airtabler

Overview

The `airtabler` package provides comprehensive tools for interacting with Airtable bases through the Airtable API. With this package, you can list bases, export data, and save exports in various formats including Excel workbooks, CSV files, and R data objects, which can be easily re-imported into Airtable or used for further analysis.

Installation

```
# Install from GitHub
devtools::install_github("n8layman/airtabler")

# Load required packages
library(airtabler)
library(tidyverse)
```

Authentication

Before using the package, you need to set up your Airtable API token:

```
# Set your Airtable API token (recommended to use .Renviron)
Sys.setenv(AIRTABLE_API_KEY="your_api_token")
```

For security, it's recommended to store your API token in your `.Renviron` file:

```
# In your .Renviron file
AIRTABLE_API_KEY=your_api_token
```

You can edit your `.Renviron` file with:

```
usethis::edit_r_environ()
```

Basic Usage

Listing Available Bases

```
# Get a list of all bases your token has access to
bases <- air_list_bases() |> pluck("bases")
```

This returns a data frame with base information including:

- `id`: The base ID
- `name`: The base name
- `permissionLevel`: Your access level to the base

Exporting Base Data

The package provides functions for exporting Airtable base data:

```
# Generate metadata for a specific base
base_metadata <- air_generate_metadata_from_api(base_id)

# Export all data from a base to an R object
base_dump <- air_dump(
  base_id,
  metadata = base_metadata,
  base_path = "path/to/output/directory",
  attachment_folder = "attachments",
  overwrite = FALSE,
  remove_original_field = TRUE,
  organize_by_table_field = TRUE
)
```

Export Format Options

To Excel Workbook

> ⚠️ **Important:** Excel has a ~30,000 character limit per cell. Fields with very long text content (such as sequence data, JSON objects, or lengthy notes) may be truncated. For data preservation, use RDS or CSV formats as described below.

```
# Export to Excel workbook
air_dump_to_xlsx(
  base_dump,
  output_file = "path/to/output.xlsx",
  base_name = "Base Name"
)
```

To CSV Files

```
# Export to CSV files (one file per table)
air_dump_to_csv(
  base_dump,
  output_dir = "path/to/output/directory",
  attachments_dir = NULL,
  overwrite = TRUE,
  output_id = "my_base_csvs",
  names_to_snake_case = TRUE
)
```

```r
# Save as R object for easier importing later
saveRDS(base_dump, file = "path/to/output.rds")
```

To R Object

Batch Processing Example

The following example demonstrates how to process and export multiple Airtable bases:

```r
# Get list of bases your token has access to
bases <- air_list_bases() |> pluck("bases")

# Create a base directory for all exports
base_export_dir <- "path/to/export/directory"
dir.create(base_export_dir, showWarnings = FALSE, recursive = TRUE)

# Process each base
pwalk(bases, function(id, name, permissionLevel) {
  # Display the current base name
  message("Processing: ", stringr::str_squish(name))

  # Create cleaned base name for filenames and directories
  cleaned_base_name <- gsub(" ", "_", stringr::str_squish(name))

  # Create output directory
  output_dir <- file.path(base_export_dir, paste0(cleaned_base_name, "_", id))
  dir.create(output_dir, showWarnings = FALSE, recursive = TRUE)

  # Try to process the base, with error handling
  tryCatch({
    # Generate metadata and export data
    base_metadata <- air_generate_metadata_from_api(id)
    base_dump <- air_dump(
      id,
      metadata = base_metadata,
      base_path = output_dir,
      attachment_folder = "attachments",
      overwrite = FALSE,
      remove_original_field = TRUE,
      organize_by_table_field = TRUE
    )

    # Base output file path
    output_file <- file.path(output_dir, cleaned_base_name)

    # Export to different formats
    # 1. Excel (WARNING: has ~30,000 character limit per cell that will truncate long data)
    air_dump_to_xlsx(
      base_dump,
      output_file = paste0(output_file, ".xlsx"),
      base_name = name
```

```
    )

    # 2. R object (preserves all data without truncation)
    saveRDS(base_dump, file = paste0(output_file, ".rds"))

    # 3. CSV files (one per table)
    air_dump_to_csv(
      base_dump,
      output_dir = output_dir,
      attachments_dir = NULL,
      overwrite = TRUE,
      output_id = paste0(cleaned_base_name, "_csv_files"),
      names_to_snake_case = TRUE
    )

    message("Successfully exported base: ", name)
  }, error = function(e) {
    message("Error processing base '", name, "' (ID: ", id, "): ", e$message)
  })
})
```

Re-importing to Airtable

The files generated by export functions are formatted for easy re-import into Airtable:

From Excel

1. Open your Airtable base in a web browser
2. Select the table where you want to import data
3. Click on the "+" button in the view bar
4. Select "Import data"
5. Choose "Upload a file" and select your Excel file
6. Follow the Airtable import wizard to map fields
7. Confirm and complete the import

> ⚠ **Warning:** When reimporting Excel files to Airtable, be aware that any fields that exceeded Excel's ~30,000 character limit were truncated during the export process. This is particularly important for data like genetic sequences, large JSON objects, or any long-form text. If your data contains such fields, consider using the CSV or RDS formats for data preservation and only use Excel for viewing or sharing.

From CSV

Similar to Excel, but import each table's CSV file separately.

From R

To re-use the R object in another R session:

```r
# Load saved base dump
base_dump <- readRDS("path/to/file.rds")

# Now you can work with the data or export it again
```

Function Reference

`air_list_bases()`

Returns a list of all Airtable bases your API token can access.

`air_generate_metadata_from_api(base_id)`

Generates metadata for a specific base, which is required for export operations.

`air_dump(base_id, metadata, ...)`

Exports all data from a base to an R object.

Parameters:

- `base_id`: The ID of the Airtable base to export
- `metadata`: Base metadata from `air_generate_metadata_from_api()`
- `base_path`: Directory used for storing attachments and other supplementary files
- `attachment_folder`: Subfolder name for attachments
- `overwrite`: Whether to overwrite existing files
- `remove_original_field`: Remove original field IDs after export
- `organize_by_table_field`: Organize exported files by table and field

`air_dump_to_xlsx(base_dump, output_file, base_name)`

Converts a base dump to an Excel workbook.

`air_dump_to_csv(base_dump, output_dir, ...)`

Exports a base dump to a series of CSV files (one per table).

Parameters:

- `base_dump`: The output from `air_dump()`
- `output_dir`: Directory to save CSV files
- `attachments_dir`: Optional directory for attachments
- `overwrite`: Whether to overwrite existing files
- `output_id`: Prefix for CSV filenames
- `names_to_snake_case`: Convert column names to snake_case

Error Handling

Most functions will return errors with informative messages if:

- Your API token is invalid or expired
- You don't have permission to access a base
- A base ID doesn't exist
- There are connection issues with the Airtable API

Handling Large Bases

For bases with large amounts of data:

- **Excel Limitations:** Excel has a strict cell character limit (~30,000 characters), which **will truncate large text fields** like genetic sequences, JSON data, or lengthy text entries
- **Data Preservation:** Always use the RDS format to preserve all data without truncation when working with R
- **CSV Option:** For sharing data outside of R, CSV files maintain the complete data (though may have their own limitations with special characters)
- **Re-importing Caution:** When re-importing to Airtable from Excel files, be aware that any truncated data cannot be recovered
- For very large bases, process tables individually

Dependencies

- tidyverse (dplyr, purrr, stringr, etc.)
- httr
- jsonlite
- openxlsx
- fs

Contributing

Contributions to improve `airtabler` are welcome. Please submit issues and pull requests on GitHub.