

Package ‘grayleafspotr’

October 28, 2026

Title Quantitative Analysis of Gray Leaf Spot Colonies from Plate Images

Version 0.99.14

Description Quantitative phenotyping of gray leaf spot (*Magnaporthe oryzae*) fungal colonies grown on petri dishes. The package segments colonies from time-lapse plate photographs using a bundled SmallUNet deep-learning model, extracts morphometric and texture features (area, eccentricity, crack coverage, radial profile), and provides tidy result objects together with template ggplot2 visualisations. Python dependencies are managed automatically through basilisk; no manual environment setup is required.

License Apache License (>= 2.0) | file LICENSE

URL <https://rotsl.github.io/grayleafspotr>,
<https://github.com/rotsl/grayleafspotr>

BugReports <https://github.com/rotsl/grayleafspotr/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.6.0)

Imports basilisk, reticulate, dplyr, ggplot2, jsonlite, readr, tibble,
png, BiocFileCache, EBImage, stats

biocViews Software, Preprocessing, Visualization, Classification,
FeatureExtraction

Suggests knitr, rmarkdown, pkgdown, magick, jpeg, tiff, testthat (>= 3.0.0), BiocStyle

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

git_url <https://git.bioconductor.org/packages/grayleafspotr>

git_branch devel

git_last_commit f4820b6

git_last_commit_date 2026-10-10

Repository Bioconductor 3.24

Date/Publication 2026-10-27

Author Rohan R [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0005-9225-1775>)

Maintainer Rohan R <phonics-tiffs1i@icloud.com>

Contents

grayleafspotr-package	2
as_grayleafspot_growth_data	3
example_grayleafspot_dir	3
example_grayleafspot_results	4
grayleafspot_analyze	4
grayleafspot_download_model	6
grayleafspot_model_path	6
grayleafspot_python_available	7
grayleafspot_python_executable	7
grayleafspot_run	8
plot_colony_expansion	9
plot_feature_heatmap	10
plot_grayleafspot_overlay	10
plot_growth_roughness	12
plot_radial_growth_area	12
plot_radial_profile	13
plot_shape_vs_stress	13
plot_stress_remodeling	14
plot_texture_organization	14
read_grayleafspot_results	15
write_grayleafspot_results	15
Index	17

grayleafspotr-package *Gray Leaf Spot Package*

Description

Self-contained R tools for gray leaf spot analysis, raw exports, and plotting. The package is intended to be used from RStudio and developed with devtools.

Author(s)

Maintainer: Rohan R <phonics-tiffs1i@icloud.com> (**ORCID**) [copyright holder]

Authors:

- Rohan R <phonics-tiffs1i@icloud.com> (**ORCID**) [copyright holder]

See Also

Useful links:

- <https://rotsl.github.io/grayleafspotr>
- <https://github.com/rotsl/grayleafspotr>
- Report bugs at <https://github.com/rotsl/grayleafspotr/issues>

`as_grayleafspot_growth_data`*Coerce a grayleafspot run object to a tidy data frame*

Description

Extracts the results table from a `grayleafspot_run` object and normalises column aliases so that downstream plotting helpers see a consistent schema.

Usage

```
as_grayleafspot_growth_data(x)
```

Arguments

`x` A `grayleafspot_run` object, a plain `data.frame` / `tibble`, or a list with a `$results` element.

Value

A `tibble::tibble()` with one row per image.

Examples

```
run <- example_grayleafspot_results()
as_grayleafspot_growth_data(run)
```

`example_grayleafspot_dir`*Path to the bundled example run directory*

Description

Resolves the installed location of the small example run shipped under `inst/extdata/example/` (falling back to the in-source path during development, before the package has been installed).

Usage

```
example_grayleafspot_dir()
```

Value

A single character path.

```
example_grayleafspot_results
```

Load the built-in example grayleafspot run

Description

Returns the small example run shipped with the package under `inst/extdata/example/`. Useful for exploring the plotting helpers without needing to run the analysis pipeline.

Usage

```
example_grayleafspot_results()
```

Value

A `grayleafspot_run` object.

Examples

```
run <- example_grayleafspot_results()
```

```
grayleafspot_analyze
```

Analyze plate images with the SmallUNet pipeline

Description

Calls the bundled Python pipeline, performs dish detection and SmallUNet segmentation on each image, and returns a `grayleafspot_run` object with tidy results and template plots.

Usage

```
grayleafspot_analyze(  
  input_dir,  
  output_dir = "outputs",  
  filenames = NULL,  
  plate_diameter_mm = 90,  
  run_name = NULL,  
  save_outputs = TRUE,  
  verbose = TRUE,  
  python = NULL,  
  engine_model = "localunet"  
)
```

Arguments

<code>input_dir</code>	Character. Path to the folder containing plate images (JPEG, PNG, BMP, TIFF, or WEBP). Images must include a day token in their filename (e.g. *_d04_* for day 4).
<code>output_dir</code>	Character. Base output directory. A timestamped sub-folder is created for each run. Defaults to "outputs".
<code>filenames</code>	Optional character vector. Names of specific image files inside <code>input_dir</code> to analyze. If NULL, all supported images are processed.
<code>plate_diameter_mm</code>	Numeric. Known petri dish diameter in mm (default 90).
<code>run_name</code>	Optional character. Human-readable suffix appended to the timestamped run folder name.
<code>save_outputs</code>	Logical. If FALSE, outputs are written to a temporary directory and deleted after the results are returned.
<code>verbose</code>	Logical. Print the saved run path to the console.
<code>python</code>	Optional character. Advanced override: path to a specific Python executable. Overrides GRAYLEAFSPOTR_PYTHON and basilisk.
<code>engine_model</code>	Character. Must be "localunet".

Details

Python dependencies are managed automatically through basilisk. No manual Python environment setup is required for normal use. The first call will download and configure the required packages (this may take a few minutes on a fresh installation). Subsequent calls use the cached environment.

Developers who maintain a local Python environment can bypass basilisk by setting the GRAYLEAFSPOTR_PYTHON environment variable to the path of their Python interpreter; this is not required for normal users.

Value

A `grayleafspot_run` S3 object with elements `$run` (manifest metadata), `$results` (per-image data frame), and `$raw_results`.

See Also

[grayleafspot_run\(\)](#) for a simpler entry point returning raw JSON.

Examples

```
img_dir <- system.file("extdata", "testdata", "06FEB", package = "grayleafspotr")
run <- grayleafspot_analyze(img_dir, output_dir = tempdir())
```

```
grayleafspot_download_model
```

Download the SmallUNet segmentation model

Description

Fetches best_area_w_0.7.pt from HuggingFace and caches it using `BiocFileCache::BiocFileCache()`. On subsequent calls the cached file is returned immediately without re-downloading.

Usage

```
grayleafspot_download_model(force = FALSE, quiet = FALSE)
```

Arguments

<code>force</code>	Logical. Re-download even if a cached copy already exists.
<code>quiet</code>	Logical. Suppress progress messages.

Details

The function looks for the model in this order:

1. models/best_area_w_0.7.pt relative to the package root (development).
2. The BiocFileCache cache keyed under grayleafspotr.
3. Downloads from HuggingFace and adds it to the cache.

Value

Invisible character string: absolute path to the downloaded model file.

Examples

```
grayleafspot_download_model()
```

```
grayleafspot_model_path
```

Return the path to the SmallUNet model, downloading it if necessary

Description

Used internally by the analysis pipeline. If the model is not found locally, `grayleafspot_download_model()` is called automatically.

Usage

```
grayleafspot_model_path(quiet = FALSE)
```

Arguments

quiet Logical. Suppress download progress messages.

Value

Character string: absolute path to best_area_w_0.7.pt.

grayleafspot_python_available

Check whether the Python ML dependencies are available

Description

When the default basilisk-managed environment is in use, this function returns TRUE as long as basilisk is installed (the environment is set up lazily on first pipeline run). When an explicit Python interpreter is configured via GRAYLEAFSPOTR_PYTHON or the python argument, the function probes that interpreter directly.

Usage

```
grayleafspot_python_available(python = NULL, engine_model = "localunet")
```

Arguments

python Optional character. Path to a specific Python executable. If NULL and GRAYLEAFSPOTR_PYTHON is not set, the basilisk-managed environment is assumed available.

engine_model Character. Currently only "localunet" is supported.

Value

Logical TRUE if the pipeline can run, FALSE otherwise.

Examples

```
grayleafspot_python_available()
```

grayleafspot_python_executable

Return the Python executable used by the grayleafspot pipeline

Description

This function is intended for **advanced use only**. Normal users do not need to call it — the pipeline resolves its Python environment automatically via basilisk.

Usage

```
grayleafspot_python_executable(python = NULL, engine_model = "localunet")
```

Arguments

python	Optional character. Path to a Python executable.
engine_model	Character. Reserved for future use; currently only "localunet" is supported.

Details

Resolves the Python interpreter in this priority order: the python argument, the GRAYLEAFSPOTR_PYTHON environment variable, the grayleafspotr.python option, and finally python3 / python from PATH.

Value

Character string: absolute path to the resolved Python executable.

Examples

```
tryCatch(
  grayleafspot_python_executable(),
  error = function(e) message("Python not found: ", conditionMessage(e))
)
```

grayleafspot_run	<i>Run the gray leaf spot pipeline — simplified entry point</i>
------------------	---

Description

A streamlined wrapper around the SmallUNet pipeline. Python dependencies are resolved automatically via basilisk — no manual environment setup is required.

Usage

```
grayleafspot_run(
  input_dir,
  output_dir,
  run_name = "run",
  engine_model = "localunet",
  plate_diameter_mm = 90
)
```

Arguments

input_dir	Character. Path to the folder containing plate images.
output_dir	Character. Directory to write outputs into (created if absent).
run_name	Character. Human-readable label for the run (default "run").
engine_model	Character. Must be "localunet".
plate_diameter_mm	Numeric. Known petri dish diameter in mm (default 90).

Details**Workflow:**

```
library(grayleafspotr)

res <- grayleafspot_run(
  input_dir = "path/to/images",
  output_dir = "outputs",
  run_name = "trial_01"
)
res$results # per-image feature table
```

Advanced: developer Python override:

Set GRAYLEAFSPOTR_PYTHON in ~/.Rprofile to use a specific interpreter (e.g. a local rvenv_arm_311). This is **not needed** for normal use.

```
# ~/.Rprofile - developer only
Sys.setenv(GRAYLEAFSPOTR_PYTHON = "/path/to/rvenv_arm_311/bin/python")
```

Value

A named list parsed from the pipeline JSON output, containing `$results` (per-image feature records) and `$run` (manifest metadata).

See Also

[grayleafspot_analyze\(\)](#) for the full-featured interface returning a `grayleafspot_run` S3 object with plotting helpers.

Examples

```
img_dir <- system.file("extdata", "testdata", "06FEB", package = "grayleafspotr")
result <- grayleafspot_run(img_dir, tempdir())
```

`plot_colony_expansion` *Plot colony expansion over time*

Description

Draws equivalent colony radius (mm) against time (days) as a line + point plot using the `ggplot2` `theme_minimal` style.

Usage

```
plot_colony_expansion(x)
```

Arguments

`x` A `grayleafspot_run` object, data frame, or list accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_colony_expansion(run)
```

plot_feature_heatmap *Pearson correlation heatmap of numeric morphology features*

Description

Pearson correlation heatmap of numeric morphology features

Usage

```
plot_feature_heatmap(x)
```

Arguments

x A grayleafspot_run object or data accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_feature_heatmap(run)
```

plot_grayleafspot_overlay
 Overlay detected regions of interest on a plate image

Description

Draws the detected dish boundary, colony outline, and any detected crack segments on top of the original plate photograph, using [EBImage::readImage\(\)](#) and EBImage's drawing primitives. This lets you visually verify that automated segmentation and crack detection were computed as expected, rather than trusting the summary statistics alone.

Usage

```
plot_grayleafspot_overlay(
  run,
  id = NULL,
  image_path = NULL,
  show_dish = TRUE,
  show_colony = TRUE,
  show_cracks = TRUE,
  dish_col = "yellow",
  colony_col = "red",
  crack_col = "cyan"
)
```

Arguments

run	A grayleafspot_run object (see grayleafspot_analyze() , read_grayleafspot_results()), a plain list of per-image raw result records, or a single record (a list with a rawAnalysis element, as produced internally by the analysis engines).
id	Optional character. id or filename of the image to overlay, matched against the records in run. Defaults to the first record.
image_path	Optional character. Explicit path to the source image. Falls back to the record's stored imageUrl, which is only a resolvable local file path for runs produced by the pure-R fallback engine; for runs produced by the Python/basilisk pipeline, pass this explicitly (e.g. a file inside the original input_dir).
show_dish, show_colony, show_cracks	Logical. Which overlays to draw.
dish_col, colony_col, crack_col	Character. Colors for each overlay, passed to grDevices::col2rgb() .

Value

An [EBImage::Image](#) with the requested overlays drawn on it. Display with [EBImage::display\(\)](#) or save with [EBImage::writeImage\(\)](#).

Examples

```
image_file <- tempfile(fileext = ".png")
png::writePNG(matrix(seq(0, 1, length.out = 32^2), 32), image_file)
decoder_available <- tryCatch({
  EBImage::readImage(image_file)
  TRUE
}, error = function(e) FALSE)
if (decoder_available) {
  record <- list(
    imageUrl = image_file,
    rawAnalysis = list(
      dish_center = list(x = 500, y = 500),
      dish_radius = 400
    )
  )
  overlay <- plot_grayleafspot_overlay(record)
}
unlink(image_file)
```

plot_growth_roughness *Plot relative growth rate and edge roughness*

Description

Plot relative growth rate and edge roughness

Usage

```
plot_growth_roughness(x)
```

Arguments

x A grayleafspot_run object or data accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_growth_roughness(run)
```

plot_radial_growth_area

Plot radial growth rate and colony area by plate over time

Description

Produces a faceted panel with colony area (mm²) on one facet and radial growth rate (mm/day) on the other. When each plate appears only once in the data, points from all plates are connected in day order.

Usage

```
plot_radial_growth_area(x)
```

Arguments

x A grayleafspot_run object or data accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_radial_growth_area(run)
```

plot_radial_profile	<i>Plot the radial intensity profile from the first image in a run</i>
---------------------	--

Description

Reads the radialProfile field from raw_results. When called with a plain data frame, the function auto-discovers the most recent analysis.json under outputs/ in the working directory.

Usage

```
plot_radial_profile(x)
```

Arguments

x	A grayleafspot_run object, a path to a run directory or analysis.json, a list of raw results, or a data frame.
---	--

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_radial_profile(run)
```

plot_shape_vs_stress	<i>Scatter plot of colony shape vs crack stress</i>
----------------------	---

Description

Plots eccentricity against crack coverage percentage, with point size proportional to colony diameter and colour encoding time (day).

Usage

```
plot_shape_vs_stress(x)
```

Arguments

x	A grayleafspot_run object or data accepted by as_grayleafspot_growth_data() .
---	---

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_shape_vs_stress(run)
```

`plot_stress_remodeling`*Plot crack coverage and crack count over time*

Description

Plot crack coverage and crack count over time

Usage

```
plot_stress_remodeling(x)
```

Arguments

`x` A grayleafspot_run object or data accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_stress_remodeling(run)
```

`plot_texture_organization`*Plot texture entropy and center-to-edge intensity delta over time*

Description

Plot texture entropy and center-to-edge intensity delta over time

Usage

```
plot_texture_organization(x)
```

Arguments

`x` A grayleafspot_run object or data accepted by [as_grayleafspot_growth_data\(\)](#).

Value

A ggplot2 object.

Examples

```
run <- example_grayleafspot_results()
plot_texture_organization(run)
```

`read_grayleafspot_results`*Read a saved grayleafspot run from disk*

Description

Loads a run directory (containing `analysis.json`, `analysis.csv`, and `manifest.json`) or a single JSON / CSV file and returns a `grayleafspot_run` object.

Usage

```
read_grayleafspot_results(path)
```

Arguments

<code>path</code>	Character. Path to a run directory or to an individual <code>analysis.json</code> / <code>analysis.csv</code> file.
-------------------	---

Value

A `grayleafspot_run` object with elements `$run`, `$results`, and `$raw_results`.

Examples

```
dir <- system.file("extdata", "example", package = "grayleafspotr")
run <- read_grayleafspot_results(dir)
```

`write_grayleafspot_results`*Write grayleafspot results to disk*

Description

Serialises a list of result records to `analysis.json`, `analysis.csv`, and `manifest.json` under `output_dir` and returns a `grayleafspot_run` object.

Usage

```
write_grayleafspot_results(
  results,
  output_dir,
  engine = "python-local",
  engine_model = "packaged-python-pipeline",
  run_name = NULL
)
```

Arguments

<code>results</code>	A list of per-image result records (as produced by the Python pipeline or <code>analyze_grayleafspot_images</code>).
<code>output_dir</code>	Character. Directory to write outputs into (created if absent).
<code>engine</code>	Character. Engine label stored in the manifest.
<code>engine_model</code>	Character. Engine-model label stored in the manifest.
<code>run_name</code>	Optional character. Human-readable run name for the manifest.

Value

A `grayleafspot_run` object.

Examples

```
run <- example_grayleafspot_results()
write_grayleafspot_results(run$raw_results, tempdir(), run_name = "demo")
```

Index

*** internal**

- example_grayleafspot_dir, [3](#)
- grayleafspot_model_path, [6](#)
- grayleafspotr-package, [2](#)

- as_grayleafspot_growth_data, [3](#)
- as_grayleafspot_growth_data(), [9](#), [10](#),
[12–14](#)

- BiocFileCache::BiocFileCache(), [6](#)

- EImage::Image, [11](#)
- EImage::readImage(), [10](#)
- example_grayleafspot_dir, [3](#)
- example_grayleafspot_results, [4](#)

- grayleafspot_analyze, [4](#)
- grayleafspot_analyze(), [9](#), [11](#)
- grayleafspot_download_model, [6](#)
- grayleafspot_model_path, [6](#)
- grayleafspot_python_available, [7](#)
- grayleafspot_python_executable, [7](#)
- grayleafspot_run, [8](#)
- grayleafspot_run(), [5](#)
- grayleafspotr (grayleafspotr-package), [2](#)
- grayleafspotr-package, [2](#)
- grDevices::col2rgb(), [11](#)

- plot_colony_expansion, [9](#)
- plot_feature_heatmap, [10](#)
- plot_grayleafspot_overlay, [10](#)
- plot_growth_roughness, [12](#)
- plot_radial_growth_area, [12](#)
- plot_radial_profile, [13](#)
- plot_shape_vs_stress, [13](#)
- plot_stress_remodeling, [14](#)
- plot_texture_organization, [14](#)

- read_grayleafspot_results, [15](#)
- read_grayleafspot_results(), [11](#)

- tibble::tibble(), [3](#)

- write_grayleafspot_results, [15](#)