putior Cheatsheet

Workflow Visualization from Code Annotations

putior

PUT + Input + Output + R

Extract beautiful workflow diagrams from your code annotations. Works with R, Python, SQL, Shell, and Julia.

Quick Start

```
# 1. Add annotation
#put label:"Load Data",
     output:"clean.csv"

# 2. Generate diagram
library(putior)
put_diagram(put("./"))
```

Annotation Syntax

Basic Format

```
#put key:"value", key:"value"
```

Minimal (label only)

```
#put label:"My Step"
```

ID auto-generated, type = "process"

Full Annotation

```
#put id:"step1", \
     label:"Load Data", \
     node_type:"input", \
     input:"config.json", \
     output:"data.csv"
```

Alternative Formats

```
#put label:"Step"   # Standard
# put label:"Step"  # Space
#put| label:"Step"  # Pipe
#put: label:"Step"  # Colon
```

Node Types & Shapes

Type	Shape	Use For
input	`([ ])`	Data sources
process	`[ ]`	Transforms
output	`[[ ]]`	Reports
decision	`{ }`	Branching
start	`([orange])`	Entry
end	`([green])`	Exit

Annotation Fields

Field	Req?	Default
id	No	Auto UUID
label	Rec.	None
node_type	No	"process"
input	No	None
output	No	File name

File Artifacts

```
# Multiple files
output:"data.csv, log.txt"

# Variable tracking
output:"result.internal"
```

.internal = in-memory only

Connecting Scripts

```
# Script A outputs file
#put label:"Fetch",
     output:"data.csv"

# Script B reads that file
#put label:"Process",
     input:"data.csv"
```

Key Functions

Core Workflow

```
# Extract annotations
workflow <- put("./src/")
workflow <- put("script.R")
workflow <- put("./",
                recursive = TRUE)

# Generate diagram
put_diagram(workflow)
put_diagram(workflow,
            theme = "github")
```

Auto-Annotation

```
# Auto-detect from code
put_auto("./src/")

# Generate annotation text
put_generate("./src/")
put_generate("./src/",
  output = "clipboard")

# Merge manual + auto
put_merge("./src/",
  merge_strategy = "supplement")
```

Output Options

```
# Console (default)
put_diagram(wf)

# Copy to clipboard
put_diagram(wf,
  output = "clipboard")

# Save to file
put_diagram(wf,
  output = "file",
  file = "diagram.md")
```

Interactive Features

```
# Show source file info
put_diagram(wf,
  show_source_info = TRUE)

# Clickable nodes (VS Code)
put_diagram(wf,
  enable_clicks = TRUE,
  click_protocol = "vscode")
```

Diagram Options

Themes

`light` | `dark` | `auto` | `github` | `minimal`

```
put_diagram(wf, theme="github")
```

Directions

`TD` (top-down) | `LR` (left-right)
`BT` (bottom-top) | `RL` (right-left)

```
put_diagram(wf, direction="LR")
```

Visualization Modes

```
# Simple (script connections)
put_diagram(wf)

# With data artifacts
put_diagram(wf,
  show_artifacts = TRUE)

# With file labels on edges
put_diagram(wf,
  show_files = TRUE)

# Workflow boundaries
put_diagram(wf,
  show_workflow_boundaries=TRUE)
```

Debugging

```
# Enable logging
set_putior_log_level("DEBUG")

# Per-call logging
workflow <- put("./",
  log_level = "INFO")

# Validate syntax
is_valid_put_annotation(
  '#put label:"Test"')
```

Detection Patterns

```
# View auto-detect patterns
get_detection_patterns("r")
get_detection_patterns("python",
  type = "input")
```

Install

```
install.packages("putior")
# or
pak::pak("pjt222/putior")
```

Supported Languages

R, Python, SQL, Shell, Julia

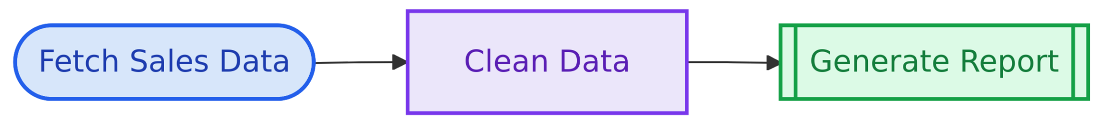
Example Workflows

Simple Linear Pipeline

```
# 01_fetch.R
#put label:"Fetch Sales Data",
     node_type:"input",
     output:"sales.csv"

# 02_clean.py
#put label:"Clean Data",
     input:"sales.csv",
     output:"clean.csv"

# 03_report.R
#put label:"Generate Report",
     node_type:"output",
     input:"clean.csv"
```



Branching & Merging

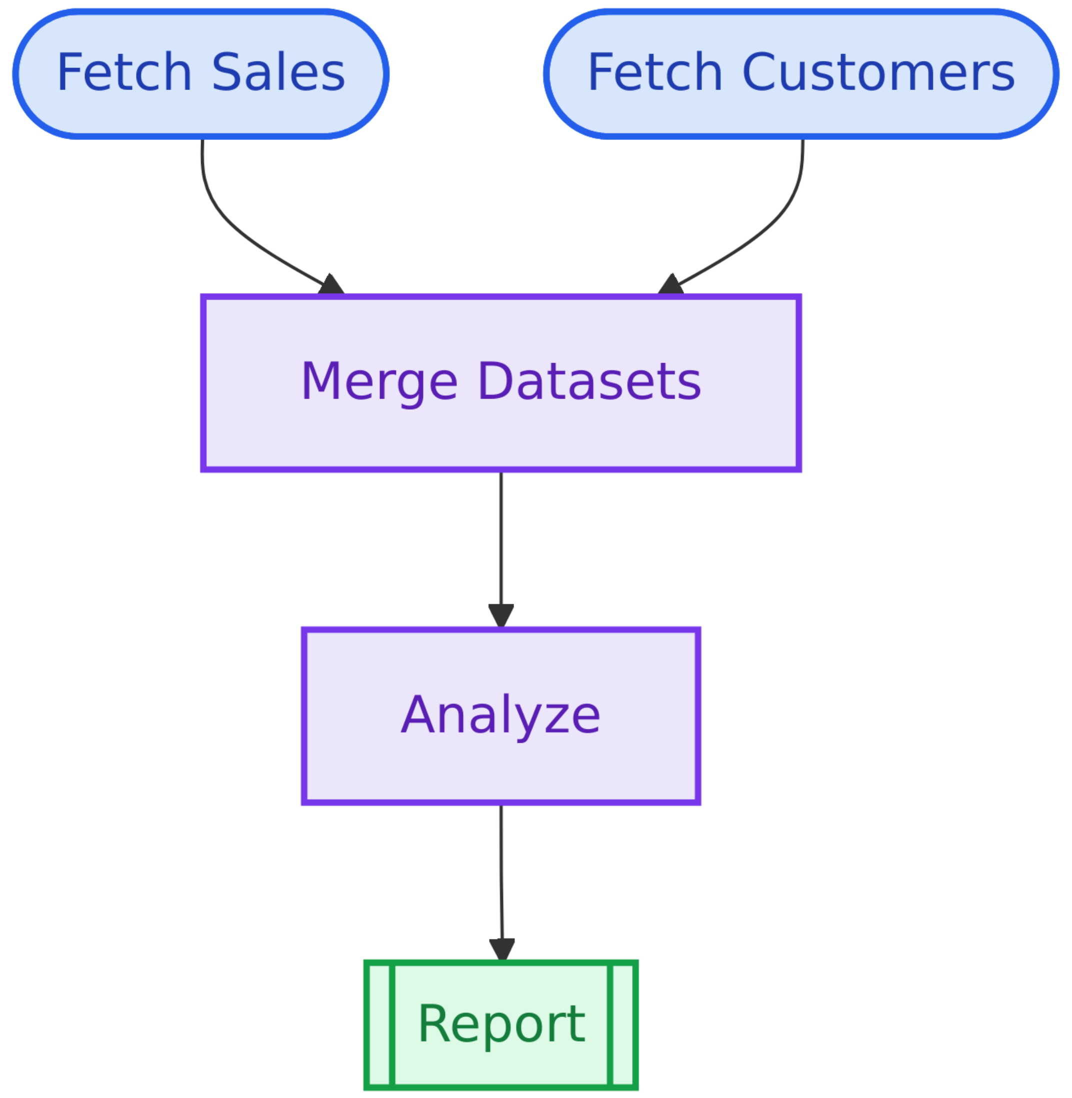
```
# 01_fetch_sales.R
#put label:"Fetch Sales",
     node_type:"input", output:"sales.csv"

# 02_fetch_customers.R
#put label:"Fetch Customers",
     node_type:"input", output:"customers.csv"

# 03_merge.R
#put label:"Merge Datasets",
     input:"sales.csv, customers.csv",
     output:"merged.csv"

# 04_analyze.py
#put label:"Analyze",
     input:"merged.csv", output:"stats.json"

# 05_report.R
#put label:"Report",
     node_type:"output", input:"stats.json"
```

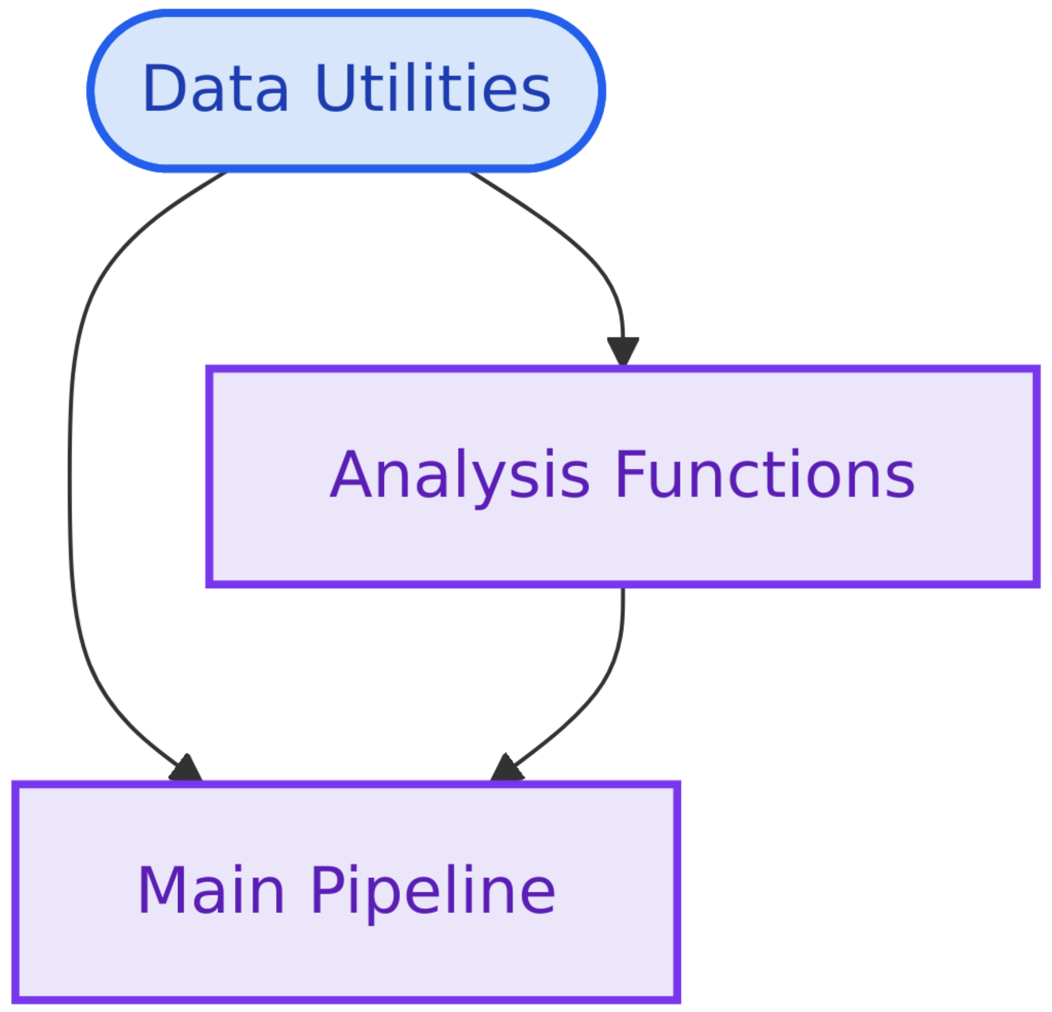


Modular source() Pattern

```
# utils.R - Utility functions
#put label:"Data Utilities",
     node_type:"input"

# analysis.R - Uses utils
#put label:"Analysis Functions",
     input:"utils.R"

# main.R - Orchestrates both
#put label:"Main Pipeline",
     input:"utils.R, analysis.R",
     output:"results.csv"
```



Decision/Branching Logic

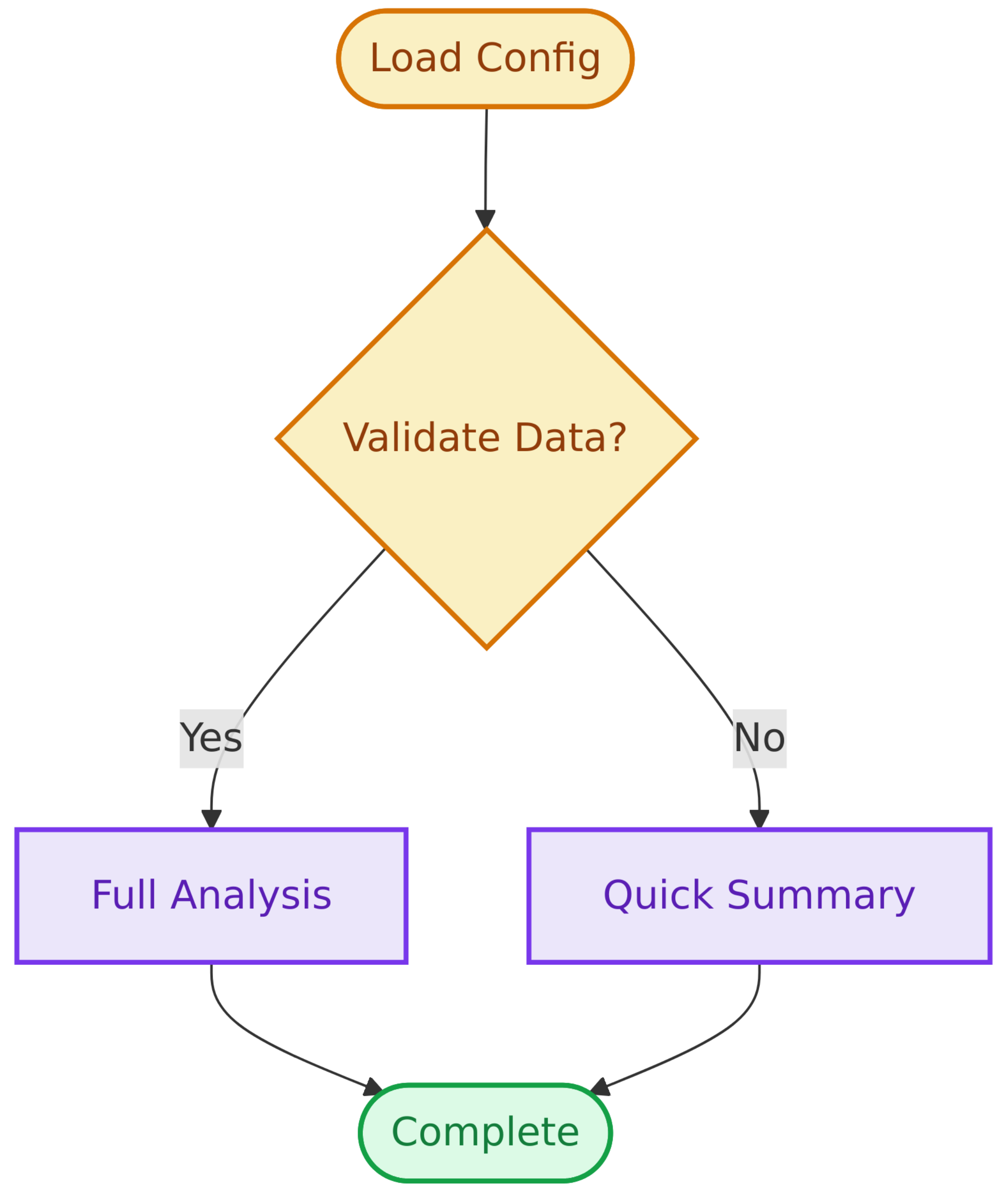
```
# start.R
#put label:"Load Config",
     node_type:"start", output:"config.json"

# check.R
#put label:"Validate Data?",
     node_type:"decision", input:"config.json"

# path_a.R
#put label:"Full Analysis",
     input:"config.json", output:"full.csv"

# path_b.R
#put label:"Quick Summary",
     input:"config.json", output:"summary.csv"

# end.R
#put label:"Complete",
     node_type:"end",
     input:"full.csv, summary.csv"
```


Load Config
Validate Data?
Yes
No
Full Analysis
Quick Summary
Complete