APPENDIX 2

This Maple program computes the power of the R-Fisher test with a standardized shift in location of delta units and a standardized shift in scale of kappa units

The program will compute the right hand tail for the associated noncentral generalized F distribution and plots the density of this distribution

Donald Ramirez

Mathematics Department

University of Virginia

der@virginia.edu

April 1, 2004

```
> restart:
```

Initialized the statistics packages

```
> with(stats):
> with(statevalf):
> with(pdf):
> with(cdf):

Warning, these names have been redefined: beta, cauchy, chisquare,
exponential, fratio, gamma, laplaced, logistic, lognormal, normald,
studentst, uniform, weibull
```

Input initial values

n = number of cases

p = rank of experimental design matrix

s = number of cases in the subset I under study

delta = shift in location

kappa = shift ratio in scale

lambda = vector of length s of canonical leverages

theta := vector of length s for the standardized shift in location of one unit, **Q'**1

```
> n := 25:
> p := 8:
> s := 2:
> delta := 2:
> kappa := 2:
> lambda := [0.3665, 0.0702]:
> theta := [1.0243, 0.9751]:
```

User defined parameters

max_N is maximum number of terms in the partial sum that will be retained

error_tolerance controls truncation error

```
> max_N := 40:
> error_tolerance := 10^(-5):
```

Start of procedure

```
> nu := n - p - 2;
```

$$\nu := 15$$

```
> setup_gen_F := proc(kappa, delta)
> local i, temp_theta;
> global nterm, lambda, theta, s, alpha, noncentrality, omega;
> temp_theta := delta*theta;
> for i from 1 to s do alpha[i] := kappa - (kappa-1)*lambda[i]
end do:
> for i from 1 to s do omega[i] := temp_theta[i]/(kappa +
> lambda[i]/(1-lambda[i]))^(1/2) end do:
> noncentrality := sum( omega['i']^2, 'i' = 1 .. s):
> end proc;
```

```
$setup\_gen\_F :=$ **proc**$(\kappa, \delta)$
**local** $i$, $temp\_theta$;
**global** $nterm, \lambda, \theta, s, \alpha, noncentrality, \omega$;
    $temp\_theta := \delta * \theta$ ;
    **for** $i$ **to** $s$ **do** $\alpha_i := \kappa - (\kappa - 1) * \lambda_i$ **end do** ;
    **for** $i$ **to** $s$ **do** $\omega_i := temp\_theta_i/(\kappa + \lambda_i/(1 - \lambda_i))^{(1/2)}$ **end do** ;
    $noncentrality := \mathrm{sum}(\omega_{'i'}^2, 'i' = 1..s)$
**end proc**
```

procedure sum_dc computes the required coefficients c[k]

```
> sum_dc := proc (k)
> local temp, i;
> global c, d;
> temp := d(k)*c[0];
> for i to k-1 do temp := temp+d(k-i)*c[i] end do;
> c[k] := simplify(temp/k)
> end proc;
```

$$
\begin{array}{l}
\mathit{sum\_dc} := \mathbf{proc}(k) \\
\mathbf{local}\, \mathit{temp},\, i; \\
\mathbf{global}\, c,\, d; \\
\quad \mathit{temp} := \mathrm{d}(k) * c_0\,; \\
\quad \mathbf{for}\, i\, \mathbf{to}\, k-1\, \mathbf{do}\;\; \mathit{temp} := \mathit{temp} + \mathrm{d}(k-i) * c_i\, \mathbf{end\ do}\,; \\
\quad c_k := \mathrm{simplify}(\mathit{temp}/k) \\
\mathbf{end\ proc}
\end{array}
$$

procedure compute_c computes the required coefficients d[k] and returns the number of terms in the partial sum to meet the error tolerance

```
> compute_c := proc (temp_alpha)
> local i, temp_A, A, temp_d, k, temp, temp1_alpha :
> global b, c, d, N, s, nu, omega, noncentrality:
> temp1_alpha := sort(temp_alpha):
> b := .99*temp1_alpha[1]:
> temp_A := sqrt(product(b/temp_alpha['i'],('i') = 1 ..  s)):
> A := temp_A*exp(-1/2*noncentrality):
> temp_d :=
> (1/2)*sum((1-b/temp_alpha['i'])^k+k*b*omega['i']^2*(1-b/temp_alpha['i'
> ])^(k-1)/temp_alpha['i'],('i') = 1 ..  s):
> d := unapply(temp_d, k):
> c[0] := A:
> for k from 1 to max_N do sum_dc(k) end do:
> k := 0:
> temp := c[0];
> while (s/(b*(s+2*(k+1)))*(1-temp) > error_tolerance and k < max_N)
> do k := k + 1;
> temp := sum(c['i'], 'i' = 0 ..  k):
> end do:
> N := k:
> end proc;
```

```
compute_c := proc(temp_alpha)
local i, temp_A, A, temp_d, k, temp, temp1_alpha;
global b, c, d, N, s, ν, ω, noncentrality;
    temp1_alpha := sort(temp_alpha) ;
    b := 0.99 * temp1_alpha_1 ;
    temp_A := sqrt(product(b/temp_alpha'_i', 'i' = 1..s)) ;
    A := temp_A * exp(−noncentrality/2) ;
    temp_d := 1/2 * sum((1 − b/temp_alpha'_i')^k
        + k * b * ω'_i'^2 * (1 − b/temp_alpha'_i')^(k−1)/temp_alpha'_i', 'i' = 1..s);
    d := unapply(temp_d, k) ;
    c_0 := A ;
    for k to max_N do sum_dc(k) end do ;
    k := 0 ;
    temp := c_0 ;
    while error_tolerance < s * (1 − temp)/(b * (s + 2 * k + 2)) and k < max_N do
        k := k + 1 ; temp := sum(c'_i', 'i' = 0..k)
    end do;
    N := k
end proc
```

procedure gen_F_pdf computes the required partial sum for the density of the generalized F distribution

```
> gen_F_pdf := proc(y)
> local temp, k:
> temp := 0;
> for k from 0 to N do temp := temp + (s/nu)*(c[k]/b)*GAMMA((2*k
+ nu +
> s)/2)*((s/nu)*y/b)^((s + 2*k - 2)/2)/ (GAMMA((s +
> 2*k)/2)*GAMMA(nu/2)*(1 + (s/nu)*y/b)^((2*k + nu + s)/2));
> end do;
> end proc;
```

```
gen_F_pdf := proc(y)
local temp, k;
    temp := 0 ;
    for k from 0 to N do temp := temp + s * c_k * Γ(k + ν/2 + s/2)*
        (s * y/(ν * b))^(s/2+k−1)/(ν * b * Γ(s/2 + k) * Γ(ν/2) * (1 + s * y/(ν * b))^(k+ν/2+s/2))
    end do
end proc
```

compute the 95-percential value y0 from the central F distribution

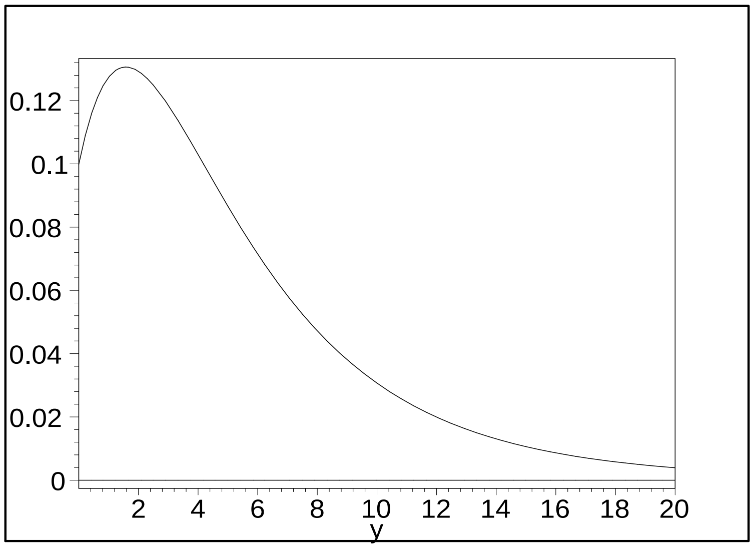
```
>  y0 := statevalf[icdf,fratio[s,nu]](0.95);
                        y0 := 3.682320344
>  setup_gen_F(kappa, delta):
>  print(alpha);
                 table([1 = 1.6335, 2 = 1.9298])
>  print(omega);
            table([1 = 1.275765662, 2 = 1.353685484])
>  number_of_terms_used := compute_c(alpha);
                 number_of_terms_used := 10
>  gen_F_cdf := x -> evalf(Int(gen_F_pdf(t), t= 0 ..  x)):
```

compute the power for ; that is Pr[W > y0]

```
>  power := 1-gen_F_cdf(y0);
                     power := 0.5517833771
```

plot the density of the non-central generalized F distribution

```
>  plot({gen_F_pdf(y), 0} , y = 0 ..  20, thickness = 2, axes =
>  'boxed', color = 'black');
```


0.12
0.1
0.08
0.06
0.04
0.02
0
2
4
6
8
10
12
14
16
18
20
y