This Maple worksheet accompanies the papers:

Di Nardo E., G. Guarino, D. Senato (2008), *A Maple algorithm for polykays and their generalizations*, Adv. Appl. Stat. Vol. 8, No. 1, 19 - 36, http://www.pphmj.com/journals/adas.htm.

Di Nardo E., G. Guarino, D. Senato (2008), *An unifying framework for k-statistics, polykays and their generalizations,* Bernoulli. Vol. 14(2), 440-468. Official Journal of the Bernoulli Society for Mathematical Statistics and Probability, http://isi.cbs.nl/bernoulli/, (download from http://www.unibas.it/utenti/dinardo/lavori.html)

Di Nardo E., G. Guarino, D. Senato (2008), *Symbolic computation of moments of sampling distributions,* Comp. Stat. Data Analysis Vol. 52, no. 11, 4909-4922, (download from http://arxiv.org/PS_cache/arxiv/pdf/0806/0806.0129v1.pdf or http://www.unibas.it/utenti/dinardo/lavori.html)

A Maple algorithm for k-statistics, polykays and their multivariate generalization

E. Di Nardo*
elvira.dinardo@unibas.it
http://www.unibas.it/utenti/dinardo/home.html;
Tel: +39 0971205890, Fax: +39 0971205896

G. Guarino**
giuseppe.guarino@asl2.potenza.it

D. Senato*
domenico.senato@unibas.it

* Dipartimento di Matematica e Informatica, Università degli Studi della Basilicata,Viale dell'Ateneo Lucano n.10, 85100 Potenza, Italy

**Medical Scool, Università del Sacro Cuore (Rome branch), Largo Agostino Gemelli n.8, 00168 Roma, Italy

Introduction

***Abstract*:** Through the classical umbral calculus, we provide a unifying syntax for single and multivariate k-statistics, polykays and multivariate polykays. Classical umbral calculus is a simbolic method for handling ordinary and exponential formal power series. A feature of the classical umbral calculus is that an umbra has the structure of a random variable but with non reference to a probability space. This brings the umbral syntax closer to statistical methods.

Application Areas/Subject: Combinatorics & algebraic methods in statistics
Keyword: umbral calculus, symmetric polynomials, set partitions, multiset, cumulants, k-statistics, polykays.
See Also: Maple algorithm [3]

Initialization

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

[*partition*, *multinomial*] **(2.1)**

Subdivision of Multiset and Augumented symmetric function to Power Sum

The following algorithm function is used for listing all subdivision of a multiset. Theese algorithms is fully discussed in [3]

Subdivision of Multiset

See [3] for complete discussion about this algorithm.

```
> nRep :=proc(u) 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 := [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:

comb:=proc(V, ptr, Y)
    if ptr = nops(V) + 1 then return Y end if;
    seq(comb(V, ptr + 1, [op(Y), L]), L = V[ptr])
 end proc:
```

The output of the following new function is a disjoint union of vectors.

```
> unionVects :=proc(U::list, V::list)
    if nops(U) = 0 then V
    elif nops(V) = 0 then U
        else [seq(seq([sort([[op(v[1]), op(u[1])]]), v[2]·u[2]], u = U), v = V)] end if
  end proc:
```

Example

```
> A := [[[a, a, a], 1], [[a, a^2], 3], [[a^3], 1]];
  B := [[[b, b], 1], [[b^2], 1]];
  DisjointUnion = unionVects(A, B)
```

$$A := [[[a, a, a], 1], [[a, a^2], 3], [[a^3], 1]]$$

$$B := [[[b, b], 1], [[b^2], 1]]$$

$$DisjointUnion = [[[a, a, a, b, b], 1], [[a, b, b, a^2], 3], [[b, b, a^3], 1], [[a, a, a, b^2], 1], [[a, b^2, a^2], 3], [[b^2, a^3], 1]] \quad \textbf{(3.1.1)}$$

The output of the following function is a multiset subdivision. This algorithm is described in [3].
Remark: here the function *maketab* is slightly different from the function *maketab* in [3]. The input parameters are changed. In [3] we just give the multiplicities of elements. In this function we also name the elements. For example if all partitions of the multiset [a,a,b] are needed we can recall *makeTab([a,a,b])* while in [3] the function calls *makeTab(2,1)*.
Note: at the end of the function we have inserted a code block in order to calculate additional values of the coefficients for every term of the final expression. This is usefull either in the

construction of the k-statistics either in the conversion between augumented symmetric functions and power sums.

```
> makeTab := proc( )
    local vParts, vEvalTo, vEvalBack, vIndets, U, V, v;
    vParts := sort( convert( args[1], multiset ), proc(x, y) evalb( y[2] < x[2] ) end proc );
    vEvalBack := [ seq( P || i = vParts[i, 1], i = 1 ..nops( vParts ) ) ];
    vEvalTo := [ seq( vParts[i, 1] = P || i, i = 1 ..nops( vParts ) ) ];
    vIndets := { op( sort( eval( args[1], vEvalTo ) ) ) };
    vParts := eval( vParts, vEvalTo );
    U := [ seq( [ seq( [ [ seq( p[1]^z, z = y ) ] ], multinomial( p[2], seq( r, r = y ) ) ], y
     = partition( p[2] ) ) ], p = vParts ) ];
    if nops( U ) = 1 then
        U := [ seq( [ [ x[1], x[2]/nRep( x[1] ) ] ], x = op( U ) ) ]
    else
        U := [ seq( URmV( op( x ) ), x = [ comb( U, 1, [ ] ) ] ) ]
    end if;
    if nargs = 1 then
    elif args[2] = 'augToPs( )' then
        U := [ seq( [ y[1], mul( ( -1 )^(degree(x) - 1) * ( degree( x ) - 1 )!, x = y[1] ) * y[2] ], y = U ) ]
    elif args[2] = 'polykays( )' then
        U := [ seq( [ y[1], ( -1 )^(nops( y[1] ) - 1) * ( nops( y[1] ) - 1 )! * y[2] ], y = U ) ]
    end if;
    eval( U, vEvalBack )
  end proc:
> makeTab( [ a, a, a ] )
```

$$[[[a, a, a], 1], [[a, a^2], 3], [[a^3], 1]] \quad \textbf{(3.1.2)}$$

```
> makeTab( [ a, a, b ] )
```

$$[[[a\,b, a], 2], [[a, a, b], 1], [[a^2\,b], 1], [[a^2, b], 1]] \quad \textbf{(3.1.3)}$$

```
> makeTab( [ a, a, a^3 ] )
```

$$[[[a^4, a], 2], [[a, a, a^3], 1], [[a^5], 1], [[a^2, a^3], 1]] \quad \textbf{(3.1.4)}$$

Augumented symmetric function to Power Sum

The output of this function is a symmetric function expressed in terms of power sum symmetric function, e.g.

$$S_r = \sum_{i=i}^{n} X_i^r$$

Note:

the symbol *[a,a,a]* represents the subdivision *{{a}{a}{a}}* and the symbol $[a^2,a\,]$ represents the

subdivision *{{a,a},{a}}*.
The number associated to each symbol represents a product of $(-1)^{k-1}\cdot(k-1)!$, where *k* is the degree of the monomial in each block, multiplied by the integer associated to the same symbol in the output of *maketab*.
For example: for the symbol $[a^2, a]$= *{{a,a},{a}}* we calculate $[(-1)^{2-1}\cdot(2-1)!]\cdot[(-1)^{1-1}\cdot(1-1)!] = [-1\cdot 1] = -1$ and we multiply -1 with 3 that is the multiplicity of the subdivision:

Example: how we calculate *augToPs([a,a,a])*

1) we recall *makeTab([a,a,a])* having following result:

$$[[[a, a, a], 1], [[a, a^2], 3], [[a^3], 1]]$$

2) we calculate:

2.1) block $[a, a, a]$:

$$[(-1)^{1-1}(1-1)!][(-1)^{1-1}(1-1)!][(-1)^{1-1}(1-1)!] = 1\cdot 1\cdot 1 = 1$$

2.2) block $[a, a^2]$:

$$[(-1)^{1-1}(1-1)!][(-1)^{2-1}(2-1)!] = -1\cdot 1 = -1$$

2.3) block $[a^3]$:

$$[(-1)^{3-1}(3-1)!] = 2$$

3) Multipling the multiplicities in the step 1) with the results found in step 2), we have:

$$[[[a, a, a], 1], [[a, a^2], -3], [[a^3], 2]]$$

4) from which we can express the following result:

$$S_1^3 - 3\,S_1\,S_2 + 2\,S_3$$

```
> augToPs := proc( )
      local vIndets;
      if nargs = 0 then
          return NULL
      elif nargs = 1 then vIndets := sort( [op( indets( args[1] ) )] )
                     else vIndets := args[2] end if;
      add( mul( S[seq(degree(x, vIndets[i]), i = 1 ..nops(vIndets))], x = y[1] ) * y[2], y = makeTab( args[1],
      'procname( )' ) )
  end proc:
```

Examples

```
> augToPs( [a, a, a] )
```

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

```
> augToPs( [a, b] )
```

$$-S_{1,1} + S_{0,1}\,S_{1,0} \qquad \textbf{(3.2.2)}$$

```
> augToPs( [a, a^2] );
```

$$-S_3 + S_1\,S_2 \qquad \textbf{(3.2.3)}$$

the step 3) in the previouse note.

> *makeTab*([*a*, *a*, *a*], '*augToPs*( )')

$$[[[a, a, a], 1], [[a, a^2], -3], [[a^3], 2]] \quad \textbf{(3.2.4)}$$

A unifying framework for k-statistics, polykays and their multivariate generalizations

The nth k-statistic is the unique symmetric unbiased estimator of the cumulant κ_n of a given statistical distribution.

$$k_n \text{ is defined so that } \mathrm{E}[k_n] = \kappa_n$$

The symmetric statistic $k_{r, s, ..}$ is defined as

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

where κ_r is a cumulant. These statistics called polykays, generalize the k-statistics. K-statistics, polykays and their multivariate generalization are commonly defined in terms of power sums, that are sums of the rth powers of the data points:

$$S_r = \sum_{i=i}^{n} X_i^r$$

Note: in polyk we calculate an additional value: the cardinality of subdivision. For every subdivision we calculate $(-1)^{k-1}(k-1)!$ where k is its cardinality.
For example $[a^2, a, b] = \{\{a, a\}, \{a\}, \{b\}\}$ has 3 blocks and we calculate $(-1)^{3-1}(3-1)! = 2$

Example: how we calculate *Polykays(3)*

1) we call *makeTab([a,a,a])* and this is the output:

$$[[[a, a, a], 1], [[a, a^2], 3], [[a^3], 1]]$$

2) we calculate:
   2.1) block $[a, a, a]$: $[(-1)^{3-1}(3-1)!] = 2$
   2.2) block $[a, a^2]$: $[(-1)^{2-1}(2-1)!] = -1$
   2.3) block $[a^3]$: $[(-1)^{1-1}(1-1)!] = 1$

3) Multipling the multiplicities in step 1) with the results found in step 2) we have:

$$[[[a, a, a], 2], [[a, a^2], -3], [[a^3], 1]]$$

4) from which we calculate:

$$\frac{2\ augToPs([a, a, a])}{n\ (n-1)\ (n-2)} - \frac{3\ augToPs([a, a^2])}{n\ (n-1)} + \frac{1\ augToPs([a^3])}{n}$$

$$= \frac{2\,S_1^3 - 6\,S_1\,S_2 + 4\,S_3}{n\,(n-1)\,(n-2)} + \frac{3\,S_3 - 3\,S_1\,S_2}{n\,(n-1)} + \frac{S_3}{n}$$

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

The following function gives the expression for k-statistics and polykays and their multivariate generalizzations depending on the input parameters:

- 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]

```
> polykays := proc( )
    local nRes, nNum, vPar, vParP, vEval, vIndets, vTab, k, i, x, T, TotN, px;
    if args[1] = [0] and nargs = 1 then return 1 end if;
    vPar := [ ]; vEval := [ ]; vParP := [ ]; vIndets := [ ];
    TotN := add(add(x, x = y), y = args) - 1;
    vIndets := [seq(X||i, i = 1 ..max(seq(nops(x), x = args)))];
    for k to nargs do
        if args[k] = [0] then next end if;
        for i to nops(args[k]) do
            vParP := [op(vParP), X||i||Y||k$args[k, i]];
            vEval := [op(vEval), X||i||Y||k = X||i]
        end do;
        vPar := [op(vPar), vParP];
        vParP := [ ]
    end do;
    vTab := makeTab(vPar[1],'procname( )');
    for x from 2 to nops(vPar) do
        vTab := unionVects(vTab, makeTab(vPar[x],'procname( )'))
    end do;
    vTab := eval(vTab, vEval);
    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;
    for x from 0 to TotN do
        T[x+1] := 0
    end do;
    for x in vTab do
        px := nops(x[1]);
```

```
        Tpx := Tpx + x2·augToPs(x1, vIndets)
    end do;
    nRes := add( T(x+1) / mul(n − t, t = 0..x), x = 0..TotN );
    nNum := numer(nRes);
    nNum := collect(nNum, `minus`(indets(nNum), {n}), distributed);
    nRes := nNum / denom(nRes);
    return nRes
end proc:
```

examples: k-statistics

> *polykays*([3])

$$\frac{S_3\,n^2 - 3\,S_1\,S_2\,n + 2\,S_1^3}{n\,(n-1)\,(n-2)} \tag{4.1}$$

example: polykays

> *polykays*([2], [1])

$$\frac{-S_1^3 - S_3\,n + (n+1)\,S_1\,S_2}{n\,(n-1)\,(n-2)} \tag{4.2}$$

example: multivariate k-statistics

> *polykays*([2, 1])

$$\frac{S_{2,1}\,n^2 - 2\,S_{1,0}\,S_{1,1}\,n - S_{2,0}\,S_{0,1}\,n + 2\,S_{1,0}^2\,S_{0,1}}{n\,(n-1)\,(n-2)} \tag{4.3}$$

example: multivariate polykays

> *polykays*([1, 1], [1])

$$\frac{-S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1}}{n\,(n-1)\,(n-2)} \tag{4.4}$$

Example: steps for k-statistics and polykays construction

k-Statistics k_3

> *vTab* := *makeTab*([*a*, *a*, *a*], '*polykays*( )')

$$vTab := [[[a, a, a], 2], [[a, a^2], -3], [[a^3], 1]] \tag{4.1.1}$$

> $kstat := \frac{2\,augToPs([a,a,a])}{n\,(n-1)\,(n-2)} - \frac{3\,augToPs([a,a^2])}{n\,(n-1)} + \frac{1\,augToPs([a^3])}{n}$

$$kstat := \frac{2\,(S_1^3 - 3\,S_1\,S_2 + 2\,S_3)}{n\,(n-1)\,(n-2)} - \frac{3\,(-S_3 + S_1\,S_2)}{n\,(n-1)} + \frac{S_3}{n} \tag{4.1.2}$$

> $k_3 := simplify(kstat)$

$$k_3 := \frac{2\,S_1^3 - 3\,S_1\,S_2\,n + S_3\,n^2}{n\,(n-1)\,(n-2)} \tag{4.1.3}$$

Test previous result

> *polykays*([3])

$$\frac{2\,S_1^3 - 3\,S_1\,S_2\,n + S_3\,n^2}{n\,(n-1)\,(n-2)} \tag{4.1.4}$$

> *evalb*(%=*k*[3])

$$true \tag{4.1.5}$$

polykays $k_{2;1}$

> *A* := *makeTab*([*a*, *a*], 'polykays( )')

$$A := [[[a, a], -1], [[a^2], 1]] \tag{4.1.6}$$

> *B* := *makeTab*([*b*], 'polykays( )')

$$B := [[[b], 1]] \tag{4.1.7}$$

> *vTab* := *unionVects*(*A*, *B*)

$$vTab := [[[a, a, b], -1], [[b, a^2], 1]] \tag{4.1.8}$$

> *vTab* := *eval*(*vTab*, [*b*=*a*])

$$vTab := [[[a, a, a], -1], [[a, a^2], 1]] \tag{4.1.9}$$

> $kstat := -\frac{1\ augToPs([a, a, a])}{n\,(n-1)\,(n-2)} + \frac{1\ augToPs([a, a^2])}{n\,(n-1)}$

$$kstat := -\frac{S_1^3 - 3\,S_1\,S_2 + 2\,S_3}{n\,(n-1)\,(n-2)} + \frac{-S_3 + S_1\,S_2}{n\,(n-1)} \tag{4.1.10}$$

> *kstat* := *simplify*(*kstat*)

$$kstat := \frac{-S_1^3 + S_1\,S_2 - S_3\,n + S_1\,S_2\,n}{n\,(n-1)\,(n-2)} \tag{4.1.11}$$

> *nNum* := *numer*(*kstat*); *nDenom* := *denom*(*kstat*)

$$nNum := -S_1^3 + S_1\,S_2 - S_3\,n + S_1\,S_2\,n$$

$$nDenom := n\,(n-1)\,(n-2) \tag{4.1.12}$$

> *nNum* := *collect*(*nNum*, *indets*(*nNum*) \ {*n*}, *distributed*)

$$nNum := -S_1^3 - S_3\,n + (1+n)\,S_1\,S_2 \tag{4.1.13}$$

> $k[`2;1`] := \frac{nNum}{nDenom}$

$$k_{2;1} := \frac{-S_1^3 - S_3\,n + (1+n)\,S_1\,S_2}{n\,(n-1)\,(n-2)} \tag{4.1.14}$$

test previous result

> *polykays*([2], [1])

$$\frac{-S_1^3 - S_3\, n + (1+n)\, S_1\, S_2}{n\,(n-1)\,(n-2)} \tag{4.1.15}$$

> $evalb(\%=k[\,`2;1`\,])$

$$true \tag{4.1.16}$$

multivariate k-Statistics $k_{2,\,1}$

> $vTab := makeTab([a, a, b], 'polykays(\,)')$

$$vTab := [[[a\,b, a], -2], [[a, a, b], 2], [[a^2\,b], 1], [[a^2, b], -1]] \tag{4.1.17}$$

> $kstat := -\frac{2\,augToPs([a\,b, a])}{n\,(n-1)} + \frac{2\,augToPs([a, a, b])}{n\,(n-1)\,(n-2)} + \frac{1\,augToPs([a^2\,b])}{n} - \frac{1\,augToPs([a^2, b])}{n\,(n-1)}$

$$kstat := -\frac{2\,(-S_{2,\,1} + S_{1,\,0}\,S_{1,\,1})}{n\,(n-1)} + \frac{2\,(-2\,S_{1,\,0}\,S_{1,\,1} + S_{1,\,0}^2\,S_{0,\,1} + 2\,S_{2,\,1} - S_{2,\,0}\,S_{0,\,1})}{n\,(n-1)\,(n-2)} + \frac{S_{2,\,1}}{n} - \frac{-S_{2,\,1} + S_{2,\,0}\,S_{0,\,1}}{n\,(n-1)} \tag{4.1.18}$$

> $k[2, 1] := simplify(kstat)$

$$k_{2,\,1} := \frac{-2\,S_{1,\,0}\,S_{1,\,1}\,n + 2\,S_{1,\,0}^2\,S_{0,\,1} + S_{2,\,1}\,n^2 - S_{2,\,0}\,S_{0,\,1}\,n}{n\,(n-1)\,(n-2)} \tag{4.1.19}$$

test previous result

> $polykays([2, 1])$

$$\frac{-2\,S_{1,\,0}\,S_{1,\,1}\,n + 2\,S_{1,\,0}^2\,S_{0,\,1} + S_{2,\,1}\,n^2 - S_{2,\,0}\,S_{0,\,1}\,n}{n\,(n-1)\,(n-2)} \tag{4.1.20}$$

> $evalb(\%=k[2, 1])$

$$true \tag{4.1.21}$$

multivariate polykays $k_{1,1;1,0}$

> $A := makeTab([a, b], 'polykays(\,)')$

$$A := [[[a\,b], 1], [[b, a], -1]] \tag{4.1.22}$$

> $B := makeTab([a], 'polykays(\,)')$

$$B := [[[a], 1]] \tag{4.1.23}$$

> $vTab := unionVects(A, B)$

$$vTab := [[[a, a\,b], 1], [[a, a, b], -1]] \tag{4.1.24}$$

> $kstat := \frac{1\,augToPs([a, a\,b])}{n\,(n-1)} - \frac{1\,augToPs([a, a, b])}{n\,(n-1)\,(n-2)}$

$$kstat := \frac{-S_{2,\,1} + S_{1,\,0}\,S_{1,\,1}}{n\,(n-1)} - \frac{-2\,S_{1,\,0}\,S_{1,\,1} + S_{1,\,0}^2\,S_{0,\,1} + 2\,S_{2,\,1} - S_{2,\,0}\,S_{0,\,1}}{n\,(n-1)\,(n-2)} \tag{4.1.25}$$

> $kstat := simplify(kstat)$

$$kstat := \frac{-S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1}}{n\,(n-1)\,(n-2)} \tag{4.1.26}$$

> $nNum := numer(kstat);\quad nDenom := denom(kstat)$

$$nNum := -S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1}$$

$$nDenom := n\,(n-1)\,(n-2) \tag{4.1.27}$$

> $nNum := collect(nNum, indets(nNum) \setminus \{n\}, distributed)$

$$nNum := -S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1} \tag{4.1.28}$$

> $k[\,`1,1;\ 1\,`] := \frac{nNum}{nDenom}$

$$k_{1,1;\,1} := \frac{-S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1}}{n\,(n-1)\,(n-2)} \tag{4.1.29}$$

test previous result

> $polykays([1, 1], [1])$

$$\frac{-S_{2,1}\,n + S_{1,0}\,S_{1,1}\,n - S_{1,0}^2\,S_{0,1} + S_{2,0}\,S_{0,1}}{n\,(n-1)\,(n-2)} \tag{4.1.30}$$

> $evalb(\% = k[\,`1,1;\ 1\,`])$

$$true \tag{4.1.31}$$

Replacing symbols with numerical data

Sums of the rth powers of the data points:

> $powS :=$ **proc**( )
>
> **if** $nargs = 1$ **then**
>
> $Sum\left('X_i^{args}{}_1,\ i = 1\,..'n'\right)$
>
> **else**
>
> $Sum\left(mul\left('X_{i,j}^{args}{}_j, j = 1\,..nargs\right), i = 1\,..'n'\right)$
>
> **end if**
>
> **end proc**:

Example

> $powS(5, 3, 1)$

$$\sum_{i=1}^{n} X_{i,1}^5\,X_{i,2}^3\,X_{i,3} \tag{4.2.1}$$

> $powS(9);$

$$\sum_{i=1}^{n} X_i^9 \tag{4.2.2}$$

This function allows us to process a k-statistic or polykay replacing the simbols with numerical data. The parameter is the following:

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

```
> npolyk := proc(V, data)
      local res, ind, N, vE;
      if nops(V) = 1 then
          if nops(op(V)) = 1 then
              N := nops(op(data))
          else
              N := nops(data) end if:
      elif add(`if`(nops(x) = 1, 0, 1), x = V) = 0 then
          N := nops(op(data))
      else
          N := nops(data)
      end if;
      res := polykays(op(V));
      ind := `minus`(indets(res), {n});
      vE := seq(y[1] = powS(op(y[2])), y = seq([x, [op(x)]], x = ind));
      eval(res, [eval(evalf(eval(vE, [n = N]))), [X = op(data)]), n = N])
  end proc:
```

Examples: k-statistics and polykays

```
> 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]]
```

$$data := [[16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08, 19.43, 8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83, 16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11]] \quad \textbf{(4.2.3)}$$

The estimator for the mean is given by

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

$$14.02166667 \quad \textbf{(4.2.4)}$$

The estimator for the variance is given by

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

$$12.65006954 \quad \textbf{(4.2.5)}$$

The estimator for the skewness is given by $k_3 / \sqrt{k_2^3}$

```
> npolyk([[3]], data) / sqrt(npolyk([[2]], data)^3)
```

$$-0.03216238591 \quad \textbf{(4.2.6)}$$

The estimator for the kurtosis is given by k_4 / k_2^2

```
> npolyk([[4]], data)/npolyk([[2]], data)^2
```

$$-0.8852921296 \quad \textbf{(4.2.7)}$$

The estimator for the $\kappa_3\ \kappa_2$ is given by

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

$$-15.56133309 \quad \textbf{(4.2.8)}$$

Examples: multivariate k-ktatistics and multivariate polykays

```
> 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]]
```

$$data := [[5.31, 11.16], [3.26, 3.26], [2.35, 2.35], [8.32, 14.34], [13.48, 49.45], [6.25, 15.05], [7.01, 7.01], [8.52, 8.52], [0.45, 0.45], [12.08, 12.08], [19.39, 10.42]] \quad \textbf{(4.2.9)}$$

The estimator for the $\kappa_{2,\,1}$ is given by

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

$$-23.73790606 \quad \textbf{(4.2.10)}$$

The estimator for the $\kappa_{2,\,1}\ \kappa_{1,\,0}$is given by

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

$$294.2657622 \quad \textbf{(4.2.11)}$$

The estimator for the $\kappa_{2,\,1}\ \kappa_{2,\,0}\kappa_{1,\,0}$is given by

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

$$12369.47450 \quad \textbf{(4.2.12)}$$

Conclusions

Umbral formulae for k-statistics and polykays, either in single or multivariate cases, share a common algorithm to construct multiset subdivisions. When the multiset has the form {a^(i)}, an efficient way is to resort integer partitions.
In general we may construct multiset subdivisions by using suitable set partitions, but this procedure result non efficient from computational point of view [4].
Indeed, subdivision may occur more than one time in the same formula so that it is necessary to built a procedure generating only different subdivision with their multiplicity i.e. the number of corresponding set partitions.
To accomplish this task, the algorithm makeTab takes into account the connection between multisets and integer partitions, reducing the overall computational complexity. To accomplish this task, the algorithm makeTab takes into account the connection between multisets and integer partitions, reducing the overall computational complexity. It is possible to build very fast algorithms for k-statistics, polikays and their generalizzations by forfeiting the elegant idea of producing only one algorithm for the whole subject see [5],[6].

References

[1] Di Nardo E., G. Guarino, D. Senato (2008) A Maple algorithm for polykays and their generalizations. Adv. Appl. Stat. Vol. 8, No. 1, 19 - 36, http://www.pphmj.com/journals/adas.htm.

[2] Di Nardo E., G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. Vol. 14(2), 440-468. Official Journal of the Bernoulli Society for Mathematical Statistics and Probability, http://isi.cbs.nl/bernoulli/, (download from http://www.unibas.it/utenti/dinardo/lavori.html)

[3] Di Nardo E., G. Guarino, D. Senato*, Multiset Subdivision,* source Maple algorithm located in www.maplesoft.com *(submitted)*

[4] Di Nardo E., G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis Vol. 52, no. 11, 4909-4922, (download from http://arxiv.org/PS_cache/arxiv/pdf/0806/0806.0129v1.pdf or http://www.unibas.it/utenti/dinardo/lavori.html)

[5] Di Nardo E., G. Guarino, D. Senato (2009), *A new method for fast computing unbiased estimators of cumulants.* Statistics and Computing Vol. 19, 155-165. (download from http://www.unibas.it/utenti/dinardo/lavori.html)

[6] Di Nardo E., G. Guarino, D. Senato*, Fast Maple algorithms for k-statistics, polykays and their multivariate generalization ,* source Maple algorithm located in www.maplesoft.com *(submitted)*

>