

Package: vayr (via r-universe)

September 21, 2026

Title Extensions for 'ggplot2' to Visualize as You Randomize

Version 1.1.0

Description Extensions for 'ggplot2' that implement the ``visualize as you randomize" principles of Coppock (2021) [doi:10.1017/9781108777919.022](https://doi.org/10.1017/9781108777919.022), which can be especially useful when plotting experimental data. Provides position adjustments that arrange over-plotted points so that a statistical model can be shown in data-space, and a helper for graphing extreme value bounds when an experiment encounters attrition.

License GPL-2 | file LICENSE

URL <https://alexandercoppock.com/vayr/>

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.0.0), packcircles (>= 0.3.7), withr (>= 2.1.1)

Suggests dplyr, estimatr, knitr, marginalesffects, patchwork, randomizr, rmarkdown, sessioninfo, testthat, tibble, tidyr, tidyverse

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Repository <https://acoppock.r-universe.dev>

Date/Publication 2026-09-21 00:32:57 UTC

RemoteUrl <https://github.com/acoppock/vayr>

RemoteRef HEAD

RemoteSha 800275125317c21c667404e82e27f20eea378b54

Contents

attrition_experiment	2
blocked_experiment	3
clustered_experiment	4
continuous_interaction	5
covariate_adjustment	6
impute_extreme_values	6
noncompliance_experiment	8
patriot_act	9
position_bluenoise	9
position_bluenoisedodge	11
position_circlepack	12
position_circlepackdodge	14
position_honeycomb	15
position_honeycombdodge	16
position_jitter_ellipse	17
position_jitterdodge_ellipse	19
position_sunflower	20
position_sunflowerdodge	21
sunflower	22
two_arm_trial	24
vayr	25
Index	26

attrition_experiment	<i>Simulated experiment encountering attrition</i>
----------------------	--

Description

Nineteen of the 200 subjects have a missing outcome, and missingness is related to the potential outcomes, so dropping those subjects conditions the analysis on a post-treatment variable. Use with [impute_extreme_values\(\)](#), which imputes the best and worst cases the observed data admit. The outcome is a seven-point Likert item, so its logical range is 1 to 7 even though no subject in this sample answered 1.

Usage

attrition_experiment

Format

- attrition_experiment:
- A tibble with 200 rows and 10 columns:
- ID** Subject identifier
 - U** Unobserved subject-level shock

Y_Z_0, Y_Z_1 Likert potential outcomes under control and treatment
R_Z_0, R_Z_1 Potential response indicators, 1 if the subject would report an outcome
Z Random assignment, 1 for treatment and 0 for control
Z_cond_prob Probability of the assigned condition, 0.5 throughout
R Revealed response indicator, 1 when Y is observed
Y Revealed outcome, NA for the 19 subjects who did not report

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [blocked_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [noncompliance_experiment](#), [patriot_act](#), [two_arm_trial](#)

blocked_experiment	<i>Simulated block-randomized experiment</i>
--------------------	--

Description

Residents of two neighborhoods, with 25 residents treated in each. The neighborhoods differ in size, so assignment probabilities differ across blocks: 0.5 in the 50-resident neighborhood and 0.25 in the 100-resident one. The treatment effect also differs by neighborhood.

Usage

```
blocked_experiment
```

Format

blocked_experiment:
 A tibble with 150 rows and 9 columns:
neighborhood Block identifier, 1 or 2
lambda Poisson rate governing the outcome, 10 in neighborhood 1 and 5 in neighborhood 2
resident Subject identifier within neighborhood
Y_Z_0, Y_Z_1 Count potential outcomes under control and treatment
Z Random assignment, 1 for treatment and 0 for control
Z_cond_prob Probability of the assigned condition, which varies by block
Y Revealed outcome
condition Z labelled "Treatment" and "Control"

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [noncompliance_experiment](#), [patriot_act](#), [two_arm_trial](#)

clustered_experiment *Simulated cluster-randomized experiment*

Description

Thirty classes of varying size are assigned to treatment or control as whole clusters, so every student in a class shares its assignment. Outcomes are driven by a class-level shock as well as a student-level one, which is why uncertainty estimates that ignore the clustering are overconfident.

Usage

```
clustered_experiment
```

Format

```
clustered_experiment:
```

A tibble with 441 rows and 11 columns:

class Cluster identifier, 30 classes in all

n_per_class Number of students in the class, between 10 and 20

class_shock Class-level component of the outcome

student Student identifier within class

student_shock Student-level component of the outcome

Y_Z_0, Y_Z_1 Continuous potential outcomes under control and treatment

Z Random assignment, constant within class

Z_cond_prob Probability of the assigned condition, 0.5 throughout

Y Revealed outcome

condition Z labelled "Treatment" and "Control"

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [noncompliance_experiment](#), [patriot_act](#), [two_arm_trial](#)

`continuous_interaction`*Simulated experiment whose effect varies with a continuous covariate*

Description

The treatment effect is a nonlinear function of X , which makes this dataset useful for showing conditional average treatment effects. Subjects were also sampled with probability related to X , so the sample is not a simple random sample of the population that generated it.

Usage

`continuous_interaction`

Format

`continuous_interaction:`

A tibble with 1189 rows and 10 columns:

ID Subject identifier

noise Unobserved subject-level shock

X Continuous pretreatment covariate

Y_Z_1, Y_Z_0 Continuous potential outcomes under treatment and control

S_inclusion_prob Probability the subject was sampled, which increases in X

Z Random assignment, 1 for treatment and 0 for control

Z_cond_prob Probability of the assigned condition, 0.5 throughout

Y Revealed outcome

condition Z labelled "Treatment" and "Control"

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [clustered_experiment](#), [covariate_adjustment](#), [noncompliance_experiment](#), [patriot_act](#), [two_arm_trial](#)

covariate_adjustment *Simulated experiment with a pretreatment covariate*

Description

A completely randomized experiment in which a continuous pretreatment covariate predicts the outcome, so adjusting for it buys precision.

Usage

```
covariate_adjustment
```

Format

covariate_adjustment:

A tibble with 100 rows and 9 columns:

ID Subject identifier

U Unobserved subject-level shock

X Continuous pretreatment covariate

Y_Z_0, Y_Z_1 Continuous potential outcomes under control and treatment

Z Random assignment, 1 for treatment and 0 for control

Z_cond_prob Probability of the assigned condition, 0.5 throughout

Y Revealed outcome

condition Z labelled "Treatment" and "Control"

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [noncompliance_experiment](#), [patriot_act](#), [two_arm_trial](#)

impute_extreme_values *Impute the extreme value bounds for an outcome with attrition*

Description

Attrition is missingness in the outcome variable. Dropping the units whose outcome is missing conditions the analysis on a post-treatment variable and can induce bias. Extreme value bounds (Manski, 1999) sidestep the problem by imputing the logical best case and worst case instead: the upper bound imputes the largest possible outcome for missing treated units and the smallest possible outcome for missing control units, and the lower bound does the reverse.

Usage

```
impute_extreme_values(data, outcome, assignment, range, treated = NULL)
```

Arguments

data	A data frame containing the outcome and the assignment.
outcome	The name of the outcome column, as a string. The column must be numeric, and missing values are what gets imputed.
assignment	The name of the random assignment column, as a string. It must have exactly two distinct values and no missing values.
range	The logical minimum and maximum of the outcome, as a numeric vector of length two. For a seven-point Likert item, <code>c(1, 7)</code> .
treated	The value of assignment that denotes the treated group. Defaults to the second value in sort order, which is "Treatment" for <code>c("Control", "Treatment")</code> and 1 for <code>c(0, 1)</code> . Set it explicitly when sort order does not pick out the treated group.

Details

This function does the imputation only, so that the two scenarios can be plotted alongside the observed data. It estimates nothing. For the bounds themselves and their uncertainty, see `estimator_ev()` in the 'attrition' package, available at <https://github.com/acoppock/attrition>.

`range` has no default. The logical minimum and maximum of the outcome are the substantive input the bounds rest on, so they are stated at the call site rather than guessed from the observed data. Guessing would silently narrow the bounds whenever no respondent used an endpoint of the scale.

Value

A data frame with twice as many rows as `data`, holding both scenarios stacked. The outcome column carries the imputed values, and two columns are added: `scenario`, a factor with levels "Lower bound" and "Upper bound", suitable for faceting; and `imputed`, a factor with levels "Outcome available" and "Outcome imputed", suitable for mapping to both colour and shape so the distinction survives in grayscale.

References

Manski, C. F. (1999). *Identification Problems in the Social Sciences*. Harvard University Press.

See Also

Other Functions: [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(
  Z = rep(c("Control", "Treatment"), each = 100),
  Y = c(sample(1:7, 100, replace = TRUE), sample(1:7, 100, replace = TRUE))
)
dat$Y[sample(200, 30)] <- NA

bounded <- impute_extreme_values(dat, "Y", "Z", range = c(1, 7))

ggplot(bounded, aes(Z, Y, colour = imputed, shape = imputed)) +
  geom_point(position = position_sunflower(density = 30, aspect_ratio = 1 / 4),
    alpha = 0.5) +
  facet_wrap(~ scenario) +
  scale_y_continuous(breaks = 1:7)
```

noncompliance_experiment

Simulated experiment with two-sided noncompliance

Description

Assignment does not determine receipt in either direction: 50 of the 300 control subjects took the treatment anyway, and 100 of the 300 treated subjects did not take it. Receipt is a post-treatment variable, so conditioning on it invites post-treatment bias.

Usage

```
noncompliance_experiment
```

Format

```
noncompliance_experiment:
```

A tibble with 600 rows and 3 columns:

Z Random assignment, "Treatment" or "Control"

D Treatment receipt, 1 if the subject took the treatment. A post-treatment variable

Y Turnout, 1 if the subject voted

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [patriot_act](#), [two_arm_trial](#)

patriot_act	<i>Original and Replication data for the Patriot Act experiment described in Persuasion in Parallel</i>
-------------	---

Description

Original and Replication data for the Patriot Act experiment described in Persuasion in Parallel

Usage

```
patriot_act
```

Format

patriot_act:

A tibble with 2062 rows and 4 columns:

sample_label The original study (Chong and Druckman (2011) or the Mechanical Turk replication)

pid_3 Subject partisanship (limited to Republicans and Democrats, including leaners)

T1_content Content of assigned treatment condition: pro-Patriot act statements, anti-Patriot act statements, or a control

PA_support Post-treatment support for the Patriot Act on a 1 to 7 Likert scale

Source

[doi:10.7910/DVN/I9GSKI](https://doi.org/10.7910/DVN/I9GSKI)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [noncompliance_experiment](#), [two_arm_trial](#)

position_bluenoise	<i>Scatter over-plotted points evenly at random</i>
--------------------	---

Description

This function spreads perfectly over-plotted points across an elliptical field, like [position_jitter_ellipse\(\)](#), but places them so that no two land much closer together than the rest. Sampling uniformly at random, which is what jittering does, leaves visible knots and voids: in a draw of 200 points the closest pair typically sits about a fifteenth of the median spacing apart. A reader cannot tell those knots from real structure. The arrangement here has the even spacing of [position_sunflower\(\)](#) while still looking unstructured, so no reader mistakes a spiral arm for a finding.

Usage

```
position_bluenoise(width = NULL, height = NULL, candidates = 10, seed = NA)
```

Arguments

width, height	The dimensions of the elliptical field the points are spread across.
candidates	The number of random draws considered for each point. Larger values space the points more evenly and take longer. The default of 10 is enough to remove essentially all of the clumping.
seed	A random seed for reproducibility.

Details

The pattern is the one the eye's own photoreceptors are laid out in, known as blue noise or a Poisson-disc distribution. It is produced by Mitchell's best-candidate algorithm: each point is the best of candidates random draws, where best means farthest from every point already placed.

Value

A ggproto object of class `PositionBlueNoise`.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(x = rep(1, 400), y = rep(1, 400))

# Evenly scattered.
ggplot(dat, aes(x, y)) +
  geom_point(position = position_bluenoise(width = 0.5, height = 0.5)) +
  coord_equal(xlim = c(0, 2), ylim = c(0, 2))

# Uniformly jittered, for comparison. Note the knots and the gaps.
ggplot(dat, aes(x, y)) +
  geom_point(position = position_jitter_ellipse(width = 0.5, height = 0.5)) +
  coord_equal(xlim = c(0, 2), ylim = c(0, 2))
```

position_bluenoisedodge

Scatter over-plotted points evenly at random and dodge groups side-to-side

Description

This function dodges groups of points side-to-side and then scatters the points that share a position across an elliptical field, spacing them evenly. See the [position_bluenoise\(\)](#) documentation for more information.

Usage

```
position_bluenoisedodge(
  scatter.width = NULL,
  scatter.height = NULL,
  dodge.width = 1,
  candidates = 10,
  seed = NA,
  orientation = "x"
)
```

Arguments

scatter.width, scatter.height	The dimensions of the elliptical field the points are spread across.
dodge.width	The dodging width, which defaults to 1.
candidates	The number of random draws considered for each point.
seed	A random seed for reproducibility.
orientation	The axis along which groups are separated, either "x" (the default, side-to-side) or "y" (up and down). Matches the argument of the same name in ggplot2::position_dodge() .

Value

A ggproto object of class PositionBlueNoiseDodge.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(x = rep(1, 400), y = rep(1, 400),
                  group = sample(LETTERS[1:2], 400, replace = TRUE))

ggplot(dat, aes(x, y, shape = group, color = group)) +
  geom_point(position = position_bluenoisododge(scatter.width = 0.4,
                                              scatter.height = 0.4,
                                              dodge.width = 1)) +
  coord_cartesian(xlim = c(0, 2), ylim = c(0, 2))
```

position_circlepack	<i>Arrange over-plotted points with a circle-packing algorithm</i>
---------------------	--

Description

This function uses a circle packing algorithm from the 'packcircles' package to arrange perfectly over-plotted points of varying sizes into an elliptical area.

Usage

```
position_circlepack(density = 1, aspect_ratio = 1)
```

Arguments

density	The density of the circle pack, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image. It will also depend on the size scale.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if coord_equal() is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under coord_fixed(), set aspect_ratio to the same value as that function's ratio argument.

Value

A ggproto object of class PositionCirclePack.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisododge\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```

library(ggplot2)
library(dplyr)
library(randomizr)
library(tibble)

dat <- data.frame(
  X = c(rep(0, 200)),
  Y = rep(0, 200),
  size = runif(200, 0, 1)
)

ggplot(dat, aes(x = X, y = Y, size = size)) +
  geom_point(position = position_circlepack(density = 0.25, aspect_ratio = 1),
    alpha = 0.25) +
  coord_equal(xlim = c(-1, 1), ylim = c(-1, 1), expand = TRUE) +
  theme(legend.position = "none")

# Applied to a mock experiment with weighted groups

dat <-
  tibble(
    age_group = rep(c("young", "middle", "old"), c(100, 200, 300)),
    treatment = block_ra(age_group, block_m = c(50, 50, 50)),
    latent_outcome =
      case_when(age_group == "young" & treatment == 0 ~ 0.10,
        age_group == "young" & treatment == 1 ~ 0.20,
        age_group == "middle" & treatment == 0 ~ 0.40,
        age_group == "middle" & treatment == 1 ~ 0.45,
        age_group == "old" & treatment == 0 ~ 0.70,
        age_group == "old" & treatment == 1 ~ 0.90),
    outcome = rbinom(600, size = 1,
      prob = latent_outcome)
  )

dat <-
  dat |>
  mutate(
    treatment_prob =
      case_when(age_group == "young" ~ 50/100,
        age_group == "middle" ~ 50/200,
        age_group == "old" ~ 50/300),
    weights = 1/case_when(treatment == 1 ~ treatment_prob,
      treatment == 0 ~ 1 - treatment_prob)
  )

ggplot(dat, aes(treatment, outcome, size = weights, color = age_group)) +
  geom_point(alpha = 0.5, position = position_circlepack(density = 0.5))

```

position_circlepackdodge

Arrange over-plotted points with a circle-packing algorithm and dodge groups side-to-side

Description

This function dodges groups and uses a circle packing algorithm from the 'packcircles' package to arrange perfectly over-plotted points of varying sizes into an elliptical area.

Usage

```
position_circlepackdodge(
  width = 1,
  density = 1,
  aspect_ratio = 1,
  orientation = "x"
)
```

Arguments

width	The dodging width, which defaults to 1.
density	The density of the circle pack, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image. It will also depend on the size scale.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if coord_equal() is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under coord_fixed(), set aspect_ratio to the same value as that function's ratio argument.
orientation	The axis along which groups are separated, either "x" (the default, side-to-side) or "y" (up and down). Matches the argument of the same name in ggplot2::position_dodge() .

Value

A ggproto object of class PositionCirclePackDodge.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(
  X = c(rep(0, 200)),
  Y = rep(0, 200),
  size = runif(200, 0, 1),
  id = (rep(c("A", "B"), 100))
)

ggplot(dat, aes(x = X, y = Y, size = size, color = id)) +
  geom_point(position = position_circlepackdodge(width = 1, density = 1, aspect_ratio = 1),
    alpha = 0.25) +
  coord_equal(xlim = c(-1, 1), ylim = c(-1, 1), expand = TRUE) +
  scale_size_continuous(range = c(1, 3)) +
  theme(legend.position = "none")
```

position_honeycomb	<i>Arrange over-plotted points on a honeycomb lattice</i>
--------------------	---

Description

This function arranges perfectly over-plotted points on a hexagonal lattice, filling outward from the centre in the order of the data. The hexagonal lattice is the densest packing of equal circles in the plane, so the cluster is as compact as it can be for a given spacing.

Usage

```
position_honeycomb(density = 1, aspect_ratio = 1)
```

Arguments

density	The pattern density, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if <code>coord_equal()</code> is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under <code>coord_fixed()</code> , set <code>aspect_ratio</code> to the same value as that function's <code>ratio</code> argument.

Details

It is the crystalline sibling of [position_sunflower\(\)](#), which fills the same footprint at the same density with a spiral rather than a lattice. Choose between them on looks: the lattice reads as countable and orderly, the spiral as organic and without a preferred direction.

`position_beeswarm()` in the 'ggbeeswarm' package also offers a hexagonal method, and does a different job. A beeswarm spreads points along one axis to show the shape of a distribution, so perfectly over-plotted points come out as a line rather than a cluster, and its hexagonal and square methods move points off their true value on the data axis. Reach for a beeswarm to show a distribution, and for this to show a count.

Value

A ggproto object of class `PositionHoneycomb`.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(
  x = rep(1:3, times = 60),
  y = rep(1:3, times = 60)
)

ggplot(dat, aes(x, y)) +
  geom_point(size = 1, position = position_honeycomb(density = 4)) +
  coord_equal()
```

`position_honeycombdodge`

Arrange over-plotted points on a honeycomb lattice and dodge groups side-to-side

Description

This function applies the honeycomb position adjustment alongside the dodge position adjustment, arranging overlapping points per x, y, and group on a hexagonal lattice. See the [position_honeycomb\(\)](#) documentation for more information.

Usage

```
position_honeycombdodge(
  width = 1,
  density = 1,
  aspect_ratio = 1,
  orientation = "x"
)
```

Arguments

width	The dodging width, which defaults to 1.
density	The pattern density, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if <code>coord_equal()</code> is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under <code>coord_fixed()</code> , set <code>aspect_ratio</code> to the same value as that function's <code>ratio</code> argument.
orientation	The axis along which groups are separated, either "x" (the default, side-to-side) or "y" (up and down). Matches the argument of the same name in ggplot2::position_dodge() .

Value

A ggproto object of class `PositionHoneycombDodge`.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(
  x = rep(1, 300),
  y = rep(1, 300),
  type = factor(sample(LETTERS[1:2], 300, replace = TRUE))
)

ggplot(dat, aes(x, y, color = type, shape = type)) +
  geom_point(position = position_honeycombdodge(width = 1, density = 30)) +
  coord_equal()
```

Description

This function adds elliptical random noise to perfectly over-plotted points, offering a pleasing way to visualize many points that represent the same position. In contrast to the `position_jitter()` function, which samples from a rectangular field, the `position_jitter_ellipse()` function samples from an elliptical field. This function takes algorithmic inspiration from <https://stackoverflow.com/questions/5529148/algorithm-calculate-pseudo-random-point-inside-an-ellipse> and <https://stats.stackexchange.com/questions/120527/simulate-a-uniform-distribution-on-a-disc>.

Usage

```
position_jitter_ellipse(width = NULL, height = NULL, seed = NA)
```

Arguments

<code>width, height</code>	The dimensions of the elliptical field, from which over-plotted points are sampled.
<code>seed</code>	A random seed for reproducibility.

Value

A ggproto object of class `PositionJitterEllipse`.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflower\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

dat <- data.frame(x = rep(1, 500), y = rep(1, 500))

# Jitter on an ellipse.
ggplot(dat, aes(x, y)) +
  geom_point(position = position_jitter_ellipse(width = 0.5, height = 0.5)) +
  coord_cartesian(xlim = c(0, 2), ylim = c(0, 2))

# Jitter on a rectangle, for comparison.
ggplot(dat, aes(x, y)) +
  geom_point(position = position_jitter(width = 0.5, height = 0.5)) +
  coord_cartesian(xlim = c(0, 2), ylim = c(0, 2))
```

position_jitterdodge_ellipse

Jitter points on an ellipse and dodge groups side-to-side

Description

This function dodges groups of points side-to-side and adds elliptical random noise to perfectly over-plotted points. See the `position_jitter_ellipse()` documentation for more information.

Usage

```
position_jitterdodge_ellipse(
  jitter.width = NULL,
  jitter.height = NULL,
  dodge.width = 1,
  seed = NA,
  orientation = "x"
)
```

Arguments

jitter.width, jitter.height	The dimensions of the elliptical field, from which over-plotted points are sampled.
dodge.width	The dodging width, which defaults to 1.
seed	A random seed for reproducibility.
orientation	The axis along which groups are separated, either "x" (the default, side-to-side) or "y" (up and down). Matches the argument of the same name in <code>ggplot2::position_dodge()</code> .

Value

A ggproto object of class `PositionJitterDodgeEllipse`.

See Also

Other Functions: `impute_extreme_values()`, `position_bluenoise()`, `position_bluenoisedodge()`, `position_circlepack()`, `position_circlepackdodge()`, `position_honeycomb()`, `position_honeycombdodge()`, `position_jitter_ellipse()`, `position_sunflower()`, `position_sunflowerdodge()`, `sunflower()`

Examples

```
library(ggplot2)

dat <- data.frame(x = rep(1, 500), y = rep(1, 500),
                  group = sample(LETTERS[1:2], 500, replace = TRUE))

ggplot(dat, aes(x, y, shape = group, color = group)) +
```

```
geom_point(position = position_jitterdodge_ellipse(jitter.width = 0.5,
                                                    jitter.height = 0.5,
                                                    dodge.width = 1)) +
coord_cartesian(xlim = c(0, 2), ylim = c(0, 2))
```

position_sunflower	<i>Arrange over-plotted points in a sunflower pattern</i>
--------------------	---

Description

This function applies the sunflower algorithm, executed by [sunflower\(\)](#), as a position adjustment, arranging overlapping points at any given x and y into a sunflower pattern. See the [sunflower\(\)](#) documentation for more information.

Usage

```
position_sunflower(density = 1, aspect_ratio = 1)
```

Arguments

density	The pattern density, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if <code>coord_equal()</code> is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under <code>coord_fixed()</code> , set <code>aspect_ratio</code> to the same value as that function's <code>ratio</code> argument.

Value

A ggproto object of class `PositionSunflower`.

See Also

Other Functions: [impute_extreme_values\(\)](#), [position_bluenoise\(\)](#), [position_bluenoisedodge\(\)](#), [position_circlepack\(\)](#), [position_circlepackdodge\(\)](#), [position_honeycomb\(\)](#), [position_honeycombdodge\(\)](#), [position_jitter_ellipse\(\)](#), [position_jitterdodge_ellipse\(\)](#), [position_sunflowerdodge\(\)](#), [sunflower\(\)](#)

Examples

```
library(ggplot2)

# Use the sunflower position function to arrange N points
N <- 100

dat <- data.frame(
  x = rep(1:4, times = N),
  y = rep(1:4, times = N)
)

ggplot(dat, aes(x = x, y = y)) +
  geom_point(size = 1, position = position_sunflower(density = 1, aspect_ratio = 1)) +
  xlim(0, 5) +
  ylim(0, 5) +
  coord_equal()
```

position_sunflowerdodge

Arrange over-plotted points in a sunflower pattern and dodge groups side-to-side

Description

This function applies the sunflower position adjustment alongside the dodge position adjustment, arranging overlapping points per x, y, and group into a sunflower pattern. See the [sunflower\(\)](#) documentation for more information.

Usage

```
position_sunflowerdodge(
  width = 1,
  density = 1,
  aspect_ratio = 1,
  orientation = "x"
)
```

Arguments

width	The dodging width, which defaults to 1.
density	The pattern density, which defaults to 1 but will have to be adjusted in most cases. The desirable density will depend on both the ranges of the axes and the dimensions of the image.
aspect_ratio	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if coord_equal() is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect

ratio (e.g., 0.5) would adjust for horizontal stretching. The default aspect ratio of 1 is appropriate when no adjustment is required. Under `coord_fixed()`, set `aspect_ratio` to the same value as that function's `ratio` argument.

orientation The axis along which groups are separated, either "x" (the default, side-to-side) or "y" (up and down). Matches the argument of the same name in `ggplot2::position_dodge()`.

Value

A ggproto object of class `PositionSunflowerDodge`.

See Also

Other Functions: `impute_extreme_values()`, `position_bluenoise()`, `position_bluenoisedodge()`, `position_circlepack()`, `position_circlepackdodge()`, `position_honeycomb()`, `position_honeycombdodge()`, `position_jitter_ellipse()`, `position_jitterdodge_ellipse()`, `position_sunflower()`, `sunflower()`

Examples

```
library(ggplot2)

# Use the sunflower dodge position function to arrange and dodge N points.
N <- 300

dat <- data.frame(
  x = sample(1:2, size = N, replace = TRUE),
  y = sample(1:7, size = N, replace = TRUE),
  type = factor(sample(LETTERS[1:2], N, replace = TRUE))
)

# With coord_equal
ggplot(dat, aes(x, y, color = type, shape = type)) +
  geom_point(position = position_sunflowerdodge(width = 0.5, density = 2, aspect_ratio = 1)) +
  coord_equal()

# Without coord_equal, might want to play with aspect ratio to get a pleasing plot
ggplot(dat, aes(x, y, color = type, shape = type)) +
  geom_point(position = position_sunflowerdodge(width = 0.5, density = 10, aspect_ratio = 1/4))
```

sunflower

Distribute points using a sunflower seed algorithm

Description

This function distributes points in an ellipse via a sunflower seed algorithm as a solution for overplotting. To implement the algorithm, this function adapts the code from <https://stackoverflow.com/questions/28567166/uniformly-distribute-x-points-inside-a-circle>.

Usage

```
sunflower(x = NULL, y = NULL, density, aspect_ratio)
```

Arguments

<code>x, y</code>	The identical coordinates of multiple over-plotted points, as vectors, which will be arranged using a sunflower seed algorithm. A vector of length one is returned unchanged, since a point with nothing over-plotting it belongs at its own coordinates.
<code>density</code>	The pattern density.
<code>aspect_ratio</code>	An aspect ratio adjustment to compensate for distortion of the circular arrangement, which might occur when plotting if <code>coord_equal()</code> is not used. A wide aspect ratio (e.g., 2) would adjust for vertical stretching, whereas a tall aspect ratio (e.g., 0.5) would adjust for horizontal stretching. An aspect ratio of 1 is appropriate when no adjustment is required.

Details

This is the engine `position_sunflower()` runs on, exported so that the arrangement can be computed as data rather than at draw time. Reach for it when another layer needs the arranged coordinates, or when writing a `Position` class of your own: call it once for `x` and once for `y`, as `PositionSunflower` does. The two calls agree because the arrangement is determined by the number of over-plotted points, so the same point is placed at the same angle in both. The package's other algorithms do not decompose that way and have no equivalent function.

Value

A numeric vector of adjusted `x` or `y` positions, computed using a sunflower seed algorithm.

See Also

Other Functions: `impute_extreme_values()`, `position_bluenoise()`, `position_bluenoisedodge()`, `position_circlepack()`, `position_circlepackdodge()`, `position_honeycomb()`, `position_honeycombdodge()`, `position_jitter_ellipse()`, `position_jitterdodge_ellipse()`, `position_sunflower()`, `position_sunflowerdodge()`

Examples

```
library(ggplot2)
library(dplyr)

# Arrange the points as data, keeping the position each one came from, so
# that a second layer can draw from the cell centre out to each point. A
# position adjustment cannot do this, because it arranges at draw time and
# the arranged coordinates never reach the data.
N <- 300

dat <- data.frame(
  x = sample(1:2, size = N, replace = TRUE),
  y = sample(1:7, size = N, replace = TRUE)
```

```

) |>
  group_by(x, y) |>
  mutate(
    x_flower = sunflower(x = x, density = 1, aspect_ratio = 1),
    y_flower = sunflower(y = y, density = 1, aspect_ratio = 1)
  )

ggplot(dat, aes(x_flower, y_flower)) +
  geom_segment(aes(xend = x, yend = y), colour = "grey80") +
  geom_point() +
  coord_equal()

# To dodge groups as well, use position_sunflowerdodge(), which handles the
# dodging and the arrangement together and accepts an orientation.

```

two_arm_trial	<i>Simulated two-arm trial</i>
---------------	--------------------------------

Description

One of seven simulated datasets behind the worked examples in Coppock (2021). A completely randomized trial with a binary outcome, in which 100 of 500 subjects are treated, so the two arms have different assignment probabilities.

Usage

```
two_arm_trial
```

Format

two_arm_trial:

A tibble with 500 rows and 8 columns:

ID Subject identifier

U Unobserved subject-level shock

Y_Z_0, Y_Z_1 Binary potential outcomes under control and treatment

Z Random assignment, 1 for treatment and 0 for control

Z_cond_prob Probability of the assigned condition, 0.2 for treated and 0.8 for control

Y Revealed outcome

condition Z labelled "Treatment" and "Control"

Details

Every dataset in this family reports both potential outcomes. They are knowable here because the data are simulated, and no real experiment observes them. `Z_cond_prob` is the probability that a unit was assigned to the condition it is actually in, which is the quantity an inverse probability weight inverts.

Source

[doi:10.7910/DVN/VE6VSR](https://doi.org/10.7910/DVN/VE6VSR)

See Also

Other Data: [attrition_experiment](#), [blocked_experiment](#), [clustered_experiment](#), [continuous_interaction](#), [covariate_adjustment](#), [noncompliance_experiment](#), [patriot_act](#)

vayr

Extensions for 'ggplot2' to Visualize as You Randomize

Description

Position adjustments for 'ggplot2' to implement "visualize as you randomize" principles, which can be especially useful when plotting experimental data.

Details

The 'vayr' package provides 'ggplot2' extensions that foster "visualize as you randomize" principles. These principles should guide the visualization of experimental data. Thus far, the package includes position adjustments that avoid over-plotting, which helps organize "data-space." The principles are set out in Coppock (2021) [doi:10.1017/9781108777919.022](https://doi.org/10.1017/9781108777919.022).

Author(s)

Maintainer: Alexander Coppock <acoppock@gmail.com> ([ORCID](#)) [copyright holder]

Authors:

- Alexander Coppock <acoppock@gmail.com> ([ORCID](#)) [copyright holder]

Other contributors:

- Elias Hyde <eliasworralhyde@gmail.com> [contributor]

See Also

Useful links:

- <https://alexandercoppock.com/vayr/>

Index

* Data

attrition_experiment, [2](#)
blocked_experiment, [3](#)
clustered_experiment, [4](#)
continuous_interaction, [5](#)
covariate_adjustment, [6](#)
noncompliance_experiment, [8](#)
patriot_act, [9](#)
two_arm_trial, [24](#)

* Functions

impute_extreme_values, [6](#)
position_bluenoise, [9](#)
position_bluenoisedodge, [11](#)
position_circlepack, [12](#)
position_circlepackdodge, [14](#)
position_honeycomb, [15](#)
position_honeycombdodge, [16](#)
position_jitter_ellipse, [17](#)
position_jitterdodge_ellipse, [19](#)
position_sunflower, [20](#)
position_sunflowerdodge, [21](#)
sunflower, [22](#)

* Package

vayr, [25](#)

* datasets

attrition_experiment, [2](#)
blocked_experiment, [3](#)
clustered_experiment, [4](#)
continuous_interaction, [5](#)
covariate_adjustment, [6](#)
noncompliance_experiment, [8](#)
patriot_act, [9](#)
two_arm_trial, [24](#)

attrition_experiment, [2](#), [4–6](#), [8](#), [9](#), [25](#)

blocked_experiment, [3](#), [3–6](#), [8](#), [9](#), [25](#)

clustered_experiment, [3](#), [4](#), [4–6](#), [8](#), [9](#), [25](#)

continuous_interaction, [3](#), [4](#), [5](#), [6](#), [8](#), [9](#), [25](#)

covariate_adjustment, [3–5](#), [6](#), [8](#), [9](#), [25](#)

ggplot2::position_dodge(), [11](#), [14](#), [17](#), [19](#),
[22](#)

impute_extreme_values, [6](#)

impute_extreme_values(), [2](#), [10–12](#), [14](#),
[16–20](#), [22](#), [23](#)

noncompliance_experiment, [3–6](#), [8](#), [9](#), [25](#)

patriot_act, [3–6](#), [8](#), [9](#), [25](#)

position_bluenoise, [9](#)

position_bluenoise(), [7](#), [11](#), [12](#), [14](#), [16–20](#),
[22](#), [23](#)

position_bluenoisedodge, [11](#)

position_bluenoisedodge(), [7](#), [10](#), [12](#), [14](#),
[16–20](#), [22](#), [23](#)

position_circlepack, [12](#)

position_circlepack(), [7](#), [10](#), [11](#), [14](#),
[16–20](#), [22](#), [23](#)

position_circlepackdodge, [14](#)

position_circlepackdodge(), [7](#), [10–12](#),
[16–20](#), [22](#), [23](#)

position_honeycomb, [15](#)

position_honeycomb(), [7](#), [10–12](#), [14](#), [16–20](#),
[22](#), [23](#)

position_honeycombdodge, [16](#)

position_honeycombdodge(), [7](#), [10–12](#), [14](#),
[16](#), [18–20](#), [22](#), [23](#)

position_jitter_ellipse, [17](#)

position_jitter_ellipse(), [7](#), [9–12](#), [14](#),
[16](#), [17](#), [19](#), [20](#), [22](#), [23](#)

position_jitterdodge_ellipse, [19](#)

position_jitterdodge_ellipse(), [7](#),
[10–12](#), [14](#), [16–18](#), [20](#), [22](#), [23](#)

position_sunflower, [20](#)

position_sunflower(), [7](#), [9–12](#), [14–19](#), [22](#),
[23](#)

position_sunflowerdodge, [21](#)

position_sunflowerdodge(), [7](#), [10–12](#), [14](#),
[16–20](#), [23](#)

sunflower, [22](#)

sunflower(), [7](#), [10–12](#), [14](#), [16–22](#)

two_arm_trial, [3–6](#), [8](#), [9](#), [24](#)

vayr, [25](#)

vayr-package (vayr), [25](#)