Joint Cumulants of *polykays*

E. Di Nardo*
elvira.dinardo@unito.it
https://www.elviradinardo.it
Tel: +390116702862 Fax: +390116702878

G. Guarino**
giuseppe.guarino@rete.basilicata.it

* Mathematics Department "G. Peano", University of Turin, Turin, Italy

**Local Health Authority of Potenza, Italy

Introduction

***Abstract*:** Given a multivariate random sample, this set of functions returns the joint cumulants of multivariate polykays in terms of joint cumulants of the multivariate population underlying the sample.
Multivariate polykays are unbiased estimators with minimum variance of joint cumulant products. These estimators are usually indexed by a list of integer vectors: each vector corresponds to a joint cumulant.
When this list reduces to a list of single integers, the multivariate polykay reduces to a univariate polykay. When this list reduces to a single integer vector, the multivariate polykay reduces to a multivariate *k*-statistic. When this list contains a single integer vector with only one element, the multivariate *k*-statistic reduces to a univariate *k*-statistic.
Application Areas/Subject: Combinatorics & algebraic methods in statistics.
Keyword: umbral calculus, symmetric polynomial, set partition, multiset, cumulant, *k*-statistic, polykay.
See Also: Maple algorithm [1] and [2]
Remark: *k*-statistics, polykays and their multivariate generalization are commonly given in terms of power sums

$$S_{j,\,k,\ldots} = \sum_{i=i}^{n} X_i^j\, Y_i^k \ldots.$$

with *n* the sample size.

Initialization

```
> restart
> with(combinat, partition, multinomial, stirling2)
```

Background Functions

The *polyk* function handles four different algorithms. Depending on the input parameters, univariate or multivariate polykays as well as univariate or multivariate *k*-statistics are expressed in terms of power sums. Specifically, if the input is a single vector containing a single integer, univariate *k*-statistics are returned; if the input is a single vector containing more than one integer, multivariate *k*-statistics are returned; if the input is a list of vectors each containing a single integer, then univariate polykays are returned; if the input is a list of vectors, each containing more than one integer, multivariate polykays are returned. All these functions are in the MAPLE worksheet [1] and have been included here for user convenience. For the multivariate case, the algorithm listing all subdivisions of a multiset has also been included. This algorithm is given and discussed in the MAPLE worksheet [2]. For details on the procedures and mathematical backgroud see [3 - 6].

K-Statistics

The *n*-th *k*-statistic k_n is the unique symmetric unbiased estimator of the *n*-th cumulant κ_n of a given statistical distribution, that is

$$E[k_n] = \kappa_n$$

where *E* denotes the expectation. They are expressed in terms of power sums involving the random variables of a random sample.

```
>
> makeTab_ss :=proc(N)
  local x, y, i;

    [seq([ N! / (mul(x!^numboccur(y, x) * numboccur(y, x)!, x = {op(y)})) * mul(k_i, i = y), mul(S_i, i = y) ], y = partition(N))]
  end proc:


  makeK_s :=proc(N) local i, j;
      [seq(k_i = add(stirling2(i, j) * x^j * (-1)^(j-1) * (j-1)!, j = 1..i), i = 1..N)]
  end proc:

  fd :=proc(j, k) local i; expand(mul(n - i - k, i = 0..j - k)) end proc:
  ks :=proc(N) local u, v, i; global x;

      v := expand(eval(makeTab_ss(N), makeK_s(N)));
      u := [seq(x^i = (-1)^(i-1) * (i-1)! * fd(N-1, i), i = 1..N)];
      add(x1 * x2, x = expand(eval(v, u))) / mul((n - x), x = 0..N-1)
  end proc:
```

Exampe: The *k*-statistic of order 3 is

> $ks(3)$

$$\frac{n^2 S_3 - 3\, n\, S_1 S_2 + 2\, S_1^3}{n\,(n-1)\,(n-2)} \qquad \textbf{(3.1.1)}$$

Polykays

The symmetric statistic $k_{r,\,s,\,..}$ is an unbiased estimator of the cumulant product $\kappa_r\,\kappa_s...$, that is

$$E[k_{r,\,s,\,..}] = \kappa_r\,\kappa_s...$$

where E denotes the expectation and κ_r is the *r*-th cumulant. These statistics are called polykays or generalized *k*-statistics.

> $makeCTR\_s := \textbf{proc}(N)$

$$locali, v, k, x; \left[seq\left(k_i = add\left(\frac{mul\left(\mu_k, k = v\right) \cdot (-1)^{nops(v)-1} \cdot (nops(v) - 1)!\, i!}{mul\left((x)!^{numboccur(v,\,x)} \cdot numboccur(v, x)!, x = \{op(v)\}\right)}, v = partition(i) \right), i = 1..N \right) \right]$$

end proc:

$makeMu := \textbf{proc}()$

local u, v, N, eu, i, x, k;

$N := add(i, i = args)$;

$eu := \left[seq\left(\mu_i = 1, i = 1..N\right) \right]$;

if $nargs = 1$ **then**

$u := \left[seq\left([[x]], x = partition\left(args_1\right) \right) \right]$

else

$u := [seq([seq(x, x = partition(i))], i = args)]$

end if;

$$u := op\left(add\left(mul\left((-1)^{nops(i)-1} \cdot (nops(i) - 1)! \cdot mul\left(\mu_k, k = i\right) \cdot countP(i), i = v\right) \cdot fd(N - 1, add(nops(k), k = v)) \cdot countP([seq(op(i), i = v)])^{-1}, v = `if`(nargs = 1, comb(u, 1, [\,]), [comb(u, 1, [\,])])\right)\right);$$

$$seq\left(simplify\left(\frac{v}{eval(v, eu)}\right) = eval(v, eu), v = u\right)$$

end proc:

$comb := \textbf{proc}(V, ptr, Y)$

local l;**if** $ptr = nops(V) + 1$ **then**

return Y

end if;

$seq\left(comb(V, ptr + 1, [op(Y), L]), L = V_{ptr}\right)$

end proc:

```
countP := proc(u)
    local x;
                          add(x, x = u)!
    ------------------------------------------------------------
    mul(x^(numboccur(u, x)) · numboccur(u, x)!, x = {op(u)})
end proc:

ps := proc( )
    local u, v, N, x;
    N := add(x, x = args);
    u := expand(eval(makeTab_ss(N), makeCTR_s(N) ) );
    v := expand(eval(u, [makeMu(args), μ = 0]) );
          add(x_1 · x_2, x = v)
    ------------------------------------------------
    mul(n − x, x = 0 ..N − 1)

end proc:
```

Example : The polykay of order (2, 1) is

```
> ps(2, 1)
```

$$\frac{-S_1^3 + (1+n)\, S_1 S_2 - n\, S_3}{n\,(n-1)\,(n-2)} \qquad \textbf{(3.2.1)}$$

Multiset subdivision

The *makeTab* function lists all subdivisions of a multiset (for a thorough discussion see [2]). With multivariate *k*-statistics and/or multivariate polykays, this function speeds up the overall procedure (see [7]). For details on the procedure and mathematical background see also [3] and [4].

```
> nRep :=proc(u) local x; mul(x_2!, x = convert(u, multiset)) end proc:
```

```
URv :=proc(u, v)
    local U, ou, i, ptr, vI;
    ou := NULL; U := [ ]; vI := indets(v);
    for ptr from nops(u) by −1 to 2 do
        if has(u_ptr, v) then break end if
    end do;
    for i from ptr to nops(u) do
        if not (u_i = ou or has(u_i, vI)) then
            ou := u_i;
            U := [op(U), [op(u_1 ..i − 1), u_i · v, op(u_i + 1 ..−1)]]
        end if
    end do;
    op(U), [op(u), v]
end proc:
```

```
URV :=proc( )
```

```
    local U, V, i, u, x;
    U := [args[1, 1]]; V := args[2, 1];
    for i to nops(V) do U := [seq(URv(u, V[i]), u = U)] end do;
    seq([x, args[1, 2] .. args[2, 2]], x = U)
end proc:


URmV := proc( )
    local U, i;
    if nargs = 1 then args else
        U := URV(args[1], args[2]);
        for i from 3 to nargs do
            U := seq(URV(u, args[i]), u = [U])
        end do;
        seq([x[1], x[2]/nRep(x[1])], x = U)
    end if
end proc:


makeTab := proc( )
    local U, x, i, y, z, r;
    if add(x, x = args) = 0 then
        return 0
    end if;
    U := [seq(`if`(args[i] = 0, NULL, [seq([[seq((P || i)^z, z = y)], multinomial(args[i], seq(r,
    r = y))], y = partition(args[i]))]), i = 1 .. nargs)];
    if nops(U) = 1 then
        [seq([x[1], x[2]/nRep(x[1])], x = op(U))]
    else
        [seq(URmV(op(x)), x = [comb(U, 1, [ ])])]
    end if
end proc:
```

Multivariate *k*-statistics

The multivariate *k*-statistic is the unique symmetric unbiased estimator of the *n*-th joint cumulant $\kappa_{i,j,..}$ of a given multivariate statistical distribution, that is

$$E[k_{i,j,..}] = \kappa_{i,j,..}$$

where *E* denotes the expectation. They are expressed in terms of multivariate power sums involving the random vectors of a random sample.

```
> makeTab_sm := proc( )
    local vP, U, i, y, x;
    U := makeTab(args);
    vP := sort( [ seq( P||i, i = 1 ..nargs ) ] );
    [ seq( [ mul( k_add(degree(x, vP_i), i = 1 ..nops(vP)), x = y_1 ) ·y_2,
    mul( S_seq(degree(x, vP_i), i = 1 ..nops(vP)), x = y_1 ) ], y = U ) ]
  end proc:

  km := proc( ) local u, v, N; global x;

    N := add(x, x = args);
    v := expand(eval(makeTab_sm(args), makeK_s(N) ) );
    u := [ seq( x^i = ( -1 )^(i - 1) · ( i - 1 ) ! · fd(N - 1, i), i = 1 ..N ) ];
    add( x_1 · x_2, x = expand(eval(v, u) ) )
    ------------------------------------------
          mul(n - x, x = 0 ..N - 1 )
  end proc:
```

Example : The multivariate k-statistics of order $(2, 1)$ is

```
> km(2, 1)
```

$$\frac{n^2 S_{2,1} - n S_{0,1} S_{2,0} - 2 n S_{1,0} S_{1,1} + 2 S_{0,1} S_{1,0}^2}{n (n-1) (n-2)} \qquad \textbf{(3.4.1)}$$

Note that the power sums in (3.4.1) are:

$$S_{2,1} = \sum_{i=i}^{n} X_i^2 Y_i \quad S_{0,1} = \sum_{i=i}^{n} Y_i \quad S_{2,0} = \sum_{i=i}^{n} X_i^2 \quad S_{1,0} = \sum_{i=i}^{n} X_i \quad S_{1,1} = \sum_{i=i}^{n} X_i Y_i$$

where $(X_1,..., X_n)$ refers to the population X and $(Y_1,..., Y_n)$ refers to the population Y.

Multivariate Polykays

The multivariate polykay $k_{r, s,..; i, j,..;...}$ is an unbiased estimator of the cumulant product $\kappa_{r, s,...}\ \kappa_{i, j,...}$..., that is

$$E[k_{r, s,..; i, j,..;...}] = \kappa_{r, s,...}\ \kappa_{i, j,...}...$$

where E denotes the expectation and $\kappa_{r, s,...}\ \kappa_{i, j,...}$ are joint cumulants of order *(r,s,...)* and *(i,j,..)* respectively.

```
> makeTab_mm := proc( )
    local vP, U, y, x, i;
    U := makeTab(args);
    vP := sort( [ op( indets(U) ) ] );
    [ seq( [ mul( k_seq(degree(x, vP_i), i = 1 ..nops(vP)), x = y_1 ) ·y_2,
    mul( S_seq(degree(x, vP_i), i = 1 ..nops(vP)), x = y_1 ) ], y = U ) ]
```

end proc:

ctr :=**proc**( )
local *vP*, *U*, *i*, *v*, *x*;
$U := makeTab(args)$;
$vP := \left[seq\left(P \| i,\ i = 1 \,..nargs \right) \right]$;
$add\left(v_2 \cdot (-1)^{nops(v_1)-1} \cdot \left(nops(v_1) - 1 \right)! \cdot mul\left(\mu_{seq\left(degree\left(x,\ vP_i \right),\ i = 1\,..nops(vP) \right)},\ x = v_1 \right),\ v = makeTab(args) \right)$
end proc:

makeCTR_m :=**proc**( )
local *i*, *y*, *x*; $\left[seq\left(k_{op(i)} = ctr(op(i)),\ i = comb(\,[\,seq(\,[\,seq(x,\ x = 0\,..y)\,],\ y = args)\,],\ 1,\ [\ ]) \right) \right]$
end proc:

unionVects :=**proc**(*U*::*list*, *V*::*list*) **local** *v*, *u*;
if $nops(U) = 0$ **then** *V*
elif $nops(V) = 0$ **then** *U*
else $\left[seq\left(seq\left(\left[sort\left(\left[op\left(v_1 \right), op\left(u_1 \right) \right] \right), v_2 \cdot u_2 \right], u = U \right), v = V \right) \right]$
end if
end proc:

ricVtab :=**proc**(*v*, *V*, *N*)
local *u*, *vv*;
$vv := sort\left(v_1 \right)$;
for *u* **in** *V* **do**
if $sort\left(u_1 \right) = vv$ **then**
return $v_2 \cdot \dfrac{fd\left(N - 1,\ nops\left(v_1 \right) \right)}{u_2}$
end if
end do;
return 0
end proc:

pm :=**proc**( )
local *N*, *M*, *u*, *v*, *i*, *vK*, *vTab*, *vP*, *vParts*, *vEval*, *k*, *h*, *y*; **global** *x*;
$M := \max(seq(nops(x),\ x = args))$;
$vP := [\,seq(P \| i,\ i = 1\,..M)\,]$;
$u := op(add(k,\ k = seq(\,[\,op(h),\ seq(0,\ y = nops(h)\,..M - 1)\,],\ h = args)))$;

```
    v := expand(eval(makeTab_mm(u), makeCTR_m(u)));
    N := add(h, h = u);
    vTab := [ ];
    for i to nargs do
        vTab := unionVects(vTab, [seq([v[1], v[2]·(-1)^(nops(v[1])-1)·(nops(v[1])-1)!], v
        = makeTab(op(args[i])))])
    end do;
    if nops(vTab) > 1 then
        vTab := [seq([op(i)[1], op(i)[2]/2], i = mul(u[1]^(2·u[2]), u = vTab))]
    end if;
    vParts := makeTab(u);
    vTab := [seq([x[1], ricVtab(x, vParts, N)], x = vTab)];
    vEval := [seq(mul(μ[seq(degree(x, vP[i]), i = 1..nops(vP))], x = y[1]) = y[2], y = vTab), μ = 0];
    add(x[1]·x[2], x = eval(v, vEval))
    ------------------------------------
          mul(n - x, x = 0..N - 1)
end proc:
```

Example : The multivariate polykay of order ([1, 1], [1, 0]) is

```
> pm([1, 1], [1, 0])
```

$$\frac{n\,S_{1,0}\,S_{1,1} - S_{0,1}\,S_{1,0}^{2} - n\,S_{2,1} + S_{0,1}\,S_{2,0}}{n\,(n-1)\,(n-2)} \qquad \textbf{(3.5.1)}$$

Note that the vectors in the input list must be the same length: if any vector is "smaller," the function adds 0 until it reaches the correct length.

Master function "polyk" handling all cases

This function allows us to recall all functions for generate *k*-statistics, polykays and their multivariate generalizzations.
The input is the following:

- for generate *k*-statistics k_r the parameter is: [r]
- for generate polykays $k_{r;s}$ the parameter is: [r], [s]
- for generate multivariate *k*-statistics $k_{r,\,s}$ the parameter is: [r, s]
- for generate multivariate polykays $k_{r,s;\,u,v}$ the parameter is: [r, s], [u, v]

```
> polyk :=proc( )
    global x;
    if nargs = 1 then
        if nops(args[1]) = 1 then ks(op(args[1]))
                             else km(op(args[1]))
        end if;
    elif add(`if`(nops(x) = 1, 0, 1), x = args) = 0 then
        ps(seq(op(x), x = args))
    else pm(args)
```

```
    end if
end proc:
```

Esample: The *k*-statistic of order 3 is

```
> polyk([3])
```

$$\frac{n^2 S_3 - 3\,n\,S_1 S_2 + 2\,S_1^3}{n\,(n-1)\,(n-2)} \tag{3.6.1}$$

Example : The polykay of order $(2, 1)$ is

```
> polyk([2], [1])
```

$$\frac{-S_1^3 + (1+n)\,S_1 S_2 - n\,S_3}{n\,(n-1)\,(n-2)} \tag{3.6.2}$$

Example : The multivariate k-statistics of order $(2, 1)$ is

```
> polyk([2, 1])
```

$$\frac{n^2 S_{2,1} - n\,S_{0,1} S_{2,0} - 2\,n\,S_{1,0} S_{1,1} + 2\,S_{0,1} S_{1,0}^2}{n\,(n-1)\,(n-2)} \tag{3.6.3}$$

Example : The multivariate polykay of order $([1, 1], [1, 0])$ is

```
> polyk([1, 1], [1])
```

$$\frac{n\,S_{1,0} S_{1,1} - S_{0,1} S_{1,0}^2 - n\,S_{2,1} + S_{0,1} S_{2,0}}{n\,(n-1)\,(n-2)} \tag{3.6.4}$$

Replacing symbols with numerical data

The following functions allow the random variables of a random sample to be replaced with corresponding numerical observations.

The *powS* function returns the sum of the *r*-th powers of the observations.

```
> powS :=proc( )
      local j; if nargs = 1 then
          Sum('X'_i^args[1], i = 1 .. 'n')
      else
          Sum(mul('X'_{i,j}^args[j], j = 1 .. nargs), i = 1 .. 'n')
      end if
  end proc:
```

Example

```
> powS(5, 3, 1)
```

$$\sum_{i=1}^{n} X_{i,1}^5\, X_{i,2}^3\, X_{i,3} \tag{3.6.1.1}$$

```
> powS(9);
```

$$\sum_{i=1}^{n} X_i^9 \qquad \textbf{(3.6.1.2)}$$

The *npolyk* function computes a numerical value of a *k*-statistic or a polykay (including the multivariate cases), replacing the random variables of the random sample with the corresponding numerical observations. The parameters are the following:

- for generate *k*-statistics k_r the parameters are: [r], [[n1, n2, ...]]
- for generate polykays $k_{r;s}$ the parameters are: [[r], [s]], [[n1, n2, ...]]
- for generate multivariate *k*-statistics $k_{r,\,s}$ the parameters are: [[r , s]], [[n1a, n2a], [n1b, n2b] , ...]
- for generate multivariate polykays $k_{r,s;\,u,v}$ the parameters are: [[r , s], [u , v]], [[n1a, n2a], [n1b, n2b] , ...]

```
> npolyk:=proc(V, data)
    local res, ind, N, vE, Si, y, j, x;
    Si := false;
    if nops(V) = 1 then
        if nops(op(V)) = 1 then
            Si := true;
            N := nops(op(data));
            res := ks(op(op(V)))
        else
            N := nops(data);
            res := km(op(op(V)))
        end if;
    elif add(`if`(nops(x) = 1, 0, 1), x = V) = 0 then
        Si := true;
        N := nops(op(data));
        res := ps(seq(op(x), x = V))
    else
        N := nops(data);
        res := pm(op(V))
    end if;
    ind := `minus`(indets(res), {n});
    vE := seq(y[1] = Sum(mul('X'[`if`(Si, op([j, i]), op([i, j]))]^y[2,j], j = 1..nops(y[2])), i
     = 1..N), y = seq([x, [op(x)]], x = ind));
    eval(res, [eval(evalf(vE), [X = data]), n = N])
end proc:
```

Examples: suppose to have the following (univariate) numerical sample

```
> data := [[16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08,
      19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83,
      16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11]]
```

$$data := [[16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08, 19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83, 16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11]] \qquad \textbf{(3.6.1.3)}$$

An estimation of the mean (*k*-statistic of first order) is

> *npolyk*([[1]], *data*)

14.02166667 **(3.6.1.4)**

An estimation of the variance (*k*-statistic of second order) is

> *npolyk*([[2]], *data*)

12.65006954 **(3.6.1.5)**

An estimation of the skewness is $k_3 / \sqrt{k_2^3}$

> $\frac{npolyk([[3]], data)}{\sqrt{npolyk([[2]], data)^3}}$

−0.03216232203 **(3.6.1.6)**

An estimation of the kurtosis is k_4 / k_2^2

> $\frac{npolyk([[4]], data)}{npolyk([[2]], data)^2}$

−0.8852923202 **(3.6.1.7)**

The estimation of the product $\kappa_3\ \kappa_2$ is

> *npolyk*([[3], [2]], *data*)

−15.56147343 **(3.6.1.8)**

Examples: suppose to have the following (bivariate) numerical sample

> *data* := [[5.31, 11.16], [3.26, 3.26], [2.35, 2.35], [8.32, 14.34], [13.48, 49.45], [6.25, 15.05], [7.01, 7.01], [8.52, 8.52], [0.45, 0.45], [12.08, 12.08], [19.39, 10.42]]

data := [[5.31, 11.16], [3.26, 3.26], [2.35, 2.35], [8.32, 14.34], [13.48, 49.45], [6.25, 15.05], [7.01, 7.01], [8.52, 8.52], [0.45, 0.45], [12.08, 12.08], [19.39, 10.42]] **(3.6.1.9)**

An estimation of the joint cumulant $\kappa_{2,\,1}$ is

> *npolyk*([[2, 1]], *data*)

−23.73790404 **(3.6.1.10)**

An estimation of the product of the joint cumulant $\kappa_{2,\,1}\ \kappa_{1,\,0}$ is

> *npolyk*([[1, 1], [1, 0]], *data*)

294.2657621 **(3.6.1.11)**

An estimation of the product of the joint cumulant $\kappa_{2,\,1}\ \kappa_{2,\,0}\kappa_{1,\,0}$ is

> *npolyk*([[2, 1], [2, 0], [1, 0]], *data*)

12369.47450 **(3.6.1.12)**

▼ Joint Cumulant of polykays: auxiliary functions

The expression produced by the following set of functions corresponds to formula (4.10) pp. 96 in [8] . This formula expresses the cumulants of multivariate polykays in terms of joint cumulants of the population underlying the multivariate sample. This choice is motivated by the reduction of the overall final result complexity.
Referring to the simplest case of *k*-statistics, the procedure is as follows:
1) the cumulants of the *k*-statistics are expressed in terms of their moments using the function *ctr* ;
2) the moments of the *k*-statistics are expressed in terms of the moments of the population underlying the sample using the functions *pSS* and *E*;
3) the moments of the population underlying the sample are expressed in terms of its cumulants using the function *rct*.
The same procedure works for multivariate polykays.
In the following details are given for each step.

Cumulants and Moments

The *ctr function* (see [1]) expresses cumulants of a random vector in terms of joint moments $\mu_{i,j,\ldots}$.
Details on the procedure and mathematical background are given in [1] and [3]

Example: The joint cumulant $\kappa_{2,1}$ in terms of joint moments is

```
> ctr(2, 1)
```

$$2\,\mu_{0,1}\,\mu_{1,0}^{2} - \mu_{0,1}\,\mu_{2,0} - 2\,\mu_{1,0}\,\mu_{1,1} + \mu_{2,1} \qquad \textbf{(4.1.1)}$$

The *rct* function expresses joint moments of a random vector in terms of joint cumulants $\kappa_{i,j,\ldots}$.
Details on the procedure and mathematical background are given in [3].

```
> rtc := proc( )
  local vP, U, i, v, x;
      U := makeTab(args);
      vP := [seq(P||i, i = 1 ..nargs)];
      add(v[2]·mul(kappa[seq(degree(x, vP[i]), i = 1 ..nops(vP))], x = v[1]), v = makeTab(args))
  end proc:
```

Example: The joint moment $\mu_{2,1}$ in terms of joint cumulants is

```
> rtc(2, 1)
```

$$\kappa_{0,1}\,\kappa_{1,0}^{2} + \kappa_{0,1}\,\kappa_{2,0} + 2\,\kappa_{1,0}\,\kappa_{1,1} + \kappa_{2,1} \qquad \textbf{(4.1.2)}$$

Product of Augmented Symmetric Functions

Products of multivariate polykays return products of power sums. For example the expectation of $k_2\,k_1$ is:

$$E[k_2\,k_1] = E\left[\left(\frac{n\,S_2 - S_1^2}{n\,(n-1)}\right)\cdot\left(\frac{S_1}{n}\right)\right] = \frac{E[S_2 S_1]\,n - E[S_1^3]}{n^2\,(n-1)}$$

where E denotes the expectation. To express $E[S_2 S_1]$ and $E[S_1^3]$ in terms of moments, a fundamental result of estimation might be used if these products are transformed in augmented

symmetric functions, see [9]. For univariate sample, an augmented symmetric function involves products of variables with different indexes, that is

$$A_{[i,\,j,\,k,\ldots]} = \sum_{a \neq b \neq c \neq \ldots} X_a^i X_b^j X_c^k \ldots$$

The *pSS* function carries out this task.

```
> uRv := proc(u, v)
  local w, i;
   w := [sort([op(u[1]), v·'X']), u[2]],
    seq(`if`(has(u[1, i],'X'), NULL, [sort([op(u[1, 1..(i-1)]), u[1, i] + v·'X',
      op(u[1, (i + 1)..nops(u[1])])]), u[2]]), i = 1..nops(u[1]));
    w;
  end proc:

  combV := proc(pU, V)
      local U, v, u;
      U := [[pU, 1]];
      for v in V do
          U := [seq(uRv(u, v), u = U)]
      end do;
      add(u[2]·'A'[sort(u[1])], u = eval(U, ['X' = 1]))
  end proc:

  pS := proc(uS, vS)
      local u, v;
      u := coeffs(uS); v := coeffs(vS);
      u·v·combV([op(op(uS/u))], [op(op(vS/v))])
  end proc:

  pSS := proc()
      local u, v, U, V;
      V := sort([args], (proc(x, y) evalb(nops(x) > nops(y)) end));
      U := pS(S[V[1]], S[V[2]]);
      for v in V[3..nops(V)] do U := add(pS(u, S[v]), u = [op(U)]) od;
      U;
   end proc:
```

Example: product of power sums $S_1 \cdot S_2^3$ in terms of augmented symmetric functions:

```
> pSS([1], [2], [2], [2])
```

$$A_{[1,\,2,\,2,\,2]} + 3\,A_{[2,\,2,\,3]} + 3\,A_{[1,\,2,\,4]} + 3\,A_{[3,\,4]} + 3\,A_{[2,\,5]} + A_{[1,\,6]} + A_{[7]} \qquad \textbf{(4.2.1)}$$

Example: product of power sums $S_{2,\,1} \cdot S_{1,\,1} \cdot S_{2,\,3}$ in terms of multivariate augmented symmetric functions:

```
> pSS([[2, 1]], [[1, 1]], [[2, 3]])
```

(4.2.2)

$$A_{[[1,\,1],\,[2,\,1],\,[2,\,3]]} + A_{[[2,\,3],\,[3,\,2]]} + A_{[[1,\,1],\,[4,\,4]]} + A_{[[2,\,1],\,[3,\,4]]} + A_{[[5,\,5]]} \qquad \textbf{(4.2.2)}$$

The following is a test explaing how to transform the power sum product $S_{1,\,1} \cdot S_{1,\,2}$ with in augmented symmetric functions with $n=3$:

> $F1 := \left(\sum_{i=1}^{3} x_i y_i\right)\left(\sum_{i=1}^{3} x_i y_i^2\right) : F1 := sort(expand(evalf(F1)))$

$$F1 := x_1^2 y_1^3 + x_1 x_2 y_1^2 y_2 + x_1 x_2 y_1 y_2^2 + x_1 x_3 y_1^2 y_3 + x_1 x_3 y_1 y_3^2 + x_2^2 y_2^3 + x_2 x_3 y_2^2 y_3 + x_2 x_3 y_2 y_3^2 + x_3^2 y_3^3 \qquad \textbf{(4.2.3)}$$

> $pSS([[1,1]],[[1,2]])$

$$A_{[[1,\,1],\,[1,\,2]]} + A_{[[2,\,3]]} \qquad \textbf{(4.2.4)}$$

> $F2 := \sum_{j=1}^{3}\left(\sum_{i=1}^{3} x_i y_i x_j y_j^2\right) + \sum_{i=1}^{3} x_i^2 y_i^3 :$
> $F2 := sort\left(add\left(add\left(`if`\left(i=j,\,0,\,x_i y_i x_j y_j^2\right),\,i=1..3\right),\,j=1..3\right) + add\left(x_i^2 y_i^3,\,i=1..3\right)\right)$

$$F2 := x_1^2 y_1^3 + x_1 x_2 y_1^2 y_2 + x_1 x_2 y_1 y_2^2 + x_1 x_3 y_1^2 y_3 + x_1 x_3 y_1 y_3^2 + x_2^2 y_2^3 + x_2 x_3 y_2^2 y_3 + x_2 x_3 y_2 y_3^2 + x_3^2 y_3^3 \qquad \textbf{(4.2.5)}$$

> $evalb(F2 = F1);$

$$true \qquad \textbf{(4.2.6)}$$

Expected values of Augmented Simmetric Functions

The fundamental result of estimation we refer to allows to express expectation of augmented symmetric functions in terms of moments [10], that is

$$E\left[A_{[i,\,j,\,k,\ldots]}\right] = n(n\text{-}1)\ldots(n\text{-}s\text{+}1)\mu_i\,\mu_j\,\mu_k\ldots$$

where s denotes the number of elements in the list $[i, j, k, \ldots]$. The product $n(n\text{-}1)\ldots(n\text{-}s\text{+}1)$ is the lower factorial $(n)_s$. The df function computes the lower factorial. This result can be suitably generalized to the multivariate case.

> $df := \mathbf{proc}(m)$
> $\quad \mathbf{local}\ i;$
> $\quad mul((n-i), i=0..m-1)$
> $\mathbf{end}:$

Example: the decreasing factorial $(n)_3$

> $df(3)$

$$n\,(n-1)\,(n-2) \qquad \textbf{(4.3.1)}$$

The E function computes the expectation of augmented symmetric functions.

> $E := \mathbf{proc}()$
> $\quad \mathbf{local}\ v, vArgs, u, r;$
> $\quad \mathbf{if}\ nargs = 1\ \mathbf{then return}\ n \cdot \mu_{op(args)}\ \mathbf{end if};$
> $\quad vArgs := seq([u], u = [args]);$
> $\quad v := seq\left(\left[op(u)_1/2,\, op\left(op(u)_2\right)\right], u = [op(2 \cdot pSS(vArgs))]\right);$

```
    add( df( nops(u2) ) · u1 · mul( μ_op(r), r = u2 ), u = v )
  end proc:
```

Example: the expectation of $A_{[[1,\,1],\,[2,\,1]]} + A_{[[3,\,2]]}$ is

```
> E([1, 1], [2, 1]) + E([3, 2])
```

$$n\,(n-1)\,\mu_{1,\,1}\,\mu_{2,\,1} + 2\,n\,\mu_{3,\,2} \qquad \textbf{(4.3.2)}$$

The *fE* function repeatedly executes the *E* function on an expression.

```
> fE := proc(s)
    local sn, sd, vI, vE, u, w;
    sd := denom(s);
    sn := expand(numer(s));
    vI := indets(s) minus {n};
    vE := [seq(u = 1, u = vI)];
    if nops(vI) = 1 or type(sn, `*`) then
      sn := [seq([[seq([[op(w)], degree(u, w)], w = vI)], eval(u, vE)], u = [sn])]
    else
      sn := [seq([[seq([[op(w)], degree(u, w)], w = vI)], eval(u, vE)], u = sn)]
    end if;
    sn := [seq([[seq(`if`(w2 = 0, NULL, w), w = u1)], u2], u = sn)];
    expand( add( E( seq( w1$w2, w = u1 ) ) · u2, u = sn ) ) / sd
  end proc:
>
```

Example: the expectation of $\left(\frac{n\,S_2 - S_1^2}{n\,(n-1)}\right)\cdot\left(\frac{S_1}{n}\right)$ is

```
> fE(polyk([2]) · polyk([1]));
```

$$\frac{-n^3\,\mu_1^3 + n^3\,\mu_1\,\mu_2 + 3\,n^2\,\mu_1^3 - 4\,n^2\,\mu_1\,\mu_2 - 2\,n\,\mu_1^3 + n^2\,\mu_3 + 3\,n\,\mu_1\,\mu_2 - n\,\mu_3}{n^2\,(n-1)} \qquad \textbf{(4.3.3)}$$

Example: the expectation of $S_3^2 \cdot S_2 + S_4$ is

```
> fE( S3^2 · S2 + S4 )
```

$$n^3\,\mu_2\,\mu_3^2 - 3\,n^2\,\mu_2\,\mu_3^2 + n^2\,\mu_2\,\mu_6 + 2\,n^2\,\mu_3\,\mu_5 + 2\,n\,\mu_2\,\mu_3^2 - n\,\mu_2\,\mu_6 - 2\,n\,\mu_3\,\mu_5 + n\,\mu_4 + n\,\mu_8 \qquad \textbf{(4.3.4)}$$

Joint Cumulant of polykays: the main function

The *jcks* function returns the joint cumulants of multivariate polykays according to the procedure described in the previous section.

```
> jcks := proc(vK, vC)
  local s, vE, nv, dv, vS, evK, i, u, v;
  if (whattype(op(vK)) = 'list') then
      evK := [seq( polyk( op( vK_i ) ), i = 1 ..nops(vK) )];
      #print( seq( _polyk( op( vK_i ) ) = polyk( op( vK_i ) ), i = 1 ..nops(vK) ) );
```

```
else  evK := [seq(polyk(vK_i), i = 1 ..nops(vK))];
      #print(seq(_polyk(vK_i) = polyk(vK_i), i = 1 ..nops(vK)));
end if;
  s := ctr(op(vC));
  vE := [seq(u = fE(mul(evK_i^[op(u)], i = 1 ..nops([op(u)]))), u = indets(s))];
  s := simplify(eval(s, vE));
  vE := [seq(u = rtc(op(u)), u = [seq(`if`(op(0, u) = mu, u, NULL), u = indets(s))])];
  s := simplify(eval(s, vE));
  nv := numer(s);
  dv := denom(s);
  vS := [seq(v = 1, v = indets(s) minus {n})];
  if nops(vS) = 1 then return s end if;
  vS := [seq([u_1/u_2, u_2], u = [seq([v, eval(v, vS)], v = nv)])];
  vS := {seq([v_1, add(`if`(u_1 = v_1, u_2, 0), u = vS)], v = vS)};
  add(v_1·simplify(v_2/dv), v = vS)
end proc:
```

Examples

$\kappa_1(k_2)$ is the first cumulant of k_2, that can be computed as

```
> jcks([[2]], [1])
```

$$\kappa_2 \tag{5.1.1}$$

$\kappa_2(k_2)$ is the second cumulant of k_2, that can be computed as

```
> jcks([[2]], [2])
```

$$\frac{\kappa_4}{n} + \frac{2\,\kappa_2^2}{n-1} \tag{5.1.2}$$

$\kappa_3(k_2)$ is the third cumulant of k_2, that can be computed as

```
> jcks([[2]], [3])
```

$$\frac{\kappa_6}{n^2} + \frac{8\,\kappa_2^3}{(n-1)^2} + \frac{4\,\kappa_3^2\,(n-2)}{n\,(n-1)^2} + \frac{12\,\kappa_2\,\kappa_4}{n\,(n-1)} \tag{5.1.3}$$

In the following the variances of the first few *k*-statistics are computed, see [11]. Note that Var(X) = Cov(X, X).

```
> var(k_1) = jcks([[1], [1]], [1, 1])
```

$$var(k_1) = \frac{\kappa_2}{n} \tag{5.1.4}$$

```
> var(k_2) = jcks([[2], [2]], [1, 1])
```

$$var(k_2) = \frac{\kappa_4}{n} + \frac{2\,\kappa_2^2}{n-1} \tag{5.1.5}$$

> $var(k_3) = jcks([[3],[3]],[1,1])$

$$var(k_3) = \frac{\kappa_6}{n} + \frac{6\,\kappa_2^3\,n}{(n-1)\,(n-2)} + \frac{9\,\kappa_3^2}{n-1} + \frac{9\,\kappa_2\,\kappa_4}{n-1} \tag{5.1.6}$$

> $var(k_4) = jcks([[4],[4]],[1,1])$

$$var(k_4) = \frac{\kappa_8}{n} + \frac{24\,\kappa_2^4\,n\,(n+1)}{(n-1)\,(n-2)\,(n-3)} + \frac{34\,\kappa_4^2}{n-1} + \frac{144\,\kappa_2\,\kappa_3^2\,n}{(n-1)\,(n-2)} + \frac{16\,\kappa_2\,\kappa_6}{n-1} + \frac{72\,\kappa_2^2\,\kappa_4\,n}{(n-1)\,(n-2)} + \frac{48\,\kappa_3\,\kappa_5}{n-1} \tag{5.1.7}$$

More examples are given in [10], pag. 265.
The joint cumulant $\kappa(2^2\,1)$ of order (2, 1) of (k_2, k_1) is

> $jcks([[2],[1]],[2,1])$

$$\frac{\kappa_5}{n^2} + \frac{4\,\kappa_2\,\kappa_3}{n\,(n-1)} \tag{5.1.8}$$

The joint cumulant $\kappa(2^2\,1^2)$ of order (2, 2) of (k_2, k_1) is

> $jcks([[2],[1]],[2,2])$

$$\frac{\kappa_6}{n^3} + \frac{4\,\kappa_3^2}{n^2\,(n-1)} + \frac{4\,\kappa_2\,\kappa_4}{n^2\,(n-1)} \tag{5.1.9}$$

The joint cumulant $\kappa(3^1\,2^1)$ of order (1, 1) of (k_3, k_2) is (see[11] pag. 271)

> $jcks([[3],[2]],[1,1])$

$$\frac{\kappa_5}{n} + \frac{6\,\kappa_2\,\kappa_3}{n-1} \tag{5.1.10}$$

The joint cumulant $\kappa(4^1\,2^2)$ of order (1,2) of (k_4, k_2) is (see[11] pag. 272)

> $jcks([[4],[2]],[1,2])$

$$\frac{\kappa_8}{n^2} + \frac{4\,\kappa_4^2\,(7\,n-10)}{n\,(n-1)^2} + \frac{120\,\kappa_2\,\kappa_3^2}{(n-1)^2} + \frac{20\,\kappa_2\,\kappa_6}{n\,(n-1)} + \frac{80\,\kappa_2^2\,\kappa_4}{(n-1)^2} + \frac{8\,\kappa_3\,\kappa_5\,(5\,n-7)}{n\,(n-1)^2} \tag{5.1.11}$$

The joint cumulant of order (1,1) of $(k^{r,s}, k^{t,u})$ is (see[8] pag.94)

> $jcks([[1,1,0,0],[0,0,1,1]],[1,1])$

$$\frac{\kappa_{1,1,1,1}}{n} + \frac{\kappa_{1,0,0,1}\,\kappa_{0,1,1,0}}{n-1} + \frac{\kappa_{1,0,1,0}\,\kappa_{0,1,0,1}}{n-1} \tag{5.1.12}$$

The joint cumulant of order (1,1,1) of $\kappa_k(r, s|t, u|v, w)$ is (see[8] pag.94)

> *jcks*([[1, 1, 0, 0, 0, 0], [0, 0, 1, 1, 0, 0], [0, 0, 0, 0, 1, 1]], [1, 1, 1])

$$\frac{\kappa_{1,0,1,1,1,0}\,\kappa_{0,1,0,0,0,1}}{n\,(n-1)} + \frac{\kappa_{1,1,0,1,0,1}\,\kappa_{0,0,1,0,1,0}}{n\,(n-1)} + \frac{\kappa_{1,1,0,1,1,0}\,\kappa_{0,0,1,0,0,1}}{n\,(n-1)} \tag{5.1.13}$$

$$+ \frac{\kappa_{1,1,1,0,0,1}\,\kappa_{0,0,0,1,1,0}}{n\,(n-1)} + \frac{\kappa_{1,1,1,0,1,0}\,\kappa_{0,0,0,1,0,1}}{n\,(n-1)}$$

$$+ \frac{\kappa_{1,0,0,0,0,1}\,\kappa_{0,1,0,1,0,0}\,\kappa_{0,0,1,0,1,0}}{(n-1)^2} + \frac{\kappa_{1,0,0,0,0,1}\,\kappa_{0,1,1,0,0,0}\,\kappa_{0,0,0,1,1,0}}{(n-1)^2}$$

$$+ \frac{\kappa_{1,0,0,0,1,0}\,\kappa_{0,1,0,1,0,0}\,\kappa_{0,0,1,0,0,1}}{(n-1)^2} + \frac{\kappa_{1,0,0,0,1,0}\,\kappa_{0,1,1,0,0,0}\,\kappa_{0,0,0,1,0,1}}{(n-1)^2}$$

$$+ \frac{\kappa_{1,0,0,1,0,0}\,\kappa_{0,1,0,0,0,1}\,\kappa_{0,0,1,0,1,0}}{(n-1)^2} + \frac{\kappa_{1,0,0,1,0,0}\,\kappa_{0,1,0,0,1,0}\,\kappa_{0,0,1,0,0,1}}{(n-1)^2}$$

$$+ \frac{\kappa_{1,0,1,0,0,0}\,\kappa_{0,1,0,0,0,1}\,\kappa_{0,0,0,1,1,0}}{(n-1)^2} + \frac{\kappa_{1,0,1,0,0,0}\,\kappa_{0,1,0,0,1,0}\,\kappa_{0,0,0,1,0,1}}{(n-1)^2}$$

$$+ \frac{\kappa_{1,0,0,1,0,1}\,\kappa_{0,1,1,0,1,0}\,(n-2)}{n\,(n-1)^2} + \frac{\kappa_{1,0,0,1,1,0}\,\kappa_{0,1,1,0,0,1}\,(n-2)}{n\,(n-1)^2}$$

$$+ \frac{\kappa_{1,0,1,0,0,1}\,\kappa_{0,1,0,1,1,0}\,(n-2)}{n\,(n-1)^2} + \frac{\kappa_{1,0,1,0,1,0}\,\kappa_{0,1,0,1,0,1}\,(n-2)}{n\,(n-1)^2}$$

$$+ \frac{\kappa_{1,0,0,0,0,1}\,\kappa_{0,1,1,1,1,0}}{n\,(n-1)} + \frac{\kappa_{1,0,0,0,1,0}\,\kappa_{0,1,1,1,0,1}}{n\,(n-1)} + \frac{\kappa_{1,0,0,1,0,0}\,\kappa_{0,1,1,0,1,1}}{n\,(n-1)}$$

$$+ \frac{\kappa_{1,0,0,1,1,1}\,\kappa_{0,1,1,0,0,0}}{n\,(n-1)} + \frac{\kappa_{1,0,1,0,0,0}\,\kappa_{0,1,0,1,1,1}}{n\,(n-1)} + \frac{\kappa_{1,0,1,0,1,1}\,\kappa_{0,1,0,1,0,0}}{n\,(n-1)}$$

$$+ \frac{\kappa_{1,0,1,1,0,1}\,\kappa_{0,1,0,0,1,0}}{n\,(n-1)} + \frac{\kappa_{1,1,1,1,1,1}}{n^2}$$

$\kappa_2(k_3)$ is the second cumulant of k_3 (the *k-statistic* of order 3), that can be computed as

> *jcks*([[3]], [2])

$$\frac{\kappa_6}{n} + \frac{6\,\kappa_2^3\,n}{(n-1)\,(n-2)} + \frac{9\,\kappa_3^2}{n-1} + \frac{9\,\kappa_2\,\kappa_4}{n-1} \tag{5.1.14}$$

$\kappa_2(k_{2,1})$ is the second cumulant of

$k_{2,\,1}$ (the multivariate k-statistics of order $(2, 1)$), that can be computed as

```
> jcks([[[2, 1]]], [2])
```

$$\frac{\kappa_{4,\,2}}{n}+\frac{5\,\kappa_{2,\,1}^{2}}{n-1}+\frac{4\,\kappa_{1,\,1}\,\kappa_{3,\,1}}{n-1}+\frac{4\,\kappa_{1,\,1}^{2}\,\kappa_{2,\,0}\,n}{(n-1)\,(n-2)}+\frac{4\,\kappa_{1,\,2}\,\kappa_{3,\,0}}{n-1}+\frac{2\,\kappa_{2,\,0}^{2}\,\kappa_{0,\,2}\,n}{(n-1)\,(n-2)}+\frac{4\,\kappa_{2,\,2}\,\kappa_{2,\,0}}{n-1}+\frac{\kappa_{4,\,0}\,\kappa_{0,\,2}}{n-1} \tag{5.1.15}$$

$\kappa_2\left(k_{2;\,1}\right)$ is the second cumulant of $k_{2;\,1}$ (the polykay of order $(2, 1)$), that can be computed as

```
> jcks([[[2], [1]]], [2])
```

$$\frac{\kappa_2^{3}\left(n^{2}-n+4\right)}{n\,(n-1)\,(n-2)}+\frac{\kappa_3^{2}}{n\,(n-1)}+\frac{2\,\kappa_1^{2}\,\kappa_2^{2}}{n-1}+\frac{\kappa_1^{2}\,\kappa_4}{n}+\frac{\kappa_2\,\kappa_4}{n\,(n-1)}+\frac{2\,\kappa_1\,\kappa_2\,\kappa_3\,(n-3)}{n\,(n-1)} \tag{5.1.16}$$

$\kappa_2\left(k_{2;\,1}\right)$ is the second cumulant of
$k_{2;\,1}$ (multivariate polykay of order $([1, 1], [1, 0])$), that can be computed as

```
> jcks([[[1, 1], [1]]], [2])
```

$$\frac{\kappa_{2,\,1}^{2}}{n\,(n-1)}+\frac{\kappa_{1,\,0}^{2}\,\kappa_{2,\,2}}{n}+\frac{\kappa_{1,\,1}^{2}\,\kappa_{1,\,0}^{2}}{n-1}+\frac{\kappa_{1,\,1}^{2}\,\kappa_{2,\,0}\left(n^{2}-2\,n+4\right)}{n\,(n-1)\,(n-2)}+\frac{\kappa_{2,\,0}^{2}\,\kappa_{0,\,2}}{(n-1)\,(n-2)}+\frac{\kappa_{2,\,2}\,\kappa_{2,\,0}}{n\,(n-1)}+\frac{\kappa_{1,\,0}^{2}\,\kappa_{2,\,0}\,\kappa_{0,\,2}}{n-1}+\frac{2\,\kappa_{1,\,1}\,\kappa_{1,\,0}\,\kappa_{2,\,1}\,(n-2)}{n\,(n-1)}-\frac{2\,\kappa_{1,\,2}\,\kappa_{1,\,0}\,\kappa_{2,\,0}}{n\,(n-1)} \tag{5.1.17}$$

```
> jcks([[[1, 1], [1 , 1]]], [1, 1])
```

$$\frac{\kappa_{2,\,2}}{n}+\frac{\kappa_{1,\,1}^{2}}{n-1}+\frac{\kappa_{2,\,0}\,\kappa_{0,\,2}}{n-1} \tag{5.1.18}$$

```
> jcks([[[1, 1], [1 , 1]]], [1])
```

$$\kappa_{1,\,1}^{2} \tag{5.1.19}$$

```
>
```

Estimation of joint cumulants of polykays

Given a random numerical sample, the *njcks* function allows estimating the joint cumulants of multivariate polykays. The function invokes the *nployk* function in the background section. The first two parameters are the same as in the *njcks* function, while the data vector must be entered as third parameter. The procedure consists in expressing the joint cumulant of a multivariate polykay in terms of joint cumulants of the underlying population and then replacing the occurrences of these joint cumulants and/or their products with the corresponding multivariate numerical polykays.

```
> njcks := proc(vK, vC, vD)
    local s, u, sn, sd, vI, vE, N;
```

```
    s := jcks(vK, vC);
    if nops(vK[1]) = 1 then N := nops(op(vD)); else N := nops(vD); end if;
    sd := denom(s);
    sn := expand(numer(s));
    vI := indets(s) minus {n};
    vE := [seq(u = 1, u = vI)];
    if nops(vI) = 1 or type(sn, `*`) then
        sn := [seq([[seq([[op(w)], degree(u, w)], w = vI)], eval(u, vE)], u = [sn])]
    else
        sn := [seq([[seq([[op(w)], degree(u, w)], w = vI)], eval(u, vE)], u = sn)]
    end if;
    sn := [seq([[seq(`if`(w2 = 0, NULL, w), w = u1)], u2], u = sn)];
    eval(expand(add(npolyk([seq(w1$w2, w = u1)], vD) · u2, u = sn)) / sd, ['n' = N]);
end:
```

Example: suppose to have the following (univariate) numerical sample

```
> data1 := [[16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08,
      19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83, 16.98,
      19.92, 9.47, 11.68, 13.41, 15.35, 19.11]];
```

$$data1 := [[16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08, 19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83, 16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11]] \quad \textbf{(5.2.1)}$$

The estimation of the joint cumulant of order (2,2) of (k_2, k_1) is

```
> njcks([[2], [1]], [2, 2], data1)
```

$$0.01919574411 \quad \textbf{(5.2.2)}$$

Example: suppose to have the following (bivariate) numerical sample

```
> data2 := [[5.31, 11.16], [3.26, 3.26], [2.35, 2.35], [8.32, 14.34], [13.48, 49.45], [6.25,
      15.05], [7.01, 7.01], [8.52, 8.52], [0.45, 0.45], [12.08, 12.08], [19.39, 10.42]];
```

$$data2 := [[5.31, 11.16], [3.26, 3.26], [2.35, 2.35], [8.32, 14.34], [13.48, 49.45], [6.25, 15.05], [7.01, 7.01], [8.52, 8.52], [0.45, 0.45], [12.08, 12.08], [19.39, 10.42]] \quad \textbf{(5.2.3)}$$

An estimation of the joint cumulant of order (1,1) of $k_{1,1}\, k_{1,2}$ is

```
> njcks([[1, 1], [1, 2]], [1, 1], data2)
```

$$18250.79716 \quad \textbf{(5.2.4)}$$

```
>
```

Conclusions

The *jcks* function computes the cumulants of multivariate polykays. These routines are particularly useful for assessing the goodness of the estimate obtained through these estimators. For example, the variance of first- and fourth-order *k*-statistics provides insights into the goodness-of-fit estimation of the mean, variance, skeweness, and kurtosis, respectively. Further examples are provided in the last section of this worksheet.

References

[1] (2009) Di Nardo E., Guarino G., Senato D. Fast algorithms for *k*-statistics, polykays and their multivariate generalizations. (Worksheet Maple Software: available at https://www.maplesoft.com/Applications/Detail.aspx?id=33041)

[2] (2009) Di Nardo E., Guarino G., Senato D. Multiset subdivisions. (Worksheet Maple Software: : available at https://www.maplesoft.com/Applications/Detail.aspx?id=33039).

[3] (2015) Di Nardo E. Symbolic calculus in mathematical statistics: a review. Seminaire Lotharingien de Combinatoire Vol. 67 (B67a), pp. 72

[4] (2009) Di Nardo E., Guarino G., Senato D. A new method for fast computing unbiased estimators of cumulants. Statistics and Computing 19, 155--165.

[5] (2008) Di Nardo E., Guarino G., Senato D. A unifying framework for *k*-statistics, polykays and their multivariate generalizations. Bernoulli 14, 440--468.

[6] (2006) Di Nardo E., Senato D. A symbolic method for *k*-statistics. Applied Mathematics Letters 19, 968--975.

[7] (2011) Di Nardo E., Guarino G., Senato D. A new algorithm for computing the multivariate Faa di Bruno's formula. Applied Mathematics and Computation 217, 6286--6295

[8] P. McCullagh, Tensor Methods in Statistics, Monographs on Statistics and Applied Probability, Taylor & Francis Group, 22 December 2017

[9] (2008) Di Nardo E., Guarino G., Senato D. Symbolic computation of moments of sampling distributions. Computational Statistics and Data Analysis 52, 4909--4922.

[10] Stuart, A., Ord, J.K., 1987. Kendall’s Advanced Theory of Statistics 1. Charles Griffin and Company Limited, London.

[11] Weisstein, Eric W. "k-Statistic." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/k-Statistic.html

>

>