SymbolizeR Full Verification

1. Basic Expectation, Variance, and Covariance

Test basic symbolic manipulation without defined distributions.

```
# Linearity of Expectation
E(a * X + b)
## a * E(X) + b
E(X + Y + Z)
## E(X) + E(Y) + E(Z)

# Variance Expansion: Var(X) = E(X^2) - E(X)^2
Var(X)
## E(X^2) - E(X)^2

# Variance Properties
Var(a * X + b)
## E((a * X)^2) + 2 * (a * E(X) * b) + b^2 - (a * E(X) + b)^2

# Covariance Expansion
Cov(X, Y)
## E(X * Y) - E(X) * E(Y)
```

2. Like-Term Simplification (Enhancement)

Verify that like terms are combined, even across operations.

```
# Addition
E(X + X)           # Expect: 2 * E(X)
## 2 * E(X)
E(2*X + 3*X)       # Expect: 5 * E(X)
## (2 + 3) * E(X)

# Subtraction
E(3*X - X)         # Expect: 2 * E(X)
## (3 - 1) * E(X)

# Cancellation
E(X - X)           # Expect: 0
## [1] 0
```

3. Distribution Support

Verify moments for all supported distributions.

Continuous Distributions

```r
clear.definitions()

# Normal
define(X ~ Normal(mu, sigma))
E(X)               # Expect: mu
## mu
Var(X)             # Expect: sigma^2 (Verifies fix for cancellation)
## sigma^2

# Uniform
define(U ~ Uniform(a, b))
E(U)               # Expect: (a+b)/2
## (a + b)/2

# Exponential
define(E ~ Exponential(lambda))
E(E)               # Expect: 1/lambda
## 1/lambda
Var(E)             # Expect: 1/lambda^2
## 2/lambda^2 - (1/lambda)^2

# Gamma, Beta, etc.
define(G ~ Gamma(alpha, beta))
E(G)               # Expect: alpha/beta
## alpha/beta
```

Discrete Distributions

```r
clear.definitions()

# Poisson
define(P ~ Poisson(lambda))
E(P)               # Expect: lambda
## lambda
Var(P)             # Expect: lambda
## lambda

# Binomial
define(B ~ Binomial(n, p))
E(B)               # Expect: n*p
## n * p
```

New Distributions (High Priority Check)

```r
clear.definitions()

# Chi-squared
```

```r
define(C ~ ChiSq(k))
E(C)                # Expect: k
## k
E(C^2)              # Expect: k^2 + 2*k
## k^2 + 2 * k
Var(C)              # Expect: 2*k
## 2 * k

# Student's t
define(T ~ StudentT(nu))
E(T)                # Expect: 0
## [1] 0
Var(T)              # Expect: nu / (nu - 2)
## nu/(nu - 2)
```

4. Generic Moments (High Priority Check)

Verify `moment(X, n)` function.

```r
clear.definitions()
define(X ~ Normal(mu, sigma))

moment(X, 1)        # Expect: mu
## mu
moment(X, 2)        # Expect: sigma^2 + mu^2
## sigma^2 + mu^2
moment(X, 3)        # Expect: mu^3 + 3*mu*sigma^2
## mu^3 + 3 * (mu * sigma^2)
moment(X, 4)        # Expect: mu^4 + 6*mu^2*sigma^2 + 3*sigma^4
## mu^4 + 6 * (mu^2 * sigma^2) + 3 * sigma^4
```

5. Higher-Order Statistics (Medium Priority Check)

Verify Skewness and Kurtosis functions.

```r
clear.definitions()
define(X ~ Normal(mu, sigma))

# Skewness of Normal should be 0
Skewness(X)
## [1] 0

# Excess Kurtosis of Normal should be 0
Kurtosis(X)
## [1] 0

# Raw Kurtosis
Kurtosis(X, excess = FALSE)
## [1] 3
```

6. Complex Simplification Test (The Reported Bug)

Specifically checking the `Var(X)` simplification for Normal distribution, which involves cancelling `mu^2 - mu^2`.

```
clear.definitions()
define(X ~ Normal(mu, sigma))

# Detailed breakdown
EX <- E(X)        # mu
EX2 <- E(X^2)     # sigma^2 + mu^2
# Manually construct E[X^2] - E[X]^2 symbolically
V <- call("-", EX2, call("^", EX, 2))

# The following should evaluate strictly to sigma^2
Var(X)
## sigma^2
```

7. Derivations

Verify step-by-step derivation output.

```
# Expectation
derive.E(a * X + b)
## Derivation of: E(a * X + b)
## ==================================================
##
## Step 1: Starting expression
##   Rule: Input
##   => E(a * X + b)
##
## Step 2: Apply expectation rules
##   Rule: Linearity: E[aX + b] = aE[X] + b, E[X + Y] = E[X] + E[Y]
##   => a * mu + b
##
## Step 3: Substitute known distributions
##   Rule: If X ~ Distribution, replace E[X] with known moment
##   => a * mu + b
##
## Step 4: Final result
##   Rule: Done
##   => a * mu + b
##
## ==================================================
## Final Answer: a * mu + b

# Variance
derive.Var(X)
## Derivation of: Var(X)
## ==================================================
##
## Step 1: Starting expression
##   Rule: Input
##   => Var(X)
##
## Step 2: Apply variance definition
##   Rule: Var[X] = E[X²] - E[X]²
##   => E[X^2] - E[X]²
##
## Step 3: Compute E[X²]
##   Rule: Apply expectation rules to X²
##   => sigma^2 + mu^2 - E[X]²
##
## Step 4: Compute E[X]²
##   Rule: Apply expectation rules to X, then square
##   => sigma^2 + mu^2 - mu^2
##
## Step 5: Combine terms
##   Rule: E[X²] - E[X]²
##   => sigma^2 + mu^2 - mu^2
##
## Step 6: Simplify
##   Rule: Algebraic simplification
##   => sigma^2
##
## Step 7: Final result
##   Rule: Done
##   => sigma^2
##
## ==================================================
## Final Answer: sigma^2
```

8. LaTeX Output

Verify `to.latex()` conversion.

```r
# Basic expression
cat("$$", to.latex(E(a * X + b)), "$$\n")
```

$$a\mu + b$$

```r
# Variance
cat("$$", to.latex(Var(a * X + b)), "$$\n")
```

$$\mathbb{E}[(aX)^2] + 2a\mu b + b^2 - (a\mu + b)^2$$

```r
# Greek letters
cat("$$", to.latex(E(alpha * X + beta)), "$$\n")
```

$$\alpha\mu + \beta$$

```r
# Fractions
cat("$$", to.latex(E(X / Y)), "$$\n")
```

$$\mathbb{E}[\frac{X}{Y}]$$

```r
# Integrals/Moments
# Note: to.latex might operate on the result of a simplification or derivation
res <- derive.E(X)
cat(to.latex(res), "\n")
```

$$\begin{aligned}
\mathbb{E}[X] &= \mathbb{E}[X] && \text{(Starting expression)} \\
&= \mu && \text{(Apply expectation rules)} \\
&= \mu && \text{(Substitute known distributions)} \\
&= \mu && \text{(Final result)}
\end{aligned}$$