MONASH University

Advanced Data Visualisation with R

Writing $ggplot2$ extensions

Instructor: *Emi Tanaka*

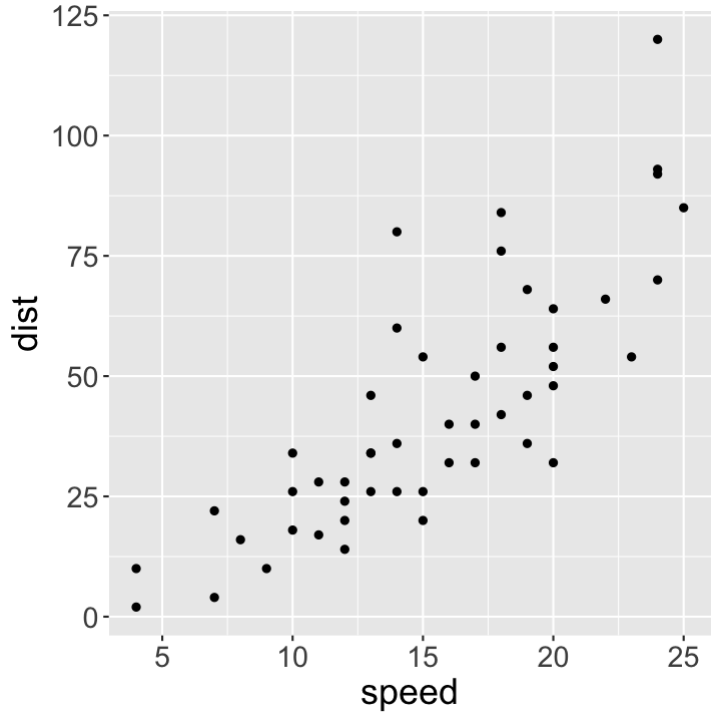
emi.tanaka@monash.edu

8th Dec 2021 @ Statistical Society of Australia Canberra Branch | Zoom

Dissecting the ggplot object

```
library(ggplot2)
g <- ggplot(cars, aes(speed, dist)) +
  geom_point()
```

```
g
```




```
str(g)
```

```
## List of 9
##  $ data       :'data.frame':    50 obs. of  2 variables:
##   ..$ speed: num [1:50] 4 4 7 7 8 9 10 10 10 11 ...
##   ..$ dist : num [1:50] 2 10 4 22 16 10 18 26 34 17 ...
##  $ layers     :List of 1
##   ..$ :Classes 'LayerInstance', 'Layer', 'ggproto', 'gg' <ggproto objec
##     aes_params: list
##     compute_aesthetics: function
##     compute_geom_1: function
##     compute_geom_2: function
##     compute_position: function
```

- Notice that a layer, `geom`, `position`, `stat`, `scales`, `coordinates` and `facet` are **`ggproto` objects**

ggproto

ggproto basics

- ggplot2 makes heavy use of **prototype-based programming**
- ggproto is a custom build class system made specically for ggplot
- The system is similar to R6Class in the R6 package that allow inheritance and method access from parent classes

```
OzCovidTracker <- ggproto("OzCovidTracker", NULL,
                  cases = 0,
                  location = "Australia",
                  last_update = NA,
                  add = function(self, cases = 0) {
                    self$cases <- self$cases + cases
                    self$last_update <- Sys.Date()
                  },
                  reset = function(self) {
                    self$cases <- 0
                  })
```

```
OzCovidTracker

## <ggproto object: Class OzCovidTracker, gg>
##     add: function
##     cases: 0
##     last_update: NA
##     location: Australia
##     reset: function

OzCovidTracker$add(cases = 219120)

OzCovidTracker$cases

## [1] 219120

OzCovidTracker$add(cases = 80)

OzCovidTracker$cases

## [1] 219200
```

ggproto inheritance

```
VicCovidTracker <- ggproto("VicCovidTracker", OzCovidTracker, location = "Victor
```

```
VicCovidTracker$reset()
VicCovidTracker
```

```
## <ggproto object: Class VicCovidTracker, OzCovidTracker, gg>
##     add: function
##     cases: 0
##     last_update: 2021-12-07
##     location: Victoria
##     reset: function
##     super:  <ggproto object: Class OzCovidTracker, gg>
```

```
VicCovidTracker$add(128849)
```

```
VicCovidTracker$cases
```

```
## [1] 128849
```

Creating a new `ggproto` object

- You should not be creating a new `ggroto` object from scratch
- You should inherit existing `ggproto` objects as outlined below:

New `ggproto`	Parent `ggproto`
`geom`	`ggplot2::Geom`
`position`	`ggplot2::Position`
`stat`	`ggplot2::Stat`
`scales`	`ggplot2::Scale`
`coordinates`	`ggplot2::Coord`
`facet`	`ggplot2::Facet`

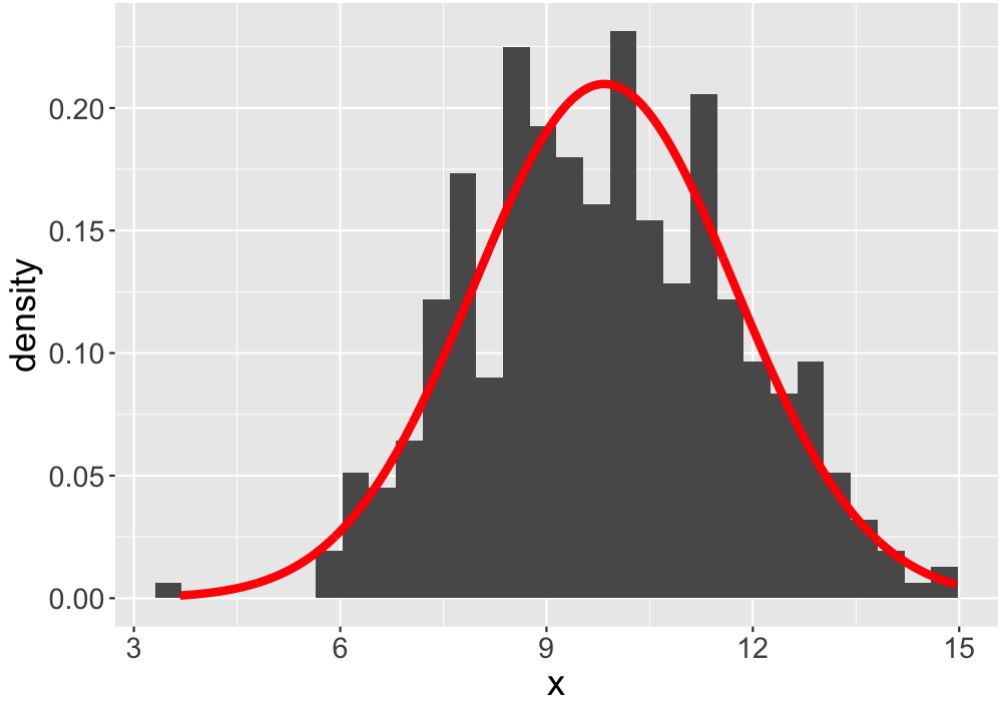
- The convention for class names is to prefix with the parent (or ancestor) `ggproto` class name and use upper camel case, e.g. `GeomPoint`.

Stat

Example: fit a normal distribution

🎯 Create a new layer called `stat_dist_normal()` which fits a normal density curve

```r
# your data
df <- data.frame(x = rnorm(400, 10, 2))

# normal fit
x <- seq(min(df$x), max(df$x),
         length.out = 1000)
fit <- data.frame(x = x,
                  y = dnorm(x, mean(df$x), sd(df$x)))

# plot
ggplot(df, aes(x)) +
  geom_histogram(aes(y = stat(density)), bins = 30) +
  geom_line(data = fit, color = "red", size = 2,
            aes(x, y))
```



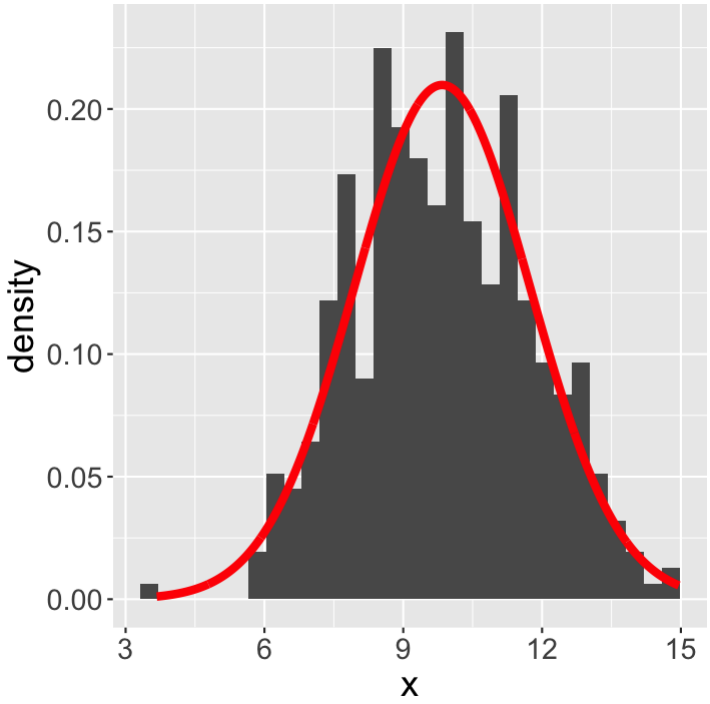
Stat ggproto

```
Stat

## <ggproto object: Class Stat, gg>
##     aesthetics: function
##     compute_group: function
##     compute_layer: function
##     compute_panel: function
##     default_aes: uneval
##     extra_params: na.rm
##     finish_layer: function
##     non_missing_aes:
##     optional_aes:
##     parameters: function
##     required_aes:
##     retransform: TRUE
##     setup_data: function
##     setup_params: function
```

```
StatDistNormal <- ggproto("StatDistNormal", Stat,
                          compute_group = function(data, scales) {
                            x <- seq(min(data$x), max(data$x), length.out = 10
                            y <- dnorm(x, mean(data$x), sd(data$x))
                            data.frame(x = x, y = y)
                          },
                          required_aes = "x")

stat_dist_normal <- function(mapping = NULL, data = NULL, geom = "line",
                             position = "identity", na.rm = FALSE, show.legend = NA,
                             inherit.aes = TRUE, ...) {
  layer(
    stat = StatDistNormal, data = data, geom = "line", position = "identity",
    show.legend = show.legend, inherit.aes = inherit.aes,
    params = list(na.rm = na.rm, ...)
    )
}
```

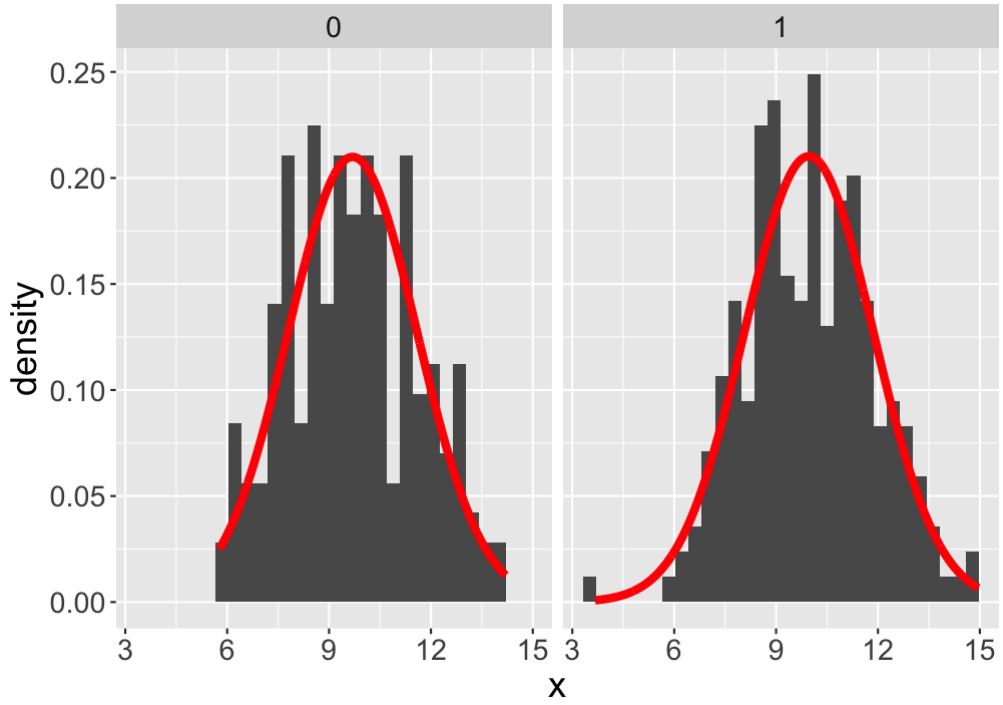
stat_dist_normal

```
ggplot(df, aes(x)) +
  geom_histogram(aes(y = stat(density)), bins = 30) +
  stat_dist_normal(color = "red", size = 2)
```



Works for facetting as well!

```
# make some artificial group
df$group <- sample(c(0, 1), replace = TRUE, size = nrow(df
ggplot(df, aes(x)) +
  geom_histogram(aes(y = stat(density)), bins = 30) +
  stat_dist_normal(color = "red", size = 2) +
  facet_wrap(~group)
```



Geom

Example: tic-tac-toe

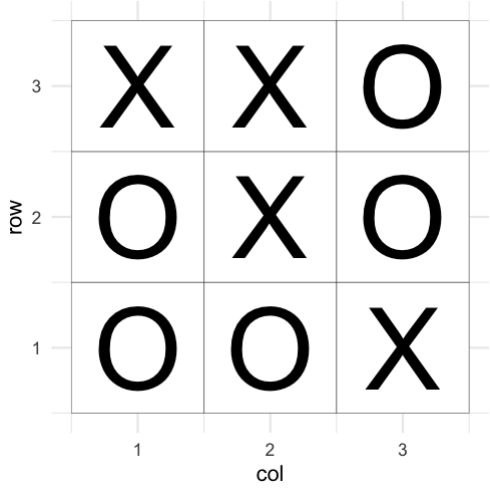
🎯 Create a new layer called `geom_tictactoe()` which draws a tic-tac-toe like board

```
game <- expand.grid(col = 1:3, row = 1:3)
game$move <- sample(rep(c("Alice", "Bob"), c(4, 5)))
```

```
game
```

```
##   col row  move
## 1   1   1   Bob
## 2   2   1   Bob
## 3   3   1 Alice
## 4   1   2   Bob
## 5   2   2 Alice
## 6   3   2   Bob
## 7   1   3 Alice
## 8   2   3 Alice
## 9   3   3   Bob
```

```
ggplot(game, aes(col, row)) +
  geom_tile(fill = "white", color = "black") +
  geom_text(aes(label = ifelse(move=="Alice", "X", "O")),
            size = 20) +
  theme_minimal()
```



Geom ggproto

Geom

```
## <ggproto object: Class Geom, gg>
##     aesthetics: function
##     default_aes: uneval
##     draw_group: function
##     draw_key: function
##     draw_layer: function
##     draw_panel: function
##     extra_params: na.rm
##     handle_na: function
##     non_missing_aes:
##     optional_aes:
##     parameters: function
##     required_aes:
##     setup_data: function
##     setup_params: function
##     use_defaults: function
```

GeomTile

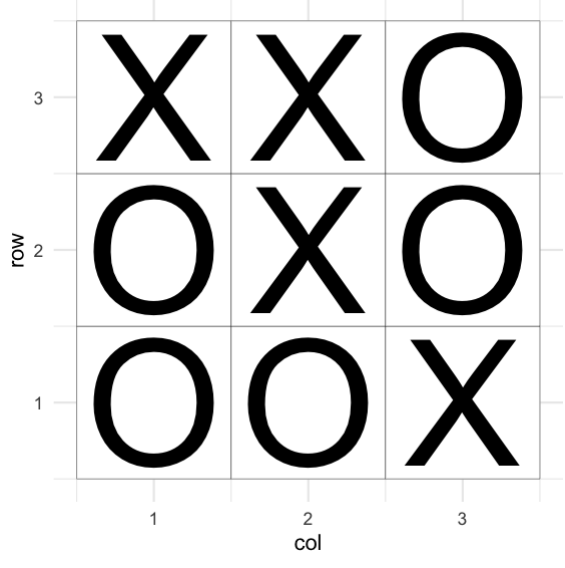
```
## <ggproto object: Class GeomTile, GeomRect, Geom, gg>
##     aesthetics: function
##     default_aes: uneval
##     draw_group: function
##     draw_key: function
##     draw_layer: function
##     draw_panel: function
##     extra_params: na.rm
##     handle_na: function
##     non_missing_aes:
##     optional_aes:
##     parameters: function
##     required_aes: x y
##     setup_data: function
##     setup_params: function
##     use_defaults: function
##     super:  <ggproto object: Class GeomRect, Geom, gg>
```

GeomTicTacToe

```
GeomTicTacToe <- ggproto("GeomTicTacToe", GeomTile,
                         draw_panel = function(data, panel_params, coord) {
                           coords <- coord$transform(data, panel_params)
                           width <- coords$xmax - coords$xmin
                           height <- coords$ymax - coords$ymin
                           tiles <- grid::rectGrob(coords$xmin, coords$ymax,
                                                   width = width,
                                                   height = height,
                                                   default.units = "native",
                                                   just = c("left", "top"),
                                                   gp = grid::gpar(col = coords$colour,
                                                                   fill = alpha(coords$fill, coords$alpha),
                                                                   lwd = coords$size * .pt, lty = coords$linetype,
                                                                   linejoin = "mitre", lineend = "square"))
                           if(length(unique(coords$label)) != 2) {
                             stop("There should be only two players in tic-tac-toe")
                           }
                           fontsize <- min(c(height, width))
                           fontsize <- grid::convertUnit(grid::unit(fontsize, "snpc"), "pt")
                           moves <- grid::textGrob(label = factor(coords$label, labels = c("X", "O")),
                                                   x = coords$x, y = coords$y,
                                                   gp = grid::gpar(fontsize = fontsize))
                           ggplot2:::ggname("geom_tictactoe", grid::gTree(children = grid::gList(tiles, moves)))
                         },
                         required_aes = c("x", "y", "label"))
```

geom_tictactoe

```
geom_tictactoe <- function(mapping = NULL, data = NULL, stat = "identity", posit
    ..., na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, color = "black", f
      list(
        layer(data = data, mapping = mapping, stat = stat, geom = GeomTicTacToe,
              position = position, show.legend = show.legend, inherit.aes = inhe
              params = list(na.rm = na.rm, color = color, fill = fill, ...)),
        theme_minimal())
    }

ggplot(game, aes(col, row,
                 label = move)) +
  geom_tictactoe()
```



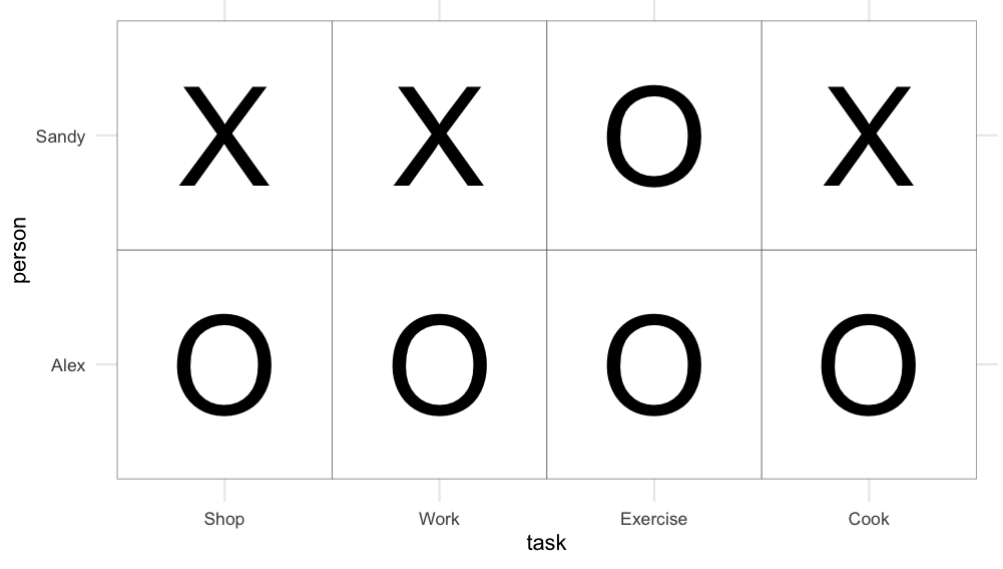
More than the tic-tac-toe game

```
checklist <- expand.grid(person = c("Alex", "Sandy"),
                         task = c("Shop", "Work", "Exercise", "Cook"))
checklist$done <- c(TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE)
```

```
checklist
```

```
##   person     task  done
## 1   Alex     Shop  TRUE
## 2  Sandy     Shop FALSE
## 3   Alex     Work  TRUE
## 4  Sandy     Work FALSE
## 5   Alex Exercise  TRUE
## 6  Sandy Exercise  TRUE
## 7   Alex     Cook  TRUE
## 8  Sandy     Cook FALSE
```

```
ggplot(checklist,
       aes(task, person, label = done)) +
  geom_tictactoe()
```



Resources

For more see the "Extending ggplot2" vignette

Check out also the 3rd edition of the ggplot2 book

https://ggplot2-book.org/extensions.html

And more also in documentation at `?ggplot2::Layout`

</> Open day1-exercise-03.Rmd

15:00

Session Information

```
devtools::session_info()
```

```
## ─ Session info  🐍  🇸🇸  😹 ──────────────────────────────────────────────
##  hash: snake, flag: South Sudan, cat with tears of joy
##
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       macOS Catalina 10.15.7
##  system   x86_64, darwin17.0
##  ui       RStudio
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2021-12-07
```