コード生成

CodeGeneration パッケージは、Maple コードを R および JavaScript に変換するための新たなサポートを提供します。さらに、コード生成の出力オプションが改良され、埋め込みテキストフィールド内で output=embed オプションを設定することで、書式付き出力を表示できるようになりました。

R

CodeGeneration[R] を使用することで、式をコードフラグメントに変換できます。

```
> with(CodeGeneration):
```

```
> R(sqrt(a^2+b^2+c^2))
```

```
cg <- sqrt(a ^ 2 + b ^ 2 + c ^ 2)
```

プロシージャや、より大きなプログラムを変換することもできます。

```
> R(m->add(i, i=1..m))
```

```
cg0 <- function(m)
{
  r <- 0
  for(i in 1 : m)
    r <- r + i
  return(r)
}
```

CodeGeneration[R] はまた、線形代数や特殊関数の多数のルーチンを含め数多くの Maple データ構造および関数を、R で使用されているものに変換します。

```
> R(Matrix([[1, 2], [3, 4]]))
```

```
cg1 <- matrix(c(1,3,2,4),nrow=2,ncol=2)
```

```
> R(piecewise(v < 1, -v^3+3*v+1, v < 2, 2*v^3-9*v^2+12*v-2, -v^3+9*v^2-24*v+22))
```

```
cg2 <- ifelse(v < 1, -v ^ 3 + 3 * v + 1, ifelse(v < 2, 2 * v ^ 3 - 9 * v ^ 2 + 12 * v - 2, -v ^ 3 + 9 * v ^ 2 - 24 * v + 22))
```

```
> R((M, n)->M - x*LinearAlgebra:-IdentityMatrix(n))
```

```
cg3 <- function(M,n) M - x * diag(n)
```

```
> R(_C1*BesselJ(nu, x)+_C2*BesselY(nu, x))
```

```
cg4 <- _C1 * besselJ(x, nu) + _C2 * besselY(x, nu)
```

Maple の R コード生成は、Statistics パッケージの主要なコマンドも変換します。

```
> R('Statistics:-Mean([5, 2, 1, 4, 3])')
```

```
cg5 <- mean(c(5,2,1,4,3))
```

```
> R('Statistics:-Median([5, 2, 1, 4, 3])')
```

```
cg6 <- median(c(5,2,1,4,3))
```

```
> R('Statistics:-StandardDeviation([5, 2, 1, 4, 3])')
```

```
cg7 <- sd(c(5,2,1,4,3))
```

```
> R('Statistics:-FivePointSummary([5, 2, 1, 4, 3])')
cg8 <- fivenum(c(5,2,1,4,3))
> R( 'Statistics:-Scale( [5, 2, 1, 4, 3], center=2, scale=1 )' );
cg9 <- scale(c(5,2,1,4,3), center = 2, scale = 1)
```

Statistics パッケージのプロットルーチンに対するサポートには制限があります。Histogram および BoxPlot などの可視化がサポートされています。

```
> R('Statistics:-Histogram([1, 3, 5, 7, 8, 4, 4], color="Red", title="Histogram", frequencyscale=absolute,
      axes=none)')
cg10 <- hist(c(1,3,5,7,8,4,4), axes = FALSE, col = "Red", freq = TRUE, main =
"Histogram")
> R('Statistics:-BoxPlot([4, 8, 15, 16, 23, 42], color="Orange", title="BoxPlot", notched=true, orientation
      =horizontal)')
cg11 <- boxplot(c(4,8,15,16,23,42), col = "Orange", notch = TRUE, horizontal = TRUE,
main = "BoxPlot")
```

Maple は、可能な場合には、多数の分布を扱うコマンドに相当するコマンドを返します。この例では、評価された確率密度関数が変換されています。

```
> R(Statistics:-PDF(LogNormal(0, 1), x))
cg12 <- ifelse(x < 0, 0, 0.1e1 / x * sqrt(0.2e1) * pi ^ (-0.1e1 / 0.2e1) * exp(-log(x)
^ 2 / 0.2e1) / 0.2e1)
```

ただし、Maple と R が名前付き確率分布を扱う方法には相違があります。Maple では、分布は Statistics コマンド内の分布名により直接参照されます。たとえば、Statistics:-CDF(Normal(..), ..) などがそうです。R では、分布はコマンド pnorm(..) そのものに含まれています。CodeGeneration[R] が名前付き分布への参照を含む式を変換しようとする場合は、常に警告メッセージが返されます。可能であれば、Maple は含まれる式を R 内の相当するコマンドに変換します。

たとえば、Maple 内の点 x = 1 において平均 = 0 および標準偏差 = 1 のパラメータを持つ LogNormal 分布の確率密度関数の値を求めるには、分布に基づくランダム変数からではなく LogNormal 分布から直接 PDF 値を取れる PDF(LogNormal(0,1), 1) を使用します。これは R における同様のコマンドに相当するものであり、これにより相当する R の変換が返されます。

```
> R('Statistics:-PDF(LogNormal(0, 1), 1)')
Warning, the function names {LogNormal} are not recognized in the target language
cg13 <- dlnorm(1, meanlog = 0, sdlog = 1)
```

場合によっては、PDF、ProbabilityFunction および CDF などのコマンドを変換することが可能です。

```
> R('Statistics:-ProbabilityFunction(Poisson(2), 1)')
Warning, the function names {Poisson} are not recognized in the target language
cg14 <- dpois(1,2)
> R('Statistics:-CDF(ChiSquare(1), 1)')
Warning, the function names {ChiSquare} are not recognized in the target language
cg15 <- pchisq(1,1)
```

さらに、samples および quantiles も返します。

```
> R('Statistics:-Sample(Uniform(1, 10), 10)');
Warning, the function names {Uniform} are not recognized in the target language
cg16 <- runif(0.10e2, min = 1, max = 10)
> R('Statistics:-Quantile(Weibull(3, 5), 0.5)');
Warning, the function names {Weibull} are not recognized in the target language
```

```
cg17 <- qweibull(0.5e0,5, scale = 3)
```

時系列オブジェクトも作成可能です。

```
> R('TimeSeriesAnalysis:-TimeSeries([10, 25, 30, 55])')
cg18 <- ts(c(10,25,30,55))
> R('TimeSeriesAnalysis:-TimeSeriesPlot( TimeSeriesAnalysis:-TimeSeries([10, 25, 30, 55]) )')
cg19 <- plot.ts(ts(c(10,25,30,55)))
```

コード生成に関して、制限付きでサポートされる Statistics コマンド のリストを確認するには、RDetails ページを参照してください。

JavaScript

CodeGeneration[JavaScript] により、式をコードフラグメントに変換できます。

```
> with(CodeGeneration):
```

$$\text{> } JavaScript\left(\sqrt{a^2+b^2+c^2}\right)$$

```
cg0 = Math.sqrt(a * a + b * b + c * c);
```

プロシージャや、より大きなプログラムを変換することもできます。

$$\text{> } JavaScript(m \rightarrow add(i, i=1..m))$$

```
function cg1(m) {
  var r;
  r = 0;
  for (i in 1...m)
   {
    r = r + i;
   }
  return(r);
}
```

その他の更新

新しいオプション output=embed により、CodeGeneration 出力を Maple テキスト領域内で生成できるようになりました。

$$CodeGeneration:\text{-}Python\left((x, y) \rightarrow \sin(x)^2 + \cos(x)^2, output = embed\right)$$

```
import math

cg = lambda x,y: math.sin(x) ** 2 + math.cos(x) ** 2
```

Pages That Link to This Page