



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to sdvplotR. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: great_tables. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

GALLERY ↗

COOKBOOKS ↗

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league`: `"nfl"` `"nba"` `"wnba"` `"mlb"` `"nhl"` `"cfb"` `"mbb"` `"wbb"` `"soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started](#) ↗

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a ggplot, a Plotly figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos](#) ↗

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: plotnine, Altair and tables give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")
```

`id_system`=`"espn"` (default, every league), `"gsis"` (NFL, via nflverse) or `"league"` (NFL gsis, NBA / WNBA Stats, MLBAM, NHL ids). An unknown id gets the CDN's silhouette.

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a {team: colour} dict any library takes (seaborn `palette=`; `team_colors()` returns one colour per value (a str, list or Series, like its input; None for a miss). Both take `which`=`"primary"` `"secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes](#) ↗

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
    ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("nfl", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity](#) ↗

WHICH ADAPTER RUNS

The target's library picks it: a matplotlib Axes or seaborn, a plotnine ggplot, a great_tables GT (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a Plotly, Altair, Bokeh, HoloViews, folium or pygal object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: matplotlib, seaborn, plotnine, Plotly, Altair. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web](#) ↗

ID SYSTEMS

`id_system`=`"auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `naaa` `pff` `cricinfo` `cfbd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries.

`strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant`=`"default"` `"dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras](#) ↗

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl uwhl pwhl phf	hockey minors & juniors
milb naaa_baseball naaa_softball	286 / 414 / 415
naaa_mhockey naaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached.

`SDV_PLOT_CACHE_DIR` moves the cache; after `SDV_PLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache](#) ↗

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors](#) ↗

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, which and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

plotnine: `scale_color_sdv("nfl")`; pygal: `team_style()`; tables: `gt_theme_sdv_team`, `reactable_sdv_team_color_bg`.



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
GeoAxes) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or https URLs
(http is refused). `axis_logos(ax, "y", mark_type=)` :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left" "right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc`,
`presort`, `no_line_below_tier`, `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from <code>marks()</code>
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
<code>add_headshots</code> has no <code>season</code> or <code>variant</code> .	

SDVPLOT.PLOTNINE

Layer	Aesthetics
<code>geom_sdv_logos</code>	<code>x y team</code> , optional
<code>geom_sdv_wordmarks</code>	<code>season</code>
<code>geom_sdv_headshots</code>	<code>x y player_id</code>
<code>geom_from_path</code>	<code>x y path</code> (any image)
<code>geom_mean_lines</code> <code>geom_median_lines</code>	<code>x0</code> and/or <code>y0</code>

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system`.

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv` : `which`, `season`, `na_value="grey"`,
`alpha`. The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref / yref` pick a subplot's axes,
`layer="above" "below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos(chart, ...)` returns a new `LayerChart` ;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type / y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height × frame_height`, else the
figure `height`. Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup`, "sdvplot
logos", for a `LayerControl`.

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style(teams, league=, which=)` : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy`.

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column(name, league=,`
`id_system="espn")`. Extra keywords go to
`ColumnDefinition`.

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), great_tables (<code>GT</code>)

SDVPLOT.TESTING

`check_adapter_contract(adapter, make_target)` and
`check_table_adapter_contract(adapter,`
`make_table)` raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()`, with the four verbs in
`__all__`, a `_drawn_marks(target)` test hook, and one
`register_adapter(...)` line in `_dispatch.py`.

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column <i>names</i> → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv
)(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#) ↗

THEMES

<code>gt_theme_sdv(gt, style="dark")</code> <code>gt_theme_sdv_team(gt, "KC", league="nfl")</code>	<code>gt_theme_sdv_dark</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95	<code>gt_theme_kenpom</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39 444.4 durian four 65100.0
sdv	sdv · dark	kenpom
<code>gt_theme_savant</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39 444.4 durian four 65100.0	<code>gt_theme_broadsheet</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39	<code>gt_theme_midnight</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95
savant	broadsheet	midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
<code>density="comfortable"</code> <code>"compact"</code> <code>"social"</code>	all
<code>accent</code>	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
<code>style="light"</code> <code>"dark"</code>	sdv, sofa, tier
<code>color · paper · stripe</code>	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)
```

`gt_theme_preview(data, themes=None, n=5)`: the first `n` rows in each theme. `pal_midnight`: a rank palette that reads on `midnight` / `terminal`.

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/Trows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

<code>(GT(df) .pipe(gt_color_ranks, "wins") .pipe(gt_legend_continuous, title="Wins", type="steps"))</code>	
Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: <code>gt_tiers</code> 's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. <code>lambda s: s > 0.7</code> ; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

```
(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))
```

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataverse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from_`. `gt_theme_preview` returns a dict; paletteeer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great_tables side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with sportsdataverse-py data, the sdvplotR name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
gt_grid(tables)	tables in a grid under one title; ncol=2, labels, gap=24
gt_stack_tables(tables)	one above another; gap=16, align
gt_snake(gt)	long table in side-by-side blocks; n_cols=2, rows_per_col
gt_snake_align(x)	reshape a parallel frame to match
gt_grid([afc, nfc], ncol=2, title="2025 standings") gt_grid(gt_theme_preview(df), ncol=3) GT(nba).pipe(gt_snake, n_cols=3)	
tables: a list of GT, or a dict (its values, in order). Both also take subtitle, caption, source_note and file=: None returns the HTML, a path writes an image. Style with title_style, subtitle_style, caption_style dicts.	

SAVE AS AN IMAGE

Function	Does · key arguments
gt_save_crop(gt, file)	PNG, trimmed, padded; whitespace=50, bg, zoom=2, width
gt_social_crop(gt, file)	pad to aspect_ratio="1:1", "4:5", ...; gravity
gt_save_batch(data, group, fn, file, dir=)	one image per group, match_width=True
gt_save_crop(tbl, "table.png", whitespace=40) gt_social_crop(tbl, "post.png", aspect_ratio="4:5") gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")	

build(df, value) returns a GT; {group} becomes the slugged value; it returns the files written. Saving runs headless Chrome through nokap, trimmed with Pillow. file=None returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
reactable_sdv_logos(league=)	a Column; height=30, include_name, season, default_img
reactable_sdv_wordmarks(league=)	a Column; as logos
reactable_sdv_headshots(league=)	a Column; height=40, id_system="espn"
reactable_sdv_cols_label(data, league=)	a list of Columns, marks as headers; mark_type
reactable_sdv_team_color_bar(data, team_col, league=)	a bar in team colour; which, max_value
reactable_sdv_team_color_bg(data, team_col, league=)	a team-colour cell fill; alpha=0.15
from reactable import Reactable from sdvplot.reactable import (reactable_sdv_logos, reactable_sdv_team_color_bar) Reactable(df, columns=[reactable_sdv_logos(league="nba", id="team"), reactable_sdv_team_color_bar(df, "team", league="nba", id="wins")])	

Other keywords (id, name, ...) go to reactable.Column. Needs sdvplot[reactable].

PLAYING SURFACES

import matplotlib.pyplot as plt fig, ax = plt.subplots() sdvplot.surface("nba", "BOS", ax=ax, center_logo=True) ax.scatter(shots["court_x"], shots["court_y"])	surface(league, team=None, *, season, ax, center_logo) draws sportypy's field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the Axes. center_logo=True is 0.25 of the Axes height, or pass a fraction. Needs [surfaces].
---	---

SURFACE LEAGUES & OPTIONS

sdvplot.surface("nhl", "BOS", center_logo=0.3) sdvplot.surface("nfl", "KC", season=2010)	26 leagues plus "fiba" (no team colours): nfl cfb ufl usfl xfl aaf nba wnba nbagl mbb wbb mlb milb ncaa_baseball nhl ahl echl ohl qmjhl whl ushl pwhl phf ncaa_hockey ncaa_hockey soccer. sportypy keywords pass through: display_range, xlim, ylim, rotation to draw(); field_updates, color_updates, units, ... to the surface. surface("nba", display_range="defense") is a half court.
Cookbook: Surfaces and shot charts ↗	

COURT COORDINATES

# stats.nba.com legacy frame -> feet shots = sdvplot.court_coords(shots) # Euroleague (cm) -> FIBA court (m) pts = sdvplot.court_coords(pts, x="coord_x", y="coord_y", provider="euroleague")	Adds court_x, court_y (Float64) on sportypy's centre-court frame; returns the frame's own type (pandas or polars). Reads x_legacy / y_legacy (LOC_X / LOC_Y) by default. ESPN coordinate_x / _y are already in feet: don't convert them. Euroleague free throws (-1, -1) become null.
--	---

PITCH COORDINATES

# soccer events -> 105 x 68 m pitch ev = sdvplot.pitch_coords(ev, provider="statsbomb") fig, ax = plt.subplots() sdvplot.surface("soccer", ax=ax) ax.scatter(ev["pitch_x"], ev["pitch_y"])	Adds pitch_x, pitch_y in metres from the centre spot, attacking +x. Providers: opta (statsperform) wyscout statsbomb uefa impect espn; tracking tracab skillcorner secondspectrum metrica also need pitch_length, pitch_width. flip=True, or a boolean column, turns rows half a turn. ESPN reads field_position_x / _y, as sportsdataverse-py's espn_soccer_game_plays returns them; its (0, 0) becomes null.
--	--

WITH SPORTSDATAVERSE DATA

import polars as pl import sportsdataverse.nba as nba box = nba.load_nba_team_boxscore([2025]) off = box.group_by("team_abbreviation").agg(pl.col("assists", "team_score").mean()) fig, ax = plt.subplots() sdvplot.add_logos(ax, off["assists"], off["team_score"], off["team_abbreviation"], league="nba", height=0.06)	Per league: NFL · CFB · NBA · WNBA · MLB · NHL · PWHL · Soccer · Cricket
--	--

FROM SDVPLOTR

sdvplotR	sdvplot
sdv_teams() · sdv_resolve() · sdv_marks()	teams() · resolve() · marks()
sdv_logo_url() · sdv_headshot_url()	logo_url() · headshot_url()
sdv_color_palette() · sdv_team_colors()	palette() · team_colors()
geom_sdv_logos() & co.	same names in sdvplot.plotnine; add_logos() anywhere
element_sdv_logo(), scale_x_sdv_headshots()	axis_logos()
ggtitle_image() · sdv_team_tiers()	title_image() · team_tiers()
gt_*() · reactable_sdv_*()	same names
Arguments match (league, id_system, season, strict, variant = "default"); only from → from_. Not ported: sdv_team_factor() (three lines over resolve()).	

LEARN MORE

GALLERY ↗	every chart and table in one place
Start: Getting started · reference	
Concepts: identity · colours · logos · seasons · cache · adapters	
Recipes: NFL EPA · CFB table · head-to-head card · NHL shot map	
Leaderboards: NFL weekly · NBA · Siblings: sdvplotR (R), @sportsdataverse/sdvplot (npm)	



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to `sdvplotR`. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: `great_tables`. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

[GALLERY ↗](#)

[COOKBOOKS ↗](#)

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfb" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started ↗](#)

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a `ggplot`, a `Plotly` figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos ↗](#)

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")

id_system="espn" (default, every league), "gsis" (NFL,
via nflverse) or "league" (NFL gsis, NBA / WNBA Stats,
MLBAM, NHL ids). An unknown id gets the CDN's silhouette.
```

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`); `team_colors()` returns one colour per value (a str, list or `Series`, like its input; `None` for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes ↗](#)

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("NFL", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity ↗](#)

WHICH ADAPTER RUNS

The target's library picks it: a `matplotlib` Axes or `seaborn`, a `plotnine` `ggplot`, a `great_tables` `GT` (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a `Plotly`, `Altair`, `Bokeh`, `HoloViews`, `folium` or `pygal` object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: `matplotlib`, `seaborn`, `plotnine`, `Plotly`, `Altair`. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web ↗](#)

ID SYSTEMS

```
id_system="auto" (the default) tries team_id espn
espn_abbr nhl nflverse mlbstats nba_api
hockeytech ncaa pff cricinfo cfbf bref sportsipy
fangraphs sdvplotr name in order; pass one to pin it, or
"nhl_id" for NHL stats ids, which "auto" never tries.
strict=True (resolve, colours, URLs, tables) raises
UnresolvedTeamError on a miss. A plot skips an unknown
team with one SdvplotWarning.
```

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default" "dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras ↗](#)

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
milb ncaa_baseball ncaa_softball	286 / 414 / 415
ncaa_mhockey ncaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache ↗](#)

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors ↗](#)

TYPE HINTS

```
from sdvplot.typing import (
  IdSystem, HeadshotIdSystem, Which,
  MarkType, AxisMarkType)
```

The Literal types behind `id_system`, which and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

```
plotnine: scale_color_sdv("nfl"); pygal:
team_style(); tables: gt_theme_sdv_team,
reactable_sdv_team_color_bg.
```



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

[GALLERY ↗](#)

[COOKBOOKS ↗](#)

ADAPTERS · ONE PER LIBRARY

SDV.PLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl`;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or `https` URLs
(http is refused). `axis_logos` (`ax`, `"y"`, `mark_type=`):
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left"` `"right"`; height in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc`,
`presort`, `no_line_below_tier`, `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from marks()
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDV.PLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system`.

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv`: which, `season`, `na_value="grey"`,
`alpha`. The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDV.PLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref` / `yref` pick a subplot's axes,
`layer="above"` `"below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDV.PLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos` (`chart`, ...) returns a new `LayerChart`;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type` / `y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDV.PLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                  league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height × frame_height`, else the
figure `height`. Factor and datetime axes work.

SDV.PLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                        league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDV.PLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup`, "sdvplot
logos", for a `LayerControl`.

SDV.PLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style` (`teams`, `league=`, `which=`): a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy`.

SDV.PLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column` (`name`, `league=`,
`id_system="espn"`). Extra keywords go to
`ColumnDefinition`.

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), great_tables (GT)

SDV.PLOT.TESTING

`check_adapter_contract` (`adapter`, `make_target`) and
`check_table_adapter_contract` (`adapter`,
`make_table`) raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()`, with the four verbs in
`__all__`, a `_drawn_marks` (`target`) test hook, and one
`register_adapter` (...) line in `_dispatch.py`.

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

[GALLERY](#)

[COOKBOOKS](#)

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column <i>names</i> → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#)

THEMES

```
gt_theme_sdv(gt, style="dark")
gt_theme_sdv_team(gt, "KC", league="nfl")
```

gt_theme_sdv	gt_theme_sdv_dark	gt_theme_kenpom
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	33.33 coconut three 1.39
2.222 banana two 17.95	2.222 banana two 17.95	444.4 durian four 65100.0

sdv

sdv · dark

kenpom

gt_theme_savant	gt_theme_broadsheet	gt_theme_midnight
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	2.222 banana two 17.95
33.33 coconut three 1.39	33.33 coconut three 1.39	33.33 coconut three 1.39
444.4 durian four 65100.0		

savant

broadsheet

midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
density="comfortable" "compact" "social"	all
accent	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
style="light" "dark"	sdv, sofa, tier
color · paper · stripe	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#)

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)
```

`gt_theme_preview(data, themes=None, n=5)`: the first `n` rows in each theme. `pal_midnight`: a rank palette that reads on `midnight` / `terminal`.

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/Trows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

```
(GT(df)
 .pipe(gt_color_ranks, "wins")
 .pipe(gt_legend_continuous,
       title="Wins", type="steps"))
```

Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: <code>gt_tiers</code> 's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. <code>lambda s: s > 0.7</code> ; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

```
(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))
```

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataVerse"}))
```

Recipe: [CFB conference table](#)

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from_`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great `tables` side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with `sportsdataverse-py` data, the `sdvplotR` name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
<code>gt_grid(tables)</code>	tables in a grid under one title; <code>ncol=2</code> , <code>labels</code> , <code>gap=24</code>
<code>gt_stack_tables(tables)</code>	one above another; <code>gap=16</code> , <code>align</code>
<code>gt_snake(gt)</code>	long table in side-by-side blocks; <code>n_cols=2</code> , <code>rows_per_col</code>
<code>gt_snake_align(x)</code>	reshape a parallel frame to match
<code>gt_grid([afc, nfc], ncol=2, title="2025 standings")</code> <code>gt_grid(gt_theme_preview(df), ncol=3)</code> <code>GT(nba).pipe(gt_snake, n_cols=3)</code>	

`tables`: a list of `GT`, or a dict (its values, in order). Both also take `subtitle`, `caption`, `source_note` and `file=`: `None` returns the HTML, a path writes an image. Style with `title_style`, `subtitle_style`, `caption_style` dicts.

SAVE AS AN IMAGE

Function	Does · key arguments
<code>gt_save_crop(gt, file)</code>	PNG, trimmed, padded; <code>whitespace=50</code> , <code>bg</code> , <code>zoom=2</code> , <code>width</code>
<code>gt_social_crop(gt, file)</code>	pad to <code>aspect_ratio="1:1"</code> , "4:5", ...; <code>gravity</code>
<code>gt_save_batch(data, group, fn, file, dir=)</code>	one image per group, <code>match_width=True</code>

<code>gt_save_crop(tbl, "table.png", whitespace=40)</code>
<code>gt_social_crop(tbl, "post.png", aspect_ratio="4:5")</code>
<code>gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")</code>

`build(df, value)` returns a `GT`; `{group}` becomes the slugged value; it returns the files written. Saving runs headless Chrome through `nokap`, trimmed with `Pillow`. `file=None` returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
<code>reactable_sdv_logos(league=)</code>	a Column; <code>height=30</code> , <code>include_name</code> , <code>season</code> , <code>default_img</code>
<code>reactable_sdv_wordmarks(league=)</code>	a Column; as logos
<code>reactable_sdv_headshots(league=)</code>	a Column; <code>height=40</code> , <code>id_system="espn"</code>
<code>reactable_sdv_cols_label(data, league=)</code>	a list of Columns, marks as headers; <code>mark_type</code>
<code>reactable_sdv_team_color_bar(data, team_col, league=)</code>	a bar in team colour; <code>which</code> , <code>max_value</code>
<code>reactable_sdv_team_color_bg(data, team_col, league=)</code>	a team-colour cell fill; <code>alpha=0.15</code>

```
from reactable import Reactable
from sdvplot.reactable import (
    reactable_sdv_logos,
    reactable_sdv_team_color_bar)
Reactable(df, columns=[
    reactable_sdv_logos(league="nba",
                        id="team"),
    reactable_sdv_team_color_bar(df, "team",
                                league="nba", id="wins")])
```

Other keywords (`id`, `name`, ...) go to `reactable.Column`. Needs `sdvplot[reactable]`.

PLAYING SURFACES

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sdvplot.surface("nba", "BOS", ax=ax,
                center_logo=True)
ax.scatter(shots["court_x"],
           shots["court_y"])
```

`surface(league, team=None, *, season, ax, center_logo)` draws `sportypy`'s field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the Axes. `center_logo=True` is 0.25 of the Axes height, or pass a fraction. Needs `[surfaces]`.

SURFACE LEAGUES & OPTIONS

```
sdvplot.surface("nhl", "BOS",
                center_logo=0.3)
sdvplot.surface("nfl", "KC", season=2010)
```

26 leagues plus "fiba" (no team colours): `nfl` `cfb` `ufl` `usfl` `xfl` `aaf` `nba` `wnba` `nbagl` `mbb` `wbb` `mlb` `milb` `ncaa_baseball` `nhl` `ahl` `echl` `ohl` `qmjhl` `whl` `ushl` `pwhl` `phf` `ncaa_hockey` `ncaa_hockey` `soccer`.

`sportypy` keywords pass through: `display_range`, `xlim`, `ylim`, `rotation` to `draw()`; `field_updates`, `color_updates`, `units`, ... to the surface. `surface("nba", display_range="defense")` is a half court.

Cookbook: [Surfaces and shot charts ↗](#)

COURT COORDINATES

```
# stats.nba.com legacy frame -> feet
shots = sdvplot.court_coords(shots)
# Euroleague (cm) -> FIBA court (m)
pts = sdvplot.court_coords(pts,
                             x="coord_x", y="coord_y",
                             provider="euroleague")
```

Adds `court_x`, `court_y` (Float64) on `sportypy`'s centre-court frame; returns the frame's own type (`pandas` or `polars`). Reads `x_legacy` / `y_legacy` (`LOC_X` / `LOC_Y`) by default. ESPN `coordinate_x` / `_y` are already in feet: don't convert them. Euroleague free throws (`-1`, `-1`) become null.

PITCH COORDINATES

```
# soccer events -> 105 x 68 m pitch
ev = sdvplot.pitch_coords(ev,
                           provider="statsbomb")
fig, ax = plt.subplots()
sdvplot.surface("soccer", ax=ax)
ax.scatter(ev["pitch_x"], ev["pitch_y"])
```

Adds `pitch_x`, `pitch_y` in metres from the centre spot, attacking +x. Providers: `opta` (`statsperform`) `wyscout` `statsbomb` `uefa` `impect` `espn`; tracking `tracab` `skillcorner` `secondspectrum` `metrica` also need `pitch_length`, `pitch_width`. `flip=True`, or a boolean column, turns rows half a turn. ESPN reads `field_position_x` / `_y`, as `sportsdataverse-py`'s `espn_soccer_game_plays` returns them; its (`0`, `0`) becomes null.

WITH SPORTSDATAVERSE DATA

```
import polars as pl
import sportsdataverse.nba as nba
box = nba.load_nba_team_boxscore([2025])
off = box.group_by("team_abbreviation").agg(
    pl.col("assists", "team_score").mean())
fig, ax = plt.subplots()
sdvplot.add_logos(ax, off["assists"],
                  off["team_score"],
                  off["team_abbreviation"],
                  league="nba", height=0.06)
```

Per league: [NFL](#) · [CFB](#) · [NBA](#) · [WNBA](#) · [MLB](#) · [NHL](#) · [PWHL](#) · [Soccer](#) · [Cricket](#)

FROM SDVPLOTR

sdvplotR	sdvplot
<code>sdv_teams()</code> · <code>sdv_resolve()</code> · <code>sdv_marks()</code>	<code>teams()</code> · <code>resolve()</code> · <code>marks()</code>
<code>sdv_logo_url()</code> · <code>sdv_headshot_url()</code>	<code>logo_url()</code> · <code>headshot_url()</code>
<code>sdv_color_palette()</code> · <code>sdv_team_colors()</code>	<code>palette()</code> · <code>team_colors()</code>
<code>geom_sdv_logos()</code> & <code>co.</code>	same names in <code>sdvplot.plotnine</code> ; <code>add_logos()</code> anywhere
<code>element_sdv_logo()</code> , <code>scale_x_sdv_headshots()</code>	<code>axis_logos()</code>
<code>ggtitle_image()</code> · <code>sdv_team_tiers()</code>	<code>title_image()</code> · <code>team_tiers()</code>
<code>gt_*()</code> · <code>reactable_sdv_*()</code>	same names

Arguments match (`league`, `id_system`, `season`, `strict`, `variant = "default"`); only `from` → `from_`. Not ported: `sdv_team_factor()` (three lines over `resolve()`).

LEARN MORE

[GALLERY ↗](#) every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** `sdvplotR` (R), [@sportsdataverse/sdvplot](#) (npm)



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to sdvplotR. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: great_tables. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

GALLERY ↗

COOKBOOKS ↗

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league`: `"nfl"` `"nba"` `"wnba"` `"mlb"` `"nhl"` `"cfl"` `"mbb"` `"wbb"` `"soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started](#) ↗

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a ggplot, a Plotly figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
        "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos](#) ↗

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: plotnine, Altair and tables give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")
```

`id_system`="espn" (default, every league), `"gsis"` (NFL, via nflverse) or `"league"` (NFL gsis, NBA / WNBA Stats, MLBAM, NHL ids). An unknown id gets the CDN's silhouette.

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a {team: colour} dict any library takes (seaborn `palette=`; `team_colors()` returns one colour per value (a str, list or Series, like its input; None for a miss). Both take `which`="primary" "secondary", `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes](#) ↗

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
    ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("nfl", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity](#) ↗

WHICH ADAPTER RUNS

The target's library picks it: a matplotlib Axes or seaborn, a plotnine ggplot, a great_tables GT (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a Plotly, Altair, Bokeh, HoloViews, folium or pygal object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: matplotlib, seaborn, plotnine, Plotly, Altair. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web](#) ↗

ID SYSTEMS

`id_system`="auto" (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `naaa` `pff` `cricinfo` `cfd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which "auto" never tries. `strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant`="default" "dark" always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras](#) ↗

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfl mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl ushl pwhl phf	hockey minors & juniors
mlb naaa_baseball naaa_softball	286 / 414 / 415
naaa_mhockey naaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache](#) ↗

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors](#) ↗

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, `which` and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

plotnine: `scale_color_sdv("nfl")`; pygal: `team_style()`; tables: `gt_theme_sdv_team`, `reactable_sdv_team_color_bg`.



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or https URLs
(http is refused). `axis_logos(ax, "y", mark_type=)` :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left" "right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc`,
`presort`, `no_line_below_tier`, `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from <code>marks()</code>
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
<code>add_headshots</code> has no <code>season</code> or <code>variant</code> .	

SDVPLOT.PLOTNINE

Layer	Aesthetics
<code>geom_sdv_logos</code>	<code>x y team</code> , optional
<code>geom_sdv_wordmarks</code>	<code>season</code>
<code>geom_sdv_headshots</code>	<code>x y player_id</code>
<code>geom_from_path</code>	<code>x y path</code> (any image)
<code>geom_mean_lines</code>	<code>x0</code> and/or <code>y0</code>
<code>geom_median_lines</code>	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system`.

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv` : `which`, `season`, `na_value="grey"`,
`alpha`. The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, <code>great_tables</code>

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref / yref` pick a subplot's axes,
`layer="above" "below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos(chart, ...)` returns a new `LayerChart`;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type / y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height × frame_height`, else the
figure `height`. Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup`, "sdvplot
logos", for a `LayerControl`.

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style(teams, league=, which=)` : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy`.

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column(name, league=,`
`id_system="espn")`. Extra keywords go to
`ColumnDefinition`.

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), <code>great_tables</code> (<code>GT</code>)

SDVPLOT.TESTING

`check_adapter_contract(adapter, make_target)` and
`check_table_adapter_contract(adapter,`
`make_table)` raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()`, with the four verbs in
`__all__`, a `_drawn_marks(target)` test hook, and one
`register_adapter(...)` line in `_dispatch.py`.

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

[GALLERY ↗](#)

[COOKBOOKS ↗](#)

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column names → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv
)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables ↗](#)

THEMES

`gt_theme_sdv(gt, style="dark")`
`gt_theme_sdv_team(gt, "KC", league="nfl")`

sdv

sdv · dark

kenpom

savant

broadsheet

midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
<code>density="comfortable"</code> <code>"compact"</code> <code>"social"</code>	all
<code>accent</code>	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
<code>style="light"</code> <code>"dark"</code>	sdv, sofa, tier
<code>color · paper · stripe</code>	drench · broadsheet · almanac

Extra keywords go to `great_tables' tab_options`.

Cookbook: [Colours and themes ↗](#)

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)

gt_theme_preview(data, themes=None, n=5): the first
n rows in each theme. pal_midnight: a rank palette that
reads on midnight / terminal.
```

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar + marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/T rows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

(GT(df) .pipe(gt_color_ranks, "wins") .pipe(gt_legend_continuous, title="Wins", type="steps"))	
Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: gt_tiers's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. lambda s: s > 0.7; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

rows: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataverse"}))
```

Recipe: [CFB conference table ↗](#)

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from is `from_`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great_tables side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with sportsdataverse-py data, the sdvplotR name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
gt_grid(tables)	tables in a grid under one title; ncol=2, labels, gap=24
gt_stack_tables(tables)	one above another; gap=16, align
gt_snake(gt)	long table in side-by-side blocks; n_cols=2, rows_per_col
gt_snake_align(x)	reshape a parallel frame to match
gt_grid([afc, nfc], ncol=2, title="2025 standings") gt_grid(gt_theme_preview(df), ncol=3) GT(nba).pipe(gt_snake, n_cols=3)	
tables: a list of GT, or a dict (its values, in order). Both also take subtitle, caption, source_note and file=: None returns the HTML, a path writes an image. Style with title_style, subtitle_style, caption_style dicts.	

SAVE AS AN IMAGE

Function	Does · key arguments
gt_save_crop(gt, file)	PNG, trimmed, padded; whitespace=50, bg, zoom=2, width
gt_social_crop(gt, file)	pad to aspect_ratio="1:1", "4:5", ...; gravity
gt_save_batch(data, group, fn, file, dir=)	one image per group, match_width=True
gt_save_crop(tbl, "table.png", whitespace=40) gt_social_crop(tbl, "post.png", aspect_ratio="4:5") gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")	

build(df, value) returns a GT; {group} becomes the slugged value; it returns the files written. Saving runs headless Chrome through nokap, trimmed with Pillow. file=None returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
reactable_sdv_logos(league=)	a Column; height=30, include_name, season, default_img
reactable_sdv_wordmarks(league=)	a Column; as logos
reactable_sdv_headshots(league=)	a Column; height=40, id_system="espn"
reactable_sdv_cols_label(data, league=)	a list of Columns, marks as headers; mark_type
reactable_sdv_team_color_bar(data, team_col, league=)	a bar in team colour; which, max_value
reactable_sdv_team_color_bg(data, team_col, league=)	a team-colour cell fill; alpha=0.15
from reactable import Reactable from sdvplot.reactable import (reactable_sdv_logos, reactable_sdv_team_color_bar) Reactable(df, columns=[reactable_sdv_logos(league="nba", id="team"), reactable_sdv_team_color_bar(df, "team", league="nba", id="wins")])	

Other keywords (id, name,...) go to reactable.Column. Needs sdvplot[reactable].

PLAYING SURFACES

import matplotlib.pyplot as plt fig, ax = plt.subplots() sdvplot.surface("nba", "BOS", ax=ax, center_logo=True) ax.scatter(shots["court_x"], shots["court_y"])	surface(league, team=None, *, season, ax, center_logo) draws sportypy's field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the Axes. center_logo=True is 0.25 of the Axes height, or pass a fraction. Needs [surfaces].
---	---

SURFACE LEAGUES & OPTIONS

sdvplot.surface("nhl", "BOS", center_logo=0.3) sdvplot.surface("nfl", "KC", season=2010)	26 leagues plus "fiba" (no team colours): nfl cfb ufl usfl xfl aaf nba wnba nbagl mbb wbb mlb milb ncaa_baseball nhl ahl echl ohl qmjhl whl ushl pwhl phf ncaa_hockey ncaa_hockey soccer. sportypy keywords pass through: display_range, xlim, ylim, rotation to draw(); field_updates, color_updates, units,... to the surface. surface("nba", display_range="defense") is a half court.
---	---

Cookbook: [Surfaces and shot charts ↗](#)

COURT COORDINATES

# stats.nba.com legacy frame -> feet shots = sdvplot.court_coords(shots) # Euroleague (cm) -> FIBA court (m) pts = sdvplot.court_coords(pts, x="coord_x", y="coord_y", provider="euroleague")	Adds court_x, court_y (Float64) on sportypy's centre-court frame; returns the frame's own type (pandas or polars). Reads x_legacy / y_legacy (LOC_X / LOC_Y) by default. ESPN coordinate_x / _y are already in feet: don't convert them. Euroleague free throws (-1, -1) become null.
--	---

PITCH COORDINATES

# soccer events -> 105 x 68 m pitch ev = sdvplot.pitch_coords(ev, provider="statsbomb") fig, ax = plt.subplots() sdvplot.surface("soccer", ax=ax) ax.scatter(ev["pitch_x"], ev["pitch_y"])	Adds pitch_x, pitch_y in metres from the centre spot, attacking +x. Providers: opta (statsperform) wyscout statsbomb uefa impect espn; tracking tracab skillcorner secondspectrum metrica also need pitch_length, pitch_width. flip=True, or a boolean column, turns rows half a turn. ESPN reads field_position_x / _y, as sportsdataverse-py's espn_soccer_game_plays returns them; its (0, 0) becomes null.
--	--

WITH SPORTSDATAVERSE DATA

import polars as pl import sportsdataverse.nba as nba box = nba.load_nba_team_boxscore([2025]) off = box.group_by("team_abbreviation").agg(pl.col("assists", "team_score").mean()) fig, ax = plt.subplots() sdvplot.add_logos(ax, off["assists"], off["team_score"], off["team_abbreviation"], league="nba", height=0.06)	
--	--

Per league: [NFL](#) · [CFB](#) · [NBA](#) · [WNBA](#) · [MLB](#) · [NHL](#) · [PWHL](#) · [Soccer](#) · [Cricket](#)

FROM SDVPLOTR

sdvplotR	sdvplot
sdv_teams() · sdv_resolve() · sdv_marks()	teams() · resolve() · marks()
sdv_logo_url() · sdv_headshot_url()	logo_url() · headshot_url()
sdv_color_palette() · sdv_team_colors()	palette() · team_colors()
geom_sdv_logos() & co.	same names in sdvplot.plotnine; add_logos() anywhere
element_sdv_logo(), scale_x_sdv_headshots()	axis_logos()
ggtitle_image() · sdv_team_tiers()	title_image() · team_tiers()
gt_*() · reactable_sdv_*()	same names

Arguments match (league, id_system, season, strict, variant = "default"); only from → from_. Not ported: sdv_team_factor() (three lines over resolve()).

LEARN MORE

[GALLERY ↗](#) every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** [sdvplotR \(R\)](#), [@sportsdataverse/sdvplot \(npm\)](#)



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to `sdvplotR`. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: `great_tables`. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

[GALLERY ↗](#)

[COOKBOOKS ↗](#)

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfb" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started ↗](#)

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a `ggplot`, a `Plotly` figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos ↗](#)

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")
```

`id_system="espn"` (default, every league), `"gsis"` (NFL, via `nflverse`) or `"league"` (NFL `gsis`, NBA / WNBA Stats, MLBAM, NHL ids). An unknown id gets the CDN's silhouette.

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`); `team_colors()` returns one colour per value (a str, list or `Series`, like its input; `None` for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes ↗](#)

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("NFL", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity ↗](#)

WHICH ADAPTER RUNS

The target's library picks it: a `matplotlib` Axes or `seaborn`, a `plotnine` `ggplot`, a `great_tables` `GT` (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a `Plotly`, `Altair`, `Bokeh`, `HoloViews`, `folium` or `pygal` object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: `matplotlib`, `seaborn`, `plotnine`, `Plotly`, `Altair`. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web ↗](#)

ID SYSTEMS

`id_system="auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `ncaa` `pff` `cricinfo` `cfbd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries. `strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default"` `"dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras ↗](#)

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
milb ncaa_baseball ncaa_softball	286 / 414 / 415
ncaa_mhockey ncaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache ↗](#)

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors ↗](#)

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, which and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

`plotnine`: `scale_color_sdv("nfl")`; `pygal`: `team_style()`; `tables`: `gt_theme_sdv_team`, `reactable_sdv_team_color_bg`.



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or `https` URLs
(http is refused). `axis_logos` (`ax` , `"y"` , `mark_type=`) :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left"` `"right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc` ,
`presort` , `no_line_below_tier` , `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from marks()
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDVPLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system` .

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)` , so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`) ,
`scale_fill_sdv` : `which` , `season` , `na_value="grey"` ,
`alpha` . The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref` / `yref` pick a subplot's axes,
`layer="above"` `"below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos` (`chart` , ...) returns a new `LayerChart` ;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type` / `y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height` × `frame_height` , else the
figure `height` . Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup` , "sdvplot
logos", for a `LayerControl` .

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style` (`teams` , `league=` , `which=`) : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy` .

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column` (`name` , `league=` ,
`id_system="espn"`) . Extra keywords go to
`ColumnDefinition` .

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>) , Altair (<code>LayerChart</code>) , HoloViews (a copy), great_tables (GT)

SDVPLOT.TESTING

`check_adapter_contract` (`adapter` , `make_target`) and
`check_table_adapter_contract` (`adapter` ,
`make_table`) raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()` , with the four verbs in
`__all__` , `a_drawn_marks` (`target`) test hook, and one
`register_adapter(...)` line in `_dispatch.py` .

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column <i>names</i> → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#) ↗

THEMES

```
gt_theme_sdv(gt, style="dark")
gt_theme_sdv_team(gt, "KC", league="nfl")
```

gt_theme_sdv Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	gt_theme_sdv_dark Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	gt_theme_kenpom Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.33</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4</td><td>durian</td><td>four</td><td>65100.0</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	33.33	coconut	three	1.39	444.4	durian	four	65100.0							
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		
33.33	coconut	three	1.39																																																		
444.4	durian	four	65100.0																																																		
sdv	sdv - dark	kenpom																																																			
gt_theme_savant Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.33</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4</td><td>durian</td><td>four</td><td>65100.0</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	33.33	coconut	three	1.39	444.4	durian	four	65100.0	gt_theme_broadsheet Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.33</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	33.33	coconut	three	1.39	gt_theme_midnight Sample table <table><tr><th>num</th><th>char</th><th>fctr</th><th>currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.222</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	num	char	fctr	currency	0.1111	apricot	one	49.95	2.222	banana	two	17.95	savant	broadsheet	midnight
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		
33.33	coconut	three	1.39																																																		
444.4	durian	four	65100.0																																																		
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		
33.33	coconut	three	1.39																																																		
num	char	fctr	currency																																																		
0.1111	apricot	one	49.95																																																		
2.222	banana	two	17.95																																																		

THEME OPTIONS

Each is `gt_theme_<name>():` almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte,plus sdv and sdv_team.

Argument	Themes
density="comfortable" "compact" "social"	all
accent	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
style="light" "dark"	sdv, sofa, tier
color · paper · stripe	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)
```

`gt_theme_preview(data, themes=None, n=5):` the first `n` rows in each theme. `pal_midnight`: a rank palette that reads on midnight / terminal.

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/Trows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

```
(GT(df)
 .pipe(gt_color_ranks, "wins")
 .pipe(gt_legend_continuous,
       title="Wins", type="steps"))
```

Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: gt_tiers 's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. <code>lambda s: s > 0.7</code> ; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

```
(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))
```

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataVerse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great `tables` side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with `sportsdataverse-py` data, the `sdvplotR` name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
<code>gt_grid(tables)</code>	tables in a grid under one title; <code>ncol=2</code> , <code>labels</code> , <code>gap=24</code>
<code>gt_stack_tables(tables)</code>	one above another; <code>gap=16</code> , <code>align</code>
<code>gt_snake(gt)</code>	long table in side-by-side blocks; <code>n_cols=2</code> , <code>rows_per_col</code>
<code>gt_snake_align(x)</code>	reshape a parallel frame to match
<code>gt_grid([afc, nfc], ncol=2, title="2025 standings")</code> <code>gt_grid(gt_theme_preview(df), ncol=3)</code> <code>GT(nba).pipe(gt_snake, n_cols=3)</code>	

`tables`: a list of `GT`, or a dict (its values, in order). Both also take `subtitle`, `caption`, `source_note` and `file=`: `None` returns the HTML, a path writes an image. Style with `title_style`, `subtitle_style`, `caption_style` dicts.

SAVE AS AN IMAGE

Function	Does · key arguments
<code>gt_save_crop(gt, file)</code>	PNG, trimmed, padded; <code>whitespace=50</code> , <code>bg</code> , <code>zoom=2</code> , <code>width</code>
<code>gt_social_crop(gt, file)</code>	pad to <code>aspect_ratio="1:1"</code> , "4:5", ...; <code>gravity</code>
<code>gt_save_batch(data, group, fn, file, dir=)</code>	one image per group, <code>match_width=True</code>

<code>gt_save_crop(tbl, "table.png", whitespace=40)</code>
<code>gt_social_crop(tbl, "post.png", aspect_ratio="4:5")</code>
<code>gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")</code>

`build(df, value)` returns a `GT`; `{group}` becomes the slugged value; it returns the files written. Saving runs headless Chrome through `nokap`, trimmed with `Pillow`. `file=None` returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
<code>reactable_sdv_logos(league=)</code>	a <code>Column</code> ; <code>height=30</code> , <code>include_name</code> , <code>season</code> , <code>default_img</code>
<code>reactable_sdv_wordmarks(league=)</code>	a <code>Column</code> ; as logos
<code>reactable_sdv_headshots(league=)</code>	a <code>Column</code> ; <code>height=40</code> , <code>id_system="espn"</code>
<code>reactable_sdv_cols_label(data, league=)</code>	a <i>list</i> of <code>Columns</code> , marks as headers; <code>mark_type</code>
<code>reactable_sdv_team_color_bar(data, team_col, league=)</code>	a bar in team colour; <code>which</code> , <code>max_value</code>
<code>reactable_sdv_team_color_bg(data, team_col, league=)</code>	a team-colour cell fill; <code>alpha=0.15</code>

```
from reactable import Reactable
from sdvplot.reactable import (
    reactable_sdv_logos,
    reactable_sdv_team_color_bar)
Reactable(df, columns=[
    reactable_sdv_logos(league="nba",
                        id="team"),
    reactable_sdv_team_color_bar(df, "team",
                                league="nba", id="wins")])
```

Other keywords (`id`, `name`, ...) go to `reactable.Column`. Needs `sdvplot[reactable]`.

PLAYING SURFACES

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sdvplot.surface("nba", "BOS", ax=ax,
                center_logo=True)
ax.scatter(shots["court_x"],
           shots["court_y"]])
```

`surface(league, team=None, *, season, ax, center_logo)` draws `sportypy`'s field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the `Axes`. `center_logo=True` is 0.25 of the `Axes` height, or pass a fraction. Needs `[surfaces]`.

SURFACE LEAGUES & OPTIONS

```
sdvplot.surface("nhl", "BOS",
               center_logo=0.3)
sdvplot.surface("nfl", "KC", season=2010)
```

26 leagues plus "fiba" (no team colours): `nfl` `cfb` `ufl` `usfl` `xfl` `aaf` `nba` `wnba` `nbagl` `mbb` `wbb` `mlb` `milb` `ncaa_baseball` `nhl` `ahl` `echl` `ohl` `qmjhl` `whl` `ushl` `pwhl` `phf` `ncaa_ohockey` `ncaa_whockey` `soccer`.

`sportypy` keywords pass through: `display_range`, `xlim`, `ylim`, `rotation` to `draw()`; `field_updates`, `color_updates`, `units`, ... to the surface. `surface("nba", display_range="defense")` is a half court.

Cookbook: [Surfaces and shot charts ↗](#)

COURT COORDINATES

```
# stats.nba.com legacy frame -> feet
shots = sdvplot.court_coords(shots)
# Euroleague (cm) -> FIBA court (m)
pts = sdvplot.court_coords(pts,
                             x="coord_x", y="coord_y",
                             provider="euroleague")
```

Adds `court_x`, `court_y` (`Float64`) on `sportypy`'s centre-court frame; returns the frame's own type (`pandas` or `polars`). Reads `x_legacy` / `y_legacy` (`LOC_X` / `LOC_Y`) by default. ESPN `coordinate_x` / `_y` are already in feet: don't convert them. Euroleague free throws (`-1`, `-1`) become null.

PITCH COORDINATES

```
# soccer events -> 105 x 68 m pitch
ev = sdvplot.pitch_coords(ev,
                           provider="statsbomb")
fig, ax = plt.subplots()
sdvplot.surface("soccer", ax=ax)
ax.scatter(ev["pitch_x"], ev["pitch_y"]])
```

Adds `pitch_x`, `pitch_y` in metres from the centre spot, attacking +x. Providers: `opta` (`statsperform`) `wyscout` `statsbomb` `uefa` `impect` `espn`; tracking `tracab` `skillcorner` `secondspectrum` `metrica` also need `pitch_length`, `pitch_width`. `flip=True`, or a boolean column, turns rows half a turn. ESPN reads `field_position_x` / `_y`, as `sportsdataverse-py`'s `espn_soccer_game_plays` returns them; its `(0, 0)` becomes null.

WITH SPORTSDATAVERSE DATA

```
import polars as pl
import sportsdataverse.nba as nba
box = nba.load_nba_team_boxscore([2025])
off = box.group_by("team_abbreviation").agg(
    pl.col("assists", "team_score").mean())
fig, ax = plt.subplots()
sdvplot.add_logos(ax, off["assists"],
                  off["team_score"],
                  off["team_abbreviation"],
                  league="nba", height=0.06)
```

Per league: [NFL](#) · [CFB](#) · [NBA](#) · [WNBA](#) · [MLB](#) · [NHL](#) · [PWHL](#) · [Soccer](#) · [Cricket](#)

FROM SDVPLOTR

sdvplotR	sdvplot
<code>sdv_teams()</code> · <code>sdv_resolve()</code> · <code>sdv_marks()</code>	<code>teams()</code> · <code>resolve()</code> · <code>marks()</code>
<code>sdv_logo_url()</code> · <code>sdv_headshot_url()</code>	<code>logo_url()</code> · <code>headshot_url()</code>
<code>sdv_color_palette()</code> · <code>sdv_team_colors()</code>	<code>palette()</code> · <code>team_colors()</code>
<code>geom_sdv_logos()</code> & <code>co.</code>	same names in <code>sdvplot.plotnine</code> ; <code>add_logos()</code> anywhere
<code>element_sdv_logo()</code> , <code>scale_x_sdv_headshots()</code>	<code>axis_logos()</code>
<code>ggtitle_image()</code> · <code>sdv_team_tiers()</code>	<code>title_image()</code> · <code>team_tiers()</code>
<code>gt_*()</code> · <code>reactable_sdv_*()</code>	same names

Arguments match (`league`, `id_system`, `season`, `strict`, `variant = "default"`); only `from` → `from_`. Not ported: `sdv_team_factor()` (three lines over `resolve()`).

LEARN MORE

GALLERY ↗

every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** `sdvplotR` (R), [@sportsdataverse/sdvplot](#) (npm)



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to sdvplotR. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: great_tables. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

GALLERY ↗

COOKBOOKS ↗

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfb" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started](#) ↗

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a ggplot, a Plotly figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos](#) ↗

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")
```

`id_system="espn"` (default, every league), `"gsis"` (NFL, via `nflverse`) or `"league"` (NFL `gsis`, NBA / WNBA Stats, MLBAM, NHL ids). An unknown id gets the CDN's silhouette.

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`; `team_colors()` returns one colour per value (a str, list or Series, like its input; None for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes](#) ↗

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (<code>team_id</code> , <code>name</code>) matches
<code>teams(league=None)</code>	polars index: <code>abbr</code> <code>name</code> <code>conference</code> <code>color_primary</code> <code>color_source</code> , ...
<code>logo_url()</code>	a mark's URL; <code>variant</code> , <code>mark_type="logo"</code> <code>"wordmark"</code>
<code>logo_image()</code>	a PIL image; <code>size</code>
<code>marks()</code>	every archived mark: <code>variant</code> <code>mark_type</code> <code>archive_url</code>
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
    ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("nfl", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity](#) ↗

WHICH ADAPTER RUNS

The target's library picks it: a matplotlib Axes or seaborn, a plotnine ggplot, a great_tables GT (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a Plotly, Altair, Bokeh, HoloViews, folium or pygal object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: matplotlib, seaborn, plotnine, Plotly, Altair. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web](#) ↗

ID SYSTEMS

`id_system="auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `naaa` `pff` `cricinfo` `cfbd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries.

`strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default" "dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras](#) ↗

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl ushl pwhl phf	hockey minors & juniors
mlb naaa_baseball naaa_softball	286 / 414 / 415
naaa_mhockey naaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached.

`SDV_PLOT_CACHE_DIR` moves the cache; after `SDV_PLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache](#) ↗

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors](#) ↗

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, `which` and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

plotnine: `scale_color_sdv("nfl")`; pygal: `team_style()`; tables: `gt_theme_sdv_team`, `reactable_sdv_team_color_bg`.



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or https URLs
(http is refused). `axis_logos(ax, "y", mark_type=)` :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left" "right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc`,
`presort`, `no_line_below_tier`, `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from <code>marks()</code>
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDVPLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system`.

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs`. `team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv` : which, `season`, `na_value="grey"`,
`alpha`. The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref / yref` pick a subplot's axes,
`layer="above" "below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos(chart, ...)` returns a new `LayerChart` ;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type / y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height × frame_height`, else the
figure `height`. Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup`, "sdvplot
logos", for a `LayerControl`.

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style(teams, league=, which=)` : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy`.

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column(name, league=,`
`id_system="espn")`. Extra keywords go to
`ColumnDefinition`.

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), great_tables (<code>GT</code>)

SDVPLOT.TESTING

`check_adapter_contract(adapter, make_target)` and
`check_table_adapter_contract(adapter,`
`make_table)` raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()`, with the four verbs in
`__all__`, a `_drawn_marks(target)` test hook, and one
`register_adapter(...)` line in `_dispatch.py`.

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column names → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#) ↗

THEMES

<code>gt_theme_sdv(gt, style="dark")</code> <code>gt_theme_sdv_team(gt, "KC", league="nfl")</code>	<code>gt_theme_sdv_dark</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95	<code>gt_theme_kenpom</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39 444.4 durian four 65100.0
sdv	sdv · dark	kenpom
<code>gt_theme_savant</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39 444.4 durian four 65100.0	<code>gt_theme_broadsheet</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95 33.33 coconut three 1.39	<code>gt_theme_midnight</code> Sample table num char fctr currency 0.1111 apricot one 49.95 2.222 banana two 17.95
savant	broadsheet	midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
<code>density="comfortable"</code> <code>"compact"</code> <code>"social"</code> <code>accent</code>	all almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
<code>style="light"</code> <code>"dark"</code>	sdv, sofa, tier
<code>color · paper · stripe</code>	drench · broadsheet · almanac

Extra keywords go to `great_tables' tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)

gt_theme_preview(data, themes=None, n=5): the first
n rows in each theme. pal_midnight: a rank palette that
reads on midnight / terminal.
```



COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar + marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/T rows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

(GT(df) .pipe(gt_color_ranks, "wins") .pipe(gt_legend_continuous, title="Wins", type="steps"))	
Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: gt_tiers's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. lambda s: s > 0.7; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

rows: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataverse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from_`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great_tables side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with sportsdataverse-py data, the sdvplotR name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
gt_grid(tables)	tables in a grid under one title; ncol=2, labels, gap=24
gt_stack_tables(tables)	one above another; gap=16, align
gt_snake(gt)	long table in side-by-side blocks; n_cols=2, rows_per_col
gt_snake_align(x)	reshape a parallel frame to match

```
gt_grid([afc, nfc], ncol=2,
        title="2025 standings")
gt_grid(gt_theme_preview(df), ncol=3)
GT(nba).pipe(gt_snake, n_cols=3)
```

tables: a list of GT, or a dict (its values, in order). Both also take subtitle, caption, source_note and file=: None returns the HTML, a path writes an image. Style with title_style, subtitle_style, caption_style dicts.

SAVE AS AN IMAGE

Function	Does · key arguments
gt_save_crop(gt, file)	PNG, trimmed, padded; whitespace=50, bg, zoom=2, width
gt_social_crop(gt, file)	pad to aspect_ratio="1:1", "4:5", ...; gravity
gt_save_batch(data, group, fn, file, dir=)	one image per group, match_width=True

```
gt_save_crop(tbl, "table.png",
             whitespace=40)
gt_social_crop(tbl, "post.png",
               aspect_ratio="4:5")
gt_save_batch(nfl, "conference", build,
              "nfl-{group}.png", dir="out")
```

build(df, value) returns a GT; {group} becomes the slugged value; it returns the files written. Saving runs headless Chrome through nokap, trimmed with Pillow. file=None returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
reactable_sdv_logos(league=)	a Column; height=30, include_name, season, default_img
reactable_sdv_wordmarks(league=)	a Column; as logos
reactable_sdv_headshots(league=)	a Column; height=40, id_system="espn"
reactable_sdv_cols_label(data, league=)	a list of Columns, marks as headers; mark_type
reactable_sdv_team_color_bar(data, team_col, league=)	a bar in team colour; which, max_value
reactable_sdv_team_color_bg(data, team_col, league=)	a team-colour cell fill; alpha=0.15

```
from reactable import Reactable
from sdvplot.reactable import (
    reactable_sdv_logos,
    reactable_sdv_team_color_bar)
Reactable(df, columns=[
    reactable_sdv_logos(league="nba",
                        id="team"),
    reactable_sdv_team_color_bar(df, "team",
                                league="nba", id="wins")])
```

Other keywords (id, name, ...) go to reactable.Column. Needs sdvplot[reactable].

PLAYING SURFACES

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sdvplot.surface("nba", "BOS", ax=ax,
                center_logo=True)
ax.scatter(shots["court_x"],
           shots["court_y"])
```

surface(league, team=None, *, season, ax, center_logo) draws sportypy's field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the Axes. center_logo=True is 0.25 of the Axes height, or pass a fraction. Needs [surfaces].

SURFACE LEAGUES & OPTIONS

```
sdvplot.surface("nhl", "BOS",
                center_logo=0.3)
sdvplot.surface("nfl", "KC", season=2010)
```

26 leagues plus "fiba" (no team colours): nfl cfb ufl usfl xfl aaf nba wnba nbagl mbb wbb mlb milb ncaa_baseball nhl ahl echl ohl qmjhl whl ushl pwhl phf ncaa_mhockey ncaa_whockey soccer.

sportypy keywords pass through: display_range, xlim, ylim, rotation to draw(); field_updates, color_updates, units, ... to the surface. surface("nba", display_range="defense") is a half court.

Cookbook: [Surfaces and shot charts ↗](#)

COURT COORDINATES

```
# stats.nba.com legacy frame -> feet
shots = sdvplot.court_coords(shots)
# Euroleague (cm) -> FIBA court (m)
pts = sdvplot.court_coords(pts,
                             x="coord_x", y="coord_y",
                             provider="euroleague")
```

Adds court_x, court_y (Float64) on sportypy's centre-court frame; returns the frame's own type (pandas or polars). Reads x_legacy / y_legacy (LOC_X / LOC_Y) by default. ESPN coordinate_x / _y are already in feet: don't convert them. Euroleague free throws (-1, -1) become null.

PITCH COORDINATES

```
# soccer events -> 105 x 68 m pitch
ev = sdvplot.pitch_coords(ev,
                           provider="statsbomb")
fig, ax = plt.subplots()
sdvplot.surface("soccer", ax=ax)
ax.scatter(ev["pitch_x"], ev["pitch_y"])
```

Adds pitch_x, pitch_y in metres from the centre spot, attacking +x. Providers: opta (statsperform) wyscout statsbomb uefa impect espn; tracking tracab skillcorner secondspectrum metrica also need pitch_length, pitch_width. flip=True, or a boolean column, turns rows half a turn. ESPN reads field_position_x / _y, as sportsdataverse-py's espn_soccer_game_plays returns them; its (0, 0) becomes null.

WITH SPORTSDATAVERSE DATA

```
import polars as pl
import sportsdataverse.nba as nba
box = nba.load_nba_team_boxscore([2025])
off = box.group_by("team_abbreviation").agg(
    pl.col("assists", "team_score").mean())
fig, ax = plt.subplots()
sdvplot.add_logos(ax, off["assists"],
                  off["team_score"],
                  off["team_abbreviation"],
                  league="nba", height=0.06)
```

Per league: [NFL](#) · [CFB](#) · [NBA](#) · [WNBA](#) · [MLB](#) · [NHL](#) · [PWHL](#) · [Soccer](#) · [Cricket](#)

FROM SDVPLOTR

sdvplotR	sdvplot
sdv_teams() · sdv_resolve() · sdv_marks()	teams() · resolve() · marks()
sdv_logo_url() · sdv_headshot_url()	logo_url() · headshot_url()
sdv_color_palette() · sdv_team_colors()	palette() · team_colors()
geom_sdv_logos() & co.	same names in sdvplot.plotnine; add_logos() anywhere
element_sdv_logo(), scale_x_sdv_headshots()	axis_logos()
ggtitle_image() · sdv_team_tiers()	title_image() · team_tiers()
gt_*() · reactable_sdv_*()	same names

Arguments match (league, id_system, season, strict, variant = "default"); only from → from_. Not ported: sdv_team_factor() (three lines over resolve()).

LEARN MORE

GALLERY ↗

every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** [sdvplotR \(R\)](#), [@sportsdataverse/sdvplot \(npm\)](#)



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to `sdvplotR`. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: `great_tables`. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

GALLERY ↗

COOKBOOKS ↗

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfb" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started](#) ↗

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a `ggplot`, a `Plotly` figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos](#) ↗

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")

id_system="espn" (default, every league), "gsis" (NFL,
via nflverse) or "league" (NFL gsis, NBA / WNBA Stats,
MLBAM, NHL ids). An unknown id gets the CDN's silhouette.
```

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`); `team_colors()` returns one colour per value (a str, list or Series, like its input; `None` for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes](#) ↗

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("NFL", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity](#) ↗

WHICH ADAPTER RUNS

The target's library picks it: a `matplotlib` Axes or `seaborn`, a `plotnine` `ggplot`, a `great_tables` `GT` (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a `Plotly`, `Altair`, `Bokeh`, `HoloViews`, `folium` or `pygal` object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: `matplotlib`, `seaborn`, `plotnine`, `Plotly`, `Altair`. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web](#) ↗

ID SYSTEMS

`id_system="auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `ncaa` `pff` `cricinfo` `cfbd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries. `strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default"` `"dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras](#) ↗

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl ushl pwhl phf	hockey minors & juniors
milb ncaa_baseball ncaa_softball	286 / 414 / 415
ncaa_mhockey ncaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache](#) ↗

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors](#) ↗

TYPE HINTS

```
from sdvplot.typing import (
  IdSystem, HeadshotIdSystem, Which,
  MarkType, AxisMarkType)
```

The `Literal` types behind `id_system`, which and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

```
plotnine: scale_color_sdv("nfl"); pygal:
team_style(); tables: gt_theme_sdv_team,
reactable_sdv_team_color_bg.
```



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or `https` URLs
(http is refused). `axis_logos` (`ax` , `"y"` , `mark_type=`) :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left"` `"right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc` ,
`presort` , `no_line_below_tier` , `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from marks()
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDVPLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system` .

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)` , so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`) ,
`scale_fill_sdv` : `which` , `season` , `na_value="grey"` ,
`alpha` . The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot") ;
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref` / `yref` pick a subplot's axes,
`layer="above"` `"below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos` (`chart` , ...) returns a new `LayerChart` ;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type` / `y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                  league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height` × `frame_height` , else the
figure `height` . Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                        league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup` , "sdvplot
logos", for a `LayerControl` .

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style` (`teams` , `league=` , `which=`) : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy` .

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column` (`name` , `league=` ,
`id_system="espn"`) . Extra keywords go to
`ColumnDefinition` .

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>) , Altair (<code>LayerChart</code>) , HoloViews (a copy), great_tables (GT)

SDVPLOT.TESTING

`check_adapter_contract` (`adapter` , `make_target`) and
`check_table_adapter_contract` (`adapter` ,
`make_table`) raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()` , with the four verbs in
`__all__` , a `_drawn_marks` (`target`) test hook, and one
`register_adapter(...)` line in `_dispatch.py` .

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column <i>names</i> → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#) ↗

THEMES

```
gt_theme_sdv(gt, style="dark")
gt_theme_sdv_team(gt, "KC", league="nfl")
```

gt_theme_sdv	gt_theme_sdv_dark	gt_theme_kenpom
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	33.33 coconut three 1.39
2.222 banana two 17.95	2.222 banana two 17.95	444.4 durian four 65100.0

sdv

sdv · dark

kenpom

gt_theme_savant	gt_theme_broadsheet	gt_theme_midnight
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	0.1111 apricot one 49.95
33.33 coconut three 1.39	33.33 coconut three 1.39	2.222 banana two 17.95
444.4 durian four 65100.0	33.33 coconut three 1.39	

savant

broadsheet

midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
density="comfortable" "compact" "social"	all
accent	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
style="light" "dark"	sdv, sofa, tier
color · paper · stripe	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)
```

`gt_theme_preview(data, themes=None, n=5)`: the first `n` rows in each theme. `pal_midnight`: a rank palette that reads on `midnight` / `terminal`.

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/T/rows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

```
(GT(df)
 .pipe(gt_color_ranks, "wins")
 .pipe(gt_legend_continuous,
       title="Wins", type="steps"))
```

Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: <code>gt_tiers</code> 's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. <code>lambda s: s > 0.7</code> ; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L-(T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

```
(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))
```

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataVerse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from`. `gt_theme_preview` returns a dict; paletteeer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great `tables` side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with `sportsdataverse-py` data, the `sdvplotR` name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
<code>gt_grid(tables)</code>	tables in a grid under one title; <code>ncol=2</code> , <code>labels</code> , <code>gap=24</code>
<code>gt_stack_tables(tables)</code>	one above another; <code>gap=16</code> , <code>align</code>
<code>gt_snake(gt)</code>	long table in side-by-side blocks; <code>n_cols=2</code> , <code>rows_per_col</code>
<code>gt_snake_align(x)</code>	reshape a parallel frame to match
<code>gt_grid([afc, nfc], ncol=2, title="2025 standings")</code> <code>gt_grid(gt_theme_preview(df), ncol=3)</code> <code>GT(nba).pipe(gt_snake, n_cols=3)</code>	

`tables`: a list of `GT`, or a dict (its values, in order). Both also take `subtitle`, `caption`, `source_note` and `file=`: `None` returns the HTML, a path writes an image. Style with `title_style`, `subtitle_style`, `caption_style` dicts.

SAVE AS AN IMAGE

Function	Does · key arguments
<code>gt_save_crop(gt, file)</code>	PNG, trimmed, padded; <code>whitespace=50</code> , <code>bg</code> , <code>zoom=2</code> , <code>width</code>
<code>gt_social_crop(gt, file)</code>	pad to <code>aspect_ratio="1:1"</code> , "4:5", ...; <code>gravity</code>
<code>gt_save_batch(data, group, fn, file, dir=)</code>	one image per group, <code>match_width=True</code>

<code>gt_save_crop(tbl, "table.png", whitespace=40)</code>
<code>gt_social_crop(tbl, "post.png", aspect_ratio="4:5")</code>
<code>gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")</code>

`build(df, value)` returns a `GT`; `{group}` becomes the slugged value; it returns the files written. Saving runs headless Chrome through `nokap`, trimmed with `Pillow`. `file=None` returns the PIL image.

Automation: **Social game-day graphics** ↗

REACTABLE COLUMNS

Function	Returns · key arguments
<code>reactable_sdv_logos(league=)</code>	a <code>Column</code> ; <code>height=30</code> , <code>include_name</code> , <code>season</code> , <code>default_img</code>
<code>reactable_sdv_wordmarks(league=)</code>	a <code>Column</code> ; as logos
<code>reactable_sdv_headshots(league=)</code>	a <code>Column</code> ; <code>height=40</code> , <code>id_system="espn"</code>
<code>reactable_sdv_cols_label(data, league=)</code>	a <i>list</i> of <code>Columns</code> , marks as headers; <code>mark_type</code>
<code>reactable_sdv_team_color_bar(data, team_col, league=)</code>	a bar in team colour; <code>which</code> , <code>max_value</code>
<code>reactable_sdv_team_color_bg(data, team_col, league=)</code>	a team-colour cell fill; <code>alpha=0.15</code>

```
from reactable import Reactable
from sdvplot.reactable import (
    reactable_sdv_logos,
    reactable_sdv_team_color_bar)
Reactable(df, columns=[
    reactable_sdv_logos(league="nba",
                        id="team"),
    reactable_sdv_team_color_bar(df, "team",
                                league="nba", id="wins")])
```

Other keywords (`id`, `name`, ...) go to `reactable.Column`. Needs `sdvplot[reactable]`.

PLAYING SURFACES

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sdvplot.surface("nba", "BOS", ax=ax,
                center_logo=True)
ax.scatter(shots["court_x"],
           shots["court_y"]])
```

`surface(league, team=None, *, season, ax, center_logo)` draws `sportypy`'s field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the `Axes`. `center_logo=True` is 0.25 of the `Axes` height, or pass a fraction. Needs `[surfaces]`.

SURFACE LEAGUES & OPTIONS

```
sdvplot.surface("nhl", "BOS",
                center_logo=0.3)
sdvplot.surface("nfl", "KC", season=2010)
```

26 leagues plus "fiba" (no team colours): `nfl` `cfb` `ufl` `usfl` `xfl` `aaf` `nba` `wnba` `nbagl` `mbb` `wbb` `mlb` `milb` `ncaa_baseball` `nhl` `ahl` `echl` `ohl` `qmjhl` `whl` `ushl` `pwhl` `phf` `ncaa_mhockey` `ncaa_whokey` `soccer`.

`sportypy` keywords pass through: `display_range`, `xlim`, `ylim`, `rotation` to `draw()`; `field_updates`, `color_updates`, `units`, ... to the surface. `surface("nba", display_range="defense")` is a half court.

Cookbook: **Surfaces and shot charts** ↗

COURT COORDINATES

```
# stats.nba.com legacy frame -> feet
shots = sdvplot.court_coords(shots)
# Euroleague (cm) -> FIBA court (m)
pts = sdvplot.court_coords(pts,
                             x="coord_x", y="coord_y",
                             provider="euroleague")
```

Adds `court_x`, `court_y` (`Float64`) on `sportypy`'s centre-court frame; returns the frame's own type (`pandas` or `polars`). Reads `x_legacy` / `y_legacy` (`LOC_X` / `LOC_Y`) by default. `ESPN` `coordinate_x` / `_y` are already in feet: don't convert them. `Euroleague` free throws (`-1`, `-1`) become null.

PITCH COORDINATES

```
# soccer events -> 105 x 68 m pitch
ev = sdvplot.pitch_coords(ev,
                           provider="statsbomb")
fig, ax = plt.subplots()
sdvplot.surface("soccer", ax=ax)
ax.scatter(ev["pitch_x"], ev["pitch_y"]])
```

Adds `pitch_x`, `pitch_y` in metres from the centre spot, attacking +x. Providers: `opta` (`statsperform`) `wyscout` `statsbomb` `uefa` `impect` `espn`; tracking `tracab` `skillcorner` `secondspectrum` `metrica` also need `pitch_length`, `pitch_width`. `flip=True`, or a boolean column, turns rows half a turn. `ESPN` reads `field_position_x` / `_y`, as `sportsdataverse-py`'s `espn_soccer_game_plays` returns them; its (`0`, `0`) becomes null.

WITH SPORTSDATAVERSE DATA

```
import polars as pl
import sportsdataverse.nba as nba
box = nba.load_nba_team_boxscore([2025])
off = box.group_by("team_abbreviation").agg(
    pl.col("assists", "team_score").mean())
fig, ax = plt.subplots()
sdvplot.add_logos(ax, off["assists"],
                  off["team_score"],
                  off["team_abbreviation"],
                  league="nba", height=0.06)
```

Per league: **NFL** · **CFB** · **NBA** · **WNBA** · **MLB** · **NHL** · **PWHL** · **Soccer** · **Cricket**

FROM SDVPLOTR

sdvplotR	sdvplot
<code>sdv_teams()</code> · <code>sdv_resolve()</code> · <code>sdv_marks()</code>	<code>teams()</code> · <code>resolve()</code> · <code>marks()</code>
<code>sdv_logo_url()</code> · <code>sdv_headshot_url()</code>	<code>logo_url()</code> · <code>headshot_url()</code>
<code>sdv_color_palette()</code> · <code>sdv_team_colors()</code>	<code>palette()</code> · <code>team_colors()</code>
<code>geom_sdv_logos()</code> & <code>co.</code>	same names in <code>sdvplot.plotnine</code> ; <code>add_logos()</code> anywhere
<code>element_sdv_logo()</code> , <code>scale_x_sdv_headshots()</code>	<code>axis_logos()</code>
<code>ggtitle_image()</code> · <code>sdv_team_tiers()</code>	<code>title_image()</code> · <code>team_tiers()</code>
<code>gt_*()</code> · <code>reactable_sdv_*()</code>	same names

Arguments match (`league`, `id_system`, `season`, `strict`, `variant = "default"`); only `from` → `from_`. Not ported: `sdv_team_factor()` (three lines over `resolve()`).

LEARN MORE

GALLERY ↗

every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** `sdvplotR` (R), [@sportsdataverse/sdvplot](#) (npm)



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to sdvplotR. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: great_tables. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

GALLERY ↗

COOKBOOKS ↗

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfl" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started](#) ↗

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a ggplot, a Plotly figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
        "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos](#) ↗

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

```
sdvplot.headshot_url("3139477", "nfl")
```

`id_system="espn"` (default, every league), `"gsis"` (NFL, via `nflverse`) or `"league"` (NFL `gsis`, NBA / WNBA Stats, MLBAM, NHL ids). An unknown id gets the CDN's silhouette.

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`; `team_colors()` returns one colour per value (a str, list or Series, like its input; None for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes](#) ↗

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (<code>team_id</code> , <code>name</code>) matches
<code>teams(league=None)</code>	polars index: <code>abbr</code> <code>name</code> <code>conference</code> <code>color_primary</code> <code>color_source</code> , ...
<code>logo_url()</code>	a mark's URL; variant, <code>mark_type="logo"</code> <code>"wordmark"</code>
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: <code>variant</code> <code>mark_type</code> <code>archive_url</code>
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
    ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("nfl", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity](#) ↗

WHICH ADAPTER RUNS

The target's library picks it: a matplotlib Axes or seaborn, a plotnine ggplot, a great_tables GT (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a Plotly, Altair, Bokeh, HoloViews, folium or pygal object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: matplotlib, seaborn, plotnine, Plotly, Altair. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web](#) ↗

ID SYSTEMS

`id_system="auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `naaa` `pff` `cricinfo` `cflb` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries. `strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default"` `"dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras](#) ↗

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cflb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl ushl pwhl phf	hockey minors & juniors
mlb naaa_baseball naaa_softball	286 / 414 / 415
naaa_mhockey naaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache](#) ↗

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors](#) ↗

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, `which` and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

plotnine: `scale_color_sdv("nfl")`; pygal: `team_style()`; tables: `gt_theme_sdv_team`, `reactable_sdv_team_color_bg`.



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl` ;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or https URLs
(http is refused). `axis_logos(ax, "y", mark_type=)` :
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left" "right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc`,
`presort`, `no_line_below_tier`, `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from <code>marks()</code>
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDVPLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system`.

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv` : which, `season`, `na_value="grey"`,
`alpha`. The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref / yref` pick a subplot's axes,
`layer="above" "below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos(chart, ...)` returns a new `LayerChart` ;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type / y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height × frame_height`, else the
figure `height`. Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup`, "sdvplot
logos", for a `LayerControl`.

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style(teams, league=, which=)` : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on `XY`
charts at render time; copy a chart with `copy.deeppcopy`.

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column(name, league=,`
`id_system="espn")`. Extra keywords go to
`ColumnDefinition`.

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), great_tables (<code>GT</code>)

SDVPLOT.TESTING

`check_adapter_contract(adapter, make_target)` and
`check_table_adapter_contract(adapter,`
`make_table)` raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()`, with the four verbs in
`__all__`, a `_drawn_marks(target)` test hook, and one
`register_adapter(...)` line in `_dispatch.py`.

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column names → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv
)(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:
`sdvplot.add_logos(gt, "team", league="nfl")` is `gt_sdv_logos`.

Cookbook: [Tables](#) ↗

THEMES

`gt_theme_sdv(gt, style="dark")`
`gt_theme_sdv_team(gt, "KC", league="nfl")`

sdv

sdv · dark

kenpom

savant

broadsheet

midnight

THEME OPTIONS

Each is `gt_theme_<name>()`: almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
<code>density="comfortable"</code> <code>"compact"</code> <code>"social"</code> <code>accent</code>	all almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
<code>style="light"</code> <code>"dark"</code>	sdv, sofa, tier
<code>color · paper · stripe</code>	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)

gt_theme_preview(data, themes=None, n=5): the first
n rows in each theme. pal_midnight: a rank palette that
reads on midnight / terminal.
```



COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/Trows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

<code>(GT(df) .pipe(gt_color_ranks, "wins") .pipe(gt_legend_continuous, title="Wins", type="steps"))</code>	
Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: gt_tiers's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, eg. lambda s: s > 0.7; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

`(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))`

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataverse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from_`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great_tables side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with sportsdataverse-py data, the sdvplotR name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
gt_grid(tables)	tables in a grid under one title; ncol=2, labels, gap=24
gt_stack_tables(tables)	one above another; gap=16, align
gt_snake(gt)	long table in side-by-side blocks; n_cols=2, rows_per_col
gt_snake_align(x)	reshape a parallel frame to match
gt_grid([afc, nfc], ncol=2, title="2025 standings") gt_grid(gt_theme_preview(df), ncol=3) GT(nba).pipe(gt_snake, n_cols=3)	
tables: a list of GT, or a dict (its values, in order). Both also take subtitle, caption, source_note and file=: None returns the HTML, a path writes an image. Style with title_style, subtitle_style, caption_style dicts.	

SAVE AS AN IMAGE

Function	Does · key arguments
gt_save_crop(gt, file)	PNG, trimmed, padded; whitespace=50, bg, zoom=2, width
gt_social_crop(gt, file)	pad to aspect_ratio="1:1", "4:5", ...; gravity
gt_save_batch(data, group, fn, file, dir=)	one image per group, match_width=True
gt_save_crop(tbl, "table.png", whitespace=40) gt_social_crop(tbl, "post.png", aspect_ratio="4:5") gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")	

build(df, value) returns a GT; {group} becomes the slugged value; it returns the files written. Saving runs headless Chrome through nokap, trimmed with Pillow. file=None returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
reactable_sdv_logos(league=)	a Column; height=30, include_name, season, default_img
reactable_sdv_wordmarks(league=)	a Column; as logos
reactable_sdv_headshots(league=)	a Column; height=40, id_system="espn"
reactable_sdv_cols_label(data, league=)	a list of Columns, marks as headers; mark_type
reactable_sdv_team_color_bar(data, team_col, league=)	a bar in team colour; which, max_value
reactable_sdv_team_color_bg(data, team_col, league=)	a team-colour cell fill; alpha=0.15
from reactable import Reactable from sdvplot.reactable import (reactable_sdv_logos, reactable_sdv_team_color_bar) Reactable(df, columns=[reactable_sdv_logos(league="nba", id="team"), reactable_sdv_team_color_bar(df, "team", league="nba", id="wins")])	

Other keywords (id, name, ...) go to reactable.Column. Needs sdvplot[reactable].

PLAYING SURFACES

import matplotlib.pyplot as plt fig, ax = plt.subplots() sdvplot.surface("nba", "BOS", ax=ax, center_logo=True) ax.scatter(shots["court_x"], shots["court_y"])	surface(league, team=None, *, season, ax, center_logo) draws sportypy's field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the Axes. center_logo=True is 0.25 of the Axes height, or pass a fraction. Needs [surfaces].
---	---

SURFACE LEAGUES & OPTIONS

sdvplot.surface("nhl", "BOS", center_logo=0.3) sdvplot.surface("nfl", "KC", season=2010)	26 leagues plus "fiba" (no team colours): nfl cfb ufl usfl xfl aaf nba wnba nbagl mbb wbb mlb milb ncaa_baseball nhl ahl echl ohl qmjhl whl ushl pwhl phf ncaa_mhockey ncaa_whockey soccer.
sportypy keywords pass through: display_range, xlim, ylim, rotation to draw(); field_updates, color_updates, units, ... to the surface. surface("nba", display_range="defense") is a half court.	
Cookbook: Surfaces and shot charts ↗	

COURT COORDINATES

# stats.nba.com legacy frame -> feet shots = sdvplot.court_coords(shots) # Euroleague (cm) -> FIBA court (m) pts = sdvplot.court_coords(pts, x="coord_x", y="coord_y", provider="euroleague")	Adds court_x, court_y (Float64) on sportypy's centre-court frame; returns the frame's own type (pandas or polars). Reads x_legacy / y_legacy (LOC_X / LOC_Y) by default. ESPN coordinate_x / _y are already in feet: don't convert them. Euroleague free throws (-1, -1) become null.
--	---

PITCH COORDINATES

# soccer events -> 105 x 68 m pitch ev = sdvplot.pitch_coords(ev, provider="statsbomb") fig, ax = plt.subplots() sdvplot.surface("soccer", ax=ax) ax.scatter(ev["pitch_x"], ev["pitch_y"])	Adds pitch_x, pitch_y in metres from the centre spot, attacking +x. Providers: opta (statsperform) wyscout statsbomb uefa impect espn; tracking tracab skillcorner secondspectrum metrica also need pitch_length, pitch_width. flip=True, or a boolean column, turns rows half a turn. ESPN reads field_position_x / _y, as sportsdataverse-py's espn_soccer_game_plays returns them; its (0, 0) becomes null.
--	--

WITH SPORTSDATAVERSE DATA

import polars as pl import sportsdataverse.nba as nba box = nba.load_nba_team_boxscore([2025]) off = box.group_by("team_abbreviation").agg(pl.col("assists", "team_score").mean()) fig, ax = plt.subplots() sdvplot.add_logos(ax, off["assists"], off["team_score"], off["team_abbreviation"], league="nba", height=0.06)	Per league: NFL · CFB · NBA · WNBA · MLB · NHL · PWHL · Soccer · Cricket
FROM SDVPLOTR	
sdvplotR	sdvplot
sdv_teams() · sdv_resolve() · sdv_marks()	teams() · resolve() · marks()
sdv_logo_url() · sdv_headshot_url()	logo_url() · headshot_url()
sdv_color_palette() · sdv_team_colors()	palette() · team_colors()
geom_sdv_logos() & co.	same names in sdvplot.plotnine; add_logos() anywhere
element_sdv_logo(), scale_x_sdv_headshots()	axis_logos()
ggtitle_image() · sdv_team_tiers()	title_image() · team_tiers()
gt_*() · reactable_sdv_*()	same names
Arguments match (league, id_system, season, strict, variant = "default"); only from → from_. Not ported: sdv_team_factor() (three lines over resolve()).	
LEARN MORE	
GALLERY ↗ every chart and table in one place	
Start: Getting started · reference	
Concepts: identity · colours · logos · seasons · cache · adapters	
Recipes: NFL EPA · CFB table · head-to-head card · NHL shot map	
Leaderboards: NFL weekly · NBA · Siblings: sdvplotR (R), @sportsdataverse/sdvplot (npm)	



sdvplot :: CHEAT SHEET

Team logos, wordmarks, player headshots and team colours for Python, across **28 leagues** (pro, college, minor-league and junior). One front door, `add_logos()`, draws on **matplotlib** (and seaborn), **plotnine**, **plotly**, **altair**, **bokeh**, **holoviews**, **folium** and **pygal**. Resolves abbreviations, names and 19 provider id systems, and picks the right era's mark for a season. The Python counterpart to `sdvplotR`. Page 1: identity, colours, the front door. Page 2: every adapter. Page 3: `great_tables`. Page 4: table layout, export, reactable, playing surfaces.

sdvplot v0.1.0

Python ≥ 3.10 · polars · extras per library

[GALLERY ↗](#)

[COOKBOOKS ↗](#)

FRONT DOOR · ANY LIBRARY

INSTALL & IMPORT

```
pip install sdvplot # or: uv add sdvplot
pip install "sdvplot[mpl,tables]"
import sdvplot
```

Extras, one per library: `mpl` `plotnine` `plotly` `altair` `bokeh` `holoviews` `tables` `reactable` `folium` `pygal` `plottable` `surfaces` `svg`, or `all`.

Every call takes `league: "nfl" "nba" "wnba" "mlb" "nhl" "cfb" "mbb" "wbb" "soccer"` and 19 more (Coverage, col 3).

Cookbook: [Getting started ↗](#)

LOGOS, WORDMARKS & HEADSHOTS

Function	Draws
<code>add_logos(target, x, y, teams)</code>	team logos at points
<code>add_wordmarks()</code>	wordmarks (NFL, MLB, MiLB, PHF)
<code>add_headshots(.., players)</code>	player photos
<code>axis_logos(target, "x")</code>	logos as tick labels

The front door picks the adapter from `target` (an Axes, a `ggplot`, a `Plotly` figure, ...). Keywords: `league` (required), `height` (share of the axis), `alpha`, `season`, `variant`, `id_system`.

```
import matplotlib.pyplot as plt
seeds = ["KC", "BUF", "BAL", "HOU",
         "LAC", "PIT", "DEN"]
x = list(range(1, 8))
fig, ax = plt.subplots(figsize=(8, 1.6))
sdvplot.add_logos(ax, x, [0] * 7, seeds,
                  league="nfl", height=0.7)
ax.set(xlim=(0.5, 7.5), ylim=(-1, 1))
```

Cookbook: [Matplotlib logos ↗](#)

SAME CALL, OTHER LIBRARIES

```
import plotly.graph_objects as go
fig = go.Figure(go.Bar(x=seeds, y=x))
sdvplot.axis_logos(fig, "x", league="nfl")
```

Every `add_*` returns what it drew on: `plotnine`, `Altair` and `tables` give a new object, so keep it.

HEADSHOTS

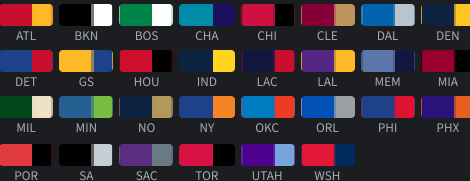
```
sdvplot.headshot_url("3139477", "nfl")

id_system="espn" (default, every league), "gsis" (NFL,
via nflverse) or "league" (NFL gsis, NBA / WNBA Stats,
MLBAM, NHL ids). An unknown id gets the CDN's silhouette.
```

TEAM COLOURS

```
sdvplot.palette("nfl", teams=["LV", "KC"])
#> {'LV': '#000000', 'KC': '#e31837'}
sdvplot.team_colors("nba", df["team"],
                    which="secondary")
```

`palette()` is a `{team: colour}` dict any library takes (seaborn `palette=`); `team_colors()` returns one colour per value (a str, list or `Series`, like its input; `None` for a miss). Both take `which="primary" "secondary"`, `season`, `id_system`, `strict`. Without `teams`, `palette` covers the whole league.



Cookbook: [Colours and themes ↗](#)

TEAM IDENTITY & MARKS

Function	Returns
<code>resolve(values, league)</code>	team ids from abbrs, names, provider ids
<code>suggest(value, league, n=5)</code>	closest (team_id, name) matches
<code>teams(league=None)</code>	polars index: abbr name conference color_primary color_source,...
<code>logo_url()</code>	a mark's URL; variant, mark_type="logo" "wordmark"
<code>logo_image()</code>	a PIL image; size
<code>marks()</code>	every archived mark: variant mark_type archive_url
<code>headshot_url()</code>	player photo URL

```
sdvplot.resolve(
    ["LV", "OAK", "Las Vegas Raiders"], "nfl")
#> ['13', '13', '13']
sdvplot.suggest("Golden St", "nba")
#> [['9', 'Golden State Warriors']]
sdvplot.teams("nfl", include_conferences=True)
```

`include_conferences` adds college conferences and AFC / NFC / NFL rows. `teams()` with no league is every league.

Concepts: [Team identity ↗](#)

WHICH ADAPTER RUNS

The target's library picks it: a `matplotlib` Axes or `seaborn`, a `plotnine` `ggplot`, a `great_tables` `GT` (`add_logos(gt, "team", league=)` is `gt_sdv_logos`), a `Plotly`, `Altair`, `Bokeh`, `HoloViews`, `folium` or `pygal` object. Any other raises `UnsupportedTargetError`; a missing extra, `OptionalDependencyError`.

`axis_logos` needs axes: `matplotlib`, `seaborn`, `plotnine`, `Plotly`, `Altair`. Elsewhere it raises `TypeError`. Page 2 has every module.

Cookbook: [Interactive web ↗](#)

ID SYSTEMS

`id_system="auto"` (the default) tries `team_id` `espn` `espn_abbr` `nfl` `nflverse` `mlbstats` `nba_api` `hockeytech` `ncaa` `pff` `cricinfo` `cfbd` `bref` `sportsipy` `fangraphs` `sdvplotr` `name` in order; pass one to pin it, or `"nhl_id"` for NHL stats ids, which `"auto"` never tries. `strict=True` (resolve, colours, URLs, tables) raises `UnresolvedTeamError` on a miss. A plot skips an unknown team with one `SdvplotWarning`.

SEASONS & VARIANTS

```
# the Oakland-era mark
sdvplot.logo_url("OAK", "nfl", season=2010)
sdvplot.logo_url("LV", "nfl", variant="dark")
sdvplot.marks("LV", "nfl") # every variant
```

`season=` works everywhere and picks the mark in use that year (15 leagues are season-dated). `variant="default"` `"dark"` always work; the archive adds `scoreboard`, `grayscale`, `on_light`, ... (the error lists a league's set).

Concepts: [Seasons and eras ↗](#)

COVERAGE

League	Teams
nfl nba wnba mlb nhl	32 / 30 / 15 / 30 / 32
cfb mbb wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjl whl ushl pwhl phf	hockey minors & juniors
milb ncaa_baseball ncaa_softball	286 / 414 / 415
ncaa_mhockey ncaa_whokey nbagl ufl usfl xfl aaf	110 / 48 / 32 / 11 / 9 / 11 / 8

Wordmarks: NFL, MLB, MiLB, PHF. Headshots: ESPN ids in every league.

CACHE & VERSIONS

```
sdvplot.versions()
#> keys: sdvplot, index,
#> manifest_last_modified
sdvplot.clear_cache()
```

The team index ships inside the package, so `resolve`, `palette` and `teams` never touch the network. Logos, the manifest and headshots download once and are cached. `SDVPLOT_CACHE_DIR` moves the cache; after `SDVPLOT_CACHE_TTL` days (default 7) the manifest is rechecked.

Concepts: [Cache ↗](#)

ERRORS & WARNINGS

Class	When
<code>SdvplotError</code>	base of every error below
<code>InputError</code>	a bad argument (a <code>ValueError</code>)
<code>UnresolvedTeamError</code>	a miss under <code>strict=True</code>
<code>OfflineError</code>	not cached, not downloadable
<code>DownloadError · Integrity Error</code>	CDN error status · sha256 mismatch
<code>OptionalDependencyError</code>	the extra is missing
<code>UnsupportedTargetError</code>	no adapter for the target
<code>UnsafeDownloadError</code>	over the byte cap, past the deadline, or off https
<code>UnsafeCachePathError</code>	a file would land outside the cache

Warnings: `SdvplotWarning`, `SdvplotDeprecationWarning`.

Reference: [Errors ↗](#)

TYPE HINTS

```
from sdvplot.typing import (
    IdSystem, HeadshotIdSystem, Which,
    MarkType, AxisMarkType)
```

The Literal types behind `id_system`, which and `mark_type`; the package ships `py.typed`.

COLOURS ANYWHERE

```
import seaborn as sns
sns.barplot(df, x="team", y="wins",
            hue="team",
            palette=sdvplot.palette("nfl"))
```

```
plotnine: scale_color_sdv("nfl"); pygal:
team_style(); tables: gt_theme_sdv_team,
reactable_sdv_team_color_bg.
```



sdvplot :: ADAPTERS

One module per plotting library, each with the same four verbs (`add_logos` , `add_wordmarks` , `add_headshots` , `axis_logos`) plus whatever that library needs. The front door on page 1 calls these for you; import a module directly for its extras or to skip the dispatch. Each needs its extra: `pip install "sdvplot[plotly]"` , and so on. Every web adapter takes `embed=True` to inline images as data URIs, for offline HTML and static export.

sdvplot v0.1.0

9 plot libraries · one contract

GALLERY ↗

COOKBOOKS ↗

ADAPTERS · ONE PER LIBRARY

SDVPLOT.MATPLOTLIB

`add_logos` `add_wordmarks` `add_headshots` `axis_logos`
`add_images` `title_image` `team_tiers`. Extra `mpl`;
seaborn routes here.

`height` : share of the Axes height, in (0, 1], default 0.1.
`zorder=3` draws above lines and markers. `transform` : a
Cartopy CRS such as `ccrs.PlateCarree()` (required on a
`GeoAxes`) or a `matplotlib` `Transform`.

```
from sdvplot.matplotlib import add_images
add_images(ax, [1, 2], [0, 0],
            ["a.png", "b.png"], height=0.2)
```

`add_images` takes local paths, `file://` URIs or `https` URLs
(http is refused). `axis_logos`(`ax`, `"y"`, `mark_type=`):
"logo" "wordmark" "headshot".

Reference: [sdvplot.matplotlib ↗](#)

TITLES & TIERS

```
from sdvplot.matplotlib import (
    title_image, team_tiers)
title_image(ax, "KC", "Chiefs offense",
            league="nfl", height=28)
team_tiers(tiers, "nba", title="Tiers",
            theme="light")
```

`title_image` : a team (with `league`) or an image URL or
path; `side="left"` `"right"`; `height` in points.

`team_tiers` returns a `Figure`. `tiers` has `tier_no` (1 is
top) and `team`, optionally `tier_rank`. Also `tier_desc` ,
`presort` , `no_line_below_tier` , `devel=True` (text, no
downloads), `variant="auto"` (dark marks on the dark
theme).

Recipe: [College hoops tiers ↗](#)

SHARED KEYWORDS

Keyword	Meaning
league	required, keyword only
height=0.1	share of the plot height
alpha=1	opacity, 0 to 1
season	the mark in use that year
variant	"default" "dark" or one from marks()
id_system	"auto" ; headshots "espn"
embed=False	web adapters: inline the images
add_headshots	has no season or variant.

SDVPLOT.PLOTNINE

Layer	Aesthetics
geom_sdv_logos	x y team, optional
geom_sdv_wordmarks	season
geom_sdv_headshots	x y player_id
geom_from_path	x y path (any image)
geom_mean_lines	x0 and/or y0
geom_median_lines	

Geom parameters: `league` `height=0.1` `alpha` `variant`
`id_system` .

```
from plotnine import aes, ggplot
from sdvplot.plotnine import (
    geom_sdv_logos, geom_mean_lines,
    title_image)
(ggplot(df, aes("epa", "sr", team="team"))
 + geom_mean_lines(aes(x0="epa", y0="sr"))
 + geom_sdv_logos(league="nfl",
                  height=0.12)
 + title_image("KC", "Chiefs",
               league="nfl", height=20))
```

`title_image` is added with `+` and replaces
`labs(title=)`, so add it after `labs.team_tiers(data,`
`league)` returns a `ggplot`.

TEAM COLOUR SCALES

```
from plotnine import geom_point
from sdvplot.plotnine import scale_color_sdv
(ggplot(df, aes("epa", "sr", color="team"))
 + geom_point() + scale_color_sdv("nfl"))
```

`scale_color_sdv` (alias `scale_colour_sdv`),
`scale_fill_sdv` : which, `season`, `na_value="grey"` ,
`alpha` . The `plotnine` module also has the four front-door
verbs.

Cookbook: [plotnine ↗](#)

LOGOS ON AN AXIS

```
sdvplot.axis_logos(ax, "x", league="nfl")
sdvplot.axis_logos(ax, "y", league="nba",
                    mark_type="headshot")
```

The tick labels must be teams (player ids for "headshot");
labels that are not stay as text, with one warning.

axis_logos	Libraries
draws	matplotlib, seaborn, plotnine, Plotly (paper coordinates), Altair
raises <code>Unsupported</code> <code>TargetError</code>	Bokeh, HoloViews, folium, pygal, great_tables

SDVPLOT.PLOTLY

```
import sdvplot.plotly as sp
sp.add_logos(fig, x, y, teams,
             league="nfl", xref="x2",
             yref="y2", layer="below",
             embed=True)
```

Layout images: `xref` / `yref` pick a subplot's axes,
`layer="above"` `"below"` the traces. `sdvplot` pins the axis
ranges so each image is exactly `height` of the plot.
`embed=True` also makes `kaleido` export work.

SDVPLOT.ALTAIR

```
import altair as alt
import sdvplot.altair as salt
pts = alt.Chart(df).mark_point().encode(
    x="epa", y="sr")
alt.layer(pts, salt.logo_layer(
    df["epa"], df["sr"], df["team"],
    league="nfl", height=0.12))
```

`add_logos`(`chart`, ...) returns a new `LayerChart`;
`logo_layer` is the layer itself. Its `chart_height` (px,
default 300), `x_type` / `y_type` ("quantitative"
"nominal" "ordinal" "temporal"). `embed=True` for vl-
convert PNGs.

SDVPLOT.BOKEH

```
from bokeh.plotting import figure
p = figure(frame_width=400,
           frame_height=300)
sdvplot.add_logos(p, x, y, teams,
                 league="nfl", height=0.15)
```

One `image_url` glyph per call, sized in screen pixels (logos
keep their size on zoom): `height` × `frame_height` , else the
figure `height` . Factor and datetime axes work.

SDVPLOT.HOLOVIEWS

```
import holoviews as hv
hv.extension("bokeh")
pts = hv.Scatter(list(zip(x, y))).opts(
    frame_height=300)
pts = sdvplot.add_logos(pts, x, y, teams,
                       league="nfl")
```

A Bokeh plot hook (Bokeh backend only); teams resolve, and
warn, at the call.

SDVPLOT.FOLIUM

```
import folium
m = folium.Map(location=[39, -95],
               zoom_start=4, height=600)
sdvplot.add_logos(m, [-94.48, -78.79],
                  [39.05, 42.77], ["KC", "BUF"],
                  league="nfl", height=0.06)
```

`x` is longitude, `y` latitude; `height` is a share of the map's
pixel height. Markers go in one `FeatureGroup` , "sdvplot
logos", for a `LayerControl` .

SDVPLOT.PYGAL

```
import pygal
import sdvplot.pygal as spg
chart = pygal.Bar(style=spg.team_style(
    ["KC", "BUF"], league="nfl"))
```

`team_style`(`teams`, `league=`, `which=`) : a `pygal` `Style`
in the teams' colours, one per series. `add_*` draw on XY
charts at render time; copy a chart with `copy.deeppcopy` .

SDVPLOT.PLOTTABLE

```
from plottable import Table
from sdvplot.plottable import logo_column
Table(df, column_definitions=[
    logo_column("team", league="nfl",
               title="")])
```

Also `headshot_column`(`name`, `league=`,
`id_system="espn"`) . Extra keywords go to
`ColumnDefinition` .

WHAT ADD_* RETURNS

Returns	Libraries
the target, drawn on	matplotlib, Plotly, Bokeh, folium, pygal
a new object; the target is unchanged	plotnine (<code>ggplot</code>), Altair (<code>LayerChart</code>), HoloViews (a copy), great_tables (GT)

SDVPLOT.TESTING

`check_adapter_contract`(`adapter`, `make_target`) and
`check_table_adapter_contract`(`adapter`,
`make_table`) raise an `AssertionError` naming the broken
rule.

A new adapter: a module named after the library, importing
it inside `requires_extra()` , with the four verbs in
`__all__` , `_drawn_marks`(`target`) test hook, and one
`register_adapter(...)` line in `_dispatch.py` .

Adapters: [Add an adapter ↗](#) · [contract ↗](#)



sdvplot :: TABLES

sdvplot.great_tables ports sdvplotR's table toolkit (gtUtils, by Andrew Weatherman) to **great_tables**, under the same names: team images, 20 themes, legends, colouring, annotations, layouts and image export. Each helper takes a `GT` first and returns a new one, so they chain with `GT.pipe`. Needs `sdvplot[tables]`. Page 4: layout, saving images, reactable.

sdvplot v0.1.0

great_tables ≥ 1.0 · nokap to save

GALLERY ↗

COOKBOOKS ↗

GREAT_TABLES · IMAGES, THEMES

TEAM IMAGES IN CELLS

Function	Does · key arguments
<code>gt_sdv_logos(gt, columns, league=)</code>	team keys → logos; height=30 (px), include_name, season, locations
<code>gt_sdv_wordmarks()</code>	keys → wordmarks; height, season
<code>gt_sdv_headshots()</code>	player ids → photos; id_system="espn"
<code>gt_sdv_cols_label()</code>	column <i>names</i> → images; mark_type="logo" "wordmark" "headshot"
<code>gt_merge_stack_team_color(gt, col1, col2, team_col)</code>	stack two columns, the lower in team colour; font_size_top=14, font_size_bottom=12

```
from great_tables import GT
from sdvplot.great_tables import (
    gt_sdv_logos, gt_theme_sdv)
(GT(df)
 .pipe(gt_sdv_logos, "team",
       league="nfl", height=28)
 .pipe(gt_theme_sdv))
```

Values resolve like `resolve()`, so abbreviations, names and provider ids all work; `strict=True` raises on a miss.

The front door works on a table too:

```
sdvplot.add_logos(gt, "team", league="nfl") is
gt_sdv_logos.
```

Cookbook: [Tables](#) ↗

THEMES

```
gt_theme_sdv(gt, style="dark")
gt_theme_sdv_team(gt, "KC", league="nfl")
```

gt_theme_sdv	gt_theme_sdv_dark	gt_theme_kenpom
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	33.33 coconut three 1.39
2.222 banana two 17.95	2.222 banana two 17.95	444.4 durian four 65100.0

sdv

sdv · dark

kenpom

gt_theme_savant	gt_theme_broadsheet	gt_theme_midnight
Sample table	Sample table	Sample table
num char fctr currency	num char fctr currency	num char fctr currency
0.1111 apricot one 49.95	0.1111 apricot one 49.95	0.1111 apricot one 49.95
2.222 banana two 17.95	2.222 banana two 17.95	2.222 banana two 17.95
33.33 coconut three 1.39	33.33 coconut three 1.39	33.33 coconut three 1.39
444.4 durian four 65100.0	33.33 coconut three 1.39	2.222 banana two 17.95

savant

broadsheet

midnight

THEME OPTIONS

Each is `gt_theme_<name>():` almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte, plus `sdv` and `sdv_team`.

Argument	Themes
density="comfortable" "compact" "social"	all
accent	almanac booktabs broadsheet brutalist midnight scoreboard swiss terminal tufte
style="light" "dark"	sdv, sofa, tier
color · paper · stripe	drench · broadsheet · almanac

Extra keywords go to `great_tables'` `tab_options`.

Cookbook: [Colours and themes](#) ↗

PREVIEW & PALETTES

```
gt_theme_preview(df) # {name: GT}
gt_theme_preview(df, ["kenpom", "pl"], n=8)
gt_color_ranks(gt, "wins",
               palette=pal_midnight)
```

`gt_theme_preview(data, themes=None, n=5)`: the first `n` rows in each theme. `pal_midnight`: a rank palette that reads on `midnight` / `terminal`.

COLOUR BY VALUE

Function	Does · key arguments
<code>gt_color_ranks()</code>	fill cells on a palette; palette, domain, reverse, rows
<code>gt_color_pills()</code>	value in a coloured pill; digits, format_type, suffix
<code>gt_percentile_bar()</code>	bar+marker; domain=(0, 100), decimals
<code>gt_color_results()</code>	W/L/T/rows; result_column, win_color, loss_color
<code>gt_row_accent(gt, column)</code>	bar on each row's edge; palette, side="left"
<code>gt_indicator_boxes()</code>	0/1 as boxes; indicator_vals, color_yes
<code>gt_tiers(gt, levels)</code>	colour a tier column; colors, tier_column="tier"

LEGENDS

```
(GT(df)
 .pipe(gt_color_ranks, "wins")
 .pipe(gt_legend_continuous,
       title="Wins", type="steps"))
```

Function	Does · key arguments
<code>gt_legend_continuous()</code>	colour bar; type="continuous" "steps" "blocks", n_bins=5, location="bottom"
<code>gt_legend_discrete(gt, key_info)</code>	{label: colour} key (default: <code>gt_tiers</code> 's); heading, shape

The legend reads the columns, palette and domain the colouring step recorded.

HIGHLIGHT & FOCUS

Function	Does · key arguments
<code>gt_highlight_cells(gt, columns, condition)</code>	fill where condition, e.g. <code>lambda s: s > 0.7</code> ; fill, bold
<code>gt_highlight_na()</code>	style missing values; missing_text, na_strings
<code>gt_outliers()</code>	method="iqr" "sd" "bounds", threshold, side, note
<code>gt_spotlight(gt, rows)</code>	emphasise rows, dim the rest; accent_color
<code>gt_bold_rows(gt, rows)</code>	bold rows; highlight_color
<code>gt_group_stripes()</code>	band alternate row groups; color, start=2

```
(GT(df)
 .pipe(gt_highlight_cells, "wins",
       lambda s: s >= 12, bold=True)
 .pipe(gt_spotlight, [0, 1]))
```

`rows`: 0-based positions, a polars expression, or a function of the frame.

BORDERS & RULES

Function	Does · key arguments
<code>gt_border_grid()</code>	full grid; color, weight=1
<code>gt_borderBars_top(gt, colors) / _bottom</code>	colour bars; bar_height=10, img, text
<code>gt_cutline(gt, after)</code>	a line after row after; label, style="dashed"

FORMAT

Function	Does · key arguments
<code>gt_delta(gt, from_, to)</code>	a change column; percent, arrows, after
<code>gt_fmt_rank()</code>	1 → 1st; superscript=True
<code>gt_fmt_tally()</code>	W-L(-T) in one cell; separator, share
<code>gt_significance(gt, columns, p_columns)</code>	stars; levels=(0.01, 0.05, 0.1), hide_p
<code>gt_scale_note()</code>	divide and say so; divisor=1000
<code>gt_wrap_labels()</code>	break labels; width=12
<code>gt_set_font(gt, font_family)</code>	from_google_font=True

```
(GT(df)
 .pipe(gt_fmt_tally, ["wins", "losses"])
 .pipe(gt_delta, "last_wins", "wins"))
```

HEADERS & ANNOTATIONS

Function	Does · key arguments
<code>gt_title_header(gt, title)</code>	kicker, subtitle, date, *_style
<code>gt_column_subheaders()</code>	col={"heading": .., "subtitle": ..}
<code>gt_538_caption()</code>	top_caption, bottom_caption
<code>gt_marginalia(gt, columns)</code>	a column as margin notes; width=220
<code>gt_social_tag(gt, accounts)</code>	handles with icons; stack, caption
<code>gt_watermark()</code>	text or image; opacity=0.06, angle

```
(GT(df)
 .pipe(gt_title_header, "NFC North",
       kicker="2025 SEASON",
       subtitle="Final standings")
 .pipe(gt_cutline, 1, label="Champion")
 .pipe(gt_social_tag,
       {"x": "@SportsDataVerse"}))
```

Recipe: [CFB conference table](#) ↗

TIPS & FROM R

`columns` takes a name, a list or a polars selector. Apply a theme before cell fills.

From R: `sport` is `league=`, `id_type` is `id_system`, from `is from`. `gt_theme_preview` returns a dict; paletteer names in `gt_legend_continuous` are not ported.



sdvplot :: LAYOUT, EXPORT & SURFACES

Put several great `tables` side by side and save them as trimmed PNGs for social posts; build **reactable** columns of logos, headshots and team-coloured bars; and draw a league's field, court, rink or pitch in a team's colours with **sportypy**, then move provider shot and event coordinates onto it. Ends with `sportsdataverse-py` data, the `sdvplotR` name map and where to read more.

sdvplot v0.1.0

[tables] · [reactable] · [surfaces]

GALLERY ↗

COOKBOOKS ↗

TABLES · LAYOUT & EXPORT

LAYOUT

Function	Does · key arguments
<code>gt_grid(tables)</code>	tables in a grid under one title; <code>ncol=2</code> , <code>labels</code> , <code>gap=24</code>
<code>gt_stack_tables(tables)</code>	one above another; <code>gap=16</code> , <code>align</code>
<code>gt_snake(gt)</code>	long table in side-by-side blocks; <code>n_cols=2</code> , <code>rows_per_col</code>
<code>gt_snake_align(x)</code>	reshape a parallel frame to match
<code>gt_grid([afc, nfc], ncol=2, title="2025 standings")</code> <code>gt_grid(gt_theme_preview(df), ncol=3)</code> <code>GT(nba).pipe(gt_snake, n_cols=3)</code>	

`tables`: a list of `GT`, or a dict (its values, in order). Both also take `subtitle`, `caption`, `source_note` and `file=`: `None` returns the HTML, a path writes an image. Style with `title_style`, `subtitle_style`, `caption_style` dicts.

SAVE AS AN IMAGE

Function	Does · key arguments
<code>gt_save_crop(gt, file)</code>	PNG, trimmed, padded; <code>whitespace=50</code> , <code>bg</code> , <code>zoom=2</code> , <code>width</code>
<code>gt_social_crop(gt, file)</code>	pad to <code>aspect_ratio="1:1"</code> , <code>"4:5"</code> , ...; <code>gravity</code>
<code>gt_save_batch(data, group, fn, file, dir=)</code>	one image per group, <code>match_width=True</code>

<code>gt_save_crop(tbl, "table.png", whitespace=40)</code>
<code>gt_social_crop(tbl, "post.png", aspect_ratio="4:5")</code>
<code>gt_save_batch(nfl, "conference", build, "nfl-{group}.png", dir="out")</code>

`build(df, value)` returns a `GT`; `{group}` becomes the slugged value; it returns the files written. Saving runs headless Chrome through `nokap`, trimmed with `Pillow`. `file=None` returns the PIL image.

Automation: [Social game-day graphics ↗](#)

REACTABLE COLUMNS

Function	Returns · key arguments
<code>reactable_sdv_logos(league=)</code>	a <code>Column</code> ; <code>height=30</code> , <code>include_name</code> , <code>season</code> , <code>default_img</code>
<code>reactable_sdv_wordmarks(league=)</code>	a <code>Column</code> ; as logos
<code>reactable_sdv_headshots(league=)</code>	a <code>Column</code> ; <code>height=40</code> , <code>id_system="espn"</code>
<code>reactable_sdv_cols_label(data, league=)</code>	a <i>list</i> of <code>Columns</code> , marks as headers; <code>mark_type</code>
<code>reactable_sdv_team_color_bar(data, team_col, league=)</code>	a bar in team colour; <code>which</code> , <code>max_value</code>
<code>reactable_sdv_team_color_bg(data, team_col, league=)</code>	a team-colour cell fill; <code>alpha=0.15</code>

```
from reactable import Reactable
from sdvplot.reactable import (
    reactable_sdv_logos,
    reactable_sdv_team_color_bar)
Reactable(df, columns=[
    reactable_sdv_logos(league="nba",
                        id="team"),
    reactable_sdv_team_color_bar(df, "team",
                                league="nba", id="wins")])
```

Other keywords (`id`, `name`, ...) go to `reactable.Column`. Needs `sdvplot[reactable]`.

PLAYING SURFACES

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sdvplot.surface("nba", "BOS", ax=ax,
                center_logo=True)
ax.scatter(shots["court_x"],
           shots["court_y"])
```

`surface(league, team=None, *, season, ax, center_logo)` draws `sportypy`'s field, court, rink or pitch in a team's colours (end zones, lane and apron, centre line, boards) and returns the `Axes`. `center_logo=True` is 0.25 of the `Axes` height, or pass a fraction. Needs `[surfaces]`.

SURFACE LEAGUES & OPTIONS

```
sdvplot.surface("nhl", "BOS",
                center_logo=0.3)
sdvplot.surface("nfl", "KC", season=2010)
```

26 leagues plus "fiba" (no team colours): `nfl` `cfb` `ufl` `usfl` `xfl` `aaf` `nba` `wnba` `nbagl` `mbb` `wbb` `mlb` `milb` `ncaa_baseball` `nhl` `ahl` `echl` `ohl` `qmjhl` `whl` `ushl` `pwhl` `phf` `ncaa_hockey` `ncaa_hockey` `soccer`.

`sportypy` keywords pass through: `display_range`, `xlim`, `ylim`, `rotation` to `draw()`; `field_updates`, `color_updates`, `units`, ... to the surface. `surface("nba", display_range="defense")` is a half court.

Cookbook: [Surfaces and shot charts ↗](#)

COURT COORDINATES

```
# stats.nba.com legacy frame -> feet
shots = sdvplot.court_coords(shots)
# Euroleague (cm) -> FIBA court (m)
pts = sdvplot.court_coords(pts,
                           x="coord_x", y="coord_y",
                           provider="euroleague")
```

Adds `court_x`, `court_y` (`Float64`) on `sportypy`'s centre-court frame; returns the frame's own type (`pandas` or `polars`). Reads `x_legacy` / `y_legacy` (`LOC_X` / `LOC_Y`) by default. `ESPN` `coordinate_x` / `_y` are already in feet: don't convert them. `Euroleague` free throws (`-1`, `-1`) become null.

PITCH COORDINATES

```
# soccer events -> 105 x 68 m pitch
ev = sdvplot.pitch_coords(ev,
                           provider="statsbomb")
fig, ax = plt.subplots()
sdvplot.surface("soccer", ax=ax)
ax.scatter(ev["pitch_x"], ev["pitch_y"])
```

Adds `pitch_x`, `pitch_y` in metres from the centre spot, attacking +x. Providers: `opta` (`statsperform`) `wyscout` `statsbomb` `uefa` `impect` `espn`; tracking `tracab` `skillcorner` `secondspectrum` `metrica` also need `pitch_length`, `pitch_width`. `flip=True`, or a boolean column, turns rows half a turn. `ESPN` reads `field_position_x` / `_y`, as `sportsdataverse-py`'s `espn_soccer_game_plays` returns them; its (`0`, `0`) becomes null.

WITH SPORTSDATAVERSE DATA

```
import polars as pl
import sportsdataverse.nba as nba
box = nba.load_nba_team_boxscore([2025])
off = box.group_by("team_abbreviation").agg(
    pl.col("assists", "team_score").mean())
fig, ax = plt.subplots()
sdvplot.add_logos(ax, off["assists"],
                  off["team_score"],
                  off["team_abbreviation"],
                  league="nba", height=0.06)
```

Per league: [NFL](#) · [CFB](#) · [NBA](#) · [WNBA](#) · [MLB](#) · [NHL](#) · [PWHL](#) · [Soccer](#) · [Cricket](#)

FROM SDVPLOTR

sdvplotR	sdvplot
<code>sdv_teams()</code> · <code>sdv_resolve()</code> · <code>sdv_marks()</code>	<code>teams()</code> · <code>resolve()</code> · <code>marks()</code>
<code>sdv_logo_url()</code> · <code>sdv_headshot_url()</code>	<code>logo_url()</code> · <code>headshot_url()</code>
<code>sdv_color_palette()</code> · <code>sdv_team_colors()</code>	<code>palette()</code> · <code>team_colors()</code>
<code>geom_sdv_logos()</code> & <code>co.</code>	same names in <code>sdvplot.plotnine</code> ; <code>add_logos()</code> anywhere
<code>element_sdv_logo()</code> , <code>scale_x_sdv_headshots()</code>	<code>axis_logos()</code>
<code>ggtitle_image()</code> · <code>sdv_team_tiers()</code>	<code>title_image()</code> · <code>team_tiers()</code>
<code>gt_*()</code> · <code>reactable_sdv_*()</code>	same names

Arguments match (`league`, `id_system`, `season`, `strict`, `variant = "default"`); only `from` → `from_`. Not ported: `sdv_team_factor()` (three lines over `resolve()`).

LEARN MORE

[GALLERY ↗](#) every chart and table in one place

Start: [Getting started](#) · [reference](#)

Concepts: [identity](#) · [colours](#) · [logos](#) · [seasons](#) · [cache](#) · [adapters](#)

Recipes: [NFL EPA](#) · [CFB table](#) · [head-to-head card](#) · [NHL shot map](#)

Leaderboards: [NFL weekly](#) · [NBA](#) · **Siblings:** `sdvplotR` (R), [@sportsdataverse/sdvplot](#) (npm)