(f(U_i, β))
       | augment(U, vy)
```



{data
{FIT

$time(1) - t0 = 2.688\ s$