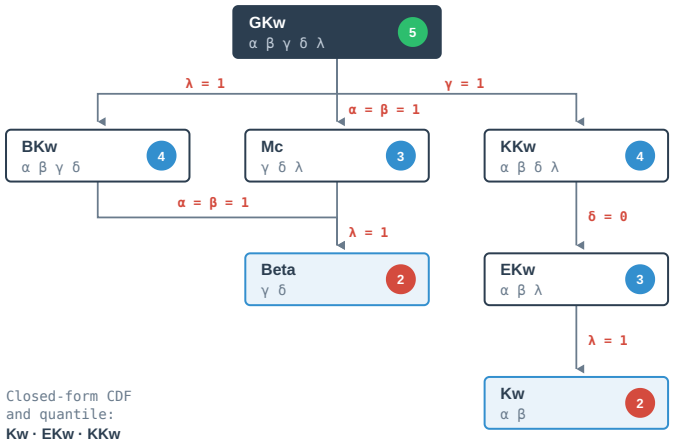
gkwdist :: cheat sheet

Seven **nested** distributions for bounded continuous data on the open interval (0, 1) — proportions, rates, shares, indices. Densities, CDFs, quantiles, random generation, and **analytical** log-likelihood, score and Hessian, all in C++.

version 1.1.7
evandeilton.github.io/gkwdist
`install.packages("gkwdist")`

THE FAMILY one distribution, seven models

Every model below is **GKw with parameters held fixed**. Follow an edge to fix a parameter and drop to a simpler model; the number in the badge is how many free parameters remain.



Uniform is `GKw(1, 1, 1, 0, 1)` — the neutral value of every parameter. **Kw** and **Beta** are *not* nested in one another, and neither are **EKw**/**Mc** or **BKw**/**KKw**.

WHAT EACH PARAMETER DOES

α > 0	Exponent on x. Governs the density near **0**.
β > 0	Exponent on 1−x^α. Governs the density near **1**.
γ > 0 **δ ≥ 0**	The `Beta(γ, δ+1)` generator that reshapes the Kumaraswamy core. **δ = 0** — not 1 — is the neutral value.
λ > 0	Exponentiation (power) parameter.

Boundary rule — what happens at x = 0 and x = 1

Each endpoint is decided by the sign of a single exponent. `d*()` returns the **limit**, as base R does, not 0.

AT	EXPONENT	> 0	= 0	< 0
x=0	α·γ·λ − 1	0	const	**Inf**
x=1	β·(δ+1) − 1	0	const	**Inf**

Each sub-family inherits the rule through its own fixed values: for `Kw(α,β)` they read α−1 and β−1. `dkw(0, 0.5, 1)` is `Inf`; `dbeta_(1, 2, 0)` is 2.

49 FUNCTIONS, ONE NAMING RULE

prefix + family code. Row = your model, column = the job. blue distribution API red likelihood API.

FAMILY (code)	D PDF	P CDF	Q QUANT	R RAND	LL −LOGL	GR −SCORE	HS −HESS
Gen. Kumaraswamy gkw · 5p	dgkw	pgkw	qgkw	rgkw	llgkw	grgkw	hsgkw
Beta-Kumaraswamy bkw · 4p	dbkw	pbkw	qbkw	rbkw	llbkw	grbkw	hsbkw
Kum.-Kumaraswamy kkw · 4p	dkkw	pkkw	qkkw	rkkw	llkkw	grkkw	hskkw
Exp. Kumaraswamy ekw · 3p	dekw	pekw	qekw	rekw	llekw	grekw	hsekw
McDonald mc · 3p	dmc	pmc	qmc	rmc	llmc	grmc	hsmc
Kumaraswamy kw · 2p	dkw	pkw	qkw	rkw	llkw	grkw	hskw
Beta beta_ · 2p	dbeta_	pbeta_	qbeta_	rbeta_	llbeta	grbeta	hsbeta

Mind the underscore. Only the Beta row is irregular: `d/p/q/r` keep the trailing `_` so they do not mask `stats::dbeta` — the likelihood trio does not. `llbeta`, **not** `llbeta_`.

DISTRIBUTION FUNCTIONS d · p · q · r

```
dgkw(x, alpha, beta, gamma, delta, lambda, log = FALSE)
pgkw(q, ...,  lower.tail = TRUE, log.p = FALSE)
qgkw(p, ...,  lower.tail = TRUE, log.p = FALSE)
rgkw(n, ...)
```

A sub-family drops the parameters it fixes: `dkw(x, alpha, beta, log = FALSE)`.

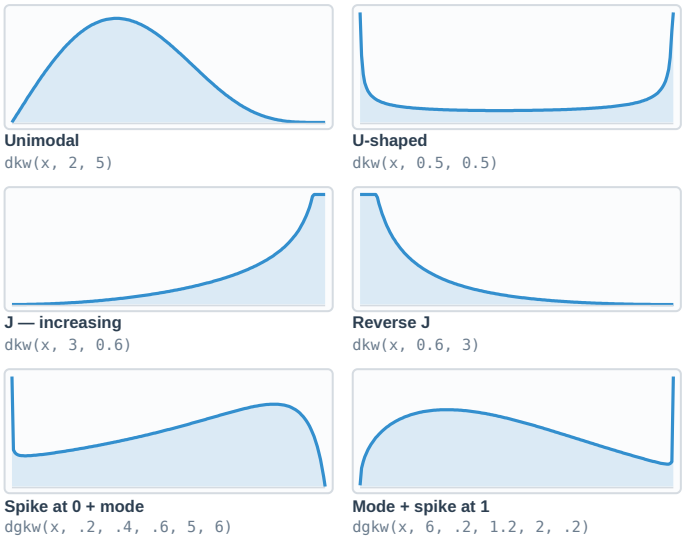
`log`, **`log.p`**	Work on the log scale. Use them instead of `log(d…)` near the boundaries.
`lower.tail`	`FALSE` gives `P(X > q)`, computed directly rather than as `1 − p`.
`n`	`length(n) > 1` means `length(n)` draws; `n = 0` gives `numeric(0)`.

The base-R contract these keep

- **Recycling.** x and every parameter are recycled to a common length: `dkw(c(.3,.6), c(1,2,3,4), 2)` gives 4 values.
- **Attributes.** `dim`, `dimnames` and `names` of the first argument survive when the lengths agree, as in `stats::dbeta`.
- **Outside [0,1]** the density is 0; at the two endpoints it is the limit (box, left).
- **An invalid parameter is an error** — `α ≤ 0`, `δ < 0`, `NA` all `stop()`. They do not return `NaN`.
- **Defaults are the neutral values**, so `dgkw(x)` is the uniform density.

SHAPES THE FAMILY REACHES

Real output. The boundary rule (column 1) predicts each end.



What it will not do. At most **one interior mode** — a second peak is always a divergence at 0 or 1, as in the bottom row. Two interior humps need a mixture, not a bigger GKw.

Where this Beta differs from `stats`

δ = 0 is neutral, so the second shape sits at δ + 1:

```
dbeta_(x, gamma, delta) == stats::dbeta(x, gamma, delta + 1)
```

That shape is therefore always **≥ 1**: this Beta **cannot be U-shaped**. Use Kw, EKw or GKw for a bathtub on (0,1).

PICK A STARTING FAMILY

YOUR DATA	START WITH
Single hump, roughly symmetric	**Beta** 2p
Single hump, skewed	**Kw** 2p — closed-form CDF & quantile
Monotone (J or reverse J)	**Kw** or **Beta**
U-shaped / mass at both ends	**Kw**, **EKw** (α, β < 1)
Heavy tail on one side only	**EKw** or **Mc** 3p
An explicit power transform	**Mc** λ is that power
Stubborn residual shape	**BKw**, **KKw**, then **GKw**

Fit up, not down. Add a parameter only when a likelihood-ratio test (page 2) says it earns its place.

FIT BY MAXIMUM LIKELIHOOD

Objective, gradient and Hessian drop straight into `stats::optim`.

```
x <- rekw(2000, 2, 3, 1.5)        # data on (0,1)

start <- gkwgetstartvalues(x, family = "ekw")

fit <- optim(start,
             fn     = llekw,   # −logL
             gr     = grekw,   # −score
             data   = x,       # passed through
             method = "BFGS",
             hessian = TRUE)

est <- fit$par                    # MLE
logLik <- -fit$value              # note the sign
se  <- sqrt(diag(solve(fit$hessian)))
ci  <- est + outer(se, c(-1.96, 1.96))
```

`fn` is already the *negative* log-likelihood, so `optim`'s default minimisation is the right direction and `fit$hessian` is the **observed information** — invert it directly, no sign flip.

For the exact information matrix, use the analytical Hessian:

```
se <- sqrt(diag(solve( hsekw(est, x) )))
```

They agree to ~5e−7 relative: `optim`'s is a finite difference.

STARTING VALUES

```
gkwgetstartvalues(x, family = "gkw", n_starts = 5)
```

Method of moments by Nelder–Mead, returning a **named** vector already in the `par` order that family's `ll*` expects. `family` is one of `"gkw" "bkw" "kkw" "ekw" "mc" "kw" "beta"` (case-insensitive).

- **Deterministic.** `set.seed()` has no effect and it never touches `.Random.seed`. Widen the search with `n_starts`; there is no seed.
- Four fixed starting points are always used, so `n_starts < 4` behaves as 4; cost grows linearly and the objective is non-increasing in `n_starts`.
- Results are clipped to a box — α, β ∈ (0.1, 50), γ ∈ (0.1, 10) but (0.1, 50) for `"beta"`, δ ∈ (0.01, 10), λ ∈ (0.1, 20) — so a value sitting exactly on a bound is a **hint that the box bit**, not an estimate.
- Values outside (0,1) are truncated **with a warning**. Treat that warning as a scale problem in the data — a 0–100 percentage — not as noise.

RETURN VALUES UNDER FAILURE

SITUATION	LL*	GR* / HS*
Bad `par` (≤ 0, wrong length, NA)	**`Inf`**	**`NaN`**
Data outside (0,1)	**`Inf`** + warning	**`NaN`**

An infinite objective gives the optimiser no direction — it sits there reporting convergence. **Screen first:** `stopifnot(all(x > 0 & x < 1))`. An exact 0 or 1 is the usual culprit; squeeze it in with `(x*(n-1) + 0.5)/n`.

LIKELIHOOD FUNCTIONS ll · gr · hs

```
llgkw(par, data)   → scalar   −ℓ(θ)
grgkw(par, data)   → vector   −∇ℓ(θ)
hsgkw(par, data)   → matrix   −∇²ℓ(θ)
```

All three carry a **minus sign**: minimise `ll*`, and `hs*(est, x)` *is* the observed information. `par` is a plain numeric vector — the names are ignored, only the **order** matters.

⚠ `par` order differs by family

The vector is **positional**. A mis-ordered `par` is not an error — it silently fits the wrong model.

FAMILY	LEN	PAR = C(…)
`gkw`	5	alpha, beta, gamma, delta, lambda
`bkw`	4	alpha, beta, gamma, delta — no λ
`kkw`	4	alpha, beta, delta, lambda — no γ
`ekw`	3	alpha, beta, lambda
`mc`	3	gamma, delta, lambda
`kw`	2	alpha, beta
`beta`	2	gamma, delta

`bkw` and **`kkw`** are the trap: both take four, and they are not the same four. Let `gkwgetstartvalues()` build the vector and the order comes out right for free.

WHY ANALYTICAL DERIVATIVES

One pass over the data instead of a finite-difference stencil per parameter. Measured at `n = 20 000`, GKw, against `numDeriv`:

	GR*()	NUMDERIV	
score	2.8 ms	61 ms	**≈22×**
Hessian	4.0 ms	177 ms	**≈44×**

Agreement with `numDeriv` to ~9e−6 absolute — the difference is the finite-difference truncation error, not the analytical form.

FIT ALL SEVEN, THEN RANK

```
fam <- c("gkw","bkw","kkw","ekw","mc","kw","beta")
ll  <- list(llgkw, llbkw, llkkw, llekw, llmc, llkw, llbeta)
gr  <- list(grgkw, grbkw, grkkw, grekw, grmc, grkw, grbeta)

tab <- Map(function(f, l, g) {
  s <- gkwgetstartvalues(x, family = f)
  o <- optim(s, l, g, data = x, method = "BFGS")
  c(k = length(s), AIC = 2*o$value + 2*length(s))
}, fam, ll, gr)

tab <- do.call(rbind, tab); tab[order(tab[, "AIC"]), ]
```

The three lists run in the same order as page 1's matrix, and `gkwgetstartvalues()` returns each `par` already in the order its family expects — nothing is aligned by hand.

COMPARE NESTED MODELS

Comparable by likelihood-ratio test only if one is the other with parameters fixed; `df` counts them.

FULL	REDUCED	CONSTRAINT	DF
gkw	bkw	λ = 1	1
gkw	kkw	γ = 1	1
gkw	mc	α = β = 1	2
bkw	beta	α = β = 1	2
kkw	ekw	δ = 0 †	1
mc	beta	λ = 1	1
ekw	kw	λ = 1	1
gkw	ekw	γ = 1, δ = 0 †	2
gkw	kw	γ = 1, δ = 0, λ = 1 †	3
gkw	beta	α = β = λ = 1	3

```
lrt <- 2 * (fit_red$value - fit_full$value)
pchisq(lrt, df, lower.tail = FALSE)
```

† δ = 0 is on the **boundary**, so χ²(df) is conservative: the null is 0.5·χ²(df−1) + 0.5·χ²(df).

Not nested: kw vs beta, ekw vs mc, bkw vs kkw. Compare those with AIC/BIC only.

INFORMATION CRITERIA

```
k <- length(fit$par);  ll <- -fit$value
AIC <- -2*ll + 2*k
BIC <- -2*ll + k*log(length(x))
```

Lower is better; a gap under ~2 is not evidence. All seven are fitted to the *same* `x`, so they compare directly.

CHECK THE FIT

```
# PIT residuals — uniform if the model holds
u <- pekw(x, est[1], est[2], est[3])
ks.test(u, "punif")

# Q-Q against the fitted quantiles
q <- qekw(ppoints(length(x)), est[1], est[2], est[3])
qqplot(q, sort(x)); abline(0, 1, col = 2)
```

Estimating from the same `x` makes the KS p-value optimistic.

Traps worth a second look

- δ = 0, not 1, is neutral — `dgkw(x, 1,1,1,0,1)` is uniform.
- `llbeta` has no underscore, `dbeta_` does — and `dbeta_(x, g, d)` is `stats::dbeta(x, g, d+1)`.
- `ll*` returns `+Inf`, never an error, on a contaminated sample.
- `gkwgetstartvalues()` ignores `set.seed()`.
- `%>%` is re-exported but **deprecated** — take it from magrittr, or use `|>`.

Carrasco, Ferrari & Cordeiro (2010) · Jones (2009) · Kumaraswamy (1980) · McDonald (1984)