



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 for **Observable Plot**, **D3**, **React**, **Chart.js**, **Plotly**, **Vega-Lite**, **ECharts**), [@sportsdataverse/sporty](#) (playing surfaces) and [@sportsdataverse/sdvtables](#) (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗

GUIDES ↗

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 }) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly](#), [Vega-Lite](#), [ECharts](#), [Chart.js](#) ↗

FROM SDVPLOTR / SDVPLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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
mlb 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 v0.1.0

7 chart subpaths · 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_esp_n_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
<code>axisLogos</code> (axis, o)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; anchor <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	Plot scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js ↗](#)

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> field
<code>withHeadshots</code>	<code>player</code> field; <code>idSystem</code> "esp_n" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (urls)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbb wbb · nhl whl pwhl ahl echl ohl qmjhl ushl phf ncaa_mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces](#) ↗

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthy, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21 kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell Kinds](#) ↗

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](#) ↗

THEMES

theme("sdv") Sample table	theme("sdv") Sample table	theme("kenpom") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				

theme("savant") Sample table	theme("broadsheet") Sample table	theme("midnight") Sample table																																																
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	<table><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95
Num	Char	Fctr	Currency																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
33.3300	coconut	three	1.39																																															
444.4000	durian	four	65100.00																																															
NUM	CHAR	FCTR	CURRENCY																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
33.3300	coconut	three	1.39																																															
Num	Char	Fctr	Currency																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
savant	broadsheet	midnight																																																

```
sdv
sdv · dark
kenpom
```

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tuft

Density: comfortable compact social (DENSITY). THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes](#) ↗

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: [{ estimate, p }].

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot**, **D3**, **React**, **Chart.js**, **Plotly**, **Vega-Lite**, **ECharts**), [@sportsdataverse/sporty](#) (playing surfaces) and [@sportsdataverse/sdvtables](#) (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0
ESM only · Node ≥ 20.18.1 · browsers

[GALLERY ↗](#) [GUIDES ↗](#)

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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 ↗](#)

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot ↗](#)

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3 ↗](#)

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React ↗](#)

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 }) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly, Vega-Lite, ECharts, Chart.js ↗](#)

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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 mbf wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
mlb ncaa_baseball ncaa_softball ncaa_mhockey ncaa_whokey	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 v0.1.0

7 chart subpaths · 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 perpoint, 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 resolve = useResolve("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.

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

Node: pass loadImage to the image helpers.

Guide: [Chart.js ↗](#)

/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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

[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 qmjl ushl iihf ncaa nwhl phf
footballField	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
soccerPitch	fifa epl mls nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbb wbb · nhl whl pwhl ahl echl ohl qmjl ushl phf ncaa_mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

[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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthful, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21.kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell kinds ↗](#)

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 ↗](#)

THEMES

theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("kenpom") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				
theme("savant") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	theme("broadsheet") Sample table <table><thead><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("midnight") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95				
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
NUM	CHAR	FCTR	CURRENCY																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
savant	broadsheet	midnight																																																				
.theme("sdvTeam", { density: "compact", options: { league: "nfl", team: "KC" } })																																																						
Theme	Options																																																					
sdv	style: "light" "dark"																																																					
sdvTeam	league, team: header in the team's colour																																																					
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte																																																					

Density: comfortable compact social (DENSITY). THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes ↗](#)

DECORATIONS (BUILDER)

```
.titleHeader(title, { subtitle, kicker,
  date })
.caption538({ top, bottom })
.outline(after, { label, style, color })
.rowAccent(key, { palette, width, size })
.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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: { [estimate, p] }.

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_outline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot, D3, React, Chart.js, Plotly, Vega-Lite, ECharts**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗

GUIDES ↗

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 } ) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly, Vega-Lite, ECharts, Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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
mlb 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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (<code>axis</code> , <code>o</code>)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; anchor <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	Plot scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

Guide: [React ↗](#)

/CHARTJS · CHARTJS

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js ↗](#)

/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
<code>withLogos</code> <code>withWordmarks</code>	rows at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> <code>field</code> ; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (<code>urls</code>)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

[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 nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbb wbb · nhl whl pwhl ahl echl ohl qmjhl ushl phf ncaa_mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (`/export`). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

[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
<code>prepare(spec)</code>	loads every league the spec names (and gsis ids)
<code>renderHTML(spec, rows, o)</code>	sync HTML string; or <code>renderHTML(table)</code>
<code>renderHTMLAsync</code>	the same, without a separate prepare
<code>toElement(spec, rows)</code>	an <code>HTMLElement</code> (browser)
<code>styleSheet(spec)</code> · <code>fontsLinkFor(spec)</code>	host-page CSS with <code>css: "none"</code> · the fonts link
<code>renderParts</code> · <code>leaguesOf</code>	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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthy, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21 kinds; `fmtRank` and `fmtTally` alias `rank` and `tally`. `fillType` `"continuous" | "rank"`; `resultType` `"wl" | "binary"`; an image's `alt` names a column.

Guide: [Cell kinds ↗](#)

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 ↗](#)

THEMES

theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("kenpom") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv - dark	kenpom																																																				
theme("savant") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	theme("broadsheet") Sample table <table><thead><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("midnight") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95				
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
NUM	CHAR	FCTR	CURRENCY																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
savant	broadsheet	midnight																																																				

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

Theme	Options
sdv	<code>style: "light" "dark"</code>
sdvTeam	<code>league</code> , <code>team</code> : header in the team's colour
18 gtUtils	<code>almanac</code> <code>athletic</code> <code>booktabs</code> <code>broadsheet</code> <code>brutalist</code> <code>drench</code> <code>gtutils</code> <code>kenpom</code> <code>midnight</code> <code>ncaa</code> <code>pl</code> <code>savant</code> <code>scoreboard</code> <code>sofa</code> <code>swiss</code> <code>terminal</code> <code>tier</code> <code>tufte</code>

Density: `comfortable` `compact` `social` (`DENSITY`).

`THEME_NAMES` lists all 20; `themePreview(spec, rows, names)` renders one table per theme.

Guide: [Table themes ↗](#)

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

`rows`: indices or a predicate; method: `iqr` `sd` `bounds`; `pairs`: `[{ estimate, p }]`.

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`. Keys: `jkhl`, `s` sorts, `/` searches, `Enter` selects. `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` (74 names) / `aliasFor("gt_cutline")` gives `{ target, status }`. Not ported: `gt_sdv_cols_label`. `PAL_MIDNIGHT` differs from R's `pal_midnight`.

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 for JavaScript and TypeScript, across **28 leagues**, from three npm packages: [@sportsdataverse/sdvplot](#) (marks for **Observable Plot**, **D3**, **React**, **Chart.js**, **Plotly**, **Vega-Lite**, **ECharts**), [@sportsdataverse/sporty](#) (playing surfaces) and [@sportsdataverse/sdvtables](#) (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗

GUIDES ↗

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 } ) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly](#), [Vega-Lite](#), [ECharts](#), [Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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 mbf wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
mlb ncaa_baseball ncaa_softball ncaa_mhockey ncaa_whokey	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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (<code>axis</code> , <code>o</code>)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; <code>anchor</code> <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	<code>Plot</code> scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

Guide: [React ↗](#)

/CHARTJS · CHARTJS

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js ↗](#)

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> <code>field</code> ; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (<code>urls</code>)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 qmjl ushl iihf ncaa nwhl phf
footballField	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
soccerPitch	fifa epl mls nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbf wbb · nhl whl pwhl ahl echl ohl qmjl ushl phf ncaa mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthful, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21.kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell kinds ↗](#)

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 ↗](#)

THEMES

theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("sdv") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("kenpom") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				

theme("savant") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	theme("broadsheet") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	theme("midnight") Sample table <table><thead><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr></thead><tbody><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></tbody></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95
Num	Char	Fctr	Currency																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
33.3300	coconut	three	1.39																																															
444.4000	durian	four	65100.00																																															
Num	Char	Fctr	Currency																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
33.3300	coconut	three	1.39																																															
Num	Char	Fctr	Currency																																															
0.1111	apricot	one	49.95																																															
2.2220	banana	two	17.95																																															
savant	broadsheet	midnight																																																

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

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte

Density: comfortable compact social (DENSITY).

THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes ↗](#)

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: [{ estimate, p }].

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot, D3, React, Chart.js, Plotly, Vega-Lite, ECharts**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗

GUIDES ↗

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 })] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly, Vega-Lite, ECharts, Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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
mlb 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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (<code>axis</code> , <code>o</code>)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; anchor <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	Plot scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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="313947" league="nfl"
  fallback="initials" />
const colors = useTeamColors("nfl");
const resolve = useResolve("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.

Guide: [React](#) ↗

/CHARTJS · CHARTJS

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js](#) ↗

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> field; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (<code>urls</code>)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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`, `colorUpdates` and every `sporty` option. `SURFACES` maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbb wbb · nhl whl pwhl ahl echl ohl qmjhl ushl phf ncaa_mhockey ncaa_whoockey. `colorUpdates`(`sport`, `primary`, `secondary`) is the team painting; `SURFACE_BASE` the floor colours. Same courts from `/d3` `appendSurface` and `/chartjs/surface`.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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: { <code>radius</code> <code>side</code> , <code>x</code> , <code>y</code> } (/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)	{ <code>r</code> , <code>steps</code> , <code>cap</code> , <code>size</code> }; <code>rule</code> "sqrt-p95" "linear-cap"
diffScale(o)	FG% diff colours: <code>palette</code> "rdbu" "master", <code>theme</code>
fgPctByDistance · vsLeague · signaturePoints	the shooting-signature pipeline
statsByZone · statsBySide	splits by zone · left/centre/right
shrunkDiff(makes, att, lg, k)	shrunk FG% diff; <code>LEAGUE_PRIOR_ATTEMPTS</code> = 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({ <code>hover</code> , <code>selected</code> , <code>predicate</code> , <code>cursor</code> })	also <code>getState</code> <code>subscribe</code> <code>clear</code>
linkSelection(store, o)	wires a figure and/or table; <code>hover</code> , <code>select</code>
brushFilter(fig, store, o)	brush to select; empty "dim" "clear"; <code>.move()</code>
nearestHover(root, store, o)	light the nearest point: <code>points</code> <code>radius</code> <code>label</code>
linkCursor(root, store, o)	a shared cursor on field: <code>shape</code> <code>snap</code> <code>label</code>
tooltip(root, o)	.show(<code>x</code> , <code>y</code> , <code>lines</code> , <code>swatch</code>) <code>.hide()</code>
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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthy, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21 kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell Kinds](#) ↗

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](#) ↗

THEMES

theme("sdv") Sample table	theme("sdv") Sample table	theme("kenpom") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				
theme("savant") Sample table	theme("broadsheet") Sample table	theme("midnight") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	<table><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95				
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
NUM	CHAR	FCTR	CURRENCY																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
savant	broadsheet	midnight																																																				

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

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tuft

Density: comfortable compact social (DENSITY).

THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes](#) ↗

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: [{ estimate, p }].

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot**, **D3**, **React**, **Chart.js**, **Plotly**, **Vega-Lite**, **ECharts**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

sdvplot-js v0.1.0

ESM only · Node ≥ 20.18.1 · browsers

GALLERY ↗

GUIDES ↗

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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");
```

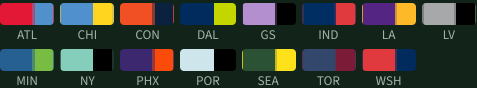
Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 }) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly](#), [Vega-Lite](#), [ECharts](#), [Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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 mbf wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
mlb ncaa_baseball ncaa_softball ncaa_mhockey ncaa_whokey	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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (axis, o)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; anchor <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	<code>Plot</code> scale options: <code>values</code> <code>legend</code> which <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js ↗](#)

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> <code>field</code> ; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (urls)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 qmjl ushl iihf ncaa nwhl phf
footballField	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
soccerPitch	fifa epl mls nwsf ncaa
baseballField	mlb milb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbf wbb · nhl whl pwhl ahl echl ohl qmjl ushl phf ncaa mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthful, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21.kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell kinds ↗](#)

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 ↗](#)

THEMES

theme("sdv") Sample table	theme("sdv") Sample table	theme("kenpom") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				
theme("savant") Sample table	theme("broadsheet") Sample table	theme("midnight") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	<table><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95				
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
NUM	CHAR	FCTR	CURRENCY																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
savant	broadsheet	midnight																																																				

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

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte

Density: comfortable compact social (DENSITY). THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes ↗](#)

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: { [estimate, p] }.

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot, D3, React, Chart.js, Plotly, Vega-Lite, ECharts**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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 the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `d3.scaleColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 }) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly, Vega-Lite, ECharts, Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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 mbf wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
mlb ncaa_baseball ncaa_softball ncaa_mhockey ncaa_whokey	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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (<code>axis</code> , <code>o</code>)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; anchor <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	Plot scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

Guide: [React](#) ↗

/CHARTJS · CHARTJS

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js](#) ↗

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> field; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (<code>urls</code>)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 nwsf ncaa
baseballField	mlb mlb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbb wbb · nhl whl pwhl ahl echl ohl qmjhl ushl phf ncaa_mhockey ncaa_whoockey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthy, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21 kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell Kinds](#) ↗

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](#) ↗

THEMES

theme("sdv") Sample table	theme("sdv") Sample table	theme("kenpom") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
sdv	sdv · dark	kenpom																																																				
theme("savant") Sample table	theme("broadsheet") Sample table	theme("midnight") Sample table																																																				
<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr><tr><td>444.4000</td><td>durian</td><td>four</td><td>65100.00</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	444.4000	durian	four	65100.00	<table><tr><th>NUM</th><th>CHAR</th><th>FCTR</th><th>CURRENCY</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr><tr><td>33.3300</td><td>coconut</td><td>three</td><td>1.39</td></tr></table>	NUM	CHAR	FCTR	CURRENCY	0.1111	apricot	one	49.95	2.2220	banana	two	17.95	33.3300	coconut	three	1.39	<table><tr><th>Num</th><th>Char</th><th>Fctr</th><th>Currency</th></tr><tr><td>0.1111</td><td>apricot</td><td>one</td><td>49.95</td></tr><tr><td>2.2220</td><td>banana</td><td>two</td><td>17.95</td></tr></table>	Num	Char	Fctr	Currency	0.1111	apricot	one	49.95	2.2220	banana	two	17.95				
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
444.4000	durian	four	65100.00																																																			
NUM	CHAR	FCTR	CURRENCY																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
33.3300	coconut	three	1.39																																																			
Num	Char	Fctr	Currency																																																			
0.1111	apricot	one	49.95																																																			
2.2220	banana	two	17.95																																																			
savant	broadsheet	midnight																																																				

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

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tuft

Density: comfortable compact social (DENSITY).

THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes](#) ↗

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: [{ estimate, p }].

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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 for **Observable Plot**, **D3**, **React**, **Chart.js**, **Plotly**, **Vega-Lite**, **ECharts**), **@sportsdataverse/sporty** (playing surfaces) and **@sportsdataverse/sdvtables** (tables). The TypeScript port of sdvplot and sdvplotR. Page 1: identity and marks. Page 2: chart libraries in depth. Page 3: surfaces, shots, interactivity, export. Page 4: tables.

MARKS · SEVEN CHART 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 with an optional peer: `/plot` (Plot + sporty), `/d3` (d3 + sporty), `/react`, `/chartjs`, `/plotly`, `/vega`, `/echarts` (no peer), `/export` (resvg).

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");
```

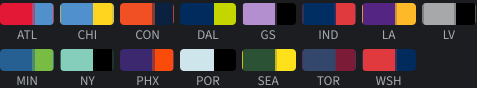
Calls are async and load a league's data on first use. After `await loadLeague("nba")` (or `preloadAll()`) the `*Sync` twins work: `resolveSync`, `logoUrlSync`, `teamColorsSync`. `teams(league)` lists ids, names, conferences and colours; ids resolve over the same 19 systems as Python (PRIORITY).

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](#) ↗

OBSERVABLE PLOT

```
import * as Plot from "@observablehq/plot";
import { logos, meanLines }
  from "@sportsdataverse/sdvplot/plot";
await loadLeague("nfl");
Plot.plot({ marks: [
  ...meanLines(afc, { x: "wins", y: "pf" }),
  logos(afc, { league: "nfl", x: "wins",
    y: "pf", team: "team", height: 0.15 }),
] });
```

Mark	Does
logos wordmarks headshots	image marks; tip, href, facets
axisLogos("x", o)	logos as tick labels
teamColor / teamFill	colour scale options
meanLines medianLines	reference rules
titleImage · teamTiers	logo title · tier plot

Guide: [Observable Plot](#) ↗

D3

```
import { appendLogos, teamColorScale }
  from "@sportsdataverse/sdvplot/d3";
const svg = d3.select(el).append("svg");
appendLogos(svg, xs, ys, teams, { league:
  "nba", x, y, frameHeight: 400 });
```

`x`, `y` are your scales; `height` is a share of `frameHeight`. Also `appendWordmarks`, `appendHeadshots`, `appendSurface`, `appendLegend`, `appendSignature`; `teamColorScale(league, { values })` is a `d3.scaleOrdinal`.

Guide: [D3](#) ↗

TIERS & TITLES (PLOT)

```
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" }));
titleImage(fig, { image: "KC",
  league: "nfl", title: "Chiefs" });
```

`teamTiers` also takes `presort`, `noLineBelowTier`, `subtitle`, `caption`.

REACT

```
import { TeamLogo, Headshot, useTeamColors }
  from "@sportsdataverse/sdvplot/react";
<TeamLogo team="KC" league="nfl" size={32} />
<Headshot playerId="3139477" league="nfl" />
const colors = useTeamColors("nfl");
```

Also `<Wordmark />`, `useResolve`, `useSelection`; props `season`, `variant`, `idSystem`, `fallback="initials"`.

Guide: [React](#) ↗

CHART.JS

```
import { axisLogos, teamColor }
  from "@sportsdataverse/sdvplot/chartjs";
new Chart(canvas, { type: "bar",
  data: { labels: teams, datasets: [{ data,
    backgroundColor: teamColor(teams, "nfl")
  } ] },
  plugins: [axisLogos("x",
    { league: "nfl", size: 28 }) ] });
```

Points: `logoPoints` `wordmarkPoints` `headshotPoints` (+ the `pointImages` plugin); `logoWatermarks`, `teamFill`; a court plugin from `/chartjs/surface`.

PLOTLY · VEGA-LITE · ECHARTS

```
import { withLogos }
  from "@sportsdataverse/sdvplot/plotly";
const fig = withLogos(base, rows, { x: "pf",
  y: "pa", team: "team", league: "nfl" });
```

Same verbs from `/plotly`, `/vega`, `/echarts`: `withLogos` `withWordmarks` `withHeadshots` `withAxisLogos` return a new figure / spec / option. Colours: `teamColorway`, `teamColorScale`, `teamColorPalette`.

Guide: [Plotly](#), [Vega-Lite](#), [ECharts](#), [Chart.js](#) ↗

FROM SDVPLOTR / SDV PLOT

R · Python	JavaScript
league · season · which · variant	same names
id_system	idSystem
geom_sdv_logos · add_logos	logos, appendLogos, withLogos
scale_color_sdv	teamColor, teamColorScale
sdv_surface · surface	surface (sdvplot), sporty builders

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 mbf wbb	687 / 366 / 364
soccer · cricket	2,631 · 111
ahl echl ohl qmjhl whl ushl pwhl phf	hockey minors & juniors
mlb ncaa_baseball ncaa_softball ncaa_mhockey ncaa_whokey	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 v0.1.0

7 chart subpaths · 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
<code>axisLogos</code> (<code>axis</code> , <code>o</code>)	<code>x</code> <code>y</code> <code>fx</code> <code>fy</code> ; <code>anchor</code> <code>tickSize</code> <code>label</code> <code>markType</code> <code>season</code> <code>variant</code>
<code>teamColor</code> · <code>teamFill</code>	<code>Plot</code> scale options: <code>values</code> <code>legend</code> <code>which</code> <code>alpha</code> <code>naValue</code> <code>strict</code>
<code>meanLines</code> <code>medianLines</code>	an array of rules: <code>x</code> <code>y</code> <code>stroke</code> <code>strokeDasharray</code> <code>strokeWidth</code>

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

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

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 resolve = useResolve("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.

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

Node: pass `loadImage` to the image helpers.

Guide: [Chart.js ↗](#)

/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
<code>withLogos</code> <code>withWordmarks</code>	<code>rows</code> at <code>x</code> , <code>y</code> fields, <code>team</code> <code>field</code>
<code>withHeadshots</code>	<code>player</code> <code>field</code> ; <code>idSystem</code> "espn" "gsis"
<code>withAxisLogos</code>	logos under an axis's ticks
<code>team colours</code>	<code>teamColorway</code> · <code>teamColorScale</code> · <code>teamColorPalette</code>
<code>extras</code>	— · <code>logoLayer</code> · <code>renderLogo</code>
<code>embedSources</code> (<code>urls</code>)	URL → data URI map for <code>embed</code>
<code>drawnMarks</code> <code>drawnAxisMarks</code> <code>visibleAxisLabels</code>	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 nine 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 v0.1.0

sporty 0.1.0 · no runtime dependencies

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 qmjl ushl iihf ncaa nwhl phf
footballField	nfl ncaa cfl nfhs11 nfhs9 nfhs8 nfhs6
soccerPitch	fifa epl mls nwsf ncaa
baseballField	mlb milb ncaa nfhs pony "little league"
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, colorUpdates and every sporty option. SURFACES maps the 22 leagues: nfl aaf ufl usfl xfl cfb · nba nbagl wnba mbf wbb · nhl whl pwhl ahl echl ohl qmjl ushl phf ncaa mhockey ncaa_whokey. colorUpdates(sport, primary, secondary) is the team painting; SURFACE_BASE the floor colours. Same courts from /d3 appendSurface and /chartjs/surface.

Guide: [Playing surfaces ↗](#)

SDVPLLOT · 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, or to PNG (/export). It ports sdvplotR's gt toolkit (gtUtils by Andrew Weatherman) and its 20 themes.

sdvplot-js v0.1.0

sdvtables 0.1.0 · peer sdvplot

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, suffixSize }),
c.delta(from, to, { percent, decimals,
  arrows, color, colorPositive,
  colorNegative, forceSign }),
c.tally(keys, { separator, share,
  shareOf, shareDecimals, shareLabel }),
c.logo(key, { league, season, variant,
  idSystem, height, includeName }),
c.wordmark(key, { league, season,
  variant, height }),
c.headshot(key, { league, idSystem,
  height }),
c.colorPills(key, { palette, domain,
  fillType, rankOrder, digits,
  formatType, reverse, pillHeight }),
c.colorRanks(key, { palette, domain,
  reverse }),
c.colorResults(key, { winColor,
  lossColor, tieColor, resultType }),
c.percentileBar(key, { domain, scale,
  palette, trackColor, markerSize,
  decimals, naLabel }),
c.indicatorBox(key, { truthy, fill,
  neutral, size, showOnly }),
c.highlight(key, when, { fill,
  textColor, bold }),
c.highlightNa(key, { missingText,
  naStrings, italic, ignoreCase }),
c.mergeStackTeamColor(key, stack, team,
  { league, fontSizeTop, fontSizeBottom }),
c.teamColorBar(key, { league, which,
  barWidth, naColor }),
c.teamColorBg(key, { league, which,
  alpha, naColor }),
c.image(key, { height: "40px", alt }),
```

21.kinds; fmtRank and fmtTally alias rank and tally. fillType "continuous" | "rank"; resultType "wl" | "binary"; an image's alt names a column.

Guide: [Cell kinds ↗](#)

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 ↗](#)

THEMES

theme("sdv")
Sample table

Num	Char	Fctr	Currency
0.1111	apricot	one	49.95
2.2220	banana	two	17.95
33.3300	coconut	three	1.39

theme("sdv")
Sample table

Num	Char	Fctr	Currency
0.1111	apricot	one	49.95
2.2220	banana	two	17.95
33.3300	coconut	three	1.39

theme("kenpom")
Sample table

Num	Char	Fctr	Currency
0.1111	apricot	one	49.95
2.2220	banana	two	17.95
33.3300	coconut	three	1.39
444.4000	durian	four	65100.00

sdv

sdv · dark

kenpom

theme("savant")
Sample table

Num	Char	Fctr	Currency
0.1111	apricot	one	49.95
2.2220	banana	two	17.95
33.3300	coconut	three	1.39
444.4000	durian	four	65100.00

theme("broadsheet")
Sample table

NUM	CHAR	FCTR	CURRENCY
0.1111	apricot	one	49.95
2.2220	banana	two	17.95
33.3300	coconut	three	1.39
444.4000	durian	four	65100.00

theme("midnight")
Sample table

Num	Char	Fctr	Currency
0.1111	apricot	one	49.95
2.2220	banana	two	17.95

savant

broadsheet

midnight

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

Theme	Options
sdv	style: "light" "dark"
sdvTeam	league, team: header in the team's colour
18 gtUtils	almanac athletic booktabs broadsheet brutalist drench gtutils kenpom midnight ncaa pl savant scoreboard sofa swiss terminal tier tufte

Density: comfortable compact social (DENSITY). THEME_NAMES lists all 20; themePreview(spec, rows, names) renders one table per theme.

Guide: [Table themes ↗](#)

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 })
.snake({ nCols })
.tiers(levels, tierKey, imageColumns)
.marginalia(cols, { label, width })
.scaleNote(cols, { divisor })
.significance(pairs, { hideP })
.socialTag(accounts, { caption })
.watermark({ text, angle, opacity })
.wrapLabels({ width })
.font(family, { weight })
```

rows: indices or a predicate; method: iqr sd bounds; pairs: [{ estimate, p }].

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. Keys: j k h l, s sorts, / searches, Enter selects. 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 (74 names) / aliasFor("gt_cutline") gives { target, status }. Not ported: gt_sdv_cols_label. PAL_MIDNIGHT differs from R's pal_midnight.

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](#)