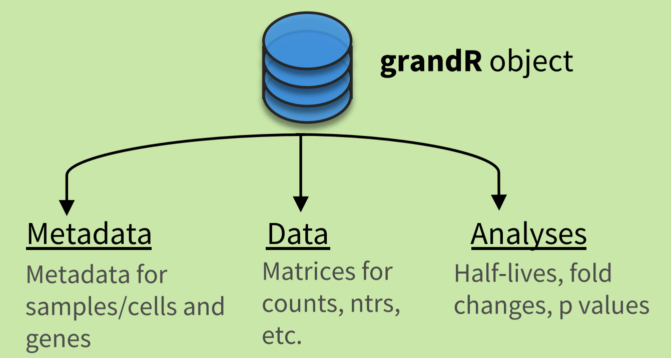
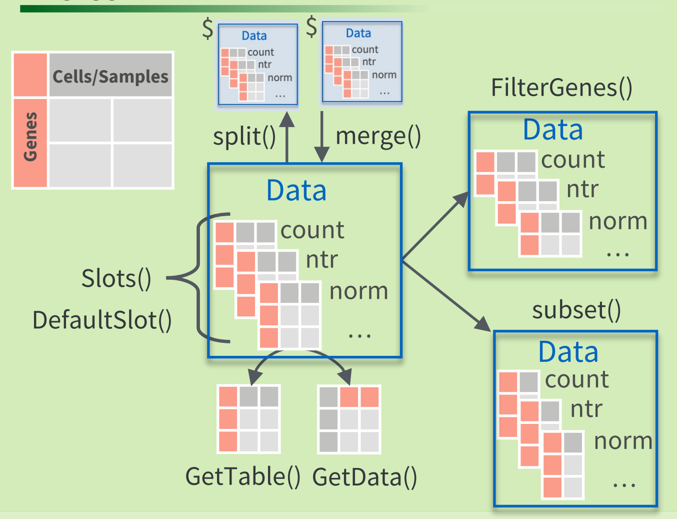
Conversion-seq analysis with grandR : : CHEAT SHEET

grandR object

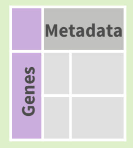

grandR object
Metadata
Metadata for samples/cells and genes
Data
Matrices for counts, ntrs, etc.
Analyses
Half-lives, fold changes, p values

Data



Metadata

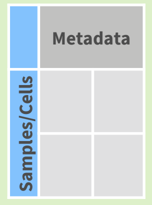
Gene metadata



GeneInfo()

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

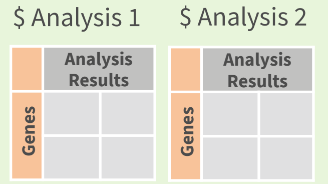
Columns metadata



Coldata()

Stores information per sample/cell such as labeling duration, experimental condition, replicate, genotype, etc.
Access a list of conditions: Condition()

Analyses



GetAnalysisTable()

Access a list of Analyses: Analyses()
Check for valid analysis name:
check.analysis()

Workflow

General

Defining samples/cell metadata:
- Using systematic sample names:
Mock.2h.A

```
ReadGRAND(prefix,design = c("Condition",
"duration.4sU", "Replicate"), ...)
```

- Using a metadata table

```
FilterGenes(data,mode.slot = "count",minval
=100,mincol=4)
```

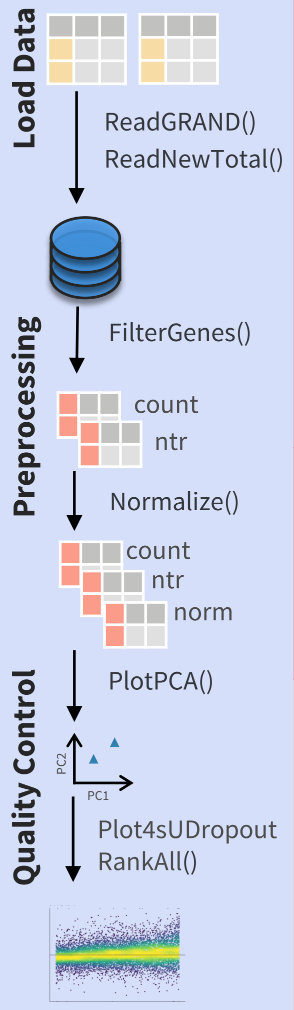
>= 100 counts in 4 samples/cells

```
FilterGenes(data, mode. slot = "tpm",minval
=10,mincond=1)
```

>= 10 TPM in 1 condition

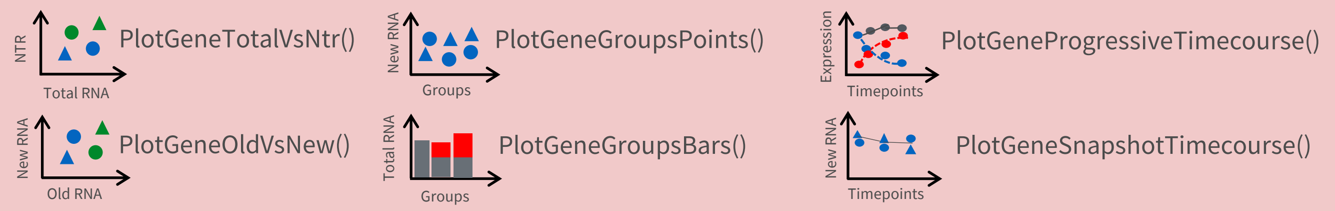
Normalize(): size factor normalization (e.g., DESeq2)
Alternatives: NormalizeTPM(), NormalizeFPKM(), NormalizeRPM(), NormalizeBaseline()

Toxicity test:
Findno4sUPairs(): Find corresponding no4sU sample for each 4sU sample.
Plot4sUDropoutRankAll(): Compare half-lives or NTR ranks against log fold changes 4sU vs. no4sU.



Visualization

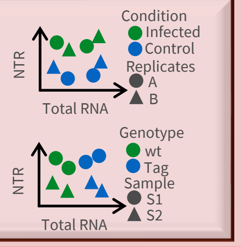
Gene-wise



Adapt the aesthetic mappings using Coldata columns:

```
PlotGeneTotalVsNtr(data,"gene" ,aest = aes(color=Condition,shape = Replicates))
```

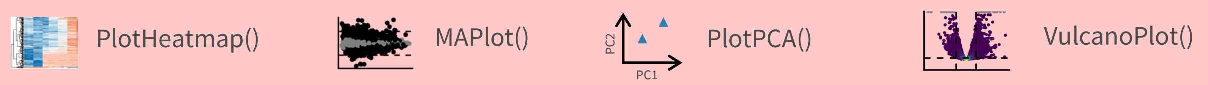
```
PlotGeneTotalVsNtr(data,"gene", aest = aes(color=Genotype,shape = Condition))
```



Global

PlotScatter()

Scatter two variables (expression values, analysis results). Genes can be highlighted (highlight = "ISG15") and labeled (label="MYC").

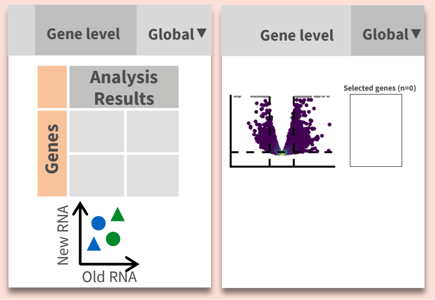


Web-based

ServeGrandR()

Defining gene level plots for a selected gene from the table (first tab):

```
data ⟵ AddGenePlot(data,
"plot1", PlotGeneOldVsNew)
```



Define global plots (second tab):

```
data ⟵ AddGlobalPlot(data,"plot2",
VulcanoPlot)
```

Differential Expression

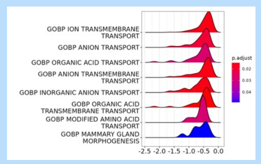
LFC()
PairwiseDESeq2()

Generate Contrast Matrix for pairwise DE Analysis:

```
GetContrasts(data,contrasts=c("Condition",...),
group="Timepoint")
```

contrasts = c("Condition"): All pairwise comparisons among condition
contrasts = c("Condition", "Control"): Each other condition vs. control
contrasts = c("Condition", "Infected", "Control"): Infected vs. control
group = "Timepoint": Comparisons per timepoint

GetSignificantGenes()
AnalyzeGeneSets()



```
GetSignificantGenes(data,criteria = Q< 0.05 & LFC>1)
```

Gene names (significant, > 2-fold upregulated)

```
GetSignificantGenes(data,criteria=abs(LFC)>1,as.table=TRUE)
```

Gene table (> 2-fold regulated)

```
GetSignificantGenes(data,criteria=LFC)
```

All gene names (ordered by fold change)

Kinetic modeling

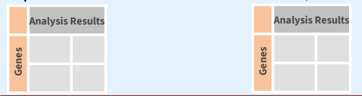
FitKinetics()

```
FitKinetics(data,name.prefix=
kinetics,type = "nlls")
```

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

Snapshot

FindReferences()
FitKineticsSnapshot()
EstimateRegulation()



```
FindReferences(data,columns="0h",group=
"Condition")
```

Define all zero-hour samples as reference sample per condition.

Calibrate Times

CalibrateEffectiveLabelingTimesKineticFit()
For progressive labeling experiments, infer effective labeling times by jointly optimizing kinetic fits for all genes.
CalibrateEffectiveLabelingTimesMatchHalflives()
If reference half-lives are known for some genes fit effective labeling time to match these half-lives.