sportypy :: CHEAT SHEET

Draw scaled `matplotlib` representations of sports playing surfaces to rule-book specification, as the base layer for tracking / shot / event data. Eight surface families share one constructor and one `draw()`; data-plotting methods clip to the surface boundary. Part of the SportsDataverse; R original: sportyR.

v1.1.0

numpy · scipy · pandas · matplotlib
sportypy.sportsdataverse.org

GET STARTED

```
pip install sportypy

from sportypy.surfaces.hockey import NHLRink
nhl = NHLRink()
nhl.draw()           # full rink, rule-book spec
nhl.draw(display_range = "ozone")  # zoom a region
```

Every surface is a class; instantiate, then `draw()` returns a matplotlib `Axes` you can keep plotting onto.

SURFACE CLASSES

Module	Base + league classes
`baseball`	`BaseballField` · MLB MiLB NCAA LittleLeague NFHS Pony
`basketball`	`BasketballCourt` · NBA WNBA NCAA FIBA NBAGLeague NFHS
`curling`	`CurlingSheet` · WCF
`football`	`FootballField` · NFL CFL NCAA NFHS
`hockey`	`HockeyRink` · NHL AHL ECHL IIHF NCAA OHL QMJHL USHL PHF NWHL
`soccer`	`SoccerPitch` · FIFA EPL MLS NWSL NCAA
`tennis`	`TennisCourt` · ATP WTA ITF ITA NCAA USTA
`volleyball`	`VolleyballCourt` · FIVB NCAA USAVolleyball

League class = base class pre-loaded with that rule book (`NHLRink()` ≡ `HockeyRink(league_code = "nhl")`).

SHARED CONSTRUCTOR

```
NHLRink(
  rink_updates  = {},   # resize any feature
  color_updates = {},   # recolor any feature
  rotation = 90.0,      # degrees
  x_trans = 0.0,        # shift the origin
  y_trans = 0.0,
  units = "ft")         # override rule-book units
```

The dimension-updates kwarg is named per sport: `field_updates` (baseball, football) · `court_updates` (basketball, tennis, volleyball) · `rink_updates` · `pitch_updates` · `sheet_updates`.

Origin: `(0, 0)` is the center of the surface. Shift with `x_trans` / `y_trans` (e.g. NHL data with origin at a corner: `x_trans = 100, y_trans = 42.5`).

WHAT CAN I...? HELPERS

```
NHLRink().cani_plot_leagues()
NBACourt().cani_color_features()
MLBField().cani_change_dimensions()
```

Print the supported league codes, every `color_updates` key, and every dimension-updates key for that surface.

CUSTOMIZE

```
rink = NHLRink(rotation = 90,
  color_updates = {
    "plot_background": "#ffffff00",
    "boards": "#000000"})
```

DRAW() & DISPLAY_RANGE

```
court = NBACourt()
ax = court.draw()          # matplotlib Axes
court.draw(display_range = "offense")
NFLField().draw(display_range = "red zone")
WCFSheet().draw(display_range = "house")
nhl.draw(display_range = "dzone")
```

Ranges are per sport — `"full"` (default), halves (`"offense"` / `"defense"`), zones (`"ozone"` `"nzone"` `"dzone"`, `"redzone"`, `"house"` …); spaced and underscored spellings both work. `xlim` / `ylim` crop arbitrarily.

PLOT DATA ON THE SURFACE

```
rink = NHLRink(x_trans = 100, y_trans = 42.5)
ax = rink.draw()
rink.scatter(df.x, df.y)         # shot chart
rink.heatmap(df.x, df.y, values = df.xg)
rink.hexbin(df.x, df.y)
rink.contour(df.x, df.y, fill = True)
rink.arrow(df.x1, df.y1, df.x2, df.y2)  # passes
rink.plot(df.x, df.y)            # path / trace
```

Every method clips to the surface boundary by default — pass `is_constrained = False` to disable.
`binned_stat_2d(x, y, values, statistic = "sum")` powers custom 2-D aggregations.

SMALL MULTIPLES

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1, 2)
NHLRink().draw(ax = axs[0])
IIHFRink().draw(ax = axs[1])   # ft vs m, same fig
```

UNITS

Dimensions default to the rule book's own units (NHL feet, IIHF metres). Pass `units = "m"` / `"ft"` to convert the whole surface — data must be in the same units as the plotted surface.

TYPICAL WORKFLOW

- Pull tracking / pbp coordinates (e.g. sportsdataverse-py loaders).
- Match the provider's coordinate frame with `x_trans` / `y_trans` / `rotation`.
- `draw()` the surface, then layer `scatter` / `heatmap` / `contour`.

SPORTSDATAVERSE SIBLINGS

sportyR — the R original (`ggplot2`, `geom_{sport}()`) · sportsdataverse-py — data loaders for the coordinates you plot here · mlbplotR / cfbplotR — logo plotting (R).

sportypy :: CHEAT SHEET

Draw scaled `matplotlib` representations of sports playing surfaces to rule-book specification, as the base layer for tracking / shot / event data. Eight surface families share one constructor and one `draw()`; data-plotting methods clip to the surface boundary. Part of the SportsDataverse; R original: sportyR.

v1.1.0

numpy · scipy · pandas · matplotlib
sportypy.sportsdataverse.org

GET STARTED

```
pip install sportypy

from sportypy.surfaces.hockey import NHLRink
nhl = NHLRink()
nhl.draw()           # full rink, rule-book spec
nhl.draw(display_range = "ozone")  # zoom a region
```

Every surface is a class; instantiate, then `draw()` returns a matplotlib `Axes` you can keep plotting onto.

SURFACE CLASSES

Module	Base + league classes
`baseball`	`BaseballField` · MLB MiLB NCAA LittleLeague NFHS Pony
`basketball`	`BasketballCourt` · NBA WNBA NCAA FIBA NBAGLeague NFHS
`curling`	`CurlingSheet` · WCF
`football`	`FootballField` · NFL CFL NCAA NFHS
`hockey`	`HockeyRink` · NHL AHL ECHL IIHF NCAA OHL QMJHL USHL PHF NWHL
`soccer`	`SoccerPitch` · FIFA EPL MLS NWSL NCAA
`tennis`	`TennisCourt` · ATP WTA ITF ITA NCAA USTA
`volleyball`	`VolleyballCourt` · FIVB NCAA USAVolleyball

League class = base class pre-loaded with that rule book (`NHLRink()` ≡ `HockeyRink(league_code = "nhl")`).

SHARED CONSTRUCTOR

```
NHLRink(
  rink_updates  = {},   # resize any feature
  color_updates = {},   # recolor any feature
  rotation = 90.0,      # degrees
  x_trans = 0.0,        # shift the origin
  y_trans = 0.0,
  units = "ft")         # override rule-book units
```

The dimension-updates kwarg is named per sport: `field_updates` (baseball, football) · `court_updates` (basketball, tennis, volleyball) · `rink_updates` · `pitch_updates` · `sheet_updates`.

Origin: `(0, 0)` is the center of the surface. Shift with `x_trans` / `y_trans` (e.g. NHL data with origin at a corner: `x_trans = 100, y_trans = 42.5`).

WHAT CAN I...? HELPERS

```
NHLRink().cani_plot_leagues()
NBACourt().cani_color_features()
MLBField().cani_change_dimensions()
```

Print the supported league codes, every `color_updates` key, and every dimension-updates key for that surface.

CUSTOMIZE

```
rink = NHLRink(rotation = 90,
  color_updates = {
    "plot_background": "#ffffff00",
    "boards": "#000000"})
```

DRAW() & DISPLAY_RANGE

```
court = NBACourt()
ax = court.draw()          # matplotlib Axes
court.draw(display_range = "offense")
NFLField().draw(display_range = "red zone")
WCFSheet().draw(display_range = "house")
nhl.draw(display_range = "dzone")
```

Ranges are per sport — `"full"` (default), halves (`"offense"` / `"defense"`), zones (`"ozone"` `"nzone"` `"dzone"`, `"redzone"`, `"house"` …); spaced and underscored spellings both work. `xlim` / `ylim` crop arbitrarily.

PLOT DATA ON THE SURFACE

```
rink = NHLRink(x_trans = 100, y_trans = 42.5)
ax = rink.draw()
rink.scatter(df.x, df.y)        # shot chart
rink.heatmap(df.x, df.y, values = df.xg)
rink.hexbin(df.x, df.y)
rink.contour(df.x, df.y, fill = True)
rink.arrow(df.x1, df.y1, df.x2, df.y2)  # passes
rink.plot(df.x, df.y)           # path / trace
```

Every method clips to the surface boundary by default — pass `is_constrained = False` to disable.
`binned_stat_2d(x, y, values, statistic = "sum")` powers custom 2-D aggregations.

SMALL MULTIPLES

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1, 2)
NHLRink().draw(ax = axs[0])
IIHFRink().draw(ax = axs[1])   # ft vs m, same fig
```

UNITS

Dimensions default to the rule book's own units (NHL feet, IIHF metres). Pass `units = "m"` / `"ft"` to convert the whole surface — data must be in the same units as the plotted surface.

TYPICAL WORKFLOW

- Pull tracking / pbp coordinates (e.g. sportsdataverse-py loaders).
- Match the provider's coordinate frame with `x_trans` / `y_trans` / `rotation`.
- `draw()` the surface, then layer `scatter` / `heatmap` / `contour`.

SPORTSDATAVERSE SIBLINGS

sportyR — the R original (`ggplot2`, `geom_{sport}()`) · sportsdataverse-py — data loaders for the coordinates you plot here · mlbplotR / cfbplotR — logo plotting (R).