

Remove legend.

```
main="simple plot\n series 1 (red) : series 2 (green)\n"
```

```
rename pointer in html to just movement_plots.png.
```

```
plot(.....frame = TRUE, axes=FALSE ...)
```

```
#override default x-labels with user defined x-axis labels:
```

```
axis(1, axTicks(1), labels=c("one", "two", "three", "four"), las=2) #2 => labels vertical
```

```
#leave y-axis untouched:
```

```
axis(2)
```

```
-----  
#!/usr/bin/perl
```

```
use Statistics::Distributions;
```

```
# normal distribution with sigmal = 1, mu = 0. This subroutine  
# returns Pr(x > $value)
```

```
$value = 1;  
$Pnormal = Statistics::Distributions::uprob($value);
```

```
print "\nUpper Prob. for normal value $value = $Pnormal\n";
```

```
# All tests supported:
```

```
$chis=Statistics::Distributions::chisqrddistr (2,.05);  
print "\nChi-squared-crit (2 degrees of freedom, 95th percentile "  
."= 0.05 level) = $chis\n";
```

```
$u=Statistics::Distributions::udistr (.05);  
print "\nu-crit (95th percentile = 0.05 level) = $u\n";
```

```
$t=Statistics::Distributions::tdistr (1,.005);  
print "\nt-crit (1 degree of freedom, 99.5th percentile = 0.005 level) "  
."= $t\n";
```

```
$f=Statistics::Distributions::fdistr (1,3,.01);  
print "\nF-crit (1 degree of freedom in numerator, 3 degrees of freedom "  
."in denominator, 99th percentile = 0.01 level) = $f\n";
```

```
$uprob=Statistics::Distributions::uprob (-0.85);  
print "\nupper probability of the u distribution (u = -0.85): Q(u) "  
."= 1-G(u) = $uprob\n";
```

```
$chisprob=Statistics::Distributions::chisqrprob (3,6.25);  
print "\nupper probability of the chi-square distribution (3 degrees "  
."of freedom, chi-squared = 6.25): Q = 1-G = $chisprob\n";
```

```
$tprob=Statistics::Distributions::tprob (3,6.251);  
print "\nupper probability of the t distribution (3 degrees of "  
."freedom, t = 6.251): Q = 1-G = $tprob\n";
```

```
$fprob=Statistics::Distributions::fprob (3,5,.625);  
print "\nupper probability of the F distribution (3 degrees of freedom "  
."in numerator, 5 degrees of freedom in denominator, F = 6.25): "  
."Q = 1-G = $fprob\n\n";
```

```
exit(0);
```

```
-----  
package Statistics::Distributions;
```

```
use strict;
```

```
use vars qw($VERSION @ISA @EXPORT @EXPORT_OK);
```

```
use constant PI => 3.1415926536;
```

```
use constant SIGNIFICANT => 5; # number of significant digits to be returned
```

```
require Exporter;
```

```
@ISA = qw(Exporter AutoLoader);
```

```
# Items to export into callers namespace by default. Note: do not export  
# names by default without a very good reason. Use EXPORT_OK instead.
```

```
# Do not simply export all your public functions/methods/constants.
```

```
@EXPORT_OK = qw(chisqrddistr tdistr fdistr udistr uprob chisqrprob tprob fprob);
```

```
$VERSION = '1.02';
```

```
# Preloaded methods go here.
```

```
sub chisqrddistr { # Percentage points X^2(x^2,n)
```

```

my ($n, $p) = @_;
if ($n <= 0 || abs($n) - abs(int($n)) != 0) {
    die "Invalid n: $n\n"; # degree of freedom
}
if ($p <= 0 || $p > 1) {
    die "Invalid p: $p\n";
}
return precision_string(_subchisqr($n, $p));
}

sub udistr { # Percentage points N(0,1^2)
my ($p) = (@_);
if ($p > 1 || $p <= 0) {
    die "Invalid p: $p\n";
}
return precision_string(_subu($p));
}

sub tdistr { # Percentage points t(x,n)
my ($n, $p) = @_;
if ($n <= 0 || abs($n) - abs(int($n)) != 0) {
    die "Invalid n: $n\n";
}
if ($p <= 0 || $p >= 1) {
    die "Invalid p: $p\n";
}
return precision_string(_subt($n, $p));
}

sub fdistr { # Percentage points F(x,n1,n2)
my ($n, $m, $p) = @_;
if (($n<=0) || ((abs($n)-(abs(int($n))))!=0)) {
    die "Invalid n: $n\n"; # first degree of freedom
}
if (($m<=0) || ((abs($m)-(abs(int($m))))!=0)) {
    die "Invalid m: $m\n"; # second degree of freedom
}
if (($p<=0) || ($p>1)) {
    die "Invalid p: $p\n";
}
return precision_string(_subf($n, $m, $p));
}

sub uprob { # Upper probability N(0,1^2)
my ($x) = @_;
return precision_string(_subuprob($x));
}

sub chisqrprob { # Upper probability X^2(x^2,n)
my ($n,$x) = @_;
if (($n <= 0) || ((abs($n) - (abs(int($n)))) != 0)) {
    die "Invalid n: $n\n"; # degree of freedom
}
return precision_string(_subchisqrprob($n, $x));
}

sub tprob { # Upper probability t(x,n)
my ($n, $x) = @_;
if (($n <= 0) || ((abs($n) - abs(int($n))) != 0)) {
    die "Invalid n: $n\n"; # degree of freedom
}
return precision_string(_subtprob($n, $x));
}

sub fprob { # Upper probability F(x,n1,n2)
my ($n, $m, $x) = @_;
if (($n<=0) || ((abs($n)-(abs(int($n))))!=0)) {
    die "Invalid n: $n\n"; # first degree of freedom
}
if (($m<=0) || ((abs($m)-(abs(int($m))))!=0)) {
    die "Invalid m: $m\n"; # second degree of freedom
}
return precision_string(_subfprob($n, $m, $x));
}

sub _subfprob {
my ($n, $m, $x) = @_;
my $p;

if ($x<=0) {
    $p=1;
} elsif ($m % 2 == 0) {
    my $z = $m / ($m + $n * $x);
    my $a = 1;
    for (my $i = $m - 2; $i >= 2; $i -= 2) {
        $a = 1 + ($n + $i - 2) / $i * $z * $a;
    }
}

```

```

    }
    $p = 1 - ((1 - $z) ** ($n / 2) * $a);
} elseif ($n % 2 == 0) {
    my $z = $n * $x / ($m + $n * $x);
    my $a = 1;
    for (my $i = $n - 2; $i >= 2; $i -= 2) {
        $a = 1 + ($m + $i - 2) / $i * $z * $a;
    }
    $p = (1 - $z) ** ($m / 2) * $a;
} else {
    my $y = atan2(sqrt($n * $x / $m), 1);
    my $z = sin($y) ** 2;
    my $a = ($n == 1) ? 0 : 1;
    for (my $i = $n - 2; $i >= 3; $i -= 2) {
        $a = 1 + ($m + $i - 2) / $i * $z * $a;
    }
    my $b = PI;
    for (my $i = 2; $i <= $m - 1; $i += 2) {
        $b *= ($i - 1) / $i;
    }
    my $p1 = 2 / $b * sin($y) * cos($y) ** $m * $a;

    $z = cos($y) ** 2;
    $a = ($m == 1) ? 0 : 1;
    for (my $i = $m - 2; $i >= 3; $i -= 2) {
        $a = 1 + ($i - 1) / $i * $z * $a;
    }
    $p = max(0, $p1 + 1 - 2 * $y / PI
        - 2 / PI * sin($y) * cos($y) * $a);
}
return $p;
}

sub _subchisqrprob {
    my ($n, $x) = @_;
    my $p;

    if ($x <= 0) {
        $p = 1;
    } elseif ($n > 100) {
        $p = _subuprob(((($x / $n) ** (1/3)
            - (1 - 2/9/$n)) / sqrt(2/9/$n)));
    } elseif ($x > 400) {
        $p = 0;
    } else {
        my ($a, $i, $il);
        if (($n % 2) != 0) {
            $p = 2 * _subuprob(sqrt($x));
            $a = sqrt(2/PI) * exp(-$x/2) / sqrt($x);
            $il = 1;
        } else {
            $p = $a = exp(-$x/2);
            $il = 2;
        }

        for ($i = $il; $i <= ($n-2); $i += 2) {
            $a *= $x / $i;
            $p += $a;
        }
    }
    return $p;
}

sub _subu {
    my ($p) = @_;
    my $y = -log(4 * $p * (1 - $p));
    my $x = sqrt(
        $y * (1.570796288
            + $y * (.03706987906
                + $y * (-.8364353589E-3
                    + $y * (-.2250947176E-3
                        + $y * (.6841218299E-5
                            + $y * (0.5824238515E-5
                                + $y * (-.104527497E-5
                                    + $y * (.8360937017E-7
                                        + $y * (-.3231081277E-8
                                            + $y * (.3657763036E-10
                                                + $y * .6936233982E-12))))))))));
    $x = -$x if ($p > .5);
    return $x;
}

sub _subuprob {
    my ($x) = @_;
    my $p = 0; # if ($absx > 100)
    my $absx = abs($x);

```

```

if ($absx < 1.9) {
    $p = (1 +
        $absx * (.049867347
        + $absx * (.0211410061
        + $absx * (.0032776263
        + $absx * (.0000380036
        + $absx * (.0000488906
        + $absx * .000005383)))))) ** -16/2;
} elseif ($absx <= 100) {
    for (my $i = 18; $i >= 1; $i--) {
        $p = $i / ($absx + $p);
    }
    $p = exp(-.5 * $absx * $absx)
        / sqrt(2 * PI) / ($absx + $p);
}

$p = 1 - $p if ($x < 0);
return $p;
}

sub _subt {
    my ($n, $p) = @_ ;

    if ($p >= 1 || $p <= 0) {
        die "Invalid p: $p\n";
    }

    if ($p == 0.5) {
        return 0;
    } elsif ($p < 0.5) {
        return -_subt($n, 1 - $p);
    }

    my $u = _subu($p);
    my $u2 = $u ** 2;

    my $a = ($u2 + 1) / 4;
    my $b = ((5 * $u2 + 16) * $u2 + 3) / 96;
    my $c = (((3 * $u2 + 19) * $u2 + 17) * $u2 - 15) / 384;
    my $d = (((79 * $u2 + 776) * $u2 + 1482) * $u2 - 1920) * $u2 - 945)
        / 92160;
    my $e = (((27 * $u2 + 339) * $u2 + 930) * $u2 - 1782) * $u2 - 765) * $u2
        + 17955) / 368640;

    my $x = $u * (1 + ($a + ($b + ($c + ($d + $e / $n) / $n) / $n) / $n) / $n);

    if ($n <= log10($p) ** 2 + 3) {
        my $round;
        do {
            my $p1 = _subtprob($n, $x);
            my $n1 = $n + 1;
            my $delta = ($p1 - $p)
                / exp(($n1 * log($n1 / ($n + $x * $x))
                + log($n/$n1/2/PI) - 1
                + (1/$n1 - 1/$n) / 6) / 2);
            $x += $delta;
            $round = sprintf("%.5s".abs(int(log10(abs $x)-4))."f", $delta);
        } while (($x) && ($round != 0));
    }
    return $x;
}

sub _subtprob {
    my ($n, $x) = @_ ;

    my ($a, $b);
    my $w = atan2($x / sqrt($n), 1);
    my $z = cos($w) ** 2;
    my $y = 1;

    for (my $i = $n-2; $i >= 2; $i -= 2) {
        $y = 1 + ($i-1) / $i * $z * $y;
    }

    if ($n % 2 == 0) {
        $a = sin($w)/2;
        $b = .5;
    } else {
        $a = ($n == 1) ? 0 : sin($w)*cos($w)/PI;
        $b = .5 + $w/PI;
    }
    return max(0, 1 - $b - $a * $y);
}

sub _subf {

```

```

my ($n, $m, $p) = @_;
my $x;

if ($p >= 1 || $p <= 0) {
    die "Invalid p: $p\n";
}

if ($p == 1) {
    $x = 0;
} elsif ($m == 1) {
    $x = 1 / (_subt($n, 0.5 - $p / 2) ** 2);
} elsif ($n == 1) {
    $x = _subt($m, $p/2) ** 2;
} elsif ($m == 2) {
    my $u = _subchisqr($m, 1 - $p);
    my $a = $m - 2;
    $x = 1 / (($u / $m * (1 +
        ((($u - $a) / 2 +
            (((4 * $u - 11 * $a) * $u + $a * (7 * $m - 10)) / 24 +
            ((2 * $u - 10 * $a) * $u + $a * (17 * $m - 26))
            - $a * $a * (9 * $m - 6))
            )/48/$n
        )/$n
    )/$n));
} elsif ($n > $m) {
    $x = 1 / _subf2($m, $n, 1 - $p)
} else {
    $x = _subf2($n, $m, $p)
}
return $x;
}

sub _subf2 {
    my ($n, $m, $p) = @_;
    my $u = _subchisqr($n, $p);
    my $n2 = $n - 2;
    my $x = $u / $n *
        (1 +
            (($u - $n2) / 2 +
                (((4 * $u - 11 * $n2) * $u + $n2 * (7 * $n - 10)) / 24 +
                ((2 * $u - 10 * $n2) * $u + $n2 * (17 * $n -
26)) * $u
                - $n2 * $n2 * (9 * $n - 6)) / 48 / $m) /
            $m) / $m);
    my $delta;
    do {
        my $z = exp(
            (($n+$m) * log(($n+$m) / ($n * $x + $m))
            + ($n - 2) * log($x)
            + log($n * $m / ($n+$m))
            - log(4 * PI)
            - (1/$n + 1/$m - 1/($n+$m))/6
            )/2);
        $delta = (_subfprob($n, $m, $x) - $p) / $z;
        $x += $delta;
    } while (abs($delta)>3e-4);
    return $x;
}

sub _subchisqr {
    my ($n, $p) = @_;
    my $x;

    if (($p > 1) || ($p <= 0)) {
        die "Invalid p: $p\n";
    } elsif ($p == 1) {
        $x = 0;
    } elsif ($n == 1) {
        $x = _subu($p / 2) ** 2;
    } elsif ($n == 2) {
        $x = -2 * log($p);
    } else {
        my $u = _subu($p);
        my $u2 = $u * $u;

        $x = max(0, $n + sqrt(2 * $n) * $u
            + 2/3 * ($u2 - 1)
            + $u * ($u2 - 7) / 9 / sqrt(2 * $n)
            - 2/405 / $n * ($u2 * (3 * $u2 + 7) - 16));

        if ($n <= 100) {
            my ($x0, $p1, $z);
            do {
                $x0 = $x;
                if ($x < 0) {
                    $p1 = 1;
                }
            }
        }
    }
}

```

```

        } elsif ($n>100) {
            $p1 = _subuprob((( $x / $n)**(1/3) - (1 -
2/9/$n))
                / sqrt(2/9/$n));
        } elsif ($x>400) {
            $p1 = 0;
        } else {
            my ($i0, $a);
            if (($n % 2) != 0) {
                $p1 = 2 * _subuprob(sqrt($x));
                $a = sqrt(2/PI) * exp(-$x/2) / sqrt($x);
                $i0 = 1;
            } else {
                $p1 = $a = exp(-$x/2);
                $i0 = 2;
            }

            for (my $i = $i0; $i <= $n-2; $i += 2) {
                $a *= $x / $i;
                $p1 += $a;
            }

            $z = exp((($n-1) * log($x/$n) - log(4*PI*$x)
                + $n - $x - 1/$n/6) / 2);
            $x += ($p1 - $p) / $z;
            $x = sprintf("%.5f", $x);
        } while (($n < 31) && (abs($x0 - $x) > 1e-4));
    }
    return $x;
}

sub log10 {
    my $n = shift;
    return log($n) / log(10);
}

sub max {
    my $max = shift;
    my $next;
    while (@_) {
        $next = shift;
        $max = $next if ($next > $max);
    }
    return $max;
}

sub min {
    my $min = shift;
    my $next;
    while (@_) {
        $next = shift;
        $min = $next if ($next < $min);
    }
    return $min;
}

sub precision {
    my ($x) = @_;
    return abs int(log10(abs $x) - SIGNIFICANT);
}

sub precision_string {
    my ($x) = @_;
    if ($x) {
        return sprintf "%. " . precision($x) . "f", $x;
    } else {
        return "0";
    }
}

```

# Autoload methods go after =cut, and are processed by the autosplit program.

1;

END

# Below is the stub of documentation for your module. You better edit it!

=head1 NAME

Statistics::Distributions - Perl module for calculating critical values and upper probabilities of common statistical distributions

=head1 SYNOPSIS

```
use Statistics::Distributions;
```

```

$chis=Statistics::Distributions::chisqrddistr (2,.05);
print "Chi-squared-crit (2 degrees of freedom, 95th percentile "
    ". = 0.05 level) = $chis\n";

$u=Statistics::Distributions::udistr (.05);
print "u-crit (95th percentile = 0.05 level) = $u\n";

$t=Statistics::Distributions::tdistr (1,.005);
print "t-crit (1 degree of freedom, 99.5th percentile = 0.005 level) "
    ". = $t\n";

$f=Statistics::Distributions::fdistr (1,3,.01);
print "F-crit (1 degree of freedom in numerator, 3 degrees of freedom "
    ". in denominator, 99th percentile = 0.01 level) = $f\n";

$uprob=Statistics::Distributions::uprob (-0.85);
print "upper probability of the u distribution (u = -0.85): Q(u) "
    ". = 1-G(u) = $uprob\n";

$chisprob=Statistics::Distributions::chisqrprob (3,6.25);
print "upper probability of the chi-square distribution (3 degrees "
    ". of freedom, chi-squared = 6.25): Q = 1-G = $chisprob\n";

$tprob=Statistics::Distributions::tprob (3,6.251);
print "upper probability of the t distribution (3 degrees of "
    ". freedom, t = 6.251): Q = 1-G = $tprob\n";

$fprob=Statistics::Distributions::fprob (3,5,.625);
print "upper probability of the F distribution (3 degrees of freedom "
    ". in numerator, 5 degrees of freedom in denominator, F = 6.25): "
    ". Q = 1-G = $fprob\n";

=head1 DESCRIPTION

This Perl module calculates percentage points (5 significant digits) of the u
(standard normal) distribution, the student's t distribution, the chi-square
distribution and the F distribution. It can also calculate the upper probability (5
significant digits) of the u (standard normal), the chi-square, the t and the F
distribution.
These critical values are needed to perform statistical tests, like the u test, the t
test, the F test and the chi-squared test, and to calculate confidence intervals.

If you are interested in more precise algorithms you could look at:
StatLib: http://lib.stat.cmu.edu/apstat/ ;
Applied Statistics Algorithms by Griffiths, P. and Hill, I.D., Ellis Horwood:
Chichester (1985)

=head1 BUGS

This final version 1.02 has been released after more than one year without a bug
report on the previous version 0.07.
Nevertheless, if you find any bugs or oddities, please do inform the author.

=head1 INSTALLATION

See perlmodinstall for information and options on installing Perl modules.

=head1 AVAILABILITY

The latest version of this module is available from the Distribution Perl Archive
Network (CPAN). Please visit http://www.cpan.org/ to find a CPAN site near you or see
http://www.cpan.org/authors/id/M/MI/MIKEK/ .

=head1 AUTHOR

Michael Kospach <mike.perl@gmx.at>

Nice formatting, simplification and bug repair by Matthias Trautner Kromann
<mtk@id.cbs.dk>

=head1 COPYRIGHT

Copyright 2003 Michael Kospach. All rights reserved.

This library is free software; you can redistribute it and/or modify it under the same
terms as Perl itself.

=head1 SEE ALSO

Statistics::ChiSquare, Statistics::Table::t, Statistics::Table::F, perl(1).

=cut

```

