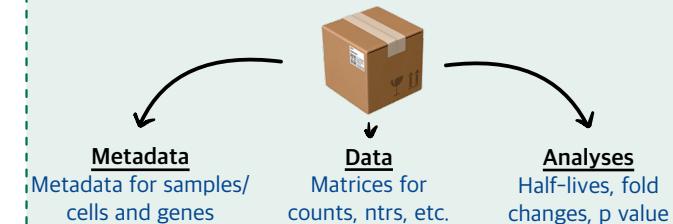
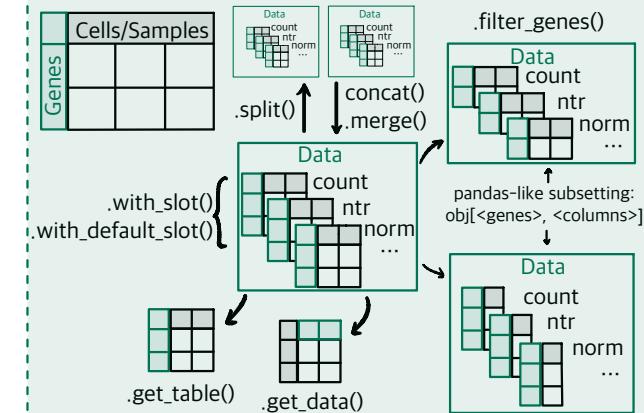


Conversion-seq analysis with GrandPy - CHEAT SHEET

GrandPy OBJECT



DATA



METADATA

Gene metadata

Genes	Metdata

Stores information per Gene such as gene IDs, gene symbols, transcript length, type, etc.
Access a list of gene names: `.genes`

Columns metadata

Samples/Cells	Metdata

Stores information per sample/cell such as labeling duration, experimental condition, replicate, genotype etc.

Access a list of conditions: `.condition`

ANALYSES

Genes	Analysis results

Access a list of Analyses:
`.analyses`

WORKFLOW

General

Defining samples/cells metadata:

- Using systematic sample names:

`Mock.2h.A`

`read_grand(prefix, design = ('Condition', 'duration.4sU', 'Replicate'), ...)`

- Using a metadata table

`obj.filter_genes(mode_slot = 'count', min_expression = 100, min_columns = 4)`

`>= 100 counts in 4 samples/cells`

`obj.filter_genes(mode_slot = 'tpm', min_expression = 10, min_condition = 1)`

`>= 10 TPM in 1 condition`

`normalize(): size factor normalization`

(e.g., DESeq2)

Alternatives: `.normalize_tpm()`, `.normalize_fpkm()`, `.normalize_rpkm()`, `.normalize_baseline()`

`plot_pca(obj)`: visualize sample clustering and detect outliers based on expression or NTR values -> identify global trends and batch effects

Differential Expression

`.compute_lfc()`
`.pairwise_DESeq2()`

Genes	Analysis results

`.get_significant_genes()`

... plots and more

`obj.get_significant_genes(criteria = 'Q < 0.05 and abs(LFC) > 1')`
Gene names (`significant`, > 2-fold upregulated)
`obj.get_significant_genes(criteria = 'abs(LFC) > 1', as_table = True)`
Gene table (> 2-fold regulated)
`obj.get_significant_genes(criteria = 'LFC')`
All gene names (ordered by fold change)

VISUALIZATION

Gene-wise

`plot_gene_total_vs_ntr()`

`plot_gene_old_vs_ntr()`

`plot_gene_groups_points()`

`plot_gene_groups_bars()`

`plot_gene_progressive_timecourse()`

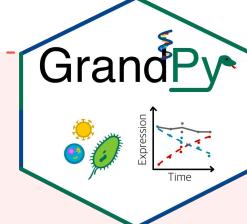
`plot_gene_snapshot_timecourse()`

`plot_heatmap()`

`plot_ma()`

`plot_pca()`

`plot_vulcano()`



Adapt aesthetic mapping using `coldata` columns:

`plot_gene_total_vs_ntr(data, 'gene', aest = {color: 'Condition', shape: 'Replicate'})`

Condition ● Infected ● A
● Control ● B

`plot_gene_total_vs_ntr(data, 'gene', aest = {color: 'Genotype', shape: 'Condition'})`

Genotype ● wt ● S1
● Tag ● S2

`plot_scatter()`

Scatter two variables (expression values, analysis results). Genes can be highlighted (highlight = 'UHMK1') and labeled (label = 'MYC').

Kinetic modeling

`.fit_kinetics()`

fit_type = 'nlls': Non-linear least square fit (steady state and non steady state)
fit_type = 'ntr': Bayesian fit (only steady state)

Calibrate Times

`.calibrate_effective_labeling_time_kinetic_fit()`

For progressive labeling experiments, infer effective labeling times by jointly optimizing kinetic fits for all genes.

Metadata	Calibrated Times
Samples/Cells	Calibrated Times