Exploratory Analysis with anyFit : : CHEATSHEET

DATA LOADING

delim2xts(file_path, time_zone, date_index = 1, strict_step = TRUE, delim = "\t", skip_rows = 0, col_names = TRUE, no_value = NA, exc_leaps = TRUE, save_Xts = FALSE, filename = NA)

Convert delimited files to xts objects. Automatically detects gaps if strict_step = TRUE. *delim2xts("data.csv", time_zone = "UTC", delim = ",")*

nc2xts(filename, varname, shapefile = NA, country = NA, continent = NA, xlim = NA, ylim = NA, projection = "+proj=longlat +datum=WGS84")

Load NetCDF gridded data with optional spatial masking by country, continent, shapefile, or bounding box (xlim/ylim). Returns an sxts object. Reads directly via ncdf4 (no raster intermediate) for speed and low memory; CRS auto-detected from file metadata. *nc2xts("era5.nc", "tp", country = "France")*

nc2xts_nn(filename, varname, coords)

Extract point time series from one or more NetCDF files at specific coordinates using nearest-neighbour (no interpolation). Returns a plain xts via indexed hyperslab read; points outside the grid yield all-NA columns. *nc2xts_nn("data.nc", "rr", coords = data.frame(x = 14.5, y = 62.6))*

TIME SERIES MANIPULATION

aggregate_xts(ts, periods = NA, FUN = 'mean', period_multiplier = NA, pstart = NA, pend = NA, season_end = NA, mseason_title = NA)

Aggregate time series to different scales. Supports custom seasons. *aggregate_xts(ts, c("months", "years"), FUN = 'sum')*

check_missing(data, periods, plot = TRUE, group_months = FALSE)

Check and visualize missing values by period. *check_missing(data, c("months","years"))*

plot_missing(ts)

Quick visual check for missing value positions. *plot_missing(ts)*

normalise_xts(ts, dist_period = 'monthly', ignore_zeros = FALSE)

Transform data to standard normal distribution. *normalise_xts(ts, 'monthly')*

period_apply_nc(data = NULL, filename = NA, varname = NA, period = "months", period_multiplier = 1, FUN = "mean", ...)

Apply aggregation function over time periods to an sxts object, a raster file, or a NetCDF path. Returns an sxts (preserving coords and projection) when the input is an sxts. *period_apply_nc(data, "years", FUN = "max")*

DISTRIBUTION FITTING

Single Distribution Fitting

fitlm_[distribution](x, ignore_zeros = FALSE, zero_threshold = 0.01)

Fit a specific distribution using L-moments. Returns parameters, theoretical L-moments, sample L-moments, and GoF statistics.

Available distribution fitting functions:

`fitlm_exp()`, `fitlm_rayleigh()`, `fitlm_gamma()`, `fitlm_gamma3()`, `fitlm_norm()`, `fitlm_lognorm()`, `fitlm_genlogi()`, `fitlm_weibull()`, `fitlm_gumbel()`, `fitlm_gev()`, `fitlm_GPD()`, `fitlm_gengamma()`, `fitlm_gengamma_loc()`, `fitlm_burr()`, `fitlm_dagum()`, `fitlm_expweibull()`

Distribution function quartet (d/p/q/r):

For each distribution, standard R functions are available:
`d[dist]()` density, `p[dist]()` CDF, `q[dist]()` quantile, `r[dist]()` random generation
e.g., dgev(), pgev(), qgev(), rgev()
The Generalized Pareto quartet is now available: `dgpd()`, `pgpd()`, `qgpd()`, `rgpd()`

Multiple Distribution Fitting

fitlm_multi(ts, candidates, ignore_zeros = FALSE, zero_threshold = 0.01, diagnostic_plots = TRUE, order = NULL)

Fit multiple distributions and compare with diagnostic plots. `order` = named list mapping a numerically-fitted candidate (gengamma, gengamma_loc, burr, dagum, expweibull) to its L-moment order vector, e.g. `list(gengamma = 1:5)`. *fitlm_multi(ts, list('exp','gamma3','gev'))*

fitlm_monthly(ts, candidates, ignore_zeros = FALSE, zero_threshold = 0.01, nrow = 4, ncol = 3, order = NULL)

Fit distributions by month for cyclo-stationary analysis. Accepts `order` for numerically-fitted candidates. *fitlm_monthly(ts, list('exp','weibull','gamma3'))*

fitlm_nxts(ts, candidates, nrow = 5, ncol = 4, ignore_zeros = FALSE, zero_threshold = 0.01, parallel = FALSE, ncores = 2, shared_memory = TRUE, diagnostic_plots = TRUE, order = NULL)

Fit distributions to multiple time series with panel output. Accepts `order` for numerically-fitted candidates. *fitlm_nxts(data, candidates, parallel = TRUE)*

fitlm_nc(data = NULL, filename = NA, varname = NA, candidates = 'norm', ignore_zeros = FALSE, zero_threshold = 0.01, parallel = FALSE, ncores = 2, shared_memory = TRUE, order = NULL, ...)

BASIC STATISTICS

basic_stats(ts, pstart = NA, pend = NA, show_label = TRUE, label_prefix = 'Station', show_table = TRUE, xpos_label = 0.1, ypos_label = 0.95, xpos_table = 0.1, ypos_table = 0.15, nbins = 30, ignore_zeros = FALSE, zero_threshold = 0.01, plot = FALSE)

Calculate comprehensive statistics and generate combined plot (timeseries, PDF, ECDF, ACF). *basic_stats(ts, pstart = '2002', pend = '2005', ignore_zeros = TRUE)*

basic_stats_nc(data = NULL, filename = NA, varname = NA, ignore_zeros = FALSE, zero_threshold = 0.01, ...)

Calculate statistics for each grid cell of gridded data (an sxts object, raster file, or NetCDF path). *basic_stats_nc(data, ignore_zeros = TRUE)*

lmom_stats(ts, ignore_zeros = FALSE, zero_threshold = 0.01)

Calculate L-moment statistics for multiple time series. *lmom_stats(data, ignore_zeros = TRUE)*

period_stats(ts, period = 'months', period_multiplier = 1)

Calculate period-based statistics (min, max, mean, variance, L-moments, quantiles). *period_stats(ts, "months", period_multiplier = 3)*

monthly_stats(ts, aggregated = FALSE, FUN = 'mean', ignore_zeros = FALSE)

Calculate and visualize monthly statistics including lag-1 correlations. *monthly_stats(ts, aggregated = TRUE, FUN = "sum")*

monthly_stats_nc(data = NULL, filename = NA, varname = NA, ignore_zeros = FALSE, zero_threshold = 0.01, parallel = FALSE, ncores = 2, ...)

Per-month gridded statistics from a NetCDF file or an sxts object; same per-cell output as basic_stats_nc for each calendar month. *monthly_stats_nc(data, parallel = TRUE, ncores = 4)*

AUTOCORRELATION FUNCTIONS

fit_ACF(ts, lag_max, type = list('CAS','HK','SRD'), ignore_zeros = FALSE, zero_threshold = 0.01)

Fit theoretical ACFs to time series data. *fit_ACF(ts, lag_max = 10)*

fit_ACF_monthly(ts, lag, type = list('CAS','HK','SRD'), nrow = 4, ncol = 3)

Fit ACFs by month with panel plot output. *fit_ACF_monthly(ts, lag = 10)*

Theoretical ACF Functions:

- `CAS_ACF(kappa, beta, lag_max)` - Cauchy-type autocorrelation structure
- `HK_ACF(H, lag_max)` - Hurst-Kolmogorov (long-term persistence)
- `SRD_ACF(kappa, lag_max)` - Short Range Dependence (exponential)

VISUALIZATION

nc_ggplot(data, title = FALSE, legend.title = NA, common_legend = FALSE, viridis.option = "viridis", ...)

Fit distributions to gridded data (an sxts object, raster file, or NetCDF path). Returns rasters of parameters, L-moments, and GoF statistics. *fitlm_nc(data, candidates = "gev", parallel = TRUE)*

fitlm_monthly_nc(data = NULL, filename = NA, varname = NA, candidates = "norm", ignore_zeros = FALSE, zero_threshold = 0.01, parallel = FALSE, ncores = 2, shared_memory = TRUE, parallel_by = c("cells","months"), order = NULL, ...)

Fit candidate distributions per calendar month to gridded data (NetCDF or sxts); per-month equivalent of fitlm_nc. *fitlm_monthly_nc(data, candidates = "gamma3", parallel = TRUE)*

Diagnostic Tools

fit_diagnostics(ts, dist = 'norm', params, ignore_zeros = FALSE, zero_threshold = 0.01)

Generate Q-Q and P-P plots with GoF statistics. *fit_diagnostics(ts, dist = 'gamma3', params = fit$Param)*

monthly_fdiagnostics(ts, distr, params, ignore_zeros = FALSE, zero_threshold = 0.01, nrow = 3, ncol = 4)

Generate diagnostic plots by month. *monthly_fdiagnostics(ts, 'gamma3', monthly_fits$params_monthly$gamma3)*

LRatio_check(ts_lmoms)

Check if sampled L-ratios fall within theoretical distribution support. *LRatio_check(lmom_stats(data))*

GRIDDED DATA WORKFLOW

Common Gridded Data Workflow:

1. Load data (returns an sxts): `data <- nc2xts("file.nc", "varname", country = "Italy")`
2. Aggregate temporally (sxts in, sxts out): `annual <- period_apply_nc(data, "years", FUN = "sum")`
3. Calculate statistics: `stats <- basic_stats_nc(annual)`
4. Visualize: `nc_ggplot(stats[["Mean"]], viridis.option = "turbo")`
5. Fit distributions: `fits <- fitlm_nc(annual, candidates = "gev", parallel = TRUE)`

Performance Tip: Enable parallel processing with `parallel = TRUE` and specify `ncores` for large gridded datasets.

KEY CONCEPTS

L-Moments: Linear combinations of ordered data values that provide robust estimates for highly skewed distributions. Less sensitive to outliers than conventional moments. Advantageous when data exhibit autocorrelation.

Create ggplot visualizations of an sxts object or any raster (Layer/Stack/Brick), including the statistic/parameter outputs of basic_stats_nc and fitlm_nc. Date-like layer names are used as titles. *nc_ggplot(data, viridis.option = "turbo", common_legend = TRUE)*

monthly_boxplots(ts, palette = 'Set1', ignore_zeros = FALSE, zero_threshold = 0.01)
Generate monthly boxplots for inter-annual variability. *monthly_boxplots(ts, palette = 'Set3', ignore_zeros = TRUE)*

monthly_violins(ts, palette = 'Set3', ignore_zeros = FALSE, zero_threshold = 0.01)
Generate monthly violin plots. *monthly_violins(ts, palette = 'Set3')*

monthly_ecdf(ts, ignore_zeros = FALSE, zero_threshold = 0.01)
Generate monthly ECDF plots. *monthly_ecdf(ts, ignore_zeros = TRUE)*

ridge_plots(ts, palette = 'Set3', ignore_zeros = FALSE, zero_threshold = 0.01)
Generate ridge plots for distributions across stations or months. Returns plot_all and plot_monthly. *ridge_plots(data[,1:5], ignore_zeros = TRUE)*

correl_plots(x, y, check_common = TRUE, ignore_zeros = FALSE, zero_threshold = 0.01)
Generate correlation plots: scatter (Pearson), copula (Spearman), and normal space (semi-correlations). *correl_plots(data[,1], data[,4], ignore_zeros = TRUE)*

Intermittency: The presence of zero values in time series (e.g., precipitation). anyFit computes probability dry (Pdr) and provides the `ignore_zeros` parameter to handle these cases.

Common Arguments:

- `ignore_zeros = TRUE` - Exclude zeros when calculating statistics/fitting
- `zero_threshold = 0.01` - Values below this are considered zero
- `parallel = TRUE` - Enable parallel processing for gridded data
- `ncores = 2` - Number of cores for parallel processing
- `pstart, pend` - Plotting date range for long time series

NAMING CONVENTIONS

Pattern	Meaning	Examples
`source2format`	Data loading	delim2xts, nc2xts
`_nc` suffix	Gridded operations (NetCDF/raster/sxts)	fitlm_nc, basic_stats_nc
`_xts` suffix	Point-based xts operations	aggregate_xts, normalise_xts
`sxts`	Spatial xts (time series + coords)	sxts, sxtsFromRaster
`FromX / fromX`	sxts ↔ raster converters	sxtsFromRaster, rasterFromSxts
`fitlm_` prefix	L-moments fitting	fitlm_gev, fitlm_gamma3
`monthly_` prefix	Monthly-scale analysis	monthly_stats, monthly_boxplots
`d/p/q/r` prefix	Distribution functions	dgev, pgev, qgev, rgev
`_ACF` suffix	Autocorrelation functions	CAS_ACF, fit_ACF
`_nn` suffix	Nearest-neighbour (no interp.)	nc2xts_nn

THE sxts CLASS

The `sxts` (Spatial eXtensible Time Series) class extends `xts`, storing coordinates and projection as attributes. An sxts behaves like an ordinary xts for every temporal operation while carrying its spatial metadata along. Gridded data loaded with `nc2xts()` is returned as an sxts.

Constructors & Converters

sxts(data, order.by, coords = NULL, projection = NULL)
Build an sxts from a data matrix, a time index, and a coords data.frame (x, y). *sxts(mat, order.by = dates, coords = xy, projection = "+proj=longlat +datum=WGS84")*

Accessors & Spatial Operations

Attribute accessors:
`coords(x)` coordinate data.frame, `projection(x)` CRS string, `elements(x)` number of locations, `is.sxts(x)` class check.

sxtsFromRaster(raster, ...)
Convert a raster (Layer/Stack/Brick) to an sxts. *sxtsFromRaster(my_stack)*

rasterFromSxts(obj, ...)
Convert an sxts back to a raster stack (one layer per time step). *rasterFromSxts(my_sxts)*

rbindlist.sxts(sxts_list)
Fast row-bind of a list of sxts objects sharing the same locations; rebuilds a single sxts in one pass. *rbindlist.sxts(list(s1, s2, s3))*

> The `print`, `summary`, `str`, `[` (subset), `lag`, `diff`, and arithmetic (`Ops`) S3 methods are defined and preserve spatial attributes.

mask.sxts(obj, xlim = NA, ylim = NA, shapefile_name = NA, mask = NA)
Subset locations by bounding box (xlim/ylim) or polygon (shapefile path or sf/Spatial* object); CRS auto-aligned. *mask.sxts(s, shapefile_name = "basin.shp")*

zonal_stats(x, shapefile = NULL, name_col = NULL, country = NA, continent = NA, FUN = mean, ...)
Aggregate the points within each polygon row-wise (per time step); returns a plain xts with one column per polygon (or per world_data country/continent). *zonal_stats(s, country = "Italy", FUN = mean, na.rm = TRUE)*