sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: Introduction ↗

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: Identity and resolution ↗

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors`, `contrast`, `onColor`.



Guide: Team colours ↗

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot`, hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig`: a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette`; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts`: `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet`: passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background`: 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: Chart libraries ↗ · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams`, `resolve`, `marks`
`geom_sdv_logos` · `add_logos`	`logos`, `appendLogos`, `withLogos`
`scale_color_sdv`	`teamColor`, `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: Migrating ↗

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()`. League CDNs: `nba`, `wnba`, `mlb`, `nhl` `HeadshotUrl`.

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default`, `dark`, `scoreboard`, `grayscale`, `on_light`, …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: Logos, wordmarks, headshots ↗

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES`.

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError`, `OfflineError`, `OptionalDependencyError`, …); `setWarningHandler`, `resetManifestCache`, `versions`.

Guides: Node and SSR ↗ · Export to PNG ↗

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole `PlotOptions`; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; `season` per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

`@sportsdataverse/sporty` builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. `sdvplot` paints those surfaces in a team's colours, bins and compares shots (`/bins`, `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0

sporty 0.3.0 · sportyR 2.4.5 parity

GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution`. Or `surface(sport, league, o)`. Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges`; `SPORTS`; the `*_SPECS` tables from `/specs`.

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)`; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

FRAMES	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy`, `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)`; options `x`, `y`, `out`. Units: `convertUnits`, `convertPoints`. Zones: `basketballZones(league, { scale, top })`, `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }`; also `season`, `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface`, `/chartjs/surface`, `/deckgl` `surfaceLayer`, `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop`.

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy`. Bins are point arrays with `x`, `y`; cells carry `attempts` `makes` `fgPct`.

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius | side, x, y }` (`/bins`)
`binner(o)`	`.bins()`, `.cell()`; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }`; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"`, `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape`, `scale`, `prior`, `dropOutside`; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table`; `hover`, `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"`; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field`: `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")`; tables through `.rowKey("team")`.

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity`, `background`. `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch" | "skip"`.

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0
sdvtables 0.3.0 · peer sdvplot ≥ 0.2
GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` | `"none"`) `fonts` `hidden` `interactive` `sort` . Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy` . Observable: `` html`${renderHTML(...)}` `` .

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable` . Conditions are data, `{ key, op, value }` with op `==` `!=` `>` `>=` `<` `<=` `in` `notIn` `isNull` `notNull` `matches` , so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally` . `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

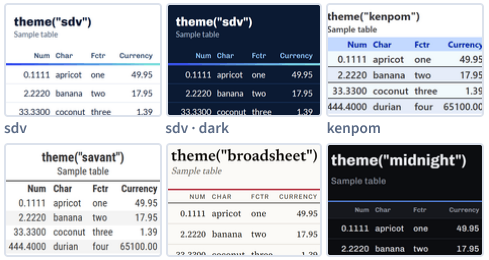
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn` ; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT` .

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league` , `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover` , immutable `getSnapshot()` , `subscribe` . `useTable(spec, rows)` returns `{ table, snapshot }` ; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background` ; `socialCrop` adds `aspect` , `gravity` . `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol` , `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium` .

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }` . Not ported: `gt_sdv_cols_label` . `PAL_MIDNIGHT` differs from R's `pal_midnight` . R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()` .

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: **Introduction ↗**

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync` , `logoUrlSync` , `teamColorsSync` . `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: **Identity and resolution ↗**

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors` , `contrast` , `onColor` .



Guide: **Team colours ↗**

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot` , hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig` : a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette` ; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts` : `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet` : passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background` : 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: **Chart libraries ↗** · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams` , `resolve` , `marks`
`geom_sdv_logos` · `add_logos`	`logos` , `appendLogos` , `withLogos`
`scale_color_sdv`	`teamColor` , `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: **Migrating ↗**

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()` . League CDNs: `nba` , `wnba` , `mlb` , `nhl` `HeadshotUrl` .

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default` , `dark` , `scoreboard` , `grayscale` , `on_light` , …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: **Logos, wordmarks, headshots ↗**

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES` .

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError` , `OfflineError` , `OptionalDependencyError` , …); `setWarningHandler` , `resetManifestCache` , `versions` .

Guides: **Node and SSR ↗** · **Export to PNG ↗**

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: **Observable Plot ↗**

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole PlotOptions; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: **Observable Plot ↗**

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; season per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: **D3 ↗**

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: **React ↗**

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: **Chart.js ↗** · **react-chartjs-2 ↗**

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: **Chart.js ↗**

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: **Plotly, Vega-Lite, ECharts ↗**

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins` , `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0

sporty 0.3.0 · sportyR 2.4.5 parity

GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution` . Or `surface(sport, league, o)` . Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges` ; `SPORTS` ; the `*_SPECS` tables from `/specs` .

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)` ; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: **Playing surfaces ↗**

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

FRAMES	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy` , `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)` ; options `x` , `y` , `out` . Units: `convertUnits` , `convertPoints` . Zones: `basketballZones(league, { scale, top })` , `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }` ; also `season` , `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface` , `/chartjs/surface` , `/deckgl` `surfaceLayer` , `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop` .

Guide: **Playing surfaces ↗**

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy` . Bins are point arrays with `x` , `y` ; cells carry `attempts` `makes` `fgPct` .

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius` | `side, x, y }` (`/bins`)
`binner(o)`	`.bins()` , `.cell()` ; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }` ; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"` , `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape` , `scale` , `prior` , `dropOutside` ; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: **Shot charts ↗** · **Shot dashboard ↗**

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table` ; `hover` , `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"` ; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field` : `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")` ; tables through `.rowKey("team")` .

Example: **Linked figures and tables ↗**

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity` , `background` . `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch"` \| `"skip"` .

Guide: **Export to PNG ↗**

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0
sdvtables 0.3.0 · peer sdvplot ≥ 0.2
GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` | `"none"`) `fonts` `hidden` `interactive` `sort` . Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy` . Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable` . Conditions are data, `{ key, op, value }` with op `==` `!=` `>` `>=` `<` `<=` `in` `notIn` `isNull` `notNull` `matches` , so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally` . `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

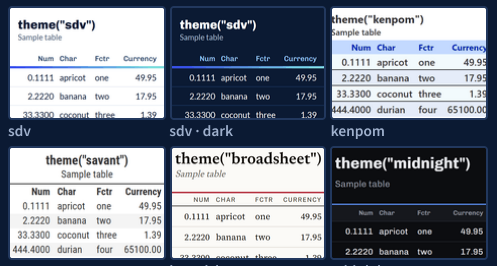
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn` ; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT` .

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league` , `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover` , immutable `getSnapshot()` , `subscribe` . `useTable(spec, rows)` returns `{ table, snapshot }` ; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background` ; `socialCrop` adds `aspect` , `gravity` . `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol` , `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium` .

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }` . Not ported: `gt_sdv_cols_label` . `PAL_MIDNIGHT` differs from R's `pal_midnight` . R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()` .

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: Introduction ↗

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: Identity and resolution ↗

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors`, `contrast`, `onColor`.



Guide: Team colours ↗

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot`, hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig`: a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette`; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts`: `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet`: passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background`: 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: Chart libraries ↗ · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams`, `resolve`, `marks`
`geom_sdv_logos` · `add_logos`	`logos`, `appendLogos`, `withLogos`
`scale_color_sdv`	`teamColor`, `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: Migrating ↗

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()`. League CDNs: `nba`, `wnba`, `mlb`, `nhl` `HeadshotUrl`.

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default`, `dark`, `scoreboard`, `grayscale`, `on_light`, …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: Logos, wordmarks, headshots ↗

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES`.

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError`, `OfflineError`, `OptionalDependencyError`, …); `setWarningHandler`, `resetManifestCache`, `versions`.

Guides: Node and SSR ↗ · Export to PNG ↗

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole `PlotOptions`; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; `season` per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins`, `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0

sporty 0.3.0 · sportyR 2.4.5 parity

GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution`. Or `surface(sport, league, o)`. Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges`; `SPORTS`; the `*_SPECS` tables from `/specs`.

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)`; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

`FRAMES`	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy`, `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)`; options `x`, `y`, `out`. Units: `convertUnits`, `convertPoints`. Zones: `basketballZones(league, { scale, top })`, `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }`; also `season`, `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface`, `/chartjs/surface`, `/deckgl` `surfaceLayer`, `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop`.

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy`. Bins are point arrays with `x`, `y`; cells carry `attempts` `makes` `fgPct`.

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius | side, x, y }` (`/bins`)
`binner(o)`	`.bins()`, `.cell()`; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }`; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"`, `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape`, `scale`, `prior`, `dropOutside`; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table`; `hover`, `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"`; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field`: `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")`; tables through `.rowKey("team")`.

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity`, `background`. `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch" | "skip"`.

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0
sdvtables 0.3.0 · peer sdvplot ≥ 0.2
GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` | `"none"`) `fonts` `hidden` `interactive` `sort`. Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy`. Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable`. Conditions are data, `{ key, op, value }` with op `== != > >= < <= in notIn isNull notNull matches`, so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally`. `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

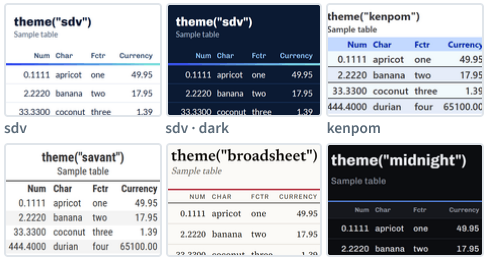
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn`; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT`.

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league`, `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover`, immutable `getSnapshot()`, `subscribe`. `useTable(spec, rows)` returns `{ table, snapshot }`; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background`; `socialCrop` adds `aspect`, `gravity`. `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol`, `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium`.

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }`. Not ported: `gt_sdv_cols_label`. `PAL_MIDNIGHT` differs from R's `pal_midnight`. R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()`.

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: Introduction ↗

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync` , `logoUrlSync` , `teamColorsSync` . `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: Identity and resolution ↗

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors` , `contrast` , `onColor` .



Guide: Team colours ↗

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot` , hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig` : a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette` ; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts` : `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet` : passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background` : 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: Chart libraries ↗ · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams` , `resolve` , `marks`
`geom_sdv_logos` · `add_logos`	`logos` , `appendLogos` , `withLogos`
`scale_color_sdv`	`teamColor` , `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: Migrating ↗

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()` . League CDNs: `nba` , `wnba` , `mlb` , `nhl` `HeadshotUrl` .

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default` , `dark` , `scoreboard` , `grayscale` , `on_light` , …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: Logos, wordmarks, headshots ↗

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES` .

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError` , `OfflineError` , `OptionalDependencyError` , …); `setWarningHandler` , `resetManifestCache` , `versions` .

Guides: Node and SSR ↗ · Export to PNG ↗

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole PlotOptions; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; season per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

`@sportsdataverse/sporty` builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. `sdvplot` paints those surfaces in a team's colours, bins and compares shots (`/bins` , `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

`sdvplot-js v0.3.0`
sporty 0.3.0 · sportyR 2.4.5 parity
GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution` . Or `surface(sport, league, o)` . Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges` ; `SPORTS` ; the `*_SPECS` tables from `/specs` .

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)` ; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

`FRAMES`	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy` , `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)` ; options `x` , `y` , `out` . Units: `convertUnits` , `convertPoints` . Zones: `basketballZones(league, { scale, top })` , `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }` ; also `season` , `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface` , `/chartjs/surface` , `/deckgl` `surfaceLayer` , `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop` .

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy` . Bins are point arrays with `x` , `y` ; cells carry `attempts` `makes` `fgPct` .

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius` | `side, x, y }` (`/bins`)
`binner(o)`	`.bins()` , `.cell()` ; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }` ; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"` , `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape` , `scale` , `prior` , `dropOutside` ; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table` ; `hover` , `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"` ; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field` : `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")` ; tables through `.rowKey("team")` .

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity` , `background` . `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch"` \| `"skip"` .

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0
sdvtables 0.3.0 · peer sdvplot ≥ 0.2
GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` \| `"none"`) `fonts` `hidden` `interactive` `sort` . Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy` . Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable` . Conditions are data, `{ key, op, value }` with op `==` `!=` `>` `>=` `<` `<=` `in` `notIn` `isNull` `notNull` `matches` , so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally` . `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

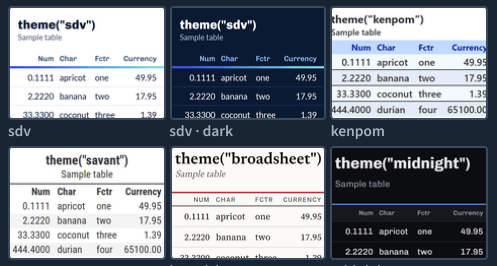
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn` ; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT` .

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league` , `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover` , immutable `getSnapshot()` , `subscribe` . `useTable(spec, rows)` returns `{ table, snapshot }` ; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background` ; `socialCrop` adds `aspect` , `gravity` . `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol` , `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium` .

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }` . Not ported: `gt_sdv_cols_label` . `PAL_MIDNIGHT` differs from R's `pal_midnight` . R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()` .

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: **Introduction ↗**

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: **Identity and resolution ↗**

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors`, `contrast`, `onColor`.



Guide: **Team colours ↗**

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot`, hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig`: a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette`; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts`: `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet`: passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background`: 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: **Chart libraries ↗** · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams`, `resolve`, `marks`
`geom_sdv_logos` · `add_logos`	`logos`, `appendLogos`, `withLogos`
`scale_color_sdv`	`teamColor`, `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: **Migrating ↗**

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()`. League CDNs: `nba`, `wnba`, `mlb`, `nhl` `HeadshotUrl`.

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default`, `dark`, `scoreboard`, `grayscale`, `on_light`, …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: **Logos, wordmarks, headshots ↗**

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES`.

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError`, `OfflineError`, `OptionalDependencyError`, …); `setWarningHandler`, `resetManifestCache`, `versions`.

Guides: **Node and SSR ↗** · **Export to PNG ↗**

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: **Observable Plot ↗**

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole `PlotOptions`; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: **Observable Plot ↗**

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; `season` per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: **D3 ↗**

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: **React ↗**

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: **Chart.js ↗** · **react-chartjs-2 ↗**

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: **Chart.js ↗**

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: **Plotly, Vega-Lite, ECharts ↗**

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins`, `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0

sporty 0.3.0 · sportyR 2.4.5 parity

GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution`. Or `surface(sport, league, o)`. Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges`; `SPORTS`; the `*_SPECS` tables from `/specs`.

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)`; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

`FRAMES`	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy`, `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)`; options `x`, `y`, `out`. Units: `convertUnits`, `convertPoints`. Zones: `basketballZones(league, { scale, top })`, `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }`; also `season`, `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface`, `/chartjs/surface`, `/deckgl` `surfaceLayer`, `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop`.

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy`. Bins are point arrays with `x`, `y`; cells carry `attempts` `makes` `fgPct`.

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius | side, x, y }` (`/bins`)
`binner(o)`	`.bins()`, `.cell()`; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }`; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"`, `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape`, `scale`, `prior`, `dropOutside`; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table`; `hover`, `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"`; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field`: `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")`; tables through `.rowKey("team")`.

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity`, `background`. `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch" | "skip"`.

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0

sdvtables 0.3.0 · peer sdvplot ≥ 0.2

GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` | `"none"`) `fonts` `hidden` `interactive` `sort`. Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy`. Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable`. Conditions are data, `{ key, op, value }` with op `== != > >= < <= in notIn isNull notNull matches`, so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally`. `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

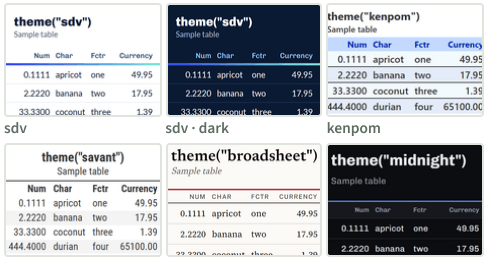
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn`; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT`.

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league`, `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover`, immutable `getSnapshot()`, `subscribe`. `useTable(spec, rows)` returns `{ table, snapshot }`; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background`; `socialCrop` adds `aspect`, `gravity`. `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol`, `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium`.

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }`. Not ported: `gt_sdv_cols_label`. `PAL_MIDNIGHT` differs from R's `pal_midnight`. R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()`.

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: **Introduction ↗**

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync` , `logoUrlSync` , `teamColorsSync` . `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: **Identity and resolution ↗**

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors` , `contrast` , `onColor` .



Guide: **Team colours ↗**

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot` , hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig` : a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette` ; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts` : `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet` : passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background` : 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: **Chart libraries ↗** · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams` , `resolve` , `marks`
`geom_sdv_logos` · `add_logos`	`logos` , `appendLogos` , `withLogos`
`scale_color_sdv`	`teamColor` , `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: **Migrating ↗**

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()` . League CDNs: `nba` , `wnba` , `mlb` , `nhl` `HeadshotUrl` .

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default` , `dark` , `scoreboard` , `grayscale` , `on_light` , …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: **Logos, wordmarks, headshots ↗**

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES` .

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError` , `OfflineError` , `OptionalDependencyError` , …); `setWarningHandler` , `resetManifestCache` , `versions` .

Guides: **Node and SSR ↗** · **Export to PNG ↗**

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole PlotOptions; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; season per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins` , `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0
sporty 0.3.0 · sportyR 2.4.5 parity
GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution` . Or `surface(sport, league, o)` . Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges` ; `SPORTS` ; the `*_SPECS` tables from `/specs` .

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)` ; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

FRAMES	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy` , `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)` ; options `x` , `y` , `out` . Units: `convertUnits` , `convertPoints` . Zones: `basketballZones(league, { scale, top })` , `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }` ; also `season` , `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface` , `/chartjs/surface` , `/deckgl` `surfaceLayer` , `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop` .

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy` . Bins are point arrays with `x` , `y` ; cells carry `attempts` `makes` `fgPct` .

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius` | `side, x, y }` (`/bins`)
`binner(o)`	`.bins()` , `.cell()` ; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }` ; `rule "sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette "rdbu"` | `"master"` , `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape` , `scale` , `prior` , `dropOutside` ; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table` ; `hover` , `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"` ; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field` : `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")` ; tables through `.rowKey("team")` .

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity` , `background` . `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch"` \| `"skip"` .

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0
sdvtables 0.3.0 · peer sdvplot ≥ 0.2
GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` \| `"none"`) `fonts` `hidden` `interactive` `sort`. Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy`. Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable`. Conditions are data, `{ key, op, value }` with op `== != > >= < <= in notIn isNull notNull matches`, so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally`. `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

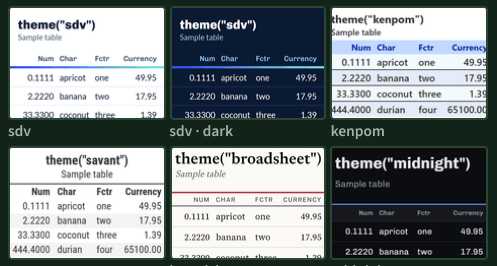
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn`; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT`.

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league`, `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover`, immutable `getSnapshot()`, `subscribe`. `useTable(spec, rows)` returns `{ table, snapshot }`; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background`; `socialCrop` adds `aspect`, `gravity`. `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol`, `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium`.

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }`. Not ported: `gt_sdv_cols_label`. `PAL_MIDNIGHT` differs from R's `pal_midnight`. R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()`.

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: Introduction ↗

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: Identity and resolution ↗

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors`, `contrast`, `onColor`.



Guide: Team colours ↗

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot`, hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig`: a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette`; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts`: `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet`: passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background`: 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: Chart libraries ↗ · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams`, `resolve`, `marks`
`geom_sdv_logos` · `add_logos`	`logos`, `appendLogos`, `withLogos`
`scale_color_sdv`	`teamColor`, `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: Migrating ↗

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()`. League CDNs: `nba`, `wnba`, `mlb`, `nhl` `HeadshotUrl`.

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default`, `dark`, `scoreboard`, `grayscale`, `on_light`, …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: Logos, wordmarks, headshots ↗

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES`.

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError`, `OfflineError`, `OptionalDependencyError`, …); `setWarningHandler`, `resetManifestCache`, `versions`.

Guides: Node and SSR ↗ · Export to PNG ↗

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; `anchor` `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole `PlotOptions`; `subtitle` `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; `season` per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; `clip: "circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins`, `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0
sporty 0.3.0 · sportyR 2.4.5 parity
GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution`. Or `surface(sport, league, o)`. Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges`; `SPORTS`; the `*_SPECS` tables from `/specs`.

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)`; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

`FRAMES`	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy`, `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)`; options `x`, `y`, `out`. Units: `convertUnits`, `convertPoints`. Zones: `basketballZones(league, { scale, top })`, `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }`; also `season`, `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface`, `/chartjs/surface`, `/deckgl` `surfaceLayer`, `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop`.

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy`. Bins are point arrays with `x`, `y`; cells carry `attempts` `makes` `fgPct`.

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius | side, x, y }` (`/bins`)
`binner(o)`	`.bins()`, `.cell()`; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }`; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"`, `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape`, `scale`, `prior`, `dropOutside`; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table`; `hover`, `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"`; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field`: `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")`; tables through `.rowKey("team")`.

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity`, `background`. `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch" | "skip"`.

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0

sdvtables 0.3.0 · peer sdvplot ≥ 0.2

GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` \| `"none"`) `fonts` `hidden` `interactive` `sort` . Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy` . Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable` . Conditions are data, `{ key, op, value }` with op `== != > >= < <= in notIn isNull notNull matches` , so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally` . `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

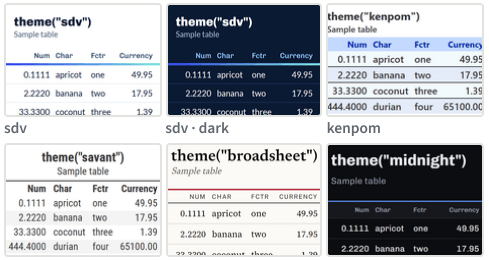
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn` ; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT` .

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league` , `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover` , immutable `getSnapshot()` , `subscribe` . `useTable(spec, rows)` returns `{ table, snapshot }` ; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background` ; `socialCrop` adds `aspect` , `gravity` . `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol` , `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium` .

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }` . Not ported: `gt_sdv_cols_label` . `PAL_MIDNIGHT` differs from R's `pal_midnight` . R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()` .

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets

sdvplot-js :: CHEAT SHEET

Team identity, colours, logos, wordmarks and headshots for JavaScript and TypeScript, across **28 leagues**, from three npm packages: **@sportsdataverse/sdvplot** (marks and team colours for **23 libraries**, from **Observable Plot**, **D3** and **React** to **Recharts**, **MUI X**, **deck.gl**, **MapLibre**, **Three.js** and **Perspective**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables and data grids). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and the library map. Page 2: the core chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.3.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗ GUIDES ↗

IDENTITY · 23 LIBRARIES

INSTALL & IMPORT

```
npm i @sportsdataverse/sdvplot
npm i @sportsdataverse/sporty \
      @observablehq/plot   # for /plot
import { resolve, palette, logoUrl }
  from "@sportsdataverse/sdvplot";
```

Each library is a subpath; the library itself is an optional peer you add. Twelve import none (`/plotly` `/vega` `/echarts` `/uplot` `/victory` `/g2` `/tanstack` `/lightweight-charts` `/maplibre` `/unovis` `/cytoscape` `/perspective`). `/plot` `/d3` `/shots` `/chartjs/surface` `/deckgl` `/three` also need sporty; `/export` needs resvg.

This sheet describes 0.3.0 (main); npm has 0.2.0.

Guide: Introduction ↗

TEAM IDENTITY

```
await resolve(
  ["LV", "OAK", "Las Vegas Raiders"], "nfl");
// ["13", "13", "13"]
await logoUrl("OAK", "nfl", { season: 2010 });
await suggest("Golden St", "nba");
```

Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync` , `logoUrlSync` , `teamColorsSync` . `teams(league)` lists ids, names, conferences and colours; ids resolve over 18 systems (`PRIORITY` + `EXPLICIT_ONLY`).

Guide: Identity and resolution ↗

TEAM COLOURS

```
await palette("nfl", ["LV", "KC"]);
// { LV: "#000000", KC: "#e31837" }
const { light, dark } = await matchupColors(
  "Alabama", "Georgia", { league: "cfb" });
// light: ["#9e1b32", "#2c2a29"]
```

`matchupColors` picks a contrast-safe pair for two teams on light and dark backgrounds. Also `teamColors` , `contrast` , `onColor` .



Guide: Team colours ↗

LIBRARY MAP

REACT & COMPONENT LIBRARIES

Subpath	Exports
`/react`	`TeamLogo` `Wordmark` `Headshot` , hooks; `LogoBump` (nivo), `LogoPoint` `LogoTick` (Victory), `SurfaceBackdrop` (Recharts), `Surface3D` `Trajectory` (R3F)
`/recharts`	`logoDot` `headshotDot` `logoTick` `teamColorPalette`
`/shadcn`	`teamChartConfig` : a `ChartConfig` with logo icons
`/mui`	MUI X Charts: `teamColors` `teamColorGetter` `logoMarker` `headshotMarker` `logoMark` `logoTickLabel`
`/nivo`	`teamColors` `logoLayer` `logoNode`
`/victory`	`teamTheme` `teamColorPalette`
`/svelte`	LayerChart: `LogoMark` `HeadshotMark` `SurfaceBackdrop`
`/tanstack`	TanStack Charts: `teamColorScale` `logoDots` `wordmarkDots` `headshotDots` `withPatterns`

CHART, SPEC & CANVAS LIBRARIES

Subpath	Exports
`/plot`	`logos` `wordmarks` `headshots` `axisLogos` `teamColor` `teamTiers` `surface` (p. 2-3)
`/d3`	`appendLogos` `appendHeadshots` `appendSurface` `teamColorScale` (p. 2)
`/chartjs`	`logoPoints` `axisLogos` `teamColorPalette` ; works through react-chartjs-2 (p. 2)
`/plotly`	and `/vega` `/echarts` : `withLogos` `withHeadshots` `withAxisLogos` (p. 2)
`/uplot`	`withTeamSeries` `logoPlugin`
`/g2`	G2 5 and Ant Design Charts: `teamColorScale` `logoMark` `wordmarkMark` `headshotMark` `axisLogos`
`/lightweight-charts`	`teamSeriesOptions` `logoMarkers` `withAttribution`

MAPS, NETWORKS, 3D, GRIDS

Subpath	Exports
`/deckgl`	`logoIconLayer` `passArcLayer` `surfaceLayer` (layer props)
`/maplibre`	`logoSymbols` `addLogoImages` `routeLines` `basemap` (keyless)
`/cytoscape`	`passNetwork` `teamStylesheet` : passing networks
`/unovis`	`teamColorScale` `logoIcons` `headshotIcons`
`/three`	`surfaceMesh` `trajectoryTube` `statcastPitchPath` (mesh buffers)
`/perspective`	`perspectiveConfig` `perspectiveLogoColumn`

```
import { logoDot }
  from "@sportsdataverse/sdvplot/recharts";
await loadLeague("nfl");
<Line data={rows} dataKey="wins"
  dot={logoDot({ team: "team",
    league: "nfl", last: true })} />
```

The adapters return plain options, props or specs and never mutate yours. Their `teamColorPalette` / `teamColorScale` take a `background` : 3:1 contrast for dark mode, teams kept distinct (CIE76 ΔE 20). Recipes without a subpath: Mosaic, Superset.

Guide: Chart libraries ↗ · each subpath links its guide

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
`league` · `season` · `which` · `variant`	same names
`id_system`	`idSystem`
`sdv_teams` `sdv_resolve` `sdv_marks` · `teams` `resolve` `marks`	`teams` , `resolve` , `marks`
`geom_sdv_logos` · `add_logos`	`logos` , `appendLogos` , `withLogos`
`scale_color_sdv`	`teamColor` , `teamColorScale`
`sdv_surface` · `surface`	`surface` (sdvplot), sporty builders
`table_theme_spec`	`tableThemeSpec` (sdvtables)

Guide: Migrating ↗

HEADSHOTS

```
headshotUrl("3139477", "nfl")
headshotUrl(gsis, "nfl", { idSystem: "gsis" })
nbaHeadshotUrl(2544)  // league CDN
```

ESPN ids by default; `"gsis"` needs `await loadGsis()` . League CDNs: `nba` , `wnba` , `mlb` , `nhl` `HeadshotUrl` .

SEASONS & VARIANTS

```
await logoUrl("LV", "nfl",
  { variant: "dark", markType: "wordmark" });
```

`season` picks the mark in use that year (season-dated for 14 leagues). `VARIANTS` holds 32 names (`default` , `dark` , `scoreboard` , `grayscale` , `on_light` , …); `marks()` lists a team's archive, `selectMark()` picks one.

Guide: Logos, wordmarks, headshots ↗

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 ncaa_mhockey ncaa_whockey	286 / 414 / 415 / 110 / 48
nbagl ufl usfl xfl aaf	32 / 11 / 9 / 11 / 8

The full list is `LEAGUES` .

NODE, EXPORT & ERRORS

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
await toPNG(socialCard(svg.outerHTML,
  { aspect: "16:9" }), { width: 1600 });
```

Plot / D3 render server-side with jsdom. Errors subclass `SdvplotError` (`UnresolvedTeamError` , `OfflineError` , `OptionalDependencyError` , …); `setWarningHandler` , `resetManifestCache` , `versions` .

Guides: Node and SSR ↗ · Export to PNG ↗

sdvplot-js :: CHART LIBRARIES

Each chart library is one subpath of **@sportsdataverse/sdvplot** with its library as an optional peer. **/plot** and **/d3** return marks and selections; **/plotly**, **/vega** and **/echarts** patch a plain figure, spec or option and never import the library; **/chartjs** returns dataset options and plugins. All are synchronous: `await loadLeague(league)` first. Image `height` is a fraction of the plot height in (0, 1], default 0.1.

sdvplot-js v0.3.0

the core 7 of 23 libraries · 1 surface plugin

GALLERY ↗ GUIDES ↗

/PLOT · OBSERVABLE PLOT

IMAGE MARKS

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(rows, { x: "wins",
    fx: "division" }),
  logos(rows, { league: "nfl", x: "wins",
    y: "pf", team: "team", fx: "division",
    height: 0.15, tip: true,
    href: (d) => `/teams/${d.team}` }),
] });
```

`logos` `wordmarks` take `team` (a channel), `league`, `season` (a value or a channel), `variant`, `idSystem`, `id`, `height`, `alpha`, `r`. Every Plot image option passes through: `tip`, `href`, `target`, `title`, `fx` / `fy` facets.

```
headshots(rows, { league: "nfl", x: "pf",
  y: "pa", player: "qb_espn_id" })
```

`player` is an ESPN id, or a GSIS id with `idSystem: "gsis"`.

```
logos(rows, { league: "nfl", x: "season",
  y: "wins", team: "team",
  season: "season" })  // era logos
```

AXES, COLOUR & RULES

```
Plot.plot({
  color: teamColor("nfl", { values:
    rows.map((r) => r.team) }),
  marks: [
    Plot.barX(rows, { x: "pf", y: "team",
      fill: "team" }),
    axisLogos("y", { league: "nfl",
      height: 0.08 })] });
```

Export	Key options
`axisLogos(axis, o)`	`x` `y` `fx` `fy`; anchor `tickSize` `label` `markType` `season` `variant`
`teamColor` · `teamFill`	Plot scale options: `values` `legend` `which` `alpha` `naValue` `strict`
`meanLines` `medianLines`	an array of rules: `x` `y` `stroke` `strokeDasharray` `strokeWidth`

Guide: Observable Plot ↗

TIERS, TITLES & MORE

```
const tiers = [{ team: "KC", tier_no: 1 },
  { team: "DET", tier_no: 1 },
  { team: "BAL", tier_no: 2 }];
Plot.plot(teamTiers(tiers, { league: "nfl",
  theme: "dark", presort: true }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

Export	Does
`teamTiers(rows, o)`	whole PlotOptions; subtitle `caption` `noLineBelowTier` `tierDesc` `tip`
`titleImage(fig, o)`	logo beside the title; `side`, `height` in px (default 24)
`linkIds(rows, "team")`	pass as `render`: tags each mark `data-sdv-id` for `/interact`

Guide: Observable Plot ↗

/D3 · D3

APPEND TO A SELECTION

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const x = d3.scaleLinear([280, 540],
  [40, 620]);
const y = d3.scaleLinear([280, 450],
  [370, 20]);
appendLogos(svg, rows.map((r) => r.pf),
  rows.map((r) => r.pa),
  rows.map((r) => r.team),
  { league: "nfl", frameHeight: 400,
    x: (v) => x(+v), y: (v) => y(+v),
    height: 0.12 });
```

Export	Notes
`appendLogos` `appendWordmarks`	`(sel, xs, ys, teams, o)`; season per point, `variant`, `alpha`, `ariaLabel`
`appendHeadshots`	`(sel, xs, ys, players, o)`; clip: `"circle"`, `ring`, `placeholder`
`appendSurface(sel, league, o)`	team court: `x` `y` `team` `displayRange`
`appendLegend` · `appendSignature`	shot legend · shooting signature
`teamColorScale(league, o)`	a `d3.scaleOrdinal`: `values` `which` `naValue`

Guide: D3 ↗

/REACT · REACT

COMPONENTS & HOOKS

```
import { TeamLogo, Headshot, useTeamColors,
  useResolve, useSelection }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="LV" league="nfl" size={32}
  season={2010} variant="dark" />
<Headshot playerId="3139477" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const { hover, selected } =
  useSelection(store);
```

`<Wordmark />` takes the `TeamLogo` props less `markType`. The hooks return `undefined` until the league loads; `useSelection` follows a `createSelection()` store. Also `<LogoBump>` (nivo), `<LogoPoint>` `<LogoTick>` (Victory), `<SurfaceBackdrop>` (Recharts court), `<Surface3D>` `<Trajectory>` (react-three-fiber, React 19).

Guide: React ↗

/CHARTJS · CHART.JS

POINTS, AXES, COLOURS

```
import { logoPoints, pointImages }
  from "@sportsdataverse/sdvplot/chartjs";
const pts = logoPoints(rows.map((r) =>
  r.team), { league: "nfl", radius: 20 });
new Chart(canvas, { type: "scatter",
  data: { datasets: [{ ...pts,
    data: rows.map((r) =>
      ({ x: r.pf, y: r.pa })) }] },
  plugins: [pointImages] });
```

Export	Returns · options
`logoPoints` `wordmarkPoints`	`{ pointStyle, pointRadius }`; `radius` `background` `fallback` `season` `variant`
`headshotPoints`	same, from ESPN player ids
`pointImages`	plugin: redraws as each image loads
`axisLogos("x", o)`	plugin: category labels as logos; `size` `scaleId` `markType`
`logoWatermarks(teams, o)`	plugin: `alpha` `inset` `size`
`teamColor` · `teamFill`	`(teams, league, o)`; `teamFill` is translucent
`teamColorPalette`	`(league, teams, o)`; `background` for dark mode

Node: pass `loadImage` to the image helpers.

Guides: Chart.js ↗ · react-chartjs-2 ↗

/CHARTJS/SURFACE

```
import { surface } from
  "@sportsdataverse/sdvplot/chartjs/surface";
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
// in the Chart config:
plugins: [court.plugin],
options: { scales: {
  x: { ...court.scales.x, display: false },
  y: { ...court.scales.y, display: false } } }
```

Returns `{ plugin, scales, scene }`; the scales keep a foot the same length on both axes. Also `xAxisID`, `yAxisID`.

Guide: Chart.js ↗

/PLOTLY · /VEGA · /ECHARTS

PATCH A FIGURE OR SPEC

```
import { withAxisLogos, teamColorway }
  from "@sportsdataverse/sdvplot/plotly";
const fig = { data: [{ type: "bar",
  x: teams, y: wins, marker: { color:
    teamColorway("nfl", teams) } }] };
const f = withAxisLogos(fig, "x",
  { league: "nfl", height: 0.1 });
Plotly.newPlot(div, f.data, f.layout);
```

```
import { withLogos, embedSources }
  from "@sportsdataverse/sdvplot/echarts";
const embed = await embedSources(urls);
chart.setOption(withLogos(option, rows, {
  league: "nfl", x: "pf", y: "pa",
  team: "team", height: 0.12, embed }));
```

Export	Plotly · Vega-Lite · ECharts
`withLogos` `withWordmarks`	rows at `x`, `y` fields, `team` field
`withHeadshots`	`player` field; `idSystem` `"espn"` | `"gsis"`
`withAxisLogos`	logos under an axis's ticks
team colours	`teamColorway` · `teamColorScale` · `teamColorPalette`
extras	— · `logoLayer` · `renderLogo`
`embedSources(urls)`	URL → data URI map for `embed`
`drawnMarks` `drawnAxisMarks` `visibleAxisLabels`	test hooks: placed images, text labels

Plotly `layer` `xref` `yref`; ECharts `xAxisIndex` `z`.

Guide: Plotly, Vega-Lite, ECharts ↗

sdvplot-js :: SURFACES, SHOTS & LINKING

@sportsdataverse/sporty builds courts, rinks and fields for ten sports as a framework-free `Scene` and draws it with SVG, canvas, Observable Plot or D3. **sdvplot** paints those surfaces in a team's colours, bins and compares shots (`/bins` , `/shots`), links charts and tables through one selection store (`/interact`) and saves SVG figures as PNG (`/export`).

sdvplot-js v0.3.0

sporty 0.3.0 · sportyR 2.4.5 parity

GALLERY ↗ GUIDES ↗

SPORTY · PLAYING SURFACES

BUILD A SCENE

```
import { basketballCourt, hockeyRink }
  from "@sportsdataverse/sporty";
import { toSVG }
  from "@sportsdataverse/sporty/svg";
const court = basketballCourt("nba",
  { displayRange: "offense" });
const rink = hockeyRink("nhl", {
  rotation: 90,
  colorUpdates: { boards: "#111111" } });
const svg = toSVG(court, { width: 600 });
```

Builder	Leagues (+ `custom`)
`basketballCourt`	nba wnba "nba g league" ncaa fiba nfhs
`hockeyRink`	nhl pwhl ahl echl ohl qmjhl ushl iihf ncaa nwhl phf
`footballField`	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
`soccerPitch`	fifa epl mls nwsl ncaa
`baseballField`	mlb milb ncaa nfhs pony "little league"
`softballField`	ncaa
`tennisCourt`	atp wta itf ita usta ncaa
`volleyballCourt`	fivb ncaa "usa volleyball"
`curlingSheet`	wcf "curling canada"
`lacrosseField`	pll nll ncaam ncaaw usam usaw "world lacrosse"

Options: `displayRange` `rotation` `xlim` / `ylim` `xTrans` / `yTrans` `units` (ft m yd in cm mm) `colorUpdates` (by colour key) `updates` (spec params) `arcResolution` . Or `surface(sport, league, o)` . Look-ups by sport: `leagues` `features` `colorKeys` `displayRanges` ; `SPORTS` ; the `*_SPECS` tables from `/specs` .

RENDER TARGETS

Subpath	Call
`/svg`	`toSVG(scene, { width, height, background })` → string
`/canvas`	`drawScene(ctx, scene, { scale })`
`/plot`	`surfaceMark(scene)` + `surfaceScales(scene)` ; `sceneToGeoJSON`
`/d3`	`appendSurface(sel, scene, x, y)`

Guide: Playing surfaces ↗

COORDINATE FRAMES

```
import { toSurfaceFrame }
  from "@sportsdataverse/sporty";
const pts = toSurfaceFrame(shots,
  { from: "nba-legacy" });
// adds surface_x, surface_y
```

FRAMES	From
`nba-legacy`	stats.nba.com: tenths of a foot, hoop origin (`x_legacy` , `y_legacy`)
`nba-legacy-vertical`	same, hoop at the bottom, for a `rotation: 90` scene
`hockeytech`	600×300 canvas → 200×85 ft (approximate)
`espn-football-0-100`	0-100 yardline → −50..50 yards

Own frame: `{ description, x, y }` or `frameBottomLeft(length, width)` ; options `x` , `y` , `out` . Units: `convertUnits` , `convertPoints` . Zones: `basketballZones(league, { scale, top })` , `basketballZoneOf(x, y, shotValue)` over `BASKETBALL_ZONES` (restricted_area paint mid_range corner_3_left corner_3_right above_break_3).

TEAM-COLOURED SURFACES

```
import { surface }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nba");
const court = surface("nba", { team: "DEN",
  displayRange: "defense" });
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  Plot.dot(pts, { x: "surface_x",
    y: "surface_y" })] });
```

Returns `{ marks, scales, scene }` ; also `season` , `centerLogo` (centre court or ice; shallow centre field on a diamond), `colorUpdates` and every sporty option. `SURFACES` has 59 keys over the ten sports: SDV leagues are painted in team colours, lacrosse, tennis, volleyball and curling drawn plain. `colorUpdates(sport, primary, secondary)` is the team painting; `SURFACE_BASE` the base colours. Same surfaces from `/d3` `appendSurface` , `/chartjs/surface` , `/deckgl` `surfaceLayer` , `/three` `surfaceMesh` and the React / Svelte `SurfaceBackdrop` .

Guide: Playing surfaces ↗

SDVPLOT · SHOT CHARTS

SHOT ROWS

`/shots` reads stats.nba.com-style rows (`ShotRow`): `shot_distance` `shot_result` `shot_value` `x_legacy` `y_legacy` . Bins are point arrays with `x` , `y` ; cells carry `attempts` `makes` `fgPct` .

/BINS & /SHOTS

Function	Does
`hexbin` · `squarebin`	bin any points: `{ radius` | `side, x, y }` (`/bins`)
`binner(o)`	`.bins()` , `.cell()` ; `hexagonPath` `squarePath` `cellPath`
`binShots(shots, cell)`	attempts, makes, FG%, zone per cell
`leagueIndex(shots, cell)`	the league's cells and zones
`cellsVsLeague` · `cellsVsDistance`	a player against the league, by cell or distance
`sizeCells(cells, o)`	`{ r, steps, cap, size }` ; `rule` `"sqrt-p95"` | `"linear-cap"`
`diffScale(o)`	FG% diff colours: `palette` `"rdbu"` | `"master"` , `theme`
`fgPctByDistance` · `vsLeague` · `signaturePoints`	the shooting-signature pipeline
`statsByZone` · `statsBySide`	splits by zone · left/centre/right
`shrunkDiff(makes, att, lg, k)`	shrunk FG% diff; `LEAGUE_PRIOR_ATTEMPTS` = 25

HEXAGONS VS THE LEAGUE

```
const court = surface("nba",
  { displayRange: "defense", rotation: 90 });
const index = leagueIndex(leagueShots, 15);
const cells = cellsVsLeague(playerShots,
                            index);
Plot.plot({ ...court.scales, marks: [
  ...court.marks,
  shotCells(cells, { tip: true,
    frame: "nba-legacy-vertical",
    r: sizeCells(cells, index).r })] });
```

`shotCells` also takes `shape` , `scale` , `prior` , `dropOutside` ; `shotZones(areas, { fill, text, stats, frame })` fills the named zones.

SHOOTING SIGNATURE

```
const pts = signaturePoints(vsLeague(
  fgPctByDistance(playerShots),
  fgPctByDistance(leagueShots)));
Plot.plot({ marks: shootingSignature(pts,
  { tip: true }) });
```

Guide: Shot charts ↗ · Shot dashboard ↗

LINKING & EXPORT

LINKED INTERACTIVITY

```
import { createSelection }
  from "@sportsdataverse/sdvplot";
import { brushFilter, linkSelection }
  from "@sportsdataverse/sdvplot/interact";
const store = createSelection();
linkSelection(store, { figure: fig,
  select: "toggle" });
linkSelection(store, { table });
brushFilter(fig, store, { data: rows,
  x: "wins", y: "net_epa", id: "team" });
```

Function	Does
`store.set({ hover, selected, predicate, cursor })`	also `getState` `subscribe` `clear`
`linkSelection(store, o)`	wires a `figure` and/or `table` ; `hover` , `select`
`brushFilter(fig, store, o)`	brush to select; `empty` `"dim"` | `"clear"` ; `.move()`
`nearestHover(root, store, o)`	light the nearest point: `points` `radius` `label`
`linkCursor(root, store, o)`	a shared cursor on `field` : `shape` `snap` `label`
`tooltip(root, o)`	`.show(x, y, lines, swatch)` `.hide()`
`highlight(root, ids)`	dims the rest; returns ids not found

Plot marks join the store through `render: linkIds(rows, "team")` ; tables through `.rowKey("team")` .

Example: Linked figures and tables ↗

SAVE AS PNG

```
import { toPNG, socialCard }
  from "@sportsdataverse/sdvplot/export";
const png = await toPNG(socialCard(svg,
  { aspect: "16:9", padding: 24 }),
  { width: 1600 });
```

`socialCard(svg, o)` pads and extends the short side, never crops: `aspect` (`"1:1"` `"16:9"` `"4:5"` `"9:16"` `"1.91:1"` or a number), `gravity` , `background` . `toPNG` (peer `@resvg/resvg-js`): `width` `scale` `background` `fonts` `images: "fetch"` \| `"skip"` .

Guide: Export to PNG ↗

sdvplot-js :: TABLES

@sportsdataverse/sdvtables is a builder for themed tables with team logos, wordmarks, headshots and colours. One serializable spec (data predicates, no functions) renders to HTML in Node, SSR or the browser (`/html`), to React (`/react`), to an interactive sort / filter / page engine, to MUI X Data Grid, AG Grid or TanStack Table, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and reactablefmtr's cell formatters, with 170 themes.

sdvplot-js v0.3.0

sdvtables 0.3.0 · peer sdvplot ≥ 0.2

GALLERY ↗ GUIDES ↗

SDVTABLES · DEFINE & RENDER

DEFINE & RENDER

```
import { defineTable }
  from "@sportsdataverse/sdvtables";
import { prepare, renderHTML }
  from "@sportsdataverse/sdvtables/html";
const spec = defineTable()
  .columns((c) => [
    c.logo("team", { league: "nfl",
                     includeName: true }),
    c.int("wins"), c.int("losses")])
  .theme("midnight")
  .title("AFC West")
  .subtitle("2024 regular season")
  .build();
await prepare(spec);  // loads leagues
const html = renderHTML(spec, rows);
```

Function	Does
`prepare(spec)`	loads every league the spec names (and gsis ids)
`renderHTML(spec, rows, o)`	sync HTML string; or `renderHTML(table)`
`renderHTMLAsync`	the same, without a separate `prepare`
`toElement(spec, rows)`	an `HTMLElement` (browser)
`styleSheet(spec)` · `fontsLinkFor(spec)`	host-page CSS with `css: "none"` · the fonts link
`renderParts` · `leaguesOf`	the pieces of the markup · leagues named

Render options: `css` (`"inline"` \| `"none"`) `fonts` `hidden` `interactive` `sort` . Builder basics: `.title` `.subtitle` `.sourceNote(html, { unsafe })` `.id` `.rowKey` `.groupBy` . Observable: `` html`${renderHTML(...)}` ``.

Guides: Tables ↗ · HTML and SSR ↗

COLUMNS IN USE

```
c.colorPills("net_epa", { digits: 3,
  domain: [-0.2, 0.2] }),
c.delta("pa", "pf", { decimals: 0,
  arrows: true }),
c.tally(["wins", "losses", "ties"],
  { label: "Record" }),
c.highlight("wins", { key: "wins",
  op: ">=", value: 11 }, { bold: true }),
c.headshot("qb_espn_id", { league: "nfl" })
```

Every kind takes `label` `subheader` `align` `width` `sortable` `filterable` . Conditions are data, `{ key, op, value }` with op `==` `!=` `>` `>=` `<` `<=` `in` `notIn` `isNull` `notNull` `matches` , so a spec serializes.

COLUMN KINDS (`c.*`)

```
c.text(key), c.int(key),
c.num(key, { digits, prefix, suffix,
  big, forceSign }),
c.pct(key, { digits, scale }),
c.rank(key, { superscript }),
c.delta(from, to, { percent, arrows }),
c.tally(keys, { separator, share }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, height }),
c.headshot(key, { league, idSystem }),
c.colorPills(key, { palette, domain,
  fillType, digits }),
c.colorRanks(key, { palette, domain }),
c.colorResults(key, { resultType }),
c.percentileBar(key, { domain, palette }),
c.indicatorBox(key, { truthy, fill }),
c.highlight(key, when, { fill, bold }),
c.highlightNa(key, { missingText }),
c.mergeStackTeamColor(key, stack, team,
  { league }),
c.teamColorBar(key, { league, which }),
c.teamColorBg(key, { league, alpha }),
c.image(key, { height: "40px", alt }),
// reactablefmtr
c.dataBar(key, { fill, gradient, posNeg,
  textPosition }),
c.iconSet(key, { icons, colors }),
c.iconAssign(key, { icon, buckets }),
c.mergeStack(key, stack, { position }),
c.bubble(key, { shape, maxSize }),
c.sparkline(series, { statline,
  bandline, showArea }),
c.sparkbar(series, { fillColor }),
c.gauge(key, { fill, showMinMax }),
```

29 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally` . `sparkline` / `sparkbar` read an array column; the reactablefmtr kinds draw inline SVG, no script.

Guides: Cell kinds ↗ · Formatters ↗

SPEC HELPERS

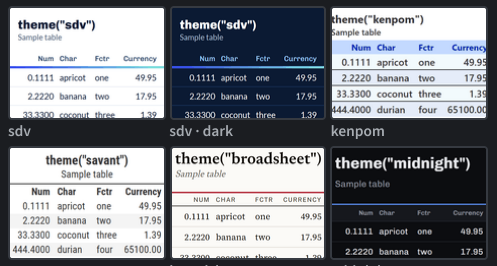
Rendering throws `TableSpecError` for a column the rows lack or an unknown theme. The engine's pure parts are exported: `applySort` `applyFilters` `paginate` `nextSortDir` `withMissingLast` `isMissing` `matches` `selectRows` `findColumn` ; colour helpers `secondaryOn` `RANK_PALETTE` `PAL_MIDNIGHT` .

Guide: Sort, filter, page ↗

DATA GRIDS

Each `(spec, rows, o)` with labels, formats, sort and image cells carried over: `/mui` `dataGridProps` (MUI X Data Grid), `/aggrid` `gridOptions` (AG Grid Community), `/tanstack-table` `tanstackTable` (TanStack Table v9).

THEMES



sdv · sdv · dark · kenpom · savant · broadsheet · midnight

```
.theme("sdvTeam", { density: "compact",
  options: { league: "nfl", team: "KC" } })
```

Theme	Options
`sdv` · `sdvTeam`	`style` light | dark · `league` , `team`
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte
27 `fmtr-*`	reactablefmtr: `fmtr-espn` `fmtr-nytimes` …; `fmtr-team`
122 registry	`datagolf` `nejm` `economist` …: `listTableThemes()`

`THEME_NAMES` lists 48 (170 with the registry); `tableThemeSpec(name, mode)` resolves one, `themePreview` shows several.

Guides: Table themes ↗ · Registry ↗

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.cutline(after, { label, style, color })
.rowAccent(key, { palette, width, side })
.boldRows(rows, { highlightColor })
.spotlight(rows, { fill, accentColor })
.outliers(cols, { method, symbol, note })
.legendContinuous({ title, digits })
.legendDiscrete(key, { heading })
.groupBy(key).groupStripes({ color })
.borderGrid({ color, includeLabels })
.borderBars(side, colors, { text })
.scaleNote(cols, { divisor }).snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width }).font(family)
.legendIcon({ iconSet, labels, title })
```

Guide: Decorations ↗

INTERACTIVE & REACT

```
const table = createTable(spec, rows,
  { pageSize: 10 });
host.innerHTML = renderHTML(table);
hydrate(host.querySelector(".sdvt"), table);
// React: sdvtables/react
<SdvTable spec={spec} rows={rows}
          interactive pageSize={10} />
```

A headless engine: `setSort` `setFilter` `setGlobalFilter` `setPage` `setPageSize` `toggleColumn` `setSelection` `setExternalFilter` `setHover` , immutable `getSnapshot()` , `subscribe` . `useTable(spec, rows)` returns `{ table, snapshot }` ; `.rowKey("team")` links rows to charts.

Guide: HTML, SSR and the host page ↗

SAVE AS AN IMAGE

```
import { tableToPNG, socialCrop }
  from "@sportsdataverse/sdvtables/export";
await tableToPNG(spec, rows,
  { file: "afc.png" });
await socialCrop(spec, rows,
  { aspect: "4:5", file: "post.png" });
```

PNG options: `file` `width` `deviceScaleFactor` `whitespace` `background` ; `socialCrop` adds `aspect` , `gravity` . `batchToPNG(rows, group, build, pattern, { dir })` writes one per group; `gridTables` (`ncol` , `labels`) and `stackTables` compose HTML; `htmlToPNG` shoots any HTML. Node only; peer `playwright` + `playwright install chromium` .

Guide: Export to PNG ↗

FROM R / PYTHON

`gt_*` helpers map to `c.*` kinds and builder methods: `GT_ALIASES` (116 names) / `aliasFor("gt_cutline")` gives `{ target, status }` . Not ported: `gt_sdv_cols_label` . `PAL_MIDNIGHT` differs from R's `pal_midnight` . R's `sdv_table_themes()` and `table_theme_spec()` are `listTableThemes()` and `tableThemeSpec()` .

Guide: Migrating ↗

LEARN MORE

GALLERY ↗ every chart and table in one place

Start: Introduction · notebooks · sample data

API: sdvplot · sporty · sdvtables

Examples: basketball · football · hockey

Coming from: sdvplot, sdvplotR, sportyR · Game on Paper · blazing-the-nets