Software for Distributions in R

David Scott\textsuperscript{1}  Diethelm Würtz\textsuperscript{2}  Christine Dong\textsuperscript{1}

\textsuperscript{1}Department of Statistics  
The University of Auckland

\textsuperscript{2}Institut für Theoretische Physik  
ETH Zürich

June 30, 2009
1 Introduction

2 Current Software for Distributions
   - Base R
   - Contributed Packages

3 Implementation in Base R

4 Design for Distribution Implementation
1. Introduction

2. Current Software for Distributions
   - Base R
   - Contributed Packages

3. Implementation in Base R

4. Design for Distribution Implementation
Outline

1. Introduction

2. Current Software for Distributions
   - Base R
     - Contributed Packages

3. Implementation in Base R

4. Design for Distribution Implementation
1. Introduction

2. Current Software for Distributions
   - Base R
   - Contributed Packages

3. Implementation in Base R

4. Design for Distribution Implementation
Outline

1. Introduction

2. Current Software for Distributions
   - Base R
   - Contributed Packages

3. Implementation in Base R

4. Design for Distribution Implementation
1 Introduction

2 Current Software for Distributions
   - Base R
   - Contributed Packages

3 Implementation in Base R

4 Design for Distribution Implementation
1 Introduction

2 Current Software for Distributions

3 Implementation in Base R

4 Design for Distribution Implementation
Distributions

- Distributions are how we model uncertainty
- There is well-established theory concerning distributions
- There are standard approaches for fitting distributions
- There are many distributions which have been found to be of interest
- Software implementation of distributions is a well-defined subject in comparison to say modelling of time-series
Distributions are how we model uncertainty

There is well-established theory concerning distributions

There are standard approaches for fitting distributions

There are many distributions which have been found to be of interest

Software implementation of distributions is a well-defined subject in comparison to say modelling of time-series
Distributions are how we model uncertainty
There is well-established theory concerning distributions
There are standard approaches for fitting distributions
There are many distributions which have been found to be of interest
Software implementation of distributions is a well-defined subject in comparison to say modelling of time-series
Distributions

- Distributions are how we model uncertainty
- There is well-established theory concerning distributions
- There are standard approaches for fitting distributions
- There are many distributions which have been found to be of interest

- Software implementation of distributions is a well-defined subject in comparison to say modelling of time-series
• Distributions are how we model uncertainty
• There is well-established theory concerning distributions
• There are standard approaches for fitting distributions
• There are many distributions which have been found to be of interest
• Software implementation of distributions is a well-defined subject in comparison to say modelling of time-series
**Introduction**

- In base **R** there are 20 distributions implemented, at least in part
- All univariate—consider univariate distributions only
- Numerous other distributions have been implemented in **R**
  - CRAN packages solely devoted to one or more distributions
  - CRAN packages which implement distributions incidentally (e.g. VGAM)
  - Implementations of distributions not on CRAN, e.g. Jim Lindsey's work
Introduction

- In base **R** there are 20 distributions implemented, at least in part
- All univariate—consider univariate distributions only
- Numerous other distributions have been implemented in **R**
  - CRAN packages solely devoted to one or more distributions
  - CRAN packages which implement distributions incidentally (e.g. VGAM)
  - implementations of distributions not on CRAN, e.g. Jim Lindsey's work
In base R there are 20 distributions implemented, at least in part.

All univariate—consider univariate distributions only.

Numerous other distributions have been implemented in R:

- CRAN packages solely devoted to one or more distributions
- CRAN packages which implement distributions incidentally (e.g. VGAM)
- implementations of distributions not on CRAN, e.g. Jim Lindsey’s work
In base R there are 20 distributions implemented, at least in part. All univariate—consider univariate distributions only. Numerous other distributions have been implemented in R.

- CRAN packages solely devoted to one or more distributions
- CRAN packages which implement distributions incidentally (e.g. VGAM)
- Implementations of distributions not on CRAN, e.g. Jim Lindsey’s work
In base **R** there are 20 distributions implemented, at least in part

All univariate—consider univariate distributions only

Numerous other distributions have been implemented in **R**

- CRAN packages solely devoted to one or more distributions
- CRAN packages which implement distributions incidentally (e.g. **VGAM**)
- implementations of distributions not on CRAN, e.g. Jim Lindsey’s work
In base **R** there are 20 distributions implemented, at least in part

All univariate—consider univariate distributions only

Numerous other distributions have been implemented in **R**

- CRAN packages solely devoted to one or more distributions
- CRAN packages which implement distributions incidentally (e.g. VGAM)
- implementations of distributions not on CRAN, e.g. Jim Lindsey’s work
There are overlaps in coverage of distributions in R.

Implementations of distributions in R are inconsistent:
- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions.
There are overlaps in coverage of distributions in R. Implementations of distributions in R are inconsistent:

- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions.
There are overlaps in coverage of distributions in \textbf{R}

Implementations of distributions in \textbf{R} are inconsistent
- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions
Introduction

- There are overlaps in coverage of distributions in \textbf{R}
- Implementations of distributions in \textbf{R} are inconsistent
  - naming of objects
  - parameterizations
    - function arguments
    - functionality
    - return structures
- It is useful to discuss some standardization of implementation of software for distributions
There are overlaps in coverage of distributions in R

Implementations of distributions in R are inconsistent

- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions
Introduction

- There are overlaps in coverage of distributions in \texttt{R}
- Implementations of distributions in \texttt{R} are inconsistent
  - naming of objects
  - parameterizations
  - function arguments
  - functionality
  - return structures
- It is useful to discuss some standardization of implementation of software for distributions
There are overlaps in coverage of distributions in R.

Implementations of distributions in R are inconsistent:
- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions.
There are overlaps in coverage of distributions in \texttt{R}.

Implementations of distributions in \texttt{R} are inconsistent:
- naming of objects
- parameterizations
- function arguments
- functionality
- return structures

It is useful to discuss some standardization of implementation of software for distributions.
Standardization

- There are benefits to a standardized approach
  - easier for users
  - easier for developers
  - fewer errors
- Deserves thought, even if not prescriptive
- Perhaps too late!
There are benefits to a standardized approach

- easier for users
  - easier for developers
  - fewer errors

Deserves thought, even if not prescriptive

Perhaps too late!
Standardization

- There are benefits to a standardized approach
  - easier for users
  - easier for developers
  - fewer errors
- Deserves thought, even if not prescriptive
- Perhaps too late!
There are benefits to a standardized approach
- easier for users
- easier for developers
- fewer errors

Deserves thought, even if not prescriptive
Perhaps too late!
Standardization

- There are benefits to a standardized approach
  - easier for users
  - easier for developers
  - fewer errors

- Deserves thought, even if not prescriptive

- Perhaps too late!
Standardization

- There are benefits to a standardized approach
  - easier for users
  - easier for developers
  - fewer errors
- Deserves thought, even if not prescriptive
- Perhaps too late!
1 Introduction

2 Current Software for Distributions

3 Implementation in Base R

4 Design for Distribution Implementation
Implementation in R is essentially the provision of \textit{dpqr} functions: the density (or probability) function, distribution function, quantile or inverse distribution function and random number generation.

The distributions are the binomial (\texttt{binom}), geometric (\texttt{geom}), hypergeometric (\texttt{hyper}), negative binomial (\texttt{nbinom}), Poisson (\texttt{pois}), Wilcoxon signed rank statistic (\texttt{signrank}), Wilcoxon rank sum statistic (\texttt{wilcox}), beta (\texttt{beta}), Cauchy (\texttt{Cauchy}), non-central chi-squared (\texttt{chisq}), exponential (\texttt{exp}), $F$ (\texttt{f}), gamma (\texttt{gamma}), log-normal (\texttt{lnorm}), logistic (\texttt{logis}), normal (\texttt{norm}), $t$ (\texttt{t}), uniform (\texttt{unif}), Weibull (\texttt{weibull}), and Tukey studentized range (\texttt{tukey}) for which only the $p$ and $q$ functions are implemented.
Distributions in Base R

- Implementation in R is essentially the provision of \textit{dpqr} functions: the density (or probability) function, distribution function, quantile or inverse distribution function and random number generation.

- The distributions are the binomial (\texttt{binom}), geometric (\texttt{geom}), hypergeometric (\texttt{hyper}), negative binomial (\texttt{nbinom}), Poisson (\texttt{pois}), Wilcoxon signed rank statistic (\texttt{signrank}), Wilcoxon rank sum statistic (\texttt{wilcox}), beta (\texttt{beta}), Cauchy (\texttt{Cauchy}), non-central chi-squared (\texttt{chisq}), exponential (\texttt{exp}), \(F\) (\texttt{f}), gamma (\texttt{gamma}), log-normal (\texttt{lnorm}), logistic (\texttt{logis}), normal (\texttt{norm}), \(t\) (\texttt{t}), uniform (\texttt{unif}), Weibull (\texttt{weibull}), and Tukey studentized range (\texttt{tukey}) for which only the \(p\) and \(q\) functions are implemented.
Many distributions are implemented in R

The following list is not complete—see the task view http://cran.r-project.org/web/views/Distributions.html

Primarily packages which deal with a particular distribution or set of related distributions

Some packages not on CRAN are nonetheless available

I know of other implementations not on CRAN and not available
Contributed Packages

- Many distributions are implemented in **R**
- The following list is not complete—see the task view [http://cran.r-project.org/web/views/Distributions.html](http://cran.r-project.org/web/views/Distributions.html)
  - Primarily packages which deal with a particular distribution or set of related distributions
  - Some packages not on CRAN are nonetheless available
  - I know of other implementations not on CRAN and not available
Contributed Packages

- Many distributions are implemented in R
- The following list is not complete—see the task view http://cran.r-project.org/web/views/Distributions.html
- Primarily packages which deal with a particular distribution or set of related distributions
  - Some packages not on CRAN are nonetheless available
  - I know of other implementations not on CRAN and not available
Contributed Packages

- Many distributions are implemented in R
- The following list is not complete—see the task view http://cran.r-project.org/web/views/Distributions.html
- Primarily packages which deal with a particular distribution or set of related distributions
- Some packages not on CRAN are nonetheless available
- I know of other implementations not on CRAN and not available
Many distributions are implemented in R

The following list is not complete—see the task view http://cran.r-project.org/web/views/Distributions.html

Primarily packages which deal with a particular distribution or set of related distributions

Some packages not on CRAN are nonetheless available

I know of other implementations not on CRAN and not available
## Packages

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actuar</td>
<td>Collection of functions and data sets related to actuarial science applications, including loss distributions</td>
</tr>
<tr>
<td>bs</td>
<td>Package for the Birnbaum-Saunders distribution</td>
</tr>
<tr>
<td>evd</td>
<td>Functions for extreme value distributions</td>
</tr>
<tr>
<td>Davies</td>
<td>The Davies quantile function</td>
</tr>
<tr>
<td>evdbayes</td>
<td>Bayesian analysis in extreme value theory</td>
</tr>
<tr>
<td>evir</td>
<td>Extreme values in R</td>
</tr>
<tr>
<td>exactRankTests</td>
<td>Exact distributions for rank and permutation tests</td>
</tr>
<tr>
<td>extRemes</td>
<td>Extreme value toolkit</td>
</tr>
<tr>
<td>ghyp</td>
<td>A package on generalized hyperbolic distributions</td>
</tr>
<tr>
<td>gld</td>
<td>Estimation and use of the generalised (Tukey) lambda distribution</td>
</tr>
<tr>
<td>HyperbolicDist</td>
<td>The hyperbolic distribution</td>
</tr>
</tbody>
</table>

David Scott, Diethelm Würtz, Christine Dong

Software for Distributions in R
<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ig</td>
<td>Package for the robust and classical versions of the inverse Gaussian distribution</td>
</tr>
<tr>
<td>ismev</td>
<td>An introduction to statistical modeling of extreme values</td>
</tr>
<tr>
<td>lmomco</td>
<td>L-moments, trimmed L-moments, L-comoments, and many distributions</td>
</tr>
<tr>
<td>Lmoments</td>
<td>L-moments and quantile mixtures</td>
</tr>
<tr>
<td>mnormt</td>
<td>The multivariate normal and t distributions</td>
</tr>
<tr>
<td>mvtnorm</td>
<td>Multivariate normal and t distribution</td>
</tr>
<tr>
<td>normalp</td>
<td>Package for exponential power distributions (EPD)</td>
</tr>
<tr>
<td>POT</td>
<td>Generalized Pareto distribution and peaks over threshold</td>
</tr>
<tr>
<td>Rmetrics</td>
<td>An environment for teaching financial engineering and computational finance with R</td>
</tr>
</tbody>
</table>
## Packages

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>skewt</td>
<td>The skewed Student-t distribution</td>
</tr>
<tr>
<td>sn</td>
<td>The skew-normal and skew-t distributions</td>
</tr>
<tr>
<td>SuppDists</td>
<td>Supplementary distributions</td>
</tr>
<tr>
<td>tdist</td>
<td>Distribution of a linear combination of independent Student’s t-variables</td>
</tr>
<tr>
<td>triangle</td>
<td>Provides the standard distribution functions for the triangle distribution</td>
</tr>
<tr>
<td>VarianceGamma</td>
<td>The variance gamma distribution</td>
</tr>
</tbody>
</table>

There are a number of packages which have implementations of a number of distributions.
<table>
<thead>
<tr>
<th>Package Name</th>
<th>Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>actuar</td>
<td>Burr, inverse Burr, generalized beta, generalized Pareto, inverse exponential,</td>
</tr>
<tr>
<td></td>
<td>inverse gamma, inverse paralogistic, inverse Pareto, inverse transformed gamma,</td>
</tr>
<tr>
<td></td>
<td>inverse Weibull, log-gamma, log-logistic, paralogistic, Pareto, transformed</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
</tr>
<tr>
<td>fBasics (Rmetrics)</td>
<td>Skew-normal, skew-t, generalized hyperbolic, hyperbolic, normal inverse</td>
</tr>
<tr>
<td></td>
<td>Gaussian, generalized hyperbolic Student-t, stable</td>
</tr>
<tr>
<td>fExtremes (Rmetrics)</td>
<td>Generalized extreme value, generalize Pareto</td>
</tr>
<tr>
<td>fGarch (Rmetrics)</td>
<td>Generalized exponential, double exponential</td>
</tr>
<tr>
<td>fOptions (Rmetrics)</td>
<td>Johnson Type IV, reciprocal gamma</td>
</tr>
<tr>
<td>Package Name</td>
<td>Distributions</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>HyperbolicDist</td>
<td>generalized hyperbolic, hyperbolic, generalized inverse Gaussian, skew-Laplace</td>
</tr>
<tr>
<td>QRMMlib</td>
<td>generalized extreme value, generalized hyperbolic, generalized Pareto, Gumbel, probit-normal</td>
</tr>
<tr>
<td>SuppDists</td>
<td>Friedman chi-squared, Johnson system (types I–IV), Kendall’s tau, Kruskal-Wallis, normal scores, generalized hypergeometric, inverse Gaussian</td>
</tr>
</tbody>
</table>
Jim Lindsey’s packages include multiple distributions also

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmutil</td>
<td>beta-binomial, Box-Cox, Burr, Consul, double binomial, double Poisson, gamma</td>
</tr>
<tr>
<td></td>
<td>count, generalized extreme value, generalized gamma, generalized inverse Gaussian,</td>
</tr>
<tr>
<td></td>
<td>generalized logistic, generalized Weibull, Hjorth, Laplace, Lévy, multiplicative binomial,</td>
</tr>
<tr>
<td></td>
<td>multiplicative Poisson, Pareto, power exponential, power variance function Poisson, simplex, skew-Laplace, two-sided power stable</td>
</tr>
<tr>
<td>stable</td>
<td>stable</td>
</tr>
</tbody>
</table>
1. Introduction

2. Current Software for Distributions

3. Implementation in Base R

4. Design for Distribution Implementation
Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.

- The argument lists for the dpqr functions are standard.
- First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.
- rwilcox is an exception using nn because n is a parameter.
- Subsequent arguments give the parameters.
- The gamma distribution is unusual, with argument list shape, rate =1, scale = 1/rate.
- This mechanism allows the user to specify the second parameter as either the scale or the rate.
Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.

The argument lists for the dpqr functions are standard:

- First argument is \( x, p, q \) and \( n \) for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.
- \( \text{rwilcox} \) is an exception using \( nn \) because \( n \) is a parameter.
- Subsequent arguments give the parameters.
- The gamma distribution is unusual, with argument list \( \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate} \).
- This mechanism allows the user to specify the second parameter as either the scale or the rate.
dpqr Functions

- Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.
- The argument lists for the dpqr functions are standard.
- First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.
- rwilcox is an exception using nn because n is a parameter.
- Subsequent arguments give the parameters.
- The gamma distribution is unusual, with argument list shape, rate = 1, scale = 1/rate.
- This mechanism allows the user to specify the second parameter as either the scale or the rate.
Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.

The argument lists for the dpqr functions are standard.

First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.

`rwilcox` is an exception using nn because n is a parameter.

Subsequent arguments give the parameters.

The gamma distribution is unusual, with argument list shape, rate =1, scale = 1/rate.

This mechanism allows the user to specify the second parameter as either the scale or the rate.
dpqr Functions

- Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.

- The argument lists for the dpqr functions are standard.

- First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.

- rwilcox is an exception using nn because n is a parameter.

- Subsequent arguments give the parameters.

  - The gamma distribution is unusual, with argument list shape, rate =1, scale = 1/rate.

  - This mechanism allows the user to specify the second parameter as either the scale or the rate.

David Scott, Diethelm Würtz, Christine Dong
Software for Distributions in R
dpqr Functions

- Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.
- The argument lists for the dpqr functions are standard.
- First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.
- rwilcox is an exception using nn because n is a parameter.
- Subsequent arguments give the parameters.
- The gamma distribution is unusual, with argument list shape, rate = 1, scale = 1/rate.
- This mechanism allows the user to specify the second parameter as either the scale or the rate.
dpqr Functions

- Any experienced R user will be aware of the naming conventions for the density, cumulative distribution, quantile and random number generation functions for the base R distributions.
- The argument lists for the dpqr functions are standard.
- First argument is x, p, q and n for respectively a vector of quantiles, a vector of quantiles, a vector of probabilities, and the sample size.
- rwilcox is an exception using nn because n is a parameter.
- Subsequent arguments give the parameters.
- The gamma distribution is unusual, with argument list shape, rate = 1, scale = 1/rate.
- This mechanism allows the user to specify the second parameter as either the scale or the rate.
dpqr Functions

- Other arguments differ among the dpqr functions
  - The d functions take the argument log, the p and q functions the argument log.p
  - These allow the extension of the range of accurate computation for these quantities
  - The p and q functions have the argument lower.tail
  - The dpqr functions are coded in C and may be found in the source software tree at /src/math/
  - They are in large part due to Ross Ihaka and Catherine Loader
  - Martin Mächler is now responsible for on-going maintenance
dpqr Functions

- Other arguments differ among the dpqr functions
- The d functions take the argument log, the p and q functions take the argument log.p
- These allow the extension of the range of accurate computation for these quantities
- The p and q functions have the argument lower.tail
- The dpqr functions are coded in C and may be found in the source software tree at /src/math/
- They are in large part due to Ross Ihaka and Catherine Loader
- Martin Mächler is now responsible for on-going maintenance
dpqr Functions

- Other arguments differ among the dpqr functions.
- The d functions take the argument `log`, the p and q functions the argument `log.p`.
- These allow the extension of the range of accurate computation for these quantities.
- The p and q functions have the argument `lower.tail`.
- The dpqr functions are coded in C and may be found in the source software tree at `/src/math/`.
- They are in large part due to Ross Ihaka and Catherine Loader.
- Martin Mächler is now responsible for on-going maintenance.
dpqr Functions

- Other arguments differ among the dpqr functions
- The d functions take the argument \( \log \), the p and q functions the argument \( \log.p \)
- These allow the extension of the range of accurate computation for these quantities
- The p and q functions have the argument \( \text{lower.tail} \)
- The dpqr functions are coded in C and may be found in the source software tree at /src/math/
- They are in large part due to Ross Ihaka and Catherine Loader
- Martin Mächler is now responsible for on-going maintenance
dpqr Functions

- Other arguments differ among the dpqr functions
- The d functions take the argument log, the p and q functions the argument log.p
- These allow the extension of the range of accurate computation for these quantities
- The p and q functions have the argument lower.tail
- The dpqr functions are coded in C and may be found in the source software tree at /src/math/
  - They are in large part due to Ross Ihaka and Catherine Loader
  - Martin Mächler is now responsible for on-going maintenance
dpqr Functions

- Other arguments differ among the dpqr functions
- The d functions take the argument log, the p and q functions the argument log.p
- These allow the extension of the range of accurate computation for these quantities
- The p and q functions have the argument lower.tail
- The dpqr functions are coded in C and may be found in the source software tree at /src/math/
- They are in large part due to Ross Ihaka and Catherine Loader
  - Martin Mächler is now responsible for on-going maintenance
dpqr Functions

- Other arguments differ among the dpqr functions
- The d functions take the argument log, the p and q functions the argument log.p
- These allow the extension of the range of accurate computation for these quantities
- The p and q functions have the argument lower.tail
- The dpqr functions are coded in C and may be found in the source software tree at /src/math/
- They are in large part due to Ross Ihaka and Catherine Loader
- Martin Mächler is now responsible for on-going maintenance
The algorithms used in the dpqr functions are well-established algorithms taken from a substantial scientific literature.

There are also tests performed, found in the directory tests in two files d-p-q-r-tests.R and p-r-random-tests.R.

Tests in d-p-q-r-tests.R are “inversion tests” which check that \( q_{\text{dist}}(p_{\text{dist}}(x)) = x \) for values \( x \) generated by \( r_{\text{dist}} \).

There are tests relying on special distribution relationships, and tests using extreme values of parameters or arguments.

For discrete distributions equality of \( \text{cumsum}(d_{\text{dist}}(.)) = p_{\text{dist}}(.) \).
The algorithms used in the dpqr functions are well-established algorithms taken from a substantial scientific literature.

There are also tests performed, found in the directory tests in two files d-p-q-r-tests.R and p-r-random-tests.R.

Tests in d-p-q-r-tests.R are “inversion tests” which check that \( q_{\text{dist}}(p_{\text{dist}}(x)) = x \) for values \( x \) generated by \( r_{\text{dist}} \).

There are tests relying on special distribution relationships, and tests using extreme values of parameters or arguments.

For discrete distributions equality of \( \text{cumsum}(d_{\text{dist}}(.)) = p_{\text{dist}}(.) \)
The algorithms used in the dpqr functions are well-established algorithms taken from a substantial scientific literature.

There are also tests performed, found in the directory tests in two files d-p-q-r-tests.R and p-r-random-tests.R.

Tests in d-p-q-r-tests.R are “inversion tests” which check that \( q_{\text{dist}}(p_{\text{dist}}(x)) = x \) for values \( x \) generated by \( r_{\text{dist}} \).

There are tests relying on special distribution relationships, and tests using extreme values of parameters or arguments.

For discrete distributions equality of \( \text{cumsum}(d_{\text{dist}}(.)) = p_{\text{dist}}(.) \).
Testing and Validation

- The algorithms used in the dpqr functions are well-established algorithms taken from a substantial scientific literature.
- There are also tests performed, found in the directory tests in two files d-p-q-r-tests.R and p-r-random-tests.R.
- Tests in d-p-q-r-tests.R are “inversion tests” which check that \( q_{\text{dist}}(p_{\text{dist}}(x)) = x \) for values \( x \) generated by \( r_{\text{dist}} \).
- There are tests relying on special distribution relationships, and tests using extreme values of parameters or arguments.
- For discrete distributions equality of \( \text{cumsum}(d_{\text{dist}}(.)) = p_{\text{dist}}(.) \).
The algorithms used in the dpqr functions are well-established algorithms taken from a substantial scientific literature.

There are also tests performed, found in the directory tests in two files d-p-q-r-tests.R and p-r-random-tests.R.

Tests in d-p-q-r-tests.R are “inversion tests” which check that \( q \text{dist}(p \text{dist}(x)) = x \) for values \( x \) generated by \( r \text{dist} \).

There are tests relying on special distribution relationships, and tests using extreme values of parameters or arguments.

For discrete distributions equality of \( \text{cumsum}(d \text{dist}(.)) = p \text{dist}(.) \).
Testing and Validation

- Tests in `p-r-random-tests.R` are based on an inequality of Massart:

\[
\Pr \left( \sup_x |\hat{F}_n(x) - F(x)| > \lambda \right) \leq 2 \exp(-2n\lambda^2)
\]

where \(\hat{F}_n\) is the empirical distribution function for a

- This is a version of the Dvoretzky-Kiefer-Wolfowitz inequality with the best possible constant, namely the leading 2 in the right hand side of the inequality.

- The inequality above is true for all distribution functions, for all \(n\) and \(\lambda\).

- Distributions are tested by generating a sample of size 10,000 and comparing the difference between the empirical distribution function and distribution function.
Testing and Validation

- Tests in `p-r-random-tests.R` are based on an inequality of Massart:

\[
\Pr \left( \sup_x |\hat{F}_n(x) - F(x)| > \lambda \right) \leq 2 \exp(-2n\lambda^2)
\]

where \( \hat{F}_n \) is the empirical distribution function for a

- This is a version of the Dvoretzky-Kiefer-Wolfowitz inequality with the best possible constant, namely the leading 2 in the right hand side of the inequality

- The inequality above is true for all distribution functions, for all \( n \) and \( \lambda \)

- Distributions are tested by generating a sample of size 10,000 and comparing the difference between the empirical distribution function and distribution function
Testing and Validation

- Tests in `p-r-random-tests.R` are based on an inequality of Massart:

\[
\Pr \left( \sup_x |\hat{F}_n(x) - F(x)| > \lambda \right) \leq 2 \exp(-2n\lambda^2)
\]

where \(\hat{F}_n\) is the empirical distribution function for a

- This is a version of the Dvoretzky-Kiefer-Wolfowitz inequality with the best possible constant, namely the leading 2 in the right hand side of the inequality

- The inequality above is true for all distribution functions, for all \(n\) and \(\lambda\)

- Distributions are tested by generating a sample of size 10,000 and comparing the difference between the empirical distribution function and distribution function
Tests in `p-r-random-tests.R` are based on an inequality of Massart:

\[
\Pr \left( \sup_x |\hat{F}_n(x) - F(x)| > \lambda \right) \leq 2 \exp\left(-2n\lambda^2\right)
\]

where \( \hat{F}_n \) is the empirical distribution function for a

- This is a version of the Dvoretzky-Kiefer-Wolfowitz inequality with the best possible constant, namely the leading 2 in the right hand side of the inequality
- The inequality above is true for all distribution functions, for all \( n \) and \( \lambda \)
- Distributions are tested by generating a sample of size 10,000 and comparing the difference between the empirical distribution function and distribution function
1. Introduction

2. Current Software for Distributions

3. Implementation in Base R

4. Design for Distribution Implementation
What Should be Provided?

- Besides the obvious dpqr functions, what else is needed?
  - moments, at least low order ones
  - the mode for unimodal distributions
  - moment generating function and characteristic function
  - functions for changing parameterisations
  - functions for fitting of distributions and fit diagnostics
  - goodness-of-fit tests
  - methods associated with fit results: print, plot, summary, print.summary and coef
  - for maximum likelihood fits, methods such as logLik and profile
Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
- methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
- methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
What Should be Provided?

Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
- methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
What Should be Provided?

- Besides the obvious dpqr functions, what else is needed?
  - moments, at least low order ones
  - the mode for unimodal distributions
  - moment generating function and characteristic function
  - functions for changing parameterisations
  - functions for fitting of distributions and fit diagnostics
  - goodness-of-fit tests
  - methods associated with fit results: print, plot, summary, print.summary and coef
  - for maximum likelihood fits, methods such as logLik and profile
Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
- methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
What Should be Provided?

Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
  - methods associated with fit results: print, plot, summary, print.summary and coef
  - for maximum likelihood fits, methods such as logLik and profile
What Should be Provided?

- Besides the obvious dpqr functions, what else is needed?
  - moments, at least low order ones
  - the mode for unimodal distributions
  - moment generating function and characteristic function
  - functions for changing parameterisations
  - functions for fitting of distributions and fit diagnostics
  - goodness-of-fit tests
  - methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
What Should be Provided?

Besides the obvious dpqr functions, what else is needed?

- moments, at least low order ones
- the mode for unimodal distributions
- moment generating function and characteristic function
- functions for changing parameterisations
- functions for fitting of distributions and fit diagnostics
- goodness-of-fit tests
- methods associated with fit results: print, plot, summary, print.summary and coef
- for maximum likelihood fits, methods such as logLik and profile
Fitting Diagnostics

- To assess the fit of a distribution, diagnostic plots should be provided

- Some useful plots are
  - a histogram or empirical density with fitted density
  - a log-histogram with fitted log-density
  - a QQ-plot with optional fitted line
  - a PP-plot with optional fitted line

- For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots
Fitting Diagnostics

- To assess the fit of a distribution, diagnostic plots should be provided.
- Some useful plots are:
  - A histogram or empirical density with fitted density.
  - A log-histogram with fitted log-density.
  - A QQ-plot with optional fitted line.
  - A PP-plot with optional fitted line.
- For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots.
Fitting Diagnostics

- To assess the fit of a distribution, diagnostic plots should be provided.

- Some useful plots are:
  - a histogram or empirical density with fitted density
  - a log-histogram with fitted log-density
  - a QQ-plot with optional fitted line
  - a PP-plot with optional fitted line

- For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots.
Fitting Diagnostics

To assess the fit of a distribution, diagnostic plots should be provided.

Some useful plots are:
- A histogram or empirical density with fitted density.
- A log-histogram with fitted log-density.
- A QQ-plot with optional fitted line.
- A PP-plot with optional fitted line.

For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots.
Fitting Diagnostics

To assess the fit of a distribution, diagnostic plots should be provided

Some useful plots are

- a histogram or empirical density with fitted density
- a log-histogram with fitted log-density
- a QQ-plot with optional fitted line
- a PP-plot with optional fitted line

For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots
Fitting Diagnostics

- To assess the fit of a distribution, diagnostic plots should be provided.
- Some useful plots are:
  - a histogram or empirical density with fitted density
  - a log-histogram with fitted log-density
  - a QQ-plot with optional fitted line
  - a PP-plot with optional fitted line
- For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots.
To assess the fit of a distribution, diagnostic plots should be provided.

Some useful plots are:
- a histogram or empirical density with fitted density
- a log-histogram with fitted log-density
- a QQ-plot with optional fitted line
- a PP-plot with optional fitted line

For maximum likelihood estimation, contour plots and perspective plots for pairs of parameters, and likelihood profile plots.
Some generic fitting routines are currently available

- `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied
- `fitdistr` from `MASS` will automatically fit most of the distributions from base `R`
- Other distributions can be fitted using `mle` by supplying the density and started values
- In designing fitting functions, the structure of the object returned and the methods available are vital aspects
- `mle` returns an S4 object of class `mle`
- `fitdistr` produces and S3 object of class `fitdistr`
Some generic fitting routines are currently available

- `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied

- `fitdistr` from `MASS` will automatically fit most of the distributions from base `R`

- Other distributions can be fitted using `mle` by supplying the density and started values

- In designing fitting functions, the structure of the object returned and the methods available are vital aspects

- `mle` returns an S4 object of class `mle`

- `fitdistr` produces and S3 object of class `fitdistr`
Some generic fitting routines are currently available

- `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied
- `fitdistr` from `MASS` will automatically fit most of the distributions from base R

Other distributions can be fitted using `mle` by supplying the density and started values.

In designing fitting functions, the structure of the object returned and the methods available are vital aspects.

- `mle` returns an S4 object of class `mle`
- `fitdistr` produces and S3 object of class `fitdistr`
Some generic fitting routines are currently available

- `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied
- `fitdistr` from `MASS` will automatically fit most of the distributions from base R

Other distributions can be fitted using `mle` by supplying the density and started values

- In designing fitting functions, the structure of the object returned and the methods available are vital aspects
- `mle` returns an S4 object of class `mle`
- `fitdistr` produces an S3 object of class `fitdistr`
Some generic fitting routines are currently available

- `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied
- `fitdistr` from `MASS` will automatically fit most of the distributions from base R

Other distributions can be fitted using `mle` by supplying the density and started values

In designing fitting functions, the structure of the object returned and the methods available are vital aspects

- `mle` returns an S4 object of class `mle`
- `fitdistr` produces an S3 object of class `fitdistr`
Some generic fitting routines are currently available

* `mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied

* `fitdistr` from `MASS` will automatically fit most of the distributions from base R

Other distributions can be fitted using `mle` by supplying the density and starting values

In designing fitting functions, the structure of the object returned and the methods available are vital aspects

* `mle` returns an S4 object of class `mle`

* `fitdistr` produces an S3 object of class `fitdistr`
Some generic fitting routines are currently available

`mle` from `stats4` can be used to fit distributions but the log likelihood and starting values must be supplied

`fitdistr` from `MASS` will automatically fit most of the distributions from base `R`

Other distributions can be fitted using `mle` by supplying the density and started values

In designing fitting functions, the structure of the object returned and the methods available are vital aspects

`mle` returns an S4 object of class `mle`

`fitdistr` produces and S3 object of class `fitdistr`
The methods available for an object of class `mle` are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>confint</td>
<td>Confidence intervals from likelihood profiles</td>
</tr>
<tr>
<td>logLik</td>
<td>Extract maximized log-likelihood</td>
</tr>
<tr>
<td>profile</td>
<td>Likelihood profile generation</td>
</tr>
<tr>
<td>show</td>
<td>Display object briefly</td>
</tr>
<tr>
<td>summary</td>
<td>Generate object summary</td>
</tr>
<tr>
<td>update</td>
<td>Update fit</td>
</tr>
<tr>
<td>vcov</td>
<td>Extract variance-covariance matrix</td>
</tr>
</tbody>
</table>

For `fitdistr` the methods are `print`, `coef`, and `logLik`.

Neither function returns the data, so a plot method which produces suitable diagnostic plots is not possible.

Ideally a fit should return and object of class `distFit` say, and the `mle` class should extend that.
The methods available for an object of class `mle` are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>confint</code></td>
<td>Confidence intervals from likelihood profiles</td>
</tr>
<tr>
<td><code>logLik</code></td>
<td>Extract maximized log-likelihood</td>
</tr>
<tr>
<td><code>profile</code></td>
<td>Likelihood profile generation</td>
</tr>
<tr>
<td><code>show</code></td>
<td>Display object briefly</td>
</tr>
<tr>
<td><code>summary</code></td>
<td>Generate object summary</td>
</tr>
<tr>
<td><code>update</code></td>
<td>Update fit</td>
</tr>
<tr>
<td><code>vcov</code></td>
<td>Extract variance-covariance matrix</td>
</tr>
</tbody>
</table>

For `fitdistr` the methods are `print`, `coef`, and `logLik`.

Neither function returns the data, so a plot method which produces suitable diagnostic plots is not possible.

Ideally a fit should return an object of class `distFit` say, and the `mle` class should extend that.
The methods available for an object of class \texttt{mle} are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>confint</td>
<td>Confidence intervals from likelihood profiles</td>
</tr>
<tr>
<td>logLik</td>
<td>Extract maximized log-likelihood</td>
</tr>
<tr>
<td>profile</td>
<td>Likelihood profile generation</td>
</tr>
<tr>
<td>show</td>
<td>Display object briefly</td>
</tr>
<tr>
<td>summary</td>
<td>Generate object summary</td>
</tr>
<tr>
<td>update</td>
<td>Update fit</td>
</tr>
<tr>
<td>vcov</td>
<td>Extract variance-covariance matrix</td>
</tr>
</tbody>
</table>

For \texttt{fitdistr} the methods are \texttt{print}, \texttt{coef}, and \texttt{logLik}

Neither function returns the data, so a plot method which produces suitable diagnostic plots is not possible

Ideally a fit should return and object of class \texttt{distFit} say, and the \texttt{mle} class should extend that
The methods available for an object of class `mle` are:

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>confint</td>
<td>Confidence intervals from likelihood profiles</td>
</tr>
<tr>
<td>logLik</td>
<td>Extract maximized log-likelihood</td>
</tr>
<tr>
<td>profile</td>
<td>Likelihood profile generation</td>
</tr>
<tr>
<td>show</td>
<td>Display object briefly</td>
</tr>
<tr>
<td>summary</td>
<td>Generate object summary</td>
</tr>
<tr>
<td>update</td>
<td>Update fit</td>
</tr>
<tr>
<td>vcov</td>
<td>Extract variance-covariance matrix</td>
</tr>
</tbody>
</table>

For `fitdistr` the methods are `print`, `coef`, and `logLik`.

Neither function returns the data, so a plot method which produces suitable diagnostic plots is not possible.

Ideally a fit should return and object of class `distFit` say, and the `mle` class should extend that.
Some Principles

- There are obvious advantages in a standard design, both for developers and for users

- Some principles are
  - the major guide to the design should be what exists in base R
  - the design should be logical with as few special cases as possible
  - the design should minimize the possibility of programming mistakes by users and developers
  - the design should simplify as much as possible the provision of the range of facilities needed to implement a distribution

- A naming scheme for functions is an important part of a standard
Some Principles

- There are obvious advantages in a standard design, both for developers and for users
- Some principles are
  - the major guide to the design should be what exists in base R
  - the design should be logical with as few special cases as possible
  - the design should minimize the possibility of programming mistakes by users and developers
  - the design should simplify as much as possible the provision of the range of facilities needed to implement a distribution
- A naming scheme for functions is an important part of a standard
Some Principles

- There are obvious advantages in a standard design, both for developers and for users.
- Some principles are:
  - The major guide to the design should be what exists in base R.
  - The design should be logical with as few special cases as possible.
  - The design should minimize the possibility of programming mistakes by users and developers.
  - The design should simplify as much as possible the provision of the range of facilities needed to implement a distribution.
- A naming scheme for functions is an important part of a standard.
Some Principles

- There are obvious advantages in a standard design, both for developers and for users.

Some principles are:

- The major guide to the design should be what exists in base R.
- The design should be logical with as few special cases as possible.
- The design should minimize the possibility of programming mistakes by users and developers.
- The design should simplify as much as possible the provision of the range of facilities needed to implement a distribution.

- A naming scheme for functions is an important part of a standard.
Some Principles

- There are obvious advantages in a standard design, both for developers and for users
- Some principles are
  - the major guide to the design should be what exists in base R
  - the design should be logical with as few special cases as possible
  - the design should minimize the possibility of programming mistakes by users and developers
  - the design should simplify as much as possible the provision of the range of facilities needed to implement a distribution
- A naming scheme for functions is an important part of a standard
Some Principles

- There are obvious advantages in a standard design, both for developers and for users
- Some principles are
  - the major guide to the design should be what exists in base R
  - the design should be logical with as few special cases as possible
  - the design should minimize the possibility of programming mistakes by users and developers
  - the design should simplify as much as possible the provision of the range of facilities needed to implement a distribution

- A naming scheme for functions is an important part of a standard
Some Principles

- There are obvious advantages in a standard design, both for developers and for users.
- Some principles are:
  - the major guide to the design should be what exists in base R.
  - the design should be logical with as few special cases as possible.
  - the design should minimize the possibility of programming mistakes by users and developers.
  - the design should simplify as much as possible the provision of the range of facilities needed to implement a distribution.
- A naming scheme for functions is an important part of a standard.
# A Possible Naming Scheme

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{dist}}$</td>
<td>Density function or probability function</td>
</tr>
<tr>
<td>$p_{\text{dist}}$</td>
<td>Cumulative distribution function</td>
</tr>
<tr>
<td>$q_{\text{dist}}$</td>
<td>Quantile function</td>
</tr>
<tr>
<td>$r_{\text{dist}}$</td>
<td>Random number generator</td>
</tr>
<tr>
<td>$\text{distMean}$</td>
<td>Theoretical mean</td>
</tr>
<tr>
<td>$\text{distVar}$</td>
<td>Theoretical variance</td>
</tr>
<tr>
<td>$\text{distSkew}$</td>
<td>Theoretical skewness</td>
</tr>
<tr>
<td>$\text{distKurt}$</td>
<td>Theoretical kurtosis</td>
</tr>
<tr>
<td>$\text{distMode}$</td>
<td>Mode of the density</td>
</tr>
<tr>
<td>$\text{distMoments}$</td>
<td>Theoretical moments (and mode)</td>
</tr>
<tr>
<td>$\text{distMGF}$</td>
<td>Moment generating function</td>
</tr>
<tr>
<td>$\text{distChFn}$</td>
<td>Characteristic function</td>
</tr>
<tr>
<td>$\text{distChangePars}$</td>
<td>Change parameterization of the distribution</td>
</tr>
<tr>
<td>$\text{distFit}$</td>
<td>Result of fitting the distribution to data</td>
</tr>
<tr>
<td>$\text{distTest}$</td>
<td>Test the distribution</td>
</tr>
</tbody>
</table>
Some Alternatives

- Use dots: `hyperb.mean`, usually deprecated because of confusion with S3 methods

- Use initial letters: `mhyperb`, but what about the mgf, multivariate distributions?
Some Alternatives

- Use dots: `hyperb.mean`, usually deprecated because of confusion with S3 methods
- Use initial letters: `mhyperb`, but what about the mgf, multivariate distributions?
Function Arguments

- Specification of parameters could allow for the parameters to be specified as a vector
- This is helpful for maximum likelihood estimation
- The approach used for the gamma distribution can be used to allow for both single parameter specification and vector parameter specification
- Separate parameters should be in the order of location, scale, skewness, kurtosis
Function Arguments

• Specification of parameters could allow for the parameters to be specified as a vector
• This is helpful for maximum likelihood estimation
• The approach used for the gamma distribution can be used to allow for both single parameter specification and vector parameter specification
• Separate parameters should be in the order of location, scale, skewness, kurtosis
Function Arguments

- Specification of parameters could allow for the parameters to be specified as a vector.
- This is helpful for maximum likelihood estimation.
- The approach used for the gamma distribution can be used to allow for both single parameter specification and vector parameter specification.
- Separate parameters should be in the order of location, scale, skewness, kurtosis.
Function Arguments

- Specification of parameters could allow for the parameters to be specified as a vector
- This is helpful for maximum likelihood estimation
- The approach used for the gamma distribution can be used to allow for both single parameter specification and vector parameter specification
- Separate parameters should be in the order of location, scale, skewness, kurtosis
I don’t have a view on whether S3 or S4 classes should be used, but probably S4 classes should be aimed for.

For a fit from a distribution, the class could be called `distfit`.

A fit for a particular distribution would add the distribution name: `hyperbfit`, `distfit` with S3 methods.

For S4 methods the class `distfit` would be extended.

Similar ideas would be used for tests: a Kolomogorov-Smirnov test would have class `kstest`, `htest` with S3 methods.
I don’t have a view on whether S3 or S4 classes should be used, but probably S4 classes should be aimed for.

For a fit from a distribution, the class could be called distfit.

A fit for a particular distribution would add the distribution name: hyperbfit, distfit with S3 methods.

For S4 methods the class distfit would be extended.

Similar ideas would be used for tests: a Kolomogorov-Smirnov test would have class kstest, htest with S3 methods.
I don’t have a view on whether S3 or S4 classes should be used, but probably S4 classes should be aimed for.

For a fit from a distribution, the class could be called `distfit`.

A fit for a particular distribution would add the distribution name: `hyperbfit`, `distfit` with S3 methods.

For S4 methods the class `distfit` would be extended.

Similar ideas would be used for tests: a Kolomogorov-Smirnov test would have class `kstest`, `htest` with S3 methods.
I don’t have a view on whether S3 or S4 classes should be used, but probably S4 classes should be aimed for.

For a fit from a distribution, the class could be called `distfit`.

A fit for a particular distribution would add the distribution name: `hyperbfit, distfit` with S3 methods.

For S4 methods the class `distfit` would be extended.

Similar ideas would be used for tests: a Kolomogorov-Smirnov test would have class `kstest, htest` with S3 methods.
I don’t have a view on whether S3 or S4 classes should be used, but probably S4 classes should be aimed for.

For a fit from a distribution, the class could be called `distfit`.

A fit for a particular distribution would add the distribution name: `hyperbfit, distfit` with S3 methods.

For S4 methods the class `distfit` would be extended.

Similar ideas would be used for tests: a Kolomogorov-Smirnov test would have class `kstest, htest` with S3 methods.
Testing

- Testing is vital to quality of software and developers should provide test data and code
- Firstly test parameter sets should be provided, which cover the range of the parameter set
- Unit tests should be provided for all functions: the RUnit package supports this
- The distributions in Rmetrics have tests of this type, although further development seems warranted
- The Massart inequality seems ideal to use in testing
Testing is vital to quality of software and developers should provide test data and code. Firstly test parameter sets should be provided, which cover the range of the parameter set. Unit tests should be provided for all functions: the RUnit package supports this. The distributions in Rmetrics have tests of this type, although further development seems warranted. The Massart inequality seems ideal to use in testing.
Testing

- Testing is vital to quality of software and developers should provide test data and code
- Firstly test parameter sets should be provided, which cover the range of the parameter set
- Unit tests should be provided for all functions: the RUnit package supports this
- The distributions in Rmetrics have tests of this type, although further development seems warranted
- The Massart inequality seems ideal to use in testing
Testing is vital to quality of software and developers should provide test data and code.

Firstly test parameter sets should be provided, which cover the range of the parameter set.

Unit tests should be provided for all functions: the RUnit package supports this.

The distributions in Rmetrics have tests of this type, although further development seems warranted.

The Massart inequality seems ideal to use in testing.
Testing is vital to quality of software and developers should provide test data and code

Firstly test parameter sets should be provided, which cover the range of the parameter set

Unit tests should be provided for all functions: the RUnit package supports this

The distributions in Rmetrics have tests of this type, although further development seems warranted

The Massart inequality seems ideal to use in testing
Final Thoughts

- The `distr` package is an object-oriented implementation of distributions
- It facilitates operations on distributions such as convolutions
- It uses sampling for calculation of moments and distribution functions
- The package `VarianceGamma` has been designed and implemented using these ideas
- Implementing the logarithm options of the dpq functions is quite difficult for the distributions in which I am interested
The `distr` package is an object-oriented implementation of distributions. It facilitates operations on distributions such as convolutions. It uses sampling for calculation of moments and distribution functions. The package `VarianceGamma` has been designed and implemented using these ideas. Implementing the logarithm options of the dpq functions is quite difficult for the distributions in which I am interested.
Final Thoughts

- The `distr` package is an object-oriented implementation of distributions.
- It facilitates operations on distributions such as convolutions.
- It uses sampling for calculation of moments and distribution functions.
- The package `VarianceGamma` has been designed and implemented using these ideas.
- Implementing the logarithm options of the dpq functions is quite difficult for the distributions in which I am interested.
Final Thoughts

- The `distr` package is an object-oriented implementation of distributions.
- It facilitates operations on distributions such as convolutions.
- It uses sampling for calculation of moments and distribution functions.
- The package `VarianceGamma` has been designed and implemented using these ideas.
- Implementing the logarithm options of the `dpq` functions is quite difficult for the distributions in which I am interested.
Final Thoughts

- The **distr** package is an object-oriented implementation of distributions
- It facilitates operations on distributions such as convolutions
- It uses sampling for calculation of moments and distribution functions
- The package **VarianceGamma** has been designed and implemented using these ideas
- Implementing the logarithm options of the dpq functions is quite difficult for the distributions in which I am interested