Package: jaspSyntax (via r-universe)

November 26, 2026

Type Package

Title Makes JASP analyses available in R

Version 1.4.7

Date 2026-11-18

Author JASP Team

Website jasp-stats.org

Maintainer JASP Team <info@jasp-stats.org>

Description Exposes the native JASP SyntaxInterface bridge to R so package code can parse module descriptions, replay QML option binding, load datasets, and read saved .jasp files using the same runtime preparation path as JASP Desktop.

License GPL-3 + file LICENSE

Encoding UTF-8

URL https://github.com/jasp-stats/jaspSyntax

BugReports https://github.com/jasp-stats/jaspSyntax/issues

SystemRequirements libcurl or wget (for downloading the SyntaxInterface library during installation), OpenGL (libglvnd; provides libOpenGL.so.0)

Imports callr, jsonlite, Rcpp (>= 1.0.5)

Suggests pkgload, testthat

LinkingTo Rcpp

RoxygenNote 7.3.3

Config/pak/sysreqs libgl1-mesa-dev wget

Repository https://jasp-stats.r-universe.dev

Date/Publication 2026-11-26 16:07:15 UTC

RemoteUrl https://github.com/jasp-stats/jaspSyntax

RemoteRef HEAD

RemoteSha b566c058ea52fb377af1d33ca1d873575b5aead6

Contents

jaspSyntax-package . . . 2
analysisSyntax . . . 3
clearQmlForms . . . 4
columnEncoderContext . . . 4
loadAnalysisDataset . . . 5
nativeBridge . . . 7
nativeBridgeProvenance . . . 9
parseQmlOptions . . . 10
readAnalysisOptionsFromJaspFile . . . 11
readAnalysisOptionsFromQml . . . 12
readAnalysisResultsFromJaspFile . . . 13
readAnalysisSyntaxFromQml . . . 14
readDatasetFromJaspFile . . . 15
readDefaultAnalysisOptions . . . 16
readModuleDescription . . . 17
resolveAnalysisQml . . . 17

Index **18**

`jaspSyntax-package` *Native JASP SyntaxInterface bridge for R*

Description

jaspSyntax exposes the native JASP SyntaxInterface bridge to R. It parses module descriptions, resolves analysis QML files, replays QML option binding, loads datasets into native state, and reads saved `.jasp` files using the same runtime preparation path as JASP Desktop.

Details

Saved `.jasp` options are read from the archive and then replayed through QML when runtime options are requested. They are not a replacement for Desktop's full archive/module upgrade workflow for old files.

Use `readModuleDescription()` and `resolveAnalysisQml()` for module metadata, `readAnalysisOptionsFromQml()` or `readDefaultAnalysisOptions()` for QML/runtime options, and `loadAnalysisDataset()`, `readLoadedDataset()`, and `readRequestedDataset()` for native dataset state. Use `readAnalysisOptionsFromJaspFile(` `readAnalysisResultsFromJaspFile()` and `readDatasetFromJaspFile()` for saved `.jasp` files. The low-level native bridge calls remain exported for compatibility, but package code should prefer the higher-level helpers. Helpers such as `parseQmlOptions()`, lifecycle controls, dataset-state readers, column mapping, and `nativeBridgeProvenance()` are native-facing/experimental integration APIs.

Author(s)

JASP Team

Maintainer: JASP Team <info@jasp-stats.org>

See Also

readAnalysisOptionsFromJaspFile, readAnalysisOptionsFromQml, readAnalysisResultsFromJaspFile, loadAnalysisDataset, nativeBridgeProvenance, readDatasetFromJaspFile, readModuleDescription

Examples

```
## Not run:
   records <- readAnalysisOptionsFromJaspFile("analysis.jasp")
   dataset <- readDatasetFromJaspFile("analysis.jasp")

## End(Not run)
```

---

analysisSyntax *Generate the R Syntax of an Analysis*

Description

Binds options to a QML form, as parseQmlOptions does, and gives the R syntax JASP shows for the analysis: the call of its R wrapper (`module::analysis(data = NULL, version = ..., ...)`) with the options that differ from the defaults of the form.

Usage

```
analysisSyntax(
  qmlFile,
  options = NULL,
  moduleName = "jaspModule",
  analysisName = NULL,
  version = "0",
  preloadData = TRUE,
  fresh = TRUE,
  isolated = FALSE
)
```

Arguments

qmlFile	Path to an analysis QML file.
options	Named list of options, a JSON object string, or NULL for defaults.
moduleName	Module name passed to the native bridge.
analysisName	Analysis name passed to the native bridge. Defaults to the QML file basename without extension.
version	Module version passed to the native bridge.
preloadData	Whether the analysis preloads data.
fresh	Whether to clear cached QML/native state before parsing. This should remain TRUE when reading defaults.
isolated	Whether to run native QML parsing in a separate R process. The data loaded in this process is not available there.

Details

Options of an older `version` are upgraded with the `Upgrades.qml` of the module first, and the syntax is then written for the version of the module of the form. A form drops variables that are not in the loaded data, so load the data of the analysis first (and keep `isolated = FALSE`) when the options name variables.

Value

The R syntax, a single string.

---

clearQmlForms *Native Bridge Lifecycle Helpers*

Description

These helpers give downstream packages explicit names for the native state they intend to clear. `clearQmlForms()` clears cached QML forms and the QML component cache, `clearDatasetState()` clears bridge-owned dataset state, and `clearNativeState()` clears both.

Usage

```
clearQmlForms()

clearDatasetState()

clearNativeState()
```

Value

Invisibly returns `NULL`.

See Also

`nativeBridge`

---

columnEncoderContext *Native Column Encoder Context*

Description

Capture and reuse the source state needed by SyntaxInterface's native `ColumnEncoder`: dataset column names/types, the namespace prefix those tokens live under, and extra QML option encodings.

Usage

```
columnEncoderContext()

decodeColumnText(text, encoderContext = NULL)
```

Arguments

`text`	Character vector to decode.
`encoderContext`	Optional context returned by `columnEncoderContext()`. When omitted, the current native bridge encoder state is used.

Details

The whole input vector goes to the native decoder unconditionally: the native lookup table resolves exactly the tokens the bridge minted and returns everything else unchanged, so plain text is a no-op round trip and decoding requires a live bridge. jaspSyntax never pattern-matches token spellings in R.

Value

`columnEncoderContext()` returns a serializable native encoder context. `decodeColumnText()` returns a character vector with native column tokens decoded.

---

loadAnalysisDataset *High-Level Native Dataset Helpers*

Description

These helpers keep native dataset preload, requested-data state, and column-name decoding behind the jaspSyntax API. They are the preferred interface for downstream packages that need loaded/requested dataset state after QML/runtime option preparation.

Usage

```
loadAnalysisDataset(
  dataset,
  modulePath,
  analysisName,
  options = NULL,
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  decode = TRUE,
  normalize = TRUE
)

readLoadedDataset(decode = TRUE, normalize = TRUE)
```

```
readRequestedDataset(decode = TRUE, normalize = TRUE)

readDatasetHeader(decode = TRUE)

decodeColumnNames(columnNames, strict = FALSE)
```

Arguments

dataset	Raw data frame supplied by the caller.
modulePath	Path to a JASP module source directory.
analysisName	Name of the analysis function.
options	Saved/QML-bound options as a named list, JSON object string, or NULL.
includeMeta	Whether to retain the .meta option in runtime options.
includeTypeOptions	Whether to retain *.types options in runtime options.
decode	Whether to decode native/encoded column names.
normalize	Whether to normalize bridge-returned factor columns back to plain character vectors while preserving numeric-looking factor labels.
columnNames	Character vector of column names.
strict	Compatibility-only flag retained for callers that pass it. Every name is handed to the native decoder; raw/non-token names come back unchanged because the native lookup table resolves only the tokens the bridge minted.

Details

loadAnalysisDataset() is the supported high-level entry point for jaspTools-style replay. The direct state readers are exported for bridge diagnostics and should be treated as native-facing integration APIs.

loadAnalysisDataset() loads the raw data frame into SyntaxInterface, replays saved/QML-bound options through readAnalysisOptionsFromQml() with fresh = TRUE, then reads both the loaded and requested native dataset state. This keeps QML option value/types interpretation in the native-backed jaspSyntax path.

The current native bridge exposes requested and decoded dataset state through R callbacks installed in .GlobalEnv. These helpers centralize that callback use. A future Desktop ABI can replace the callback implementation without changing downstream caller code.

Value

loadAnalysisDataset() returns a list with loadedDataset, requestedDataset, runtimeOptions, columnEncoderContext, modulePath, and analysisName (the former resultDecodingDataset entry was removed together with the deleted decodeAnalysisResults).

readLoadedDataset() and readRequestedDataset() return data frames. readDatasetHeader() returns a data frame with name and encodedName columns. decodeColumnNames() returns a character vector.

See Also

readAnalysisOptionsFromQml, readDatasetFromJaspFile, nativeBridge

---

nativeBridge *Low-Level SyntaxInterface Bridge*

Description

These functions expose the low-level native SyntaxInterface bridge. They are exported for compatibility and native integration work. They follow the native SyntaxInterface ABI and are not the stable package-level contract; ordinary package code should prefer the higher-level helpers such as `readModuleDescription()`, `readAnalysisOptionsFromQml()`, `readDefaultAnalysisOptions()`, `loadAnalysisDataset()`, `readLoadedDataset()`, `readRequestedDataset()`, `readAnalysisOptionsFromJaspFile()`, and `readDatasetFromJaspFile()`.

`setParameter("verbose", TRUE)` or `setParameter("verbose", "jasp")` enables native SyntaxInterface logging for debugging. For wrapped analyses, `setParameter("verbose", value)` also sets the default verbosity honored by `jaspBase` and `jaspTools`: `"all"` shows native and analysis output, `"analysis"` shows analysis messages and warnings only, `"jasp"` shows native output only, and `"none"` suppresses both. For compatibility with the old native toggle, `setParameter("verbose", FALSE)` disables native logging while keeping analysis diagnostics visible. Native logging is disabled by default and can also be enabled before bridge initialization with the `JASP_SYNTAX_VERBOSE=1` environment variable.

`cleanUp()` clears the native state and then runs the legacy cleanup hook. Use clearQmlForms, clearDatasetState, or clearNativeState when the intended lifecycle scope matters. Use nativeBridgeProvenance to inspect the recorded SyntaxInterface header/binary source for ABI diagnostics.

`loadDataSet()` returns the bridge-assigned database id of the loaded dataset invisibly; `loadDataSets()` loads a whole named list of data.frames (one DataSet per entry, first entry primary, state cleared first) and returns their ids as a named integer vector. These ids are INFORMATIONAL (logging, slicing results): the dataset selection options of a multi-dataset aware analysis take the syntax surface instead - the bare dataset title for that dataset's default filter, or "title - filterName" for one of the named filters saved in a live JASP file (a freshly loaded dataset has only its default filter, so in full R a selection is the list name). Exact strings only, never parsed, never ids; an unknown value fails with a message listing what is selectable.

`loadQmlAndParseOptionsStatus()` is `loadQmlAndParseOptions()` plus the full bridge status: a list with the parsed `options` JSON string and `multiDataSetJson` (the `{ids, names, dataSetIds, primary}` plan of the queued per-filter slices, "" for non-aware analyses) and `upgradeMsgs` ({<option name, "" for the analysis>: [messages]}, "" when there are none).

Options written for an older `version` of a module are upgraded by its `Upgrades.qml` before they are parsed. The messages of that upgrade, and the options that the form does not know and so ignores, are given as warnings by both `loadQmlAndParseOptions()` and `loadQmlAndParseOptionsStatus()`.

`decodeJsonText()` runs the engine's send-string decoder over a whole JSON payload: every DataSet's own column encoder in the bridge workspace, then the extra-options encoder, then the process-global one (each encoder only replaces the exact tokens it minted, so the sweep is safe regardless of which dataset was shown last). Non-JSON text is returned unchanged, empty input as-is, NA as NA.

Usage

```
analysisOptionsFromJaspFile(jaspFilePath, analysisNr)

cleanUp()

generateAnalysisWrapper(modulePath, analysisName)

generateModuleWrappers(modulePath)

getVariableNames()

loadDataSet(data)

loadDataSets(datasets)

loadDataSetFromJaspFile(jaspFilePath)

loadQmlAndParseOptions(
  moduleName,
  analysisName,
  qmlFile,
  options,
  version,
  preloadData
)

loadQmlAndParseOptionsStatus(
  moduleName,
  analysisName,
  qmlFile,
  options,
  version,
  preloadData
)

decodeJsonText(json)

parseDescription(modulePath)

setParameter(name, value)
```

Arguments

jaspFilePath	Path to a .jasp file.
analysisNr	Zero-based analysis index.
modulePath	Path to a JASP module.
data	A data frame-like object to load into the native bridge.

datasets	A named list of data.frames, one dataset per entry (see loadDataSets).
json	A character string holding a JSON payload (see decodeJsonText).
moduleName	Module name passed to SyntaxInterface.
analysisName	Analysis name passed to SyntaxInterface.
qmlFile	Path to an analysis QML file.
options	JSON object string with analysis options.
version	Module version passed to SyntaxInterface.
preloadData	Whether the analysis preloads data.
name	Native bridge parameter name.
value	Native bridge parameter value.

---

nativeBridgeProvenance *Read Native Bridge Provenance*

Description

Returns installation metadata for the bundled SyntaxInterface bridge, when the package was installed by a configure script that recorded it. This is a diagnostic helper for checking whether the header and native binary came from the same Desktop/build source. Recent installs also record SHA-256 hashes for the copied header and binary.

Usage

```
nativeBridgeProvenance()
```

Value

A named character vector. The path attribute points to the provenance file. An empty vector means the installed package did not record provenance.

See Also

nativeBridge

`parseQmlOptions` *Parse QML Options*

Description

Loads a QML form and parses supplied options through the native SyntaxInterface bridge. The native bridge binds the QML controls, applies option metadata, and runs the same column-name/type encoding used before JASP Desktop calls the R backend.

Usage

```
parseQmlOptions(
  qmlFile,
  options = NULL,
  moduleName = "jaspModule",
  analysisName = NULL,
  version = "0",
  preloadData = TRUE,
  fresh = TRUE,
  output = c("list", "json"),
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  isolated = TRUE
)
```

Arguments

`qmlFile`	Path to an analysis QML file.
`options`	Named list of options, a JSON object string, or `NULL` for defaults.
`moduleName`	Module name passed to the native bridge.
`analysisName`	Analysis name passed to the native bridge. Defaults to the QML file basename without extension.
`version`	Module version passed to the native bridge.
`preloadData`	Whether the analysis preloads data.
`fresh`	Whether to clear cached QML/native state before parsing. This should remain `TRUE` when reading defaults.
`output`	Return parsed R `list` output or raw `json`.
`includeMeta`	Whether to retain the `.meta` option in list output.
`includeTypeOptions`	Whether to retain `*.types` options in list output.
`isolated`	Whether to run native QML parsing in a separate R process.

Details

No analysis run is prepared: a dataset selection option of a multi-dataset form (e.g. `dataSetA`) that names no dataset stays empty instead of failing, so the defaults of such a form can be read without loading a dataset. A dataset selection that is given must name a loaded dataset.

This is a low-level/native-facing helper for diagnostics and bridge integration. Most callers should use `readAnalysisOptionsFromQml` so QML file names, module versions, and `preloadData` come from the parsed module description.

Value

A named list of parsed options, or a JSON string when `output = "json"`.

`readAnalysisOptionsFromJaspFile` *Read Analysis Options From a JASP File*

Description

Reads all saved analyses from a `.jasp` file and returns their metadata together with their saved QML-bound options. With `runtime = TRUE`, saved options are replayed through the resolved QML form and native Desktop option encoder so the result matches the R-runtime options prepared by JASP Desktop before calling the analysis. This helper reads the options stored in the archive; it does not replace Desktop's full archive/module upgrade workflow for older files.

Usage

```
readAnalysisOptionsFromJaspFile(
  jaspFilePath,
  modulePath = NULL,
  runtime = FALSE,
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  flattenTypes = FALSE,
  isolated = TRUE
)
```

Arguments

`jaspFilePath`	Path to a `.jasp` file.
`modulePath`	Optional module path, or a named list/vector of module paths keyed by module name or analysis name. Required for `runtime = TRUE` when the module is not installed.
`runtime`	Whether to replay saved options through QML and the native Desktop option encoder. The default `FALSE` returns the saved bound options from `analyses.json`.
`includeMeta`	Whether to retain the `.meta` option.

includeTypeOptions
: Whether to retain `*.types` options when present.

flattenTypes
: Whether to split the saved `{value, types}` shape of variables options into `<name>` (the value) and `<name>.types` (the types), as JASP Desktop does before it calls an analysis that does not preload its data: this runs Desktop's own walk (`ColumnEncoder::flattenTypesInOptions`). Column names stay plain (not encoded, no type suffix). An option with an `optionKey` and other members besides `optionKey`/`types`/`value` keeps its saved shape, a nested option gets its `.types` next to it. Only for `runtime = FALSE`.

isolated
: Whether to run the native `.jasp` option extraction in a separate R process. This is the default because the SyntaxInterface bridge owns process-global native state. In-process reads also clear native state before returning; use `readDatasetFromJaspFile()` for the saved dataset.

Value

A list of analysis records. Each record has `name`, `title`, `moduleName`, `moduleVersion`, and `options`.

`readAnalysisOptionsFromQml` *Read Analysis Options Through QML*

Description

Resolves an analysis in a module, loads its QML form, and parses options through the native SyntaxInterface path. The returned options are the same R-runtime JSON shape prepared for analyses by JASP Desktop: QML controls are bound, option metadata is applied, and column-name/type encoding is handled by the native `ColumnEncoder`.

Usage

```
readAnalysisOptionsFromQml(
  modulePath,
  analysisName,
  options = NULL,
  version = NULL,
  preloadData = NULL,
  fresh = TRUE,
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  isolated = TRUE
)

analysisOptionsFromQml(
  modulePath,
  analysisName,
  options = NULL,
```

```
  version = NULL,
  preloadData = NULL,
  fresh = TRUE,
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  isolated = TRUE
)
```

Arguments

`modulePath`	Path to a JASP module source directory.
`analysisName`	Name of the analysis function.
`options`	Named list of options, a JSON object string, or `NULL` for defaults.
`version`	Optional module version override. Defaults to the version from the module description.
`preloadData`	Optional preload flag override. Defaults to the analysis value from the module description.
`fresh`	Whether to clear cached QML/native state before parsing.
`includeMeta`	Whether to retain the `.meta` option in list output.
`includeTypeOptions`	Whether to retain `*.types` options in list output.
`isolated`	Whether to run native QML parsing in a separate R process.

Value

A named list of parsed options.

`readAnalysisResultsFromJaspFile` *Read Analysis Results From a JASP File*

Description

Reads the results JASP Desktop saved for every analysis in a `.jasp` file: the same results tree the analysis sent to Desktop (tables with their schema and raw, unformatted cell values, plots, collections, notes). Desktop stores them decoded, so column names are the user's.

Usage

```
readAnalysisResultsFromJaspFile(jaspFilePath)
```

Arguments

`jaspFilePath`	Path to a `.jasp` file.

Details

This only reads the archive; it does not touch the native bridge. A plot entry refers to its image under `resources/` (its `data` member), not to a plot object, so plots cannot be compared with **vdiffr** against these results.

The results are returned as stored. When JASP Desktop loads an older archive it also decodes the column names that its results may still have in their old encoded form; this reader does not.

Value

A list of analysis records in file order, named by analysis name. Each record has `id` (the analysis id, also the folder name of its files under `resources/` in the archive), `name`, `title`, `moduleName`, `moduleVersion`, `status` (e.g. `"complete"`, `"validationError"`) and `results` (`NULL` when the file holds none for that analysis). The JASP version that wrote the file and its archive version are in the `jaspVersion` and `jaspArchiveVersion` attributes.

See Also

`readAnalysisOptionsFromJaspFile`, `readDatasetFromJaspFile`

Examples

```
## Not run:
records <- readAnalysisResultsFromJaspFile("analysis.jasp")
records[[1]]$results
attr(records, "jaspVersion")

## End(Not run)
```

---

readAnalysisSyntaxFromQml

Read the R Syntax of a Module Analysis

Description

Resolves an analysis in a module, as `readAnalysisOptionsFromQml` does, and gives its R syntax with `analysisSyntax`.

Usage

```
readAnalysisSyntaxFromQml(
  modulePath,
  analysisName,
  options = NULL,
  version = NULL,
  preloadData = NULL,
  fresh = TRUE,
  isolated = FALSE
)
```

Arguments

`modulePath`	Path to a JASP module source directory.
`analysisName`	Name of the analysis function.
`options`	Named list of options, a JSON object string, or `NULL` for defaults.
`version`	Module version the options were written for. Defaults to the version from `Description.qml/DESCRIPTION`: the options are then not upgraded.
`preloadData`	Optional preload flag override. Defaults to the analysis value from the module description.
`fresh`	Whether to clear cached QML/native state before parsing.
`isolated`	Whether to run native QML parsing in a separate R process. The data loaded in this process is not available there.

Value

The R syntax, a single string.

`readDatasetFromJaspFile`

Read a Dataset from a JASP File

Description

Reads the dataset stored inside a saved JASP file and materializes it as an R `data.frame`.

Usage

```
readDatasetFromJaspFile(jaspFilePath, dataSetIndex = 1L)
```

Arguments

`jaspFilePath`	Character scalar path to a `.jasp` file.
`dataSetIndex`	1-based dataset index inside the `.jasp` file. Currently only `1L` is supported.

Details

This function isolates the native jaspSyntax dataset-loading path in a short-lived subprocess so repeated calls remain safe within the calling R session.

Dataset materialization is delegated to `readLoadedDataset` so column-name decoding and bridge factor normalization stay behind the jaspSyntax dataset API. Nominal columns are returned as factors and ordinal columns as ordered factors, with their levels in the order of the file (the order JASP shows), so that `loadDataSet` keeps that order when the `data.frame` is loaded again. Numeric columns are returned using the bridge's native conversion.

Value

Either a `data.frame` containing the dataset or `NULL` when the file does not contain tabular data.

Examples

```
## Not run:
dataset <- readDatasetFromJaspFile("path/to/analysis.jasp")
str(dataset)

## End(Not run)
```

`readDefaultAnalysisOptions` *Read Default Analysis Options*

Description

Loads an analysis QML form and returns the options produced by the native SyntaxInterface defaults.

Usage

```
readDefaultAnalysisOptions(
  modulePath,
  analysisName,
  fresh = TRUE,
  includeMeta = TRUE,
  includeTypeOptions = TRUE,
  isolated = TRUE
)
```

Arguments

`modulePath`	Path to a JASP module source directory.
`analysisName`	Name of the analysis function.
`fresh`	Whether to clear cached QML/native state before parsing.
`includeMeta`	Whether to retain the `.meta` option in list output.
`includeTypeOptions`	Whether to retain `*.types` options in list output.
`isolated`	Whether to run native QML parsing in a separate R process.

Value

A named list of default options.

readModuleDescription *Read a JASP Module Description*

Description

Reads a module's `Description.qml` metadata through the native SyntaxInterface bridge, which loads it with JASP Desktop's own module code. Defaults and derived values (an analysis' `preloadData`, the module's `requiresData`, the version string) are therefore exactly what Desktop uses.

Usage

```
parseModuleDescription(modulePath, byName = TRUE)

readModuleDescription(modulePath, byName = TRUE)
```

Arguments

`modulePath`	Path to a JASP module source directory or its `inst/Description.qml` file.
`byName`	Whether to name the returned `analyses` list by analysis name.

Value

A list with module metadata and an `analyses` list.

resolveAnalysisQml *Resolve an Analysis QML File*

Description

Resolves an analysis name to the QML file and metadata provided by the native module description parser.

Usage

```
resolveAnalysisQml(modulePath, analysisName)
```

Arguments

`modulePath`	Path to a JASP module source directory or its `inst/Description.qml` file.
`analysisName`	Name of the analysis function.

Value

A list with module description, analysis metadata, QML file path, and resolved preload flag.

Index

∗ **package**
    jaspSyntax-package, 2

analysisOptionsFromJaspFile *(nativeBridge)*, 7
analysisOptionsFromQml *(readAnalysisOptionsFromQml)*, 12
analysisSyntax, 3, *14*

cleanUp *(nativeBridge)*, 7
clearDatasetState, *7*
clearDatasetState *(clearQmlForms)*, 4
clearNativeState, *7*
clearNativeState *(clearQmlForms)*, 4
clearQmlForms, 4, *7*
columnEncoderContext, 4

decodeColumnNames *(loadAnalysisDataset)*, 5
decodeColumnText *(columnEncoderContext)*, 4
decodeJsonText *(nativeBridge)*, 7

generateAnalysisWrapper *(nativeBridge)*, 7
generateModuleWrappers *(nativeBridge)*, 7
getVariableNames *(nativeBridge)*, 7

jaspSyntax *(jaspSyntax-package)*, 2
jaspSyntax-package, 2

loadAnalysisDataset, *3*, 5
loadDataSet, *15*
loadDataSet *(nativeBridge)*, 7
loadDataSetFromJaspFile *(nativeBridge)*, 7
loadDataSets *(nativeBridge)*, 7
loadQmlAndParseOptions *(nativeBridge)*, 7
loadQmlAndParseOptionsStatus *(nativeBridge)*, 7

nativeBridge, *4*, 7, *7*, *9*
nativeBridgeProvenance, *3*, *7*, 9

parseDescription *(nativeBridge)*, 7
parseModuleDescription *(readModuleDescription)*, 17
parseQmlOptions, *3*, 10

readAnalysisOptionsFromJaspFile, *3*, 11, *14*
readAnalysisOptionsFromQml, *3*, *7*, *11*, 12, *14*
readAnalysisResultsFromJaspFile, *3*, 13
readAnalysisSyntaxFromQml, 14
readDatasetFromJaspFile, *3*, *7*, *14*, 15
readDatasetHeader *(loadAnalysisDataset)*, 5
readDefaultAnalysisOptions, 16
readLoadedDataset, *15*
readLoadedDataset *(loadAnalysisDataset)*, 5
readModuleDescription, *3*, 17
readRequestedDataset *(loadAnalysisDataset)*, 5
resolveAnalysisQml, 17

setParameter *(nativeBridge)*, 7