Package: Qploidy (via r-universe)

May 21, 2026

Title Estimation of Ploidy and Detection of Aneuploidy Using Genotyping Data

Version 1.5.2

Description Provides functions for estimating ploidy levels and detecting aneuploidy in individuals using allele intensities or allele count data from high-throughput genotyping platforms, including single nucleotide polymorphism (SNP) arrays and sequencing-based technologies. Implements method described in Taniguti et al. (2025) <doi:10.1002/tpg2.70044> an extended version of the 'PennCNV' signal standardization method by Wang et al. (2007) <doi:10.1101/gr.6861907> for higher ploidy levels. Computes B-allele frequencies (BAF), z-scores, and identifies copy number variation patterns.

License AGPL (>= 3)

Depends R (>= 3.6.0)

Imports dplyr, ggplot2, tidyr, vroom, ggpubr, multtest, vcfR, stringr, magrittr

Encoding UTF-8

URL https://github.com/Cristianetaniguti/Qploidy

BugReports https://github.com/Cristianetaniguti/Qploidy/issues

LazyData true

RoxygenNote 7.3.3

Suggests covr, spelling, updog, rmdformats, knitr (>= 1.10), rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-US

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev

Repository https://cristianetaniguti.r-universe.dev

Date/Publication 2026-05-21 20:30:56 UTC

RemoteUrl https://github.com/cristianetaniguti/qploidy

RemoteRef HEAD

RemoteSha b2456f604b9bd54fcffa2d064d5b18135b24e0ce

Contents

`all_resolutions_plots` *Plot graphics for ploidy visual inspection for each resolution*

Description

This function generates and saves plots for visual inspection of ploidy at different resolutions: chromosome, chromosome-arm, and sample levels. It is designed for parallelization purposes and supports customization of centromere positions and chromosome selection.

Usage

```
all_resolutions_plots(
  data_standardized,
  sample,
  ploidy,
  centromeres,
  types_chromosome = c("Ratio_hist", "BAF_hist", "zscore"),
  types_chromosome_arm = c("Ratio_hist", "BAF_hist", "zscore"),
  types_sample = c("Ratio_hist_overall", "BAF_hist_overall"),
  file_name = NULL,
  chr = NULL
)
```

Arguments

`data_standardized`
: An object of class 'qploidy_standardization' containing standardized data for ploidy analysis.

`sample`
: A character string specifying the sample name to be analyzed.

`ploidy`
: A numeric value indicating the expected ploidy of the sample. This parameter is required.

`centromeres`
: A named vector with centromere positions (in base pairs) for each chromosome. The names must match the chromosome IDs in the dataset. This is used for chromosome-arm level resolution.

`types_chromosome`
: A character vector defining the plot types for chromosome-level resolution. Options include: - "het": Plots heterozygous locus counts. - "BAF": Plots B-allele frequency (BAF). - "zscore": Plots z-scores. - "BAF_hist": Plots BAF histograms for each chromosome. - "ratio": Plots raw ratios for each chromosome. Default is c("Ratio_hist", "BAF_hist", "zscore").

`types_chromosome_arm`
: A character vector defining the plot types for chromosome-arm level resolution. Options include: - "het": Plots heterozygous locus counts. - "BAF": Plots B-allele frequency (BAF). - "zscore": Plots z-scores. - "BAF_hist": Plots BAF histograms for each chromosome arm. - "ratio":

	Plots raw ratios for each chromosome arm. Default is c("Ratio_hist", "BAF_hist", "zscore").
`types_sample`	A character vector defining the plot types for sample-level resolution. Options include: - "Ratio_hist_overall": Plots a histogram of raw ratios for the entire genome. - "BAF_hist_overall": Plots a BAF histogram for the entire genome. Default is c("Ratio_hist_overall", "BAF_hist_overall").
`file_name`	A character string defining the output file path and name prefix for the saved plots. The function appends resolution-specific suffixes to this prefix. If NULL, plots are not saved to files.
`chr`	A vector specifying the chromosomes to include in the analysis. If NULL, all chromosomes are included.

Details

The function generates three types of plots:

- **Chromosome-level resolution**: Plots raw ratio, BAF histograms, z-scores, heterozygous locus counts, and BAF for each chromosome. - **Chromosome-arm level resolution**: Similar to chromosome-level but splits data by chromosome arms using centromere positions. - **Sample-level resolution**: Combines all markers in the sample to generate overall raw ratio and BAF histograms.

The plots are saved as PNG files with the following suffixes: - '_res:chromosome.png' - '_res:chromosome_arm.png' - '_res:sample.png'

If 'file_name' is NULL, the plots are not saved to files but are returned in the output list.

Value

A list containing the generated plots for each resolution: - 'chromosome': Plot for chromosome-level resolution. - 'chromosome_arm': Plot for chromosome-arm level resolution (if centromeres are provided). - 'sample': Plot for sample-level resolution.

`area_estimate_ploidy` *Estimate ploidy using area method*

Description

This function estimates ploidy using the area method. It evaluates the number of copies by chromosome, sample, or chromosome arm. Note that this function does not have optimal performance, and visual inspection of the plots is required to confirm the estimated ploidy.

Usage

```
area_estimate_ploidy(
  qploidy_standardization = NULL,
  samples = "all",
  level = "chromosome",
  ploidies = NULL,
```

```
  area = 0.75,
  centromeres = NULL
)
```

Arguments

qploidy_standardization	Object of class qploidy_standardization.
samples	If "all", all samples contained in the qploidy_standardization object will be evaluated. If a vector with sample names is provided, only those will be evaluated.
level	Character identifying the level of the analysis. Must be one of "chromosome", "sample", or "chromosome-arm". If 'chromosome-arm', the analysis will be performed by chromosome arm (only if 'centromeres' argument is defined).
ploidies	Vector of ploidy levels to test. This parameter must be defined.
area	Area around the expected peak to be considered. Default is 0.75.
centromeres	Vector with centromere genomic positions in bp. The vector should be named with the chromosome IDs. This information will only be used if 'chromosome-arm' level is defined.

Value

A list of class 'qploidy_area_ploidy_estimation' containing:

- `ploidy`: Estimated ploidy by area method.
- `prop_inside_area`: Proportion of dots inside selected area.
- `diff_first_second`: Difference between first and second place in area method.
- `sd_inside_area`: Standard deviation inside area.
- `highest_correlation_modes`: Highest correlation.
- `modes_inside_area`: Modes inside areas.
- `tested`: Tested ploidies.
- `ploidy.sep`: Separated ploidy results.
- `chr`: Unique chromosomes in the dataset.
- `n.inbred`: Number of highly inbred samples.

`clean_summary` *Clean Axiom Summary File*

Description

This function removes consecutive A allele probes from an Axiom summary file.

Usage

```
clean_summary(summary_df)
```

Arguments

`summary_df`	A data frame containing A and B probe intensities.

Value

A list with cleaned A and B probes.

Examples

```
NULL
```

`find_header_line` *Find the Header Line in a File*

Description

This function scans a file to locate the first line containing a specific keyword, such as 'probeset_id'. It is useful for identifying the starting point of data in files with headers or metadata.

Usage

```
find_header_line(summary_file, word = "probeset_id", max_lines = 6000)
```

Arguments

`summary_file`	The path to the file to be scanned.
`word`	The keyword to search for in the first column. Default is "probeset_id".
`max_lines`	The maximum number of lines to scan. Default is 6000.

Value

The line number where the keyword is found.

get_aneuploids *indexes for aneuploids*

Description

indexes for aneuploids

Usage

```
get_aneuploids(ploidy_df)
```

Arguments

ploidy_df	ploidy table (chromosome in columns and individuals in rows)

Value

A logical vector where each element corresponds to an individual in the input ploidy table. The value is 'TRUE' if the individual is identified as potentially aneuploid, and 'FALSE' otherwise.

get_baf *Calculate B-Allele Frequency (BAF) from Theta Values*

Description

This function calculates the B-allele frequency (BAF) from normalized theta values, using cluster centers that represent genotype classes. BAF is computed by linearly interpolating the theta values between adjacent genotype cluster centroids.

Usage

```
get_baf(theta_subject, centers_theta, ploidy)
```

Arguments

theta_subject	A numeric vector of theta values to be standardized. These typically represent allelic ratios or normalized intensity values for a set of samples.
centers_theta	A numeric vector of length 'ploidy + 1', representing the estimated cluster centers (centroids) for each genotype class. These values should be sorted in increasing order from homozygous reference to homozygous alternative.
ploidy	An integer indicating the ploidy level of the organism (e.g., '2' for diploid).

Details

The approach is based on the methodology described by Wang et al. (2007), and is commonly used in SNP genotyping to infer allele-specific signal intensities.

Value

A numeric vector of BAF values ranging from 0 to 1

Note

The ‘centers_theta‘ vector must contain exactly ‘ploidy + 1‘ values, and must be sorted in ascending order. If ‘theta_subject‘ values fall outside the range, BAFs are capped at 0 or 1 accordingly.

References

Wang, K., Li, M., Hadley, D., Liu, R., Glessner, J., Grant, S. F. A., Hakonarson, H., & Bucan, M. (2007). PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome Research, 17*(11), 1665–1674. doi:10.1101/gr.6861907

Examples

```
theta <- c(0.1, 0.35, 0.6, 0.95)
centers <- c(0.1, 0.5, 0.9)
get_baf(theta, centers, ploidy = 2)
```

---

get_baf_par *To create baf in parallel*

Description

To create baf in parallel

Usage

```
get_baf_par(par_all_item, ploidy = 2)
```

Arguments

`par_all_item`	list containing R and theta matrices, and clusters models
`ploidy`	integer defining ploidy

Value

A list of numeric vectors, where each vector contains the BAF values for a corresponding row in the input ‘par_all_item‘ matrices. Each BAF vector has values ranging from 0 to 1, representing the standardized allelic ratios for the respective samples or markers.

`get_centers` *Estimate Cluster Centers for Genotype Dosage Classes*

Description

This function estimates the cluster centers for each genotype dosage class based on the ‘theta‘ values (e.g., allelic ratios or normalized signal intensities). It supports imputing missing clusters and optionally removing outliers.

Usage

```
get_centers(
  ratio_geno,
  ploidy,
  n.clusters.thr = NULL,
  type = c("intensities", "counts"),
  rm_outlier = TRUE,
  cluster_median = TRUE
)
```

Arguments

`ratio_geno`	A data.frame containing the following columns: - ‘MarkerName‘: Identifier for each marker. - ‘SampleName‘: Identifier for each sample. - ‘theta‘: Numeric variable representing allelic ratio or signal intensity. - ‘geno‘: Integer dosage (e.g., 0, 1, 2 for diploids).
`ploidy`	Integer specifying the organism ploidy (e.g., 2 for diploid).
`n.clusters.thr`	Integer specifying the minimum number of genotype clusters required for a marker to be retained. If fewer clusters are found, missing ones can be imputed depending on the ‘type‘. Defaults to ‘ploidy + 1‘ if ‘NULL‘.
`type`	Character string indicating the data source type: - ‘"intensities"‘: For array-based allele intensities. - ‘"counts"‘: For sequencing read counts. Default is ‘"intensities"‘.
`rm_outlier`	Logical; if ‘TRUE‘, outlier samples within genotype clusters will be identified and removed prior to center calculation (default: ‘TRUE‘).
`cluster_median`	Logical; if ‘TRUE‘, cluster centers are calculated using the median of ‘theta‘ values. If ‘FALSE‘, the mean is used (default: ‘TRUE‘).

Value

A named list with the following elements: - ‘rm‘: Integer flag: ‘0‘ (retained), ‘1‘ (no clusters found), or ‘2‘ (too few clusters). - ‘centers_theta‘: A numeric vector of cluster center positions on the theta scale. - ‘MarkerName‘: Marker identifier. - ‘n.clusters‘: Number of clusters (including imputed ones if applicable).

`get_R_theta` *Get R and Theta Values from Summary File*

Description

This function calculates R and theta values from a cleaned summary file. It optionally performs standard normalization by plate and markers.

Usage

```
get_R_theta(cleaned_summary, atan = FALSE)
```

Arguments

`cleaned_summary`	A summary object from the clean_summary function.
`atan`	Logical. If TRUE, calculates theta using atan2.

Value

A list containing the following elements: - 'R_all': A data frame where each row corresponds to a marker, and columns represent total signal intensity (R) values for each sample. - 'theta_all': A data frame where each row corresponds to a marker, and columns represent allelic ratio (theta) values for each sample. - Both data frames include a 'MarkerName' column as the first column, which contains marker identifiers.

`get_zscore` *Calculate Z-Scores for Allele Intensities or Counts*

Description

This function computes per-marker Z-scores based on the total signal intensity (R), which typically represents the sum of reference (X) and alternative (Y) allele signals. The Z-score measures how much each sample deviates from the mean intensity of that marker.

Usage

```
get_zscore(data = NULL, geno.pos = NULL)
```

Arguments

`data`	A data.frame containing signal intensity and ratio values with the following columns: **MarkerName** Marker identifiers. **SampleName** Sample identifiers. **X** Reference allele intensity or count. **Y** Alternative allele intensity or count. **R** Total signal or depth (i.e., `X + Y`). **ratio** Allelic ratio, typically `Y / (X + Y)`.
`geno.pos`	A data.frame with marker genomic positions, containing the following columns: **MarkerName** Marker identifiers. **Chromosome** Chromosome identifier where the marker is located. **Position** Genomic position (base-pair coordinate) of the marker.

Details

The function also merges positional metadata from the 'geno.pos' input, adding chromosome and physical position for each marker.

Value

A data.frame containing the following columns:

MarkerName Marker ID.

Chr Chromosome corresponding to the marker.

Position Genomic position (bp).

SampleName Sample ID.

z Z-score computed per marker across all samples.

Markers with missing chromosome or position information are excluded from the final output.

Examples

```
data <- data.frame(
  MarkerName = rep("m1", 5),
  SampleName = paste0("S", 1:5),
  X = c(100, 110, 90, 95, 85),
  Y = c(200, 190, 210, 205, 215),
  R = c(300, 300, 300, 300, 300),
  ratio = c(0.67, 0.63, 0.70, 0.68, 0.72)
)
geno.pos <- data.frame(MarkerName = "m1", Chromosome = "1", Position = 123456)
get_zscore(data, geno.pos)
```

`is_compressed_file` *Check if a File is Compressed*

Description

This function checks whether a given file is compressed by inspecting its magic number (first few bytes). It detects common compression formats such as gzip (‘.gz‘), bzip2 (‘.bz2‘), and xz (‘.xz‘).

Usage

```
is_compressed_file(file_path)
```

Arguments

`file_path`	A character string giving the path to the file to be checked.

Value

A character string indicating the compression type (‘"gzip (.gz)"‘, ‘"bzip2 (.bz2)"‘, or ‘"xz (.xz)"‘) if the file is compressed, or ‘FALSE‘ if the file is not recognized as compressed.

`merge_arms_format` *Merges chromosome-arm level analysis results into chromosome level format*

Description

Merges chromosome-arm level analysis results into chromosome level format

Usage

```
merge_arms_format(x, filter_diff = NULL)
```

Arguments

`x`	object of class qploidy_area_ploidy_estimation
`filter_diff`	filter by difference on area proportion between first and second place

Value

An updated object of class ‘qploidy_area_ploidy_estimation‘ with the following modifications:

- ‘ploidy‘: A matrix where chromosome-arm level results are merged into chromosome-level format. If ‘filter_diff‘ is provided, ploidy values with differences below the threshold are set to ‘NA‘.

The structure of the returned object remains consistent with the input, but with updated ploidy information.

---

mode *Calculate the Statistical Mode*

Description

This function returns the most frequent (modal) value in a vector. If there are multiple values with the same highest frequency, it returns the first one encountered.

Usage

```
mode(x)
```

Arguments

x	A vector of numeric, character, or factor values.

Value

A single value representing the mode of the input vector.

---

pascalTriangle *Pascal Triangle for Expected Peaks Calculation*

Description

This function generates the Pascal triangle for a given ploidy value. The Pascal triangle is used to define the expected peaks for each ploidy level, which can be useful in various genetic analyses.

Usage

```
pascalTriangle(h)
```

Arguments

h	An integer representing the ploidy value.

Value

A list where each element corresponds to a row of the Pascal triangle, up to the specified ploidy value.

`plot_baf` *Plot BAF*

Description

This function generates a BAF (B-allele frequency) plot for visualizing genomic data. It allows customization of dot size, expected and estimated peaks, centromere positions, and area colors.

Usage

```
plot_baf(
  data_sample,
  area_single,
  ploidy,
  dot.size = 1,
  add_estimated_peaks = FALSE,
  add_expected_peaks = FALSE,
  centromeres = NULL,
  add_centromeres = FALSE,
  colors = FALSE,
  font_size = 12
)
```

Arguments

`data_sample`	A data.frame containing BAF and genomic position information. Must include columns 'Chr', 'Position', and 'sample'.
`area_single`	Numeric value defining the area around the expected peak to be considered.
`ploidy`	Integer or vector specifying the expected ploidy. If a vector, it must match the number of chromosomes in 'data_sample'.
`dot.size`	Numeric value for the size of the dots in the plot. Default is 1.
`add_estimated_peaks`	Logical. If TRUE, adds lines for estimated peaks. Default is FALSE.
`add_expected_peaks`	Logical. If TRUE, adds lines for expected peaks. Default is FALSE.
`centromeres`	Named vector defining centromere positions for each chromosome. Names must match chromosome IDs in 'data_sample'.
`add_centromeres`	Logical. If TRUE, adds vertical lines at centromere positions. Default is FALSE.

colors	Logical. If TRUE, adds area colors to the plot. Default is FALSE.
font_size	Numeric value for the font size of plot labels. Default is 12.

Value

A ggplot object representing the BAF plot.

---

plot_baf_hist *Plot BAF Histogram*

Description

This function generates a histogram of BAF (B-allele frequency) values. It supports options for adding estimated and expected peaks, area colors, and filtering homozygous calls.

Usage

```
plot_baf_hist(
  data_sample,
  area_single,
  ploidy,
  colors = FALSE,
  add_estimated_peaks = TRUE,
  add_expected_peaks = FALSE,
  BAF_hist_overall = FALSE,
  ratio = FALSE,
  rm_homozygous = FALSE,
  font_size = 12
)
```

Arguments

data_sample	A data.frame containing BAF and genomic position information. Must include columns 'Chr', 'Position', and 'sample'.
area_single	Numeric value defining the area around the expected peak to be considered.
ploidy	Integer or vector specifying the expected ploidy. If a vector, it must match the number of chromosomes in 'data_sample'.
colors	Logical. If TRUE, adds area colors to the histogram. Default is FALSE.
add_estimated_peaks	Logical. If TRUE, adds lines for estimated peaks. Default is TRUE.
add_expected_peaks	Logical. If TRUE, adds lines for expected peaks. Default is FALSE.
BAF_hist_overall	Logical. If TRUE, plots the BAF histogram for the entire genome. Default is FALSE.

`ratio`	Logical. If TRUE, plots the raw ratio instead of BAF. Default is FALSE.
`rm_homozygous`	Logical. If TRUE, removes homozygous calls from the histogram. Default is FALSE.
`font_size`	Numeric value for the font size of plot labels. Default is 12.

Value

A ggplot object representing the BAF histogram.

`plot_baf_with_ploidy_guides` *Plot BAF-derived X–Y scatter with ploidy dosage guides (optional sample highlighting)*

Description

Reconstructs Cartesian coordinates from standardized B-allele frequency (BAF) and read depth ‘R‘ as $X = (1 - \text{BAF}) \cdot R$ and $Y = \text{BAF} \cdot R$, then draws a scatter plot with expected dosage guide lines for a given ploidy. You can color all samples by ‘SampleName‘ or highlight a single sample and render the rest in gray. Samples with no plottable points (non-finite coordinates) are automatically omitted from the legend.

Usage

```
plot_baf_with_ploidy_guides(
  df,
  ploidy = 2,
  fallback_to_ratio = FALSE,
  normalize_depth = TRUE,
  radius = NULL,
  sample = NULL,
  highlight_color = "tomato",
  other_color = "grey75"
)
```

Arguments

`df`	A ‘data.frame‘ with required columns: - ‘baf‘ (numeric in \[0,1\]): standardized B-allele frequency. - ‘R‘ (numeric): total read depth. - ‘SampleName‘ (character/factor): sample label used for coloring. Optional column ‘ratio‘ may be present and used when ‘fallback_to_ratio = TRUE‘.
`ploidy`	Integer (2). Ploidy used to compute dosage guide lines.
`fallback_to_ratio`	Logical. If ‘TRUE‘, fill ‘NA‘ values in ‘baf‘ with corresponding values from ‘ratio‘ (when available). Default: ‘FALSE‘.

`normalize_depth`
: Logical. If ‘TRUE‘, place all points on a common radius (see ‘radius‘); if ‘FALSE‘, use each point’s ‘R‘. Default: ‘TRUE‘.

`radius`
: Numeric scalar radius to use when ‘normalize_depth = TRUE‘. If ‘NULL‘, uses ‘stats::median(df$R, na.rm = TRUE)‘. Ignored when ‘normalize_depth = FALSE‘. Default: ‘NULL‘.

`sample`
: Character. Either ‘”all”‘ to color all samples by ‘SampleName‘, or the name of a single sample to highlight. Default: ‘”all”‘.

`highlight_color`
: Color for the highlighted sample when ‘sample != ”all”‘. Default: ‘”tomato”‘.

`other_color`
: Color for non-highlighted samples when ‘sample != ”all”‘. Default: ‘”grey75”‘.

Details

* Coordinates are computed from BAF and depth: $X = (1 - \text{BAF})R$, $Y = \text{BAF}R$. * If ‘fallback_to_ratio = TRUE‘ and ‘baf‘ is ‘NA‘, values from ‘ratio‘ are used. The effective BAF is clamped into \[0, 1\]. * When ‘normalize_depth = TRUE‘, all points are projected to the same radius (depth) given by ‘radius‘ (or ‘stats::median(R)‘ if ‘radius‘ is ‘NULL‘), which emphasizes dosage bands rather than depth variation. When ‘FALSE‘, each point uses its own ‘R‘. * Dosage guide lines are drawn for $d \in \{0, \ldots, \text{ploidy}\}$: ‘d = 0‘ $\rightarrow$ horizontal line ‘Y = 0‘; ‘d = ploidy‘ $\rightarrow$ vertical line ‘X = 0‘; intermediate dosages are lines through the origin with slope $(d/\text{ploidy})/(1 - d/\text{ploidy})$. * The legend is built from actually plotted rows only; if a requested ‘sample‘ has no plottable points, it is omitted from the legend. * Uses a fixed aspect ratio (‘coord_fixed‘) so x and y units are comparable.

Value

A **ggplot** object.

See Also

[plot_xy_with_ploidy_guides()] for plotting raw ‘X‘/‘Y‘ counts with guides.

`plot_qploidy_standardization`
Plot Method for Qploidy Standardization

Description

This function generates various plots for visualizing the results of Qploidy standardization. It supports multiple plot types, including BAF, z-score, and histograms.

Usage

```
plot_qploidy_standardization(
  x,
  sample = NULL,
  chr = NULL,
  type = c("all", "het", "BAF", "zscore", "BAF_hist", "ratio", "BAF_hist_overall",
    "Ratio_hist_overall"),
  area_single = 0.75,
  ploidy = 4,
  dot.size = 1,
  font_size = 12,
  add_estimated_peaks = FALSE,
  add_expected_peaks = FALSE,
  centromeres = NULL,
  add_centromeres = FALSE,
  colors = FALSE,
  window_size = 2e+06,
  het_interval = 0.1,
  rm_homozygous = FALSE,
  ...
)
```

Arguments

x	An object of class 'qploidy_standardization'.
sample	Character string indicating the sample ID to plot.
chr	Character or numeric vector specifying the chromosomes to plot. Default is NULL (plots all chromosomes).
type	Character vector defining the plot types. Options include: - "all": Generates all available plot types. - "het": Plots heterozygous locus counts across genomic windows. - "BAF": Plots B-allele frequency (BAF) for each chromosome. - "zscore": Plots z-scores for each chromosome. - "BAF_hist": Plots BAF histograms for each chromosome. - "BAF_hist_overall": Plots a BAF histogram for the entire genome. - "Ratio_hist_overall": Plots a histogram of raw ratios for the entire genome. - "ratio": Plots raw ratios for each chromosome. Default is "all".
area_single	Numeric value defining the area around the expected peak to be considered. Default is 0.75.
ploidy	Integer specifying the expected ploidy. Default is 4.
dot.size	Numeric value for the size of the dots in the plots. Default is 1.
font_size	Numeric value for the font size of plot labels. Default is 12.
add_estimated_peaks	Logical. If TRUE, adds lines for estimated peaks. Default is FALSE.
add_expected_peaks	Logical. If TRUE, adds lines for expected peaks. Default is FALSE.

`centromeres` Named vector defining centromere positions for each chromosome. Names must match chromosome IDs in 'x'.

`add_centromeres` Logical. If TRUE, adds vertical lines at centromere positions. Default is FALSE.

`colors` Logical. If TRUE, adds area colors to the plots. Default is FALSE.

`window_size` Numeric value defining the genomic position window for heterozygous locus counts. Default is 2000000.

`het_interval` Numeric value defining the interval to consider as heterozygous. Default is 0.1.

`rm_homozygous` Logical. If TRUE, removes homozygous calls from BAF histogram plots. Default is FALSE.

`...` Additional plot parameters.

Details

The function supports the following plot types:

- **all**: Generates all available plot types. - **het**: Plots the proportion of heterozygous loci across genomic windows, useful for identifying regions with high or low heterozygosity. - **BAF**: Plots the B-allele frequency (BAF) for each chromosome, showing the distribution of allele frequencies. - **zscore**: Plots z-scores for each chromosome, which can help identify outliers or regions with unusual data distributions. - **BAF_hist**: Plots histograms of BAF values for each chromosome, providing a summary of allele frequency distributions. - **BAF_hist_overall**: Plots a single histogram of BAF values for the entire genome, summarizing allele frequency distributions genome-wide. - **Ratio_hist_overall**: Plots a histogram of raw ratios for the entire genome, useful for visualizing overall ratio distributions. - **ratio**: Plots raw ratios for each chromosome, showing the distribution of observed ratios.

Value

A ggarrange object containing the requested plots.

`plot_xy_with_ploidy_guides` *Plot X–Y scatter with ploidy dosage guide lines and optional sample highlighting*

Description

Creates a dot plot of allele counts 'X' (A-allele) vs 'Y' (B-allele) and overlays expected dosage guide lines for a given ploidy. You can either color all samples ('sample = "all"') or highlight a single sample while rendering the others in gray. Samples that have no plotted points (e.g., due to non-finite 'X'/'Y') are automatically omitted from the legend.

Usage

```
plot_xy_with_ploidy_guides(
  df,
  ploidy = 2,
  sample = NULL,
  highlight_color = "tomato",
  other_color = "grey75"
)
```

Arguments

`df`	A ‘data.frame‘ containing at least columns ‘X‘ and ‘Y‘. If ‘sample = ”all”‘ or a specific sample is to be highlighted, ‘df‘ should also contain a ‘SampleName‘ column.
`ploidy`	Integer (2). Ploidy used to compute and draw dosage guide lines.
`sample`	Character. Either ‘”all”‘ to color all samples by ‘SampleName‘, or the name of a single sample to highlight. Default: ‘”all”‘.
`highlight_color`	Color used for the highlighted sample when ‘sample != ”all”‘. Default: ‘”tomato”‘.
`other_color`	Color used for non-highlighted samples when ‘sample != ”all”‘. Default: ‘”grey75”‘.

Details

The function: * Drops rows where ‘X‘ or ‘Y‘ are non-finite before plotting and builds the legend from the remaining points. * When ‘sample = ”all”‘, colors points by ‘SampleName‘ (requires a ‘SampleName‘ column). * When ‘sample‘ is a specific name, only that sample is colored with ‘highlight_color‘; all others use ‘other_color‘. If the requested sample has no plotted points, it is omitted from the legend. * Draws dosage guide lines for dosages ‘d = 0, …, ploidy‘: ‘d = 0‘ → horizontal line ‘Y = 0‘; ‘d = ploidy‘ → vertical line ‘X = 0‘; intermediate dosages are lines through the origin with slope $(d/ploidy)/(1 - d/ploidy)$. * Uses a fixed aspect ratio (‘coord_fixed‘) so that one unit on the x- and y-axes has the same length.

Value

A **ggplot** object.

See Also

[plot_baf_with_ploidy_guides()] for plotting standardized BAF-based coordinates.

`print.qploidy_area_ploidy_estimation`
print qploidy_area_ploidy_estimation object

Description

print qploidy_area_ploidy_estimation object

Usage

```
## S3 method for class 'qploidy_area_ploidy_estimation'
print(x, ...)
```

Arguments

`x`	qploidy_area_ploidy_estimation object
`...`	print parameters

Value

No return value, called for side effects.

`print.qploidy_standardization`
Print method for object of class 'qploidy_standardization'

Description

Print method for object of class 'qploidy_standardization'

Usage

```
## S3 method for class 'qploidy_standardization'
print(x, ...)
```

Arguments

`x`	object of class 'qploidy_standardization'
`...`	print parameters

Value

printed information about Qploidy standardization process

`qploidy_read_vcf` *Convert VCF File to Qploidy Data*

Description

This function converts a VCF file into a format compatible with Qploidy analysis. It extracts genotype and allele depth information and formats it into a data frame.

Usage

```
qploidy_read_vcf(vcf_file, geno = FALSE, geno.pos = FALSE)
```

Arguments

`vcf_file`	Path to the VCF file.
`geno`	Logical. If TRUE, the output columns will include MarkerName, SampleName, geno, and prob. If FALSE, the output will include MarkerName, SampleName, X, Y, R, and ratio.
`geno.pos`	Logical. If TRUE, the output will include MarkerName, Chromosome, and Position columns.

Value

A data frame containing the processed VCF data.

`read_axiom` *Convert Axiom Array Summary File to Qploidy Input*

Description

This function processes an Axiom array summary file and converts it into a format compatible with Qploidy and fitpoly analysis.

Usage

```
read_axiom(summary_file, ind_names = NULL, atan = FALSE)
```

Arguments

`summary_file`	Path to the Axiom summary file.
`ind_names`	Optional. A file with two columns: Plate_name (sample IDs in the summary file) and Sample_Name (desired sample names).
`atan`	Logical. If TRUE, calculates theta using atan2.

Value

A data frame formatted for Qploidy analysis, containing the following columns: - 'MarkerName': Marker identifiers. - 'SampleName': Sample identifiers (if 'ind_names' is provided, these will be updated accordingly). - 'X': Reference allele intensity (calculated if applicable). - 'Y': Alternative allele intensity (calculated if applicable). - 'R': Total signal intensity (calculated if applicable). - 'ratio': Allelic ratio (theta, calculated if applicable). - Additional columns may be included depending on the input data and processing steps.

`read_illumina_array` *Read Illumina Array Files*

Description

This function reads Illumina array files and processes them into a format suitable for Qploidy analysis. It adds a suffix to sample IDs if multiple files are provided.

Usage

```
read_illumina_array(...)
```

Arguments

`...`	One or more Illumina array filenames.

Value

A data frame containing the processed Illumina array data.

`read_qploidy_standardization` *Read a Qploidy Standardized Dataset (.tsv or .tsv.gz) with format checks*

Description

Expected layout (tab-separated): 1) "info" header line 2) one row of info values 3) "filters" header line 4) one row of numeric filter values 5) "data" header line 6+) data rows (must include at least SampleName and Chr)

Usage

```
read_qploidy_standardization(qploidy_standardization_file)
```

Arguments

`qploidy_standardization_file`	Path to the TSV/TSV.GZ file

Value

An object of class "qploidy_standardization" with fields: - info: named character vector (single row) - filters: named numeric vector (single row, typically >= 6 values) - data: data.frame/tibble with at least SampleName and Chr

---

`rm_outlier` *Identify and Remove Outliers Based on Bonferroni-Holm Adjusted P-values*

Description

This function detects and removes outlier observations from a vector of 'theta' values using externally studentized residuals and the Bonferroni-Holm adjustment for multiple testing. It is typically used during genotype cluster center estimation to clean noisy values.

Usage

```
rm_outlier(data, alpha = 0.05)
```

Arguments

`data`	A data.frame containing a 'theta' column. This is usually a subset of the full dataset, representing samples within a single genotype class.
`alpha`	Significance level for identifying outliers (default is '0.05'). Observations with adjusted p-values below this threshold will be removed.

Details

The method fits a constant model ('theta ~ 1') and computes standardized residuals. Observations with significant deviation are flagged using the Bonferroni-Holm procedure and removed if their adjusted p-value is below the defined 'alpha' threshold.

This function was originally developed by **Kaio Olympio** and incorporated into the Qploidy workflow.

Value

A data.frame containing only the non-outlier observations from the input. If fewer than two non-NA 'theta' values are present or if all values are identical, the input is returned unmodified.

Author(s)

Kaio Olympio

`simulate_axiom_summary`

Simulate an Axiom array summary file

Description

This function generates a simulated Axiom array summary file with probe IDs ending in '-A' or '-B' and sample intensities. The intensities are simulated based on the genotype of the sample: homozygous for A, homozygous for B, or heterozygous.

Usage

```
simulate_axiom_summary(file_path, n_probes = 100, n_samples = 10, seed)
```

Arguments

`file_path`	The path where the simulated summary file will be saved.
`n_probes`	Number of probes to simulate. Default is 100.
`n_samples`	Number of samples to simulate. Default is 10.
`seed`	The seed for random number generation to ensure reproducibility.

Value

None. The function writes the simulated summary content to the specified file.

`simulate_illumina_file`

Simulate an Illumina File

Description

This function generates a simulated Illumina file with SNP data for a specified number of SNPs and samples. The file includes a header section and a data section with fields such as SNP Name, Sample ID, GC Score, Theta, X, Y, X Raw, Y Raw, and Log R Ratio.

Usage

```
simulate_illumina_file(
  filepath,
  num_snps = 10,
  num_samples = 1,
  sample_id_prefix = "SAMP",
  mk_id = "MK-",
  seed = 123
)
```

Arguments

`filepath`	The path where the simulated Illumina file will be saved. Default is "simulated_summary.txt".
`num_snps`	The number of SNPs to simulate. Default is 10.
`num_samples`	The number of samples to simulate. Default is 1.
`sample_id_prefix`	The prefix for sample IDs. Default is "SAMP".
`mk_id`	The prefix for marker IDs. Default is "MK-".
`seed`	The seed for random number generation to ensure reproducibility. Default is 123.

Details

The simulated data includes random values for GC Score, Theta, X, Y, X Raw, Y Raw, and Log R Ratio. The header section provides metadata about the file, including the number of SNPs and samples.

Value

None. The function writes the simulated Illumina file to the specified path.

`simulate_standardization_input` *Simulate Genotyping Data with Flexible Ploidy*

Description

Generates synthetic genotyping and signal intensity data for a given ploidy level. Returns a structured list containing input data suitable for standardization analysis.

Usage

```
simulate_standardization_input(
  n_markers = 10,
  n_samples = 5,
  ploidy = 2,
  seed = 2025
)
```

Arguments

`n_markers`	Integer. Number of markers to simulate (default: 10).
`n_samples`	Integer. Number of individuals/samples to simulate (default: 5).
`ploidy`	Integer. Ploidy level of the organism (e.g., 2 for diploid, 4 for tetraploid).
`seed`	Integer. Random seed for reproducibility (default: 2025).

Value

A named list with:

sample_data Allelic signal intensities (X, Y, R, ratio).

geno_data Genotype dosage and probability data.

geno_pos Genomic coordinates for each marker.

standardization_input Merged input data with theta and genotype.

`simulate_vcf` *Simulate a VCF file with GT, DP, and AD format fields for 2 chromosomes*

Description

Simulate a VCF file with GT, DP, and AD format fields for 2 chromosomes

Usage

```
simulate_vcf(
  file_path,
  seed,
  n_tetraploid = 35,
  n_diploid = 5,
  n_triploid = 10,
  n_markers = 100
)
```

Arguments

`file_path`	The path where the simulated VCF file will be saved.
`seed`	The seed for random number generation to ensure reproducibility.
`n_tetraploid`	Number of tetraploid samples. Default is 35.
`n_diploid`	Number of diploid samples. Default is 5.
`n_triploid`	Number of triploid samples. Default is 10.
`n_markers`	Number of markers to simulate. Default is 100.

Value

None. The function writes the simulated VCF content to the specified file.

`standardize` *Standardize Allelic Ratio Data and Compute BAF and Z-Scores*

Description

This function performs signal standardization of genotype data by aligning ‘theta‘ values (allelic ratios or normalized intensities) to expected genotype clusters. It outputs standardized BAF (B-allele frequency) and Z-scores per sample and marker.

Usage

```
standardize(
  data = NULL,
  genos = NULL,
  geno.pos = NULL,
  threshold.missing.geno = 0.9,
  threshold.geno.prob = 0.8,
  ploidy.standardization = NULL,
  threshold.n.clusters = NULL,
  n.cores = 1,
  out_filename = NULL,
  type = "intensities",
  multidog_obj = NULL,
  parallel.type = "PSOCK",
  verbose = TRUE,
  rm_outlier = TRUE,
  cluster_median = TRUE
)
```

Arguments

`data`	A ‘data.frame‘ containing the full dataset with the following columns: - MarkerName: Marker identifiers. - SampleName: Sample identifiers. - X: Reference allele intensity or count. - Y: Alternative allele intensity or count. - R: Total signal intensity or read depth (X + Y). - ratio: Allelic ratio, typically Y / (X + Y).
`genos`	A ‘data.frame‘ containing genotype dosage information for the reference panel. - MarkerName: Marker identifiers. - SampleName: Sample identifiers. - geno: Estimated dosage (0, 1, 2, ...). - prob: Genotype call probability (used for filtering low-confidence genotypes).
`geno.pos`	A ‘data.frame‘ with marker position metadata. - MarkerName: Marker identifiers. - Chromosome: Chromosome names. - Position: Base-pair positions on the genome.
`threshold.missing.geno`	Numeric (0–1). Maximum fraction of missing genotype data allowed per marker. Markers with a higher fraction will be removed.

`threshold.geno.prob`
: Numeric (0–1). Minimum genotype call probability threshold. Genotypes with lower probability will be treated as missing.

`ploidy.standardization`
: Integer. The ploidy level of the reference panel used for standardization.

`threshold.n.clusters`
: Integer. Minimum number of expected dosage clusters per marker. For diploid data, this is typically 3 (corresponding to genotypes 0, 1, and 2).

`n.cores`
: Integer. Number of cores to use in parallel computations (e.g., for cluster center estimation and BAF generation).

`out_filename`
: Optional. Path to save the final standardized dataset to disk as a CSV file (suitable for Qploidy).

`type`
: Character. Type of data used for clustering: - "intensities": For array-based allele intensity data. - "counts": For sequencing data. - "updog": Automatically set when 'multidog_obj' is provided.

`multidog_obj`
: Optional. An object of class 'multidog' from the 'updog' package, containing model fits and estimated biases. If provided, this will override the 'type' parameter and use 'updog''s expected cluster positions.

`parallel.type`
: Character. Parallel backend to use ("FORK" or "PSOCK"). "FORK" is faster but only works on Unix-like systems.

`verbose`
: Logical. If TRUE, prints progress and filtering information to the console.

`rm_outlier`
: Logical. If TRUE, uses Bonferroni-Holm corrected residuals to remove outliers before estimating cluster centers.

`cluster_median`
: Logical. If TRUE, uses the median of theta values to estimate cluster centers. If FALSE, uses the mean.

Details

Reference genotypes are used to estimate cluster centers either from dosage data (e.g., via 'fitpoly' or 'updog') or using an 'updog' 'multidog' object directly. This function supports both array-based (intensity) and sequencing-based (count) data.

It applies marker and genotype-level quality filters, uses parallel computing to estimate BAF, and generates a final annotated output suitable for CNV or dosage variation analyses.

Filtering steps: 1. Genotype probability filter: Genotypes with probability below 'threshold.geno.prob' are set to missing. 2. Marker missingness filter: Markers with fraction of missing genotypes above 'threshold.missing.geno' are removed. 3. Cluster number filter: Markers with fewer than 'threshold.n.clusters' clusters are removed. 4. Genomic info filter: Markers lacking chromosome or position info are removed.

Merging logic: - The function merges filtered genotype and signal data, estimates cluster centers, computes BAFs in parallel, and calculates Z-scores. Results are merged into a single output data.frame containing BAF, Z-score, and genotype info.

Value

An object of class "qploidy_standardization" (list) with the following components: - info: Named vector of standardization parameters. - filters: Named vector summarizing how many markers were removed at each filtering step. - data: A data.frame containing merged BAF, Z-score, and genotype information by marker and sample.

Examples

```
# Example usage:
# data <- ... # see vignette for example data
# genos <- ...
# geno.pos <- ...
# result <- standardize(data, genos, geno.pos, ploidy.standardization=2, threshold.n.clusters=3, n.cores=
```

---

summary_to_fitpoly *Convert Summary Data to FitPoly-Compatible Format*

Description

This function processes R (total signal intensity) and theta (allelic ratio) values to generate a data frame compatible with the FitPoly tool. It calculates X and Y values (reference and alternative allele intensities, respectively) and combines them with R and theta into a long-format data frame.

Usage

```
summary_to_fitpoly(R_all, theta_all)
```

Arguments

R_all	A data frame containing total signal intensity (R) values. The first column should be 'MarkerName', and subsequent columns should represent samples.
theta_all	A data frame containing allelic ratio (theta) values. The first column should be 'MarkerName', and subsequent columns should represent samples.

Details

The function calculates X and Y values as follows: - 'X = R * (1 - theta)' - 'Y = R * theta' The resulting data frame is in a long format, where each row corresponds to a specific marker-sample combination.

Value

A data frame in long format with the following columns: - 'MarkerName': Marker identifiers. - 'SampleName': Sample identifiers. - 'X': Reference allele intensity. - 'Y': Alternative allele intensity. - 'R': Total signal intensity. - 'ratio': Allelic ratio (theta).

`updog_centers` *Estimate Centers for Standardization Using Updog Bias*

Description

This function calculates the centers for standardization based on the estimated bias from the 'updog' package. It identifies genotype dosage clusters and determines whether markers should be retained or removed based on the number of clusters.

Usage

```
updog_centers(multidog_obj, threshold.n.clusters = 2, rm.mks)
```

Arguments

`multidog_obj`	An object of class 'multidog' (from the 'updog' package), containing information about SNPs, ploidy, sequencing error rates, and bias.
`threshold.n.clusters`	An integer specifying the minimum number of dosage clusters (heterozygous classes) required for a marker to be retained for standardization. Default is '2'.
`rm.mks`	A logical vector indicating which markers should be removed. The names of the vector correspond to the marker names.

Details

The function uses the 'xi_fun' to calculate the cluster centers for each marker based on the ploidy, sequencing error rate, and bias. Markers with fewer clusters than the specified threshold are flagged for removal.

Value

A named list where each element corresponds to a marker and contains: - 'rm': An integer flag indicating whether the marker is retained ('0') or removed ('1'). - 'centers_theta': A numeric vector of cluster centers (sorted in descending order). - 'MarkerName': The name of the marker. - 'n.clusters': The number of clusters identified for the marker.

`vcf_sanity_check` *Perform a Sanity Check on a VCF File*

Description

This function performs a series of checks on a VCF file to ensure its validity and integrity. It verifies the presence of required headers, columns, and data fields, and checks for common issues such as missing or malformed data.

Usage

```
vcf_sanity_check(
  vcf_path,
  n_data_lines = 100,
  max_markers = 10000,
  depth_support_fields = c("AD", "RA", "AO", "RO", "NR", "NV", "SB", "F1R2", "F2R1"),
  verbose = FALSE
)
```

Arguments

`vcf_path`	A character string specifying the path to the VCF file. The file can be plain text or gzipped.
`n_data_lines`	An integer specifying the number of data lines to sample for detailed checks. Default is 100.
`max_markers`	An integer specifying the maximum number of markers allowed in the VCF file. Default is 10,000.
`depth_support_fields`	(Optional) A character vector of fields that are expected to be present in the FORMAT column for allele counts. Default is ‘c(”AD”, ”RA”, ”AO”, ”RO”, ”NR”, ”NV”, ”SB”, ”F1R2”, ”F2R1”)‘.
`verbose`	A logical value indicating whether to print detailed messages during the checks. Default is FALSE.

Details

The function performs the following checks: - **VCF_header**: Verifies the presence of the ‘##fileformat‘ header. - **VCF_compressed**: Checks if the VCF file is .gz compressed and if the extension is correct. - **VCF_columns**: Ensures required columns (‘#CHROM‘, ‘POS‘, ‘ID‘, ‘REF‘, ‘ALT‘, ‘QUAL‘, ‘FILTER‘, ‘INFO‘) are present. - **max_markers**: Checks if the total number of markers exceeds the specified limit. - **unique_FORMAT**: Ensures that the FORMAT fields are consistent across sampled markers. - **GT**: Verifies the presence of the ‘GT‘ (genotype) field in the FORMAT column. - **allele_counts**: Checks for allele-level count fields (e.g., ‘AD‘, ‘RA‘, ‘AO‘, ‘RO‘). - **samples**: Ensures sample/genotype columns are present. - **chrom_info** and **pos_info**: Verifies the presence of ‘CHROM‘ and ‘POS‘ columns. - **ref_alt**: Ensures ‘REF‘ and ‘ALT‘ fields contain valid nucleotide codes. - **multiallelics**: Identifies multiallelic sites (ALT field with commas). - **phased_GT**: Checks for phased genotypes (presence of ‘|‘ in the ‘GT‘ field). - **duplicated_samples**: Checks for duplicated sample IDs. - **duplicated_markers**: Checks for duplicated marker IDs.

Value

A list containing: - ‘checks‘: A named vector indicating the results of each check (TRUE or FALSE). - ‘messages‘: A data frame containing messages for each check, indicating success or failure. - ‘duplicates‘: A list containing any duplicated sample or marker IDs found in the VCF file. - ‘ploidy_max‘: The maximum ploidy detected from the genotype field, if applicable.

`write_qploidy_standardization` *Write Qploidy Standardization Object to File*

Description

This function writes a ‘qploidy_standardization‘ object to a specified file. The output file includes metadata, filtering information, and the standardized dataset.

Usage

```
write_qploidy_standardization(qploidy_standardization_object, out_filename)
```

Arguments

`qploidy_standardization_object`	An object of class ‘qploidy_standardization‘ to be written to file.
`out_filename`	A string specifying the path to the output file where the data will be saved. Specify in the file name the desired extension (e.g., ‘my_output.csv‘).

Value

None. The function writes the data to the specified file.

Index