Package ‘binfunest’

October 12, 2022

Type Package

Title Estimates Parameters of Functions Driving Binomial Random Variables

Version 0.1.0

Maintainer Philip Shea <philshea@gmail.com>

Description Provides maximum likelihood estimates of the performance parameters that drive a binomial distribution of observed errors, and takes full advantage of zero error observations. High performance communications systems typically have inherent noise sources and other performance limitations that need to be estimated. Measurements made at high signal to noise ratios typically result in zero errors due to limitation in available measurement time. Package includes theoretical performance functions for common modulation schemes (Proakis, ``Digital Communications'' (1995, <ISBN:0-07-051726-6>)), polarization shifted QPSK (Agrell & Karlsson (2009, <DOI:10.1109/JLT.2009.2029064>)), and utility functions to work with the performance functions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.2.1

Config/testthat/edition 3

Imports pracma, stats, stats4

URL https://github.com/PhilShea/binfunest

BugReports https://github.com/PhilShea/binfunest/issues

Depends R (>= 2.10)

NeedsCompilation no

Author Philip Shea [aut, cre] (<https://orcid.org/0000-0001-5460-9987>)

Repository CRAN

Date/Publication 2022-09-09 07:32:57 UTC

R topics documented:

`B2BConvert` *B2BConvert Converts a function of SNR into one of SNR, B2B, and Offset.*

Description

Creates a function `f( -dB( undB( -s) + undB( -B2B)) - offset)`

Usage

```
B2BConvert(f)
```

Arguments

`f`	A function of a single argument `f( s)`.

Details

Note that all quantities are assumed to be in Decibels.

Value

A function of three arguments `f( s, B2B, offset)`..

Examples

```
QPSKdB.B2B <- B2BConvert( QPSKdB)
```

`BERDFc`	*An example* `BERDF` *dataframe created by* `simsigs()`*, a function in a forthcoming package* `coherent`.

Description

`BERDF` is a standard `R` data frame created by the `simsigs()` function in the forthcoming `coherent` package. The observations have been `condensed`

Usage

```
BERDFc
```

Format

A dataframe with the following fields:

Name Name of constellation used to create the record.

SNR The SNR in Decibles of the observation.

Bps The number of bits per symbol. The number of bits in a simulation run is `Bps * N`

NoisePower The actual noise power in the simulation run. Since the noise is randomly generated, this is a stochastic item.

N The number of symbols in the simulation run.

SER The number of symbols errors observed in the simulation run.

BER The number of bit errors observed in the simulation run.

`mleB2B`	*mleB2B Estimates Back-to-Back "Q" and Offsets to a bit error rate function.*

Description

Bit error counts modeled as independent binary decisions result in a log-likelihood dependent on the bit error probability. This function inserts the supplied bit error probability function into the binomial log-likelihood function, and passes that to stats4::mle, which ultimately calls stats::optim. The function will optimize a binomial probability of the form $r = N * P(x_1, x_2, ..., x_n, a_1, a_2, ... a_k)$, where the x_i are variables from `data`, and the a_j are parameters to be estimated.

Usage

```
mleB2B(data = NULL, Errors, N, f, fparms, start, method = "Nelder-Mead", ...)
```

Arguments

- `data` — a data frame or list with named components. If a list, each component must be the same length (just like a data frame). This is not checked, so usual rules of recycling will apply. Partial matching not performed, so you must use full column names.
- `Errors` — A vector of error counts, or a string identifying a column of `data` from which to draw the error counts
- `N` — A single number, or a vector of the same length as `data`, or a string identifying a column of `data` specifying the number of trials used to measure the error counts in `Errors`. If a single number, then that number is used as the number of trials for all error counts.
- `f` — A function that predicts the probability of errors.
- `fparms` — a list of named components that are the arguments of `f`. Each component can be a string, a single number, or a vector. If a string that names a column of `data`, that column will be used, otherwise the string will be passed to `f`. Note the potential for errors if a column name was misspelled. A single number or vector will be passed to `f`. Between `fparms`, `start`, and function defaults, all parameters that need to be supplied to `f` should be specified, and (except for defaults) not duplicated.
- `start` — Named list of initial values for the parameters of `f` to be estimated.
- `method` — Optimization method. See `stats::optim()`.
- `...` — Optional arguments to be passed to mle.

Details

The function estimates the parameters identified in `start` in the constructed call to `f`. For a function `f` of the form `fun( SNR, x2, x3, B2B, offset)` A call of the form

```
mleB2B( data=df, Errors="r", N="trials", f=fun, fparm=list( SNR="s", x2=1, x3="noise"),
start=list(B2B=1, offset=2))
```

will construct a call to `mle` of the form:

```
mle( minuslogl=ll, start=start, nobs=length( Errors), method=method)
```

where the function `ll` is defined as

```
ll <- function( a, b) -sum( dbinom( df$r, df$n, fun( SNR=df$s, x2=1, x3=df$noise, B2B=B2B,
offset=offset), log=TRUE))
```

Value

An object of class stats4::mle with the parameters identified in `start` estimated.

See Also

`stats4::mle()`, `stats::optim()`

Examples

```
QPSKdB.B2B <- B2BConvert( QPSKdB)
O1 <- 3
B1 <- 16
s <- 0:20
N <- 1000000
r <- rbinom( length( s), N, QPSKdB.B2B( s, B1, O1))
df <- data.frame( Errors=r, SNR=s, N=N)
llsb2 <- function( b2b, offset)
         -sum( dbinom( r, N, QPSKdB.B2B( s, b2b, offset), log=TRUE))
mle1 <- stats4::mle( llsb2, start=c( b2b=20, offset=0), nobs=length(s),
                     method="Nelder-Mead")
est1 <-  mleB2B( data=df, Errors="Errors", N=N, f=QPSKdB.B2B,
                 fparms=list( x="SNR"), start=c(b2b=20, offset=0))
```

`Theoretical` *Theoretical error rate functions*

Description

Functions to calculate the theoretical performance of common modulation formats. Includes the functions `dB (x)` (returns `10log10(x)`), `undB(x)` (reverses `dB(x)`), `Q_( x)` (Markum's Q function), and `Q_Inv(x)` (returns the SNR in Decibels to get probability `x`). Also includes `mod_Inv`, which returns the SNR required for a the function `f` to reach the supplied BER (bit error rate, or bit error probability).

Usage

```
is.wholenumber(x, tol = sqrt(.Machine$double.eps))

dB(x)

undB(x)

Q_(x)

Q_Inv(perr)

QPSKdB(x)

DQPSKdB(x)

DQPSKDDdB(x)

PSQPSKdB(x)

MPSKdB(x, M)
```

```
MPSKdB.8(x)

QAMdB.8.star(x)

QAMdB(x, M)

QAMdB.16(x)

mod_Inv(f, perr, guess = Q_Inv(perr))

mod_InvV(f, pv, offset = 0)
```

Arguments

`x`	a real number
`tol`	the tolerance to test x with.
`perr`	a probability of a bit error.
`M`	The integer number of symbols > 4.
`f`	a function (usually a BER function).
`guess`	a guess for the `perr` (the default usually works).
`pv`	a vector of BERs.
`offset`	an offset in Decibels for guesses in `mod_InvV`.

Details

The rest of the functions return the probability of a bit error given the SNR in Decibels.

- `QPSKdB` is Quadrature Phase shift keyed: two bits per symbol.
- `DQPSK` is differentially detected differentially coded QPSK.
- `DQPSKDDdB` is differentially detected differential QPSK (coherently detected but differentially decoded. See `DQPSK` above.
- `PSQPSKdB` is polarization-shifted QPSK: it is dual pole, but only one pole is active at any one time, thus supplying three bits per symbol. (See Agrell & Karlsson (2009, DOI:10.1109/JLT.2009.2029064)).
- `MPSKdB(x, M)` is generic M-ary phase shift keying of `M` points in a circle.
- `MPSKdB.8` simply returns `MPSKdB(x, 8)`
- `QAMdB.8.star` is the optimal star configuration of 8-ary Quadrature Amplitude Modulation (QAM), such that the legs are at ± 1 and $\pm(1+\sqrt{3})$.
- `QAMdB(x, M)` is generic rectangular QAM constellation of `M` points.
- `QAMdB.16` Returns the BER for the rectangular QAM constellation according to Proakis Eq. 5-2-80.
- `mod_Inv` will take a function `f(x)` and return the x such that `f(x)==perr` but it does this based on the `log( f(x))` and the `log( perr)`, so `f(x)>0`.
- `mod_InvV` is a vectorized version (give it a vector of BERs and it returns a vector of SNRs).

Value

`is.wholenumber(x)` returns `TRUE` if `c-round(x) < tol`.

`dB(x)` returns `10*log10(x)`

`undB(x)` returns `10^(x/10)`

`Q_Inv(x)` returns `2*dB( -qnorm(x))`, which is the SNR (in Decibels) required to get a probability of error of x. Q_Inv(Q_(undB(x/2))) = x and Q_(undB(Q_Inv(x)/2))=x

`mod_Inv( f, x)` returns a list with the SNR in Decibels to reach the BER `perr` such that `f( mod_Inv( f, x)$x) = x`. The returned list has elements `$x` as the SNR and `$fval` as the function value.

See Also

`pracma::fzero()`

Examples

```
dB( 10) # == 10
undB( 20) # == 100
Q_Inv( Q_( undB( 10/2))) # = 10
Q_( undB( Q_Inv( 0.001)/2)) # = 0.001

mod_Inv( QPSKdB, QPSKdB( 7)) # yields 7

mod_InvV(QPSKdB, QPSKdB(c(6,7)))
```

Index