# Package: flightanalysis (via r-universe)

May 31, 2026

**Title** Find the Best Flights

**Version** 3.0.1

**Description** Tools and models for users to analyze, forecast, and collect data regarding flights and prices. Features include detailed scraping and querying tools for Google Flights.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**Depends** R (>= 3.6.0)

**Imports** progress, chromote, airportr

**Suggests** testthat (>= 3.0.0), knitr, ggplot2, scales, ggrepel

**URL** https://github.com/rempsyc/flightanalysis, https://rempsyc.github.io/flightanalysis/

**BugReports** https://github.com/rempsyc/flightanalysis/issues

**Config/pak/sysreqs** chromium make libssl-dev

**Repository** https://rempsyc.r-universe.dev

**Date/Publication** 2025-12-02 05:00:30 UTC

**RemoteUrl** https://github.com/rempsyc/flightanalysis

**RemoteRef** HEAD

**RemoteSha** 8e11ef2f7becf1bf1625e669798d6789c558aa00

## Contents

---

`airport_to_city` *Convert Airport Codes to City Names*

---

## Description

Converts IATA airport codes to city names using the airportr package. Falls back to the provided fallback value if conversion fails or package is not available.

## Usage

```
airport_to_city(airport_codes, fallback = airport_codes)
```

## Arguments

| | |
|---|---|
| `airport_codes` | Character vector of IATA airport codes |
| `fallback` | Character vector of fallback values (same length as airport_codes). Default is the original airport_codes. |

## Value

Character vector of city names

## Examples

```
airport_to_city("JFK")
airport_to_city(c("JFK", "LGA", "EWR"))
```

## city_name_to_code *Convert City Names to Airport Codes*

### Description

Converts full city names to 3-letter IATA airport codes using the airportr package. Returns all valid matching airport codes for cities with multiple airports. Automatically filters out heliports and excluded airports (those not used by Google Flights) to return only commercial airports. Throws an error if a city name is not found in the database.

### Usage

```
city_name_to_code(city_names)
```

### Arguments

| | |
|---|---|
| city_names | Character vector of city names |

### Value

Character vector of 3-letter IATA airport codes. For cities with multiple airports, all valid codes are returned (e.g., "New York" returns c("LGA", "JFK")). Heliports, excluded airports, and invalid codes are automatically filtered out.

### Examples

```
city_name_to_code("New York")
city_name_to_code(c("New York", "London"))
```

## fa_define_query *Define Flight Query*

### Description

Defines a flight query for Google Flights. Supports one-way, round-trip, chain-trip, and perfect-chain trip types.

Accepts airport codes (e.g., "JFK", "LGA"), city codes (e.g., "NYC" for all New York City airports), and full city names (e.g., "New York"). Full city names are automatically converted to their first associated airport code. Common city codes include: "NYC" (New York), "LON" (London), "PAR" (Paris), "TYO" (Tokyo), "BUE" (Buenos Aires), etc.

### Usage

```
fa_define_query(...)
```

### Arguments

| | |
|---|---|
| ... | Arguments defining the trip. Locations can be 3-letter codes or full city names. Format depends on trip type: - One-way: origin, dest, date - Round-trip: origin, dest, date_leave, date_return - Chain-trip: org1, dest1, date1, org2, dest2, date2, ... - Perfect-chain: org1, date1, org2, date2, ..., final_dest |

### Value

A flight query object (S3 class "flight_query")

### Examples

```
# One-way trip with airport codes
fa_define_query("JFK", "BOS", "2025-12-20")

# One-way trip with city codes
fa_define_query("NYC", "LON", "2025-12-20")

# One-way trip with full city names (auto-converted)
fa_define_query("New York", "Istanbul", "2025-12-20")

# Round-trip with mixed formats
fa_define_query("JFK", "Paris", "2025-12-20", "2025-12-25")

# Chain-trip
fa_define_query("JFK", "YYZ", "2025-12-20", "RDU", "LGA", "2025-12-25")
```

---

## fa_define_query_range *Define Flight Query Range*

---

### Description

Creates flight queries for multiple origin and/or destination airports/cities across a date range. This helper function generates all permutations of origins, destinations, and dates without actually fetching data. Each origin-destination pair gets its own query object. Similar to fa_define_query but for date ranges.

Supports airport codes (e.g., "JFK", "LGA"), city codes (e.g., "NYC" for all New York City airports), and full city names (e.g., "New York"). Full city names are automatically converted to all associated airport codes (excluding heliports). You can mix formats in the same vector.

### Usage

```
fa_define_query_range(origin, dest, date_min, date_max)
```

## Arguments

| | |
|---|---|
| origin | Character vector of airport codes, city codes, or full city names to search from. Can mix formats (e.g., c("JFK", "NYC", "New York")). Automatically expands city names to all associated airports (excluding heliports) and removes duplicates. |
| dest | Character vector of airport codes, city codes, or full city names to search to. Can mix formats. Multiple destinations are supported; separate query objects will be created for each origin-destination pair. |
| date_min | Character or Date. Start date in "YYYY-MM-DD" format. |
| date_max | Character or Date. End date in "YYYY-MM-DD" format. |

## Value

If single origin and destination: A flight query object containing all dates. If multiple origins and/or destinations: A named list of flight query objects, one per origin-destination pair (named as "ORIGIN-DEST").

## Examples

```
# Airport codes
fa_define_query_range(
  origin = c("BOM", "DEL"),
  dest = "JFK",
  date_min = "2025-12-18",
  date_max = "2025-12-20"
)

# City codes
fa_define_query_range(
  origin = "NYC",
  dest = "LON",
  date_min = "2025-12-18",
  date_max = "2025-12-20"
)

# Full city names (auto-converted to airport codes)
fa_define_query_range(
  origin = "New York",
  dest = "Istanbul",
  date_min = "2025-12-18",
  date_max = "2025-12-20"
)

# Mix formats - codes and city names
fa_define_query_range(
  origin = c("New York", "JFK", "BOM", "Patna"),
  dest = "JFK",
  date_min = "2025-12-18",
  date_max = "2025-12-20"
)
```

```
# Multiple destinations
fa_define_query_range(
  origin = "BOM",
  dest = c("JFK", "LON"),
  date_min = "2025-12-18",
  date_max = "2025-12-20"
)
```

---

`fa_fetch_flights` *Fetch Flight Data*

---

## Description

Fetches flight data from Google Flights using chromote. This function will automatically set up a Chrome browser connection, navigate to Google Flights URLs, and extract flight information. Uses the Chrome DevTools Protocol for reliable, driver-free browser automation. The browser runs in headless mode by default (no visible GUI).

## Usage

```
fa_fetch_flights(queries, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `queries` | A flight query object or list of query objects (from fa_define_query()) |
| `verbose` | Logical. If TRUE, shows detailed progress information (default) |

## Value

A flight_results object containing the merged flight data. Access the data via 'result$data'. **Important:** You must capture the return value to get the flight data: 'result <- fa_fetch_flights(query)'

## Examples

```
## Not run:
query <- fa_define_query("NYC", "IST", "2025-12-20", "2025-12-22")
flights <- fa_fetch_flights(query)
flights$data

## End(Not run)
```

`fa_find_best_dates` *Find Best Travel Dates*

## Description

Identifies and returns the top N dates with the cheapest average prices across all routes. This helps quickly identify the best travel dates when planning a flexible trip. Supports filtering by various criteria such as departure time, airlines, travel time, stops, and emissions.

## Usage

```
fa_find_best_dates(
  flight_results,
  n = 10,
  by = "min",
  time_min = NULL,
  time_max = NULL,
  airlines = NULL,
  price_min = NULL,
  price_max = NULL,
  travel_time_max = NULL,
  max_stops = NULL,
  max_layover = NULL,
  max_emissions = NULL,
  excluded_airports = NULL
)
```

## Arguments

| | |
|---|---|
| `flight_results` | A flight_results object from fa_fetch_flights(). This function no longer accepts data frames or query objects directly. |
| `n` | Integer. Number of best dates to return. Default is 10. |
| `by` | Character. How to calculate best dates: "mean" (average price across routes), "median", or "min" (lowest price on that date). Default is "min". |
| `time_min` | Character. Minimum departure time in "HH:MM" format (24-hour). Filters flights departing at or after this time. Default is NULL (no filter). |
| `time_max` | Character. Maximum departure time in "HH:MM" format (24-hour). Filters flights departing at or before this time. Default is NULL (no filter). |
| `airlines` | Character vector. Filter by specific airlines. Default is NULL (no filter). |
| `price_min` | Numeric. Minimum price. Default is NULL (no filter). |
| `price_max` | Numeric. Maximum price. Default is NULL (no filter). |
| `travel_time_max` | Numeric or character. Maximum travel time. If numeric, interpreted as hours. If character, use format "XX hr XX min". Default is NULL (no filter). |

| | |
|---|---|
| max_stops | Integer. Maximum number of stops. Default is NULL (no filter). |
| max_layover | Character. Maximum layover time in format "XX hr XX min". Default is NULL (no filter). |
| max_emissions | Numeric. Maximum CO2 emissions in kg. Default is NULL (no filter). |
| excluded_airports | Character vector. Airport codes to exclude from results. Default is NULL (no additional filtering beyond global excluded_airports list). |

## Value

A data frame with columns: departure_date, departure_time (or date if datetime not available), origin, price (average/median/min), n_routes, num_stops, layover, travel_time, co2_emission_kg, airlines, arrival_date, arrival_time. All column names are lowercase. Returns the top N dates with best (lowest) prices, sorted by departure time for display. Additional columns are aggregated using mean/median for numeric values and most common value for categorical. Note: arrival_date and arrival_time represent the most common values when multiple flights are aggregated and may not correspond exactly to the specific departure times shown.

## Examples

```
# Find best dates
fa_find_best_dates(sample_flight_results, n = 3, by = "min")

# With filters
fa_find_best_dates(
  sample_flight_results,
  n = 2,
  max_stops = 0
)
```

---

# fa_plot_best_dates *Plot Best Travel Dates*

---

## Description

Creates a modern visualization showing the best (cheapest) travel dates identified by fa_find_best_dates. Uses a lollipop chart style that clearly shows the price range and highlights the best options by origin and date.

Uses ggplot2 for a polished, publication-ready aesthetic with colorblind-friendly colors and clear typography.

## Usage

```
fa_plot_best_dates(
  flight_results,
  title = "Best Travel Dates by Price",
  subtitle = NULL,
```

```
  x_axis_angle = 0,
  ...
)
```

## Arguments

| | |
|---|---|
| flight_results | A flight_results object from [fa_fetch_flights()]. |
| title | Character. Plot title. Default is "Best Travel Dates by Price". |
| subtitle | Character. Plot subtitle. Default is NULL (auto-generated). |
| x_axis_angle | Numeric. Angle in degrees to rotate x-axis labels for better readability in wide figures with many dates. Common values are 45 (diagonal) or 90 (vertical). Default is 0 (horizontal labels). |
| ... | Additional arguments passed to fa_find_best_dates if best_dates is a flight_results object, including excluded_airports to filter out specific airport codes. |

## Value

A ggplot2 plot object that can be further customized or saved.

## Examples

```
## Not run:
# Plot best dates
fa_plot_best_dates(sample_flight_results, n = 5)

# With filters
fa_plot_best_dates(sample_flight_results, n = 5, max_stops = 0)

# Tilt x-axis labels diagonally for wide figures
fa_plot_best_dates(sample_flight_results, n = 10, x_axis_angle = 45)

## End(Not run)
```

---

# fa_plot_prices *Plot Price Summary*

---

## Description

Creates a modern line plot showing price trends across dates for different origins or destinations. The function automatically detects whether to group by origin or destination based on the data structure:

- When there are multiple origins and a single destination, groups by origin
- When there is a single origin and multiple destinations, groups by destination
- When there are multiple origins AND multiple destinations, you must specify the plot_by parameter to choose which dimension to use for grouping

The legend title automatically updates to "Origin" or "Destination" accordingly.

Requires flight_results objects from fa_fetch_flights. This function no longer accepts pre-summarized data or data frames.

Uses ggplot2 for a polished, publication-ready aesthetic with colorblind-friendly colors and clear typography.

## Usage

```
fa_plot_prices(
  flight_results,
  plot_by = NULL,
  title = NULL,
  subtitle = NULL,
  size_by = "travel_time",
  annotate_col = NULL,
  use_ggrepel = TRUE,
  show_max_annotation = TRUE,
  show_min_annotation = FALSE,
  x_axis_angle = 0,
  drop_empty_dates = TRUE,
  highlight_extremes = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| flight_results | A flight_results object from [fa_fetch_flights()]. |
| plot_by | Character. Specifies how to group the data: "origin" or "destination". When NULL (default), automatically detected based on data structure. Required when there are multiple origins AND multiple destinations. |
| title | Character. Plot title. Default is NULL (auto-generated with flight context). |
| subtitle | Character. Plot subtitle. Default is NULL (auto-generated with lowest price info). |
| size_by | Character. Name of column from raw flight data to use for point sizing. Can be "price", a column name like "travel_time", or NULL for uniform sizing (default). When using a column name, only works when passing raw flight data, not summary tables. Default is NULL. |
| annotate_col | Character. Name of column from raw flight data to use for point annotations (e.g., "travel_time", "num_stops"). Only works when passing raw flight data, not summary tables. Default is NULL (no annotations). |
| use_ggrepel | Logical. If TRUE, uses ggrepel for non-overlapping label positioning (requires ggrepel package). If FALSE, labels are centered on points and may overlap when there are many data points. Default is TRUE. |
| show_max_annotation | Logical. If TRUE, adds a data-journalism-style annotation for the maximum price with a horizontal bar and formatted price label. The annotation is subtle and clean (no arrows or boxes). Default is TRUE. |

`show_min_annotation`
: Logical. If TRUE, adds a data-journalism-style annotation for the minimum price with a horizontal bar and formatted price label. The annotation is subtle and clean (no arrows or boxes). Default is FALSE.

`x_axis_angle`
: Numeric. Angle in degrees to rotate x-axis labels for better readability in wide figures with many dates. Common values are 45 (diagonal) or 90 (vertical). Default is 0 (horizontal labels).

`drop_empty_dates`
: Logical. If TRUE, removes dates that have no flight data (all NA prices) from the plot. This is useful when querying multiple airports where some may not have data for certain dates. Default is TRUE.

`highlight_extremes`
: Logical. If TRUE, highlights the lowest and highest price points by filling them with distinct colorblind-friendly colors (bluish green for lowest, vermillion for highest). Default is TRUE.

`...`
: Additional arguments passed to `fa_summarize_prices`, including `excluded_airports` to filter out specific airport codes.

## Value

A ggplot2 plot object that can be further customized or saved.

## Examples

```
## Not run:
# Basic plot with auto-generated title and subtitle
fa_plot_prices(sample_flight_results)

# With point size based on travel time and annotations
fa_plot_prices(sample_flight_results,
               size_by = "travel_time",
               annotate_col = "num_stops")

# Size by number of stops
fa_plot_prices(sample_flight_results,
               size_by = "num_stops")

# With annotations centered on points (no ggrepel)
fa_plot_prices(sample_flight_results,
               size_by = "travel_time",
               annotate_col = "travel_time",
               use_ggrepel = FALSE)

# Custom title and both price annotations
fa_plot_prices(sample_flight_results,
               title = "Custom Title",
               show_max_annotation = TRUE,
               show_min_annotation = TRUE)

# Tilt x-axis labels diagonally for wide figures
fa_plot_prices(sample_flight_results, x_axis_angle = 45)
```

```
# Default behavior: filter out dates with no flight data
# Set drop_empty_dates = FALSE to keep all dates including empty ones
fa_plot_prices(sample_flight_results, drop_empty_dates = FALSE)

# Disable highlighting of lowest/highest price points
fa_plot_prices(sample_flight_results, highlight_extremes = FALSE)

## End(Not run)
```

---

`fa_summarize_prices` *Create Price Summary Table*

---

## Description

Creates a wide summary table showing prices by city/airport and date, with an average price column. When multiple flights exist for the same date, uses the minimum (cheapest) price. This is useful for visualizing price patterns across multiple dates and comparing different origin airports. Supports filtering by various criteria such as departure time, airlines, travel time, stops, and emissions.

## Usage

```
fa_summarize_prices(
  flight_results,
  include_comment = TRUE,
  currency_symbol = "$",
  round_prices = TRUE,
  time_min = NULL,
  time_max = NULL,
  airlines = NULL,
  price_min = NULL,
  price_max = NULL,
  travel_time_max = NULL,
  max_stops = NULL,
  max_layover = NULL,
  max_emissions = NULL,
  excluded_airports = NULL
)
```

## Arguments

`flight_results` A flight_results object from [fa_fetch_flights()].

`include_comment`
: Logical. If TRUE and Comment column exists, includes it in the output. Default is TRUE.

`currency_symbol`
: Character. Currency symbol to use for formatting. Default is "$".

| | |
|---|---|
| round_prices | Logical. If TRUE, rounds prices to nearest integer. Default is TRUE. |
| time_min | Character. Minimum departure time in "HH:MM" format (24-hour). Filters flights departing at or after this time. Default is NULL (no filter). |
| time_max | Character. Maximum departure time in "HH:MM" format (24-hour). Filters flights departing at or before this time. Default is NULL (no filter). |
| airlines | Character vector. Filter by specific airlines. Default is NULL (no filter). |
| price_min | Numeric. Minimum price. Default is NULL (no filter). |
| price_max | Numeric. Maximum price. Default is NULL (no filter). |
| travel_time_max | Numeric or character. Maximum travel time. If numeric, interpreted as hours. If character, use format "XX hr XX min". Default is NULL (no filter). |
| max_stops | Integer. Maximum number of stops. Default is NULL (no filter). |
| max_layover | Character. Maximum layover time in format "XX hr XX min". Default is NULL (no filter). |
| max_emissions | Numeric. Maximum CO2 emissions in kg. Default is NULL (no filter). |
| excluded_airports | Character vector. Airport codes to exclude from results. Default is NULL (no additional filtering beyond global excluded_airports list). |

## Value

A wide data frame with columns: City, Origin, Comment (optional), one column per date with prices, and an Average_Price column.

## Examples

```
# Create summary table
fa_summarize_prices(sample_flight_results)

# With filters
fa_summarize_prices(
  sample_flight_results,
  max_stops = 0
)
```

---

# print.flight_query *Print method for flight query objects*

---

## Description

Print method for flight query objects

## Usage

```
## S3 method for class 'flight_query'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A flight query object |
| ... | Additional arguments (ignored) |

---

## print.flight_record *Print method for flight_record objects*

---

### Description

Print method for flight_record objects

### Usage

```
## S3 method for class 'flight_record'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A flight_record object |
| ... | Additional arguments (ignored) |

---

## print.flight_results *Print method for flight_results objects*

---

### Description

Print method for flight_results objects

### Usage

```
## S3 method for class 'flight_results'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A flight_results object |
| ... | Additional arguments (ignored) |

`sample_flight_results` *Sample Flight Results Dataset*

## Description

A comprehensive sample dataset containing flight data from 5 Indian origins (BOM, DEL, VNS, PAT, GAY) to JFK spanning from December 18, 2025 to January 5, 2026. This dataset demonstrates realistic pricing patterns including a Christmas/New Year price spike, making it ideal for testing and demonstrating flight analysis functions.

## Usage

```
sample_flight_results
```

## Format

A flight_results object (S3 class) with the following structure:

**data** A data frame with 95 rows (5 origins × 19 days) containing:

- `departure_date`: Character, departure date in "YYYY-MM-DD" format
- `departure_time`: Character, departure time in "HH:MM" format
- `arrival_date`: Character, arrival date in "YYYY-MM-DD" format
- `arrival_time`: Character, arrival time in "HH:MM" format
- `origin`: Character, origin airport code (BOM, DEL, VNS, PAT, GAY)
- `destination`: Character, destination airport code (JFK)
- `airlines`: Character, airline name
- `travel_time`: Character, total travel time in "XX hr YY min" format
- `price`: Numeric, ticket price in USD
- `num_stops`: Integer, number of stops (0-2)
- `layover`: Character, layover information (if applicable)
- `access_date`: Character, timestamp when data was accessed
- `co2_emission_kg`: Numeric, estimated CO2 emissions in kg
- `emission_diff_pct`: Numeric, emission difference percentage

**BOM, DEL, VNS, PAT, GAY** Query objects for each origin containing the data subset and query parameters

## Details

The dataset features:

- Realistic travel times varying by origin (15.5-18.5 hours)
- Base prices varying by origin ($580-$700)
- Christmas/New Year price spike (Dec 23 - Jan 3) with 1.3x-4.5x multiplier
- Peak prices around January 1-2

- Weekend price adjustments (10% increase)
- Random variation to simulate real-world data

This dataset is particularly useful for:

- Demonstrating fa_plot_prices with seasonal patterns
- Testing fa_summarize_prices with multiple origins
- Showing fa_find_best_dates functionality
- Creating visually appealing examples with the size_by parameter

## See Also

sample_flights for a simpler data frame example, fa_plot_prices for plotting functions, fa_summarize_prices for price summary tables

## Examples

```
# Load and examine the dataset
head(sample_flight_results$data)

## Not run:
# Plot with automatic Christmas spike visualization
fa_plot_prices(sample_flight_results)

# Size points by travel time
fa_plot_prices(sample_flight_results, size_by = "travel_time")

# Create price summary table
fa_summarize_prices(sample_flight_results)

# Find best dates
fa_find_best_dates(sample_flight_results, n = 5)

## End(Not run)
```

---

# sample_flights *Sample Flight Data*

---

## Description

Sample flight data for testing analysis functions like ‘fa_find_best_dates()‘ and ‘fa_summarize_prices()‘ without internet access.

## Usage

```
sample_flights
```

### Format

A data frame with 6 rows and 14 variables:

**departure_date** Character, departure date (YYYY-MM-DD)

**departure_time** Character, departure time (HH:MM)

**arrival_date** Character, arrival date (YYYY-MM-DD)

**arrival_time** Character, arrival time (HH:MM)

**origin** Character, 3-letter origin airport code

**destination** Character, 3-letter destination airport code

**airlines** Character, airline name(s)

**travel_time** Character, total travel time

**price** Numeric, ticket price in USD

**num_stops** Integer, number of stops

**layover** Character, layover details (NA if nonstop)

**access_date** Character, when data was accessed

**co2_emission_kg** Numeric, CO2 emissions in kg

**emission_diff_pct** Numeric, emission difference percentage

### Examples

```
head(sample_flights)

# Note: sample_flights is a data frame for demonstration purposes only.
# Analysis functions now require flight_results objects from fa_fetch_flights().
# Use sample_flight_results instead:
## Not run:
fa_find_best_dates(sample_flight_results, n = 3)
fa_summarize_prices(sample_flight_results)

## End(Not run)
```

---

## sample_multi_origin *Sample Multiple Origin Queries*

---

### Description

Sample query objects for multiple origins created with ‘fa_define_query_range()‘. Demonstrates searching multiple airports over a date range.

### Usage

```
sample_multi_origin
```

### Format

A named list of 2 flight_query objects (BOM and DEL to JFK)

**BOM** query object for Mumbai (BOM) to JFK

**DEL** query object for Delhi (DEL) to JFK

### Examples

```
sample_multi_origin
```

---

## sample_query *Sample Flight Query*

---

### Description

A sample flight query object created with 'fa_define_query()'. Useful for testing and documentation examples without making API calls.

### Usage

```
sample_query
```

### Format

A flight_query object for JFK to IST round-trip

**origin** List of origin airport codes

**dest** List of destination airport codes

**dates** List of travel dates

**type** Trip type (round-trip)

**url** Google Flights URLs

### Examples

```
sample_query
```

# Index