

Human-Computer Interaction

Reporting & Writing HCI Papers

Professor Bilge Mutlu

Today's Agenda

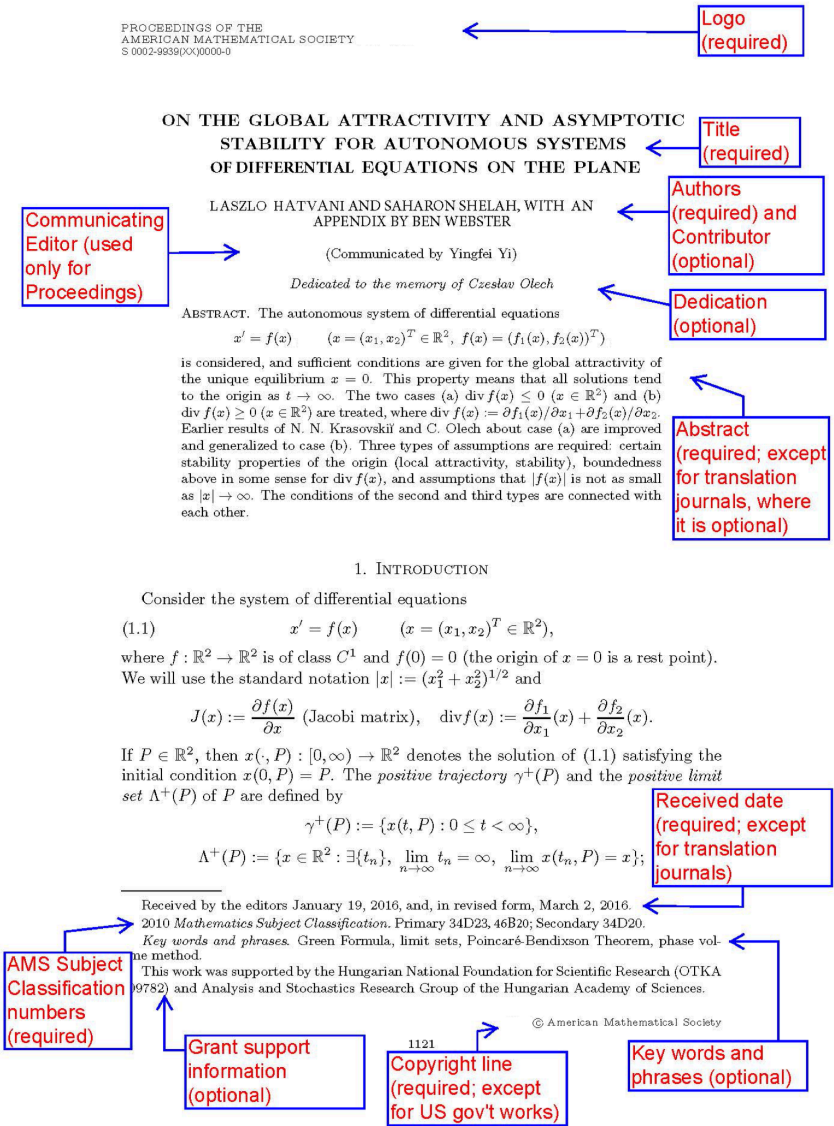
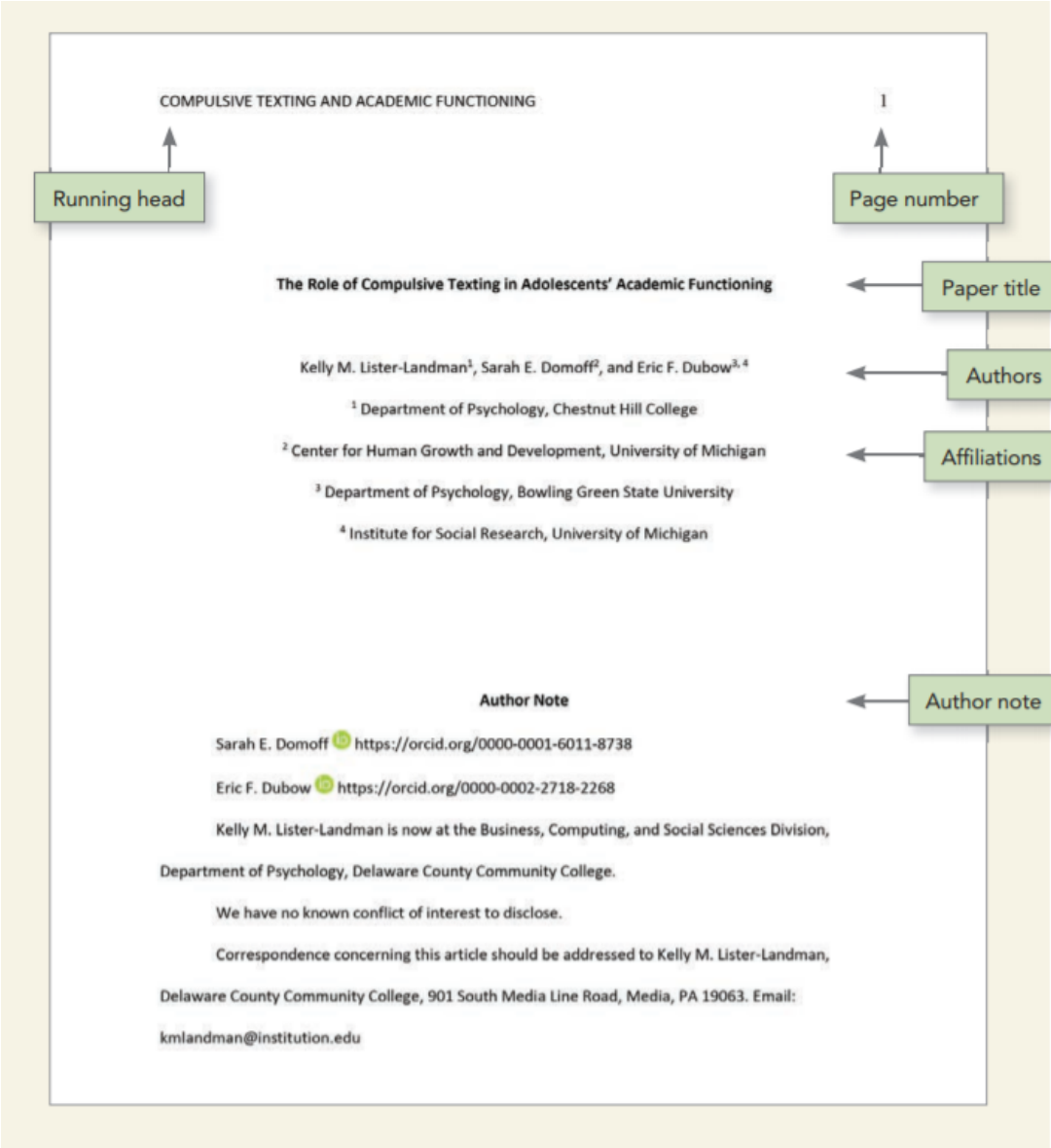
- » Overview: *Reporting Statistics, Writing*
- » General Q&A

What are reporting norms in HCI research?

Because HCI is an interdisciplinary field, the reporting norms are adopted from different fields.

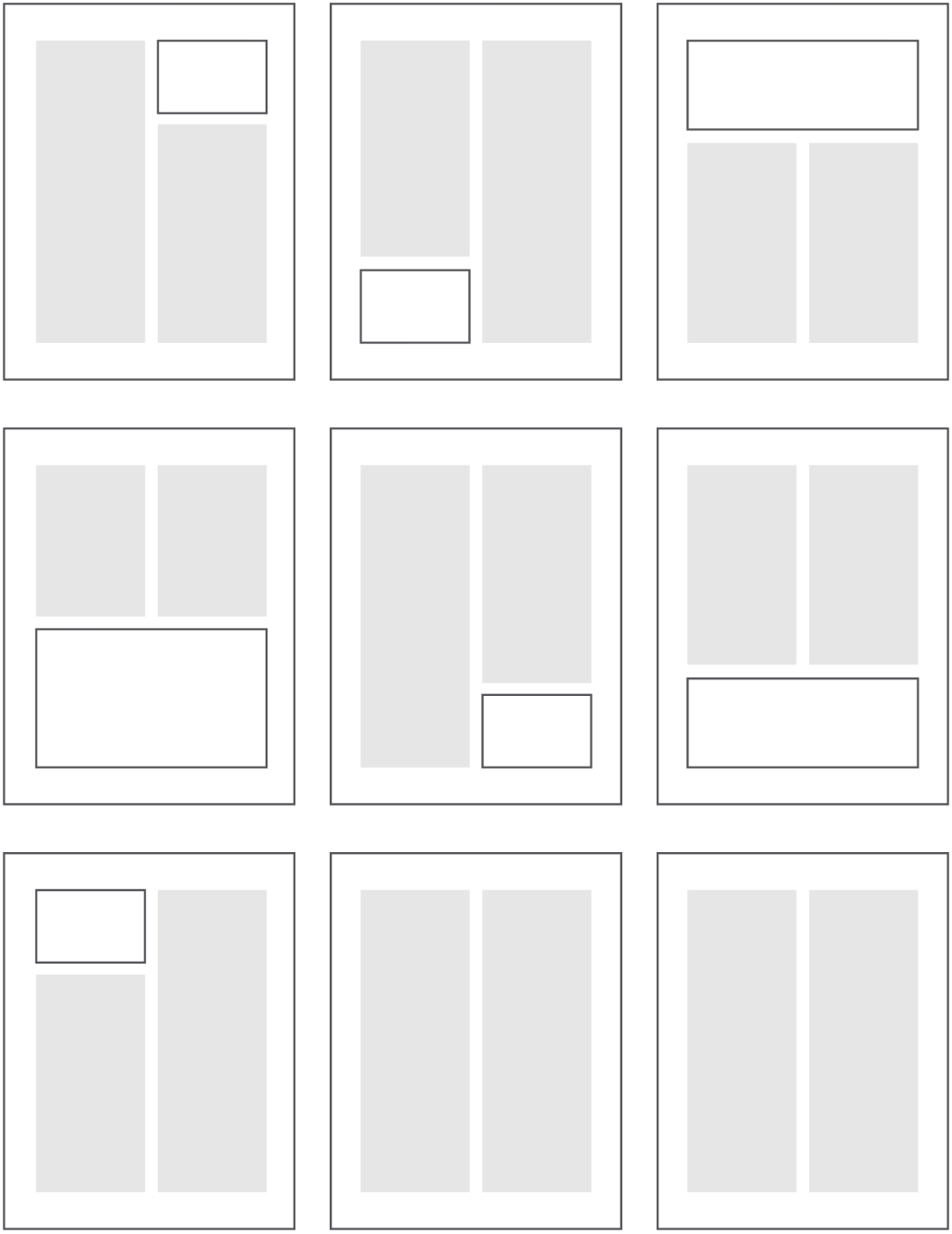
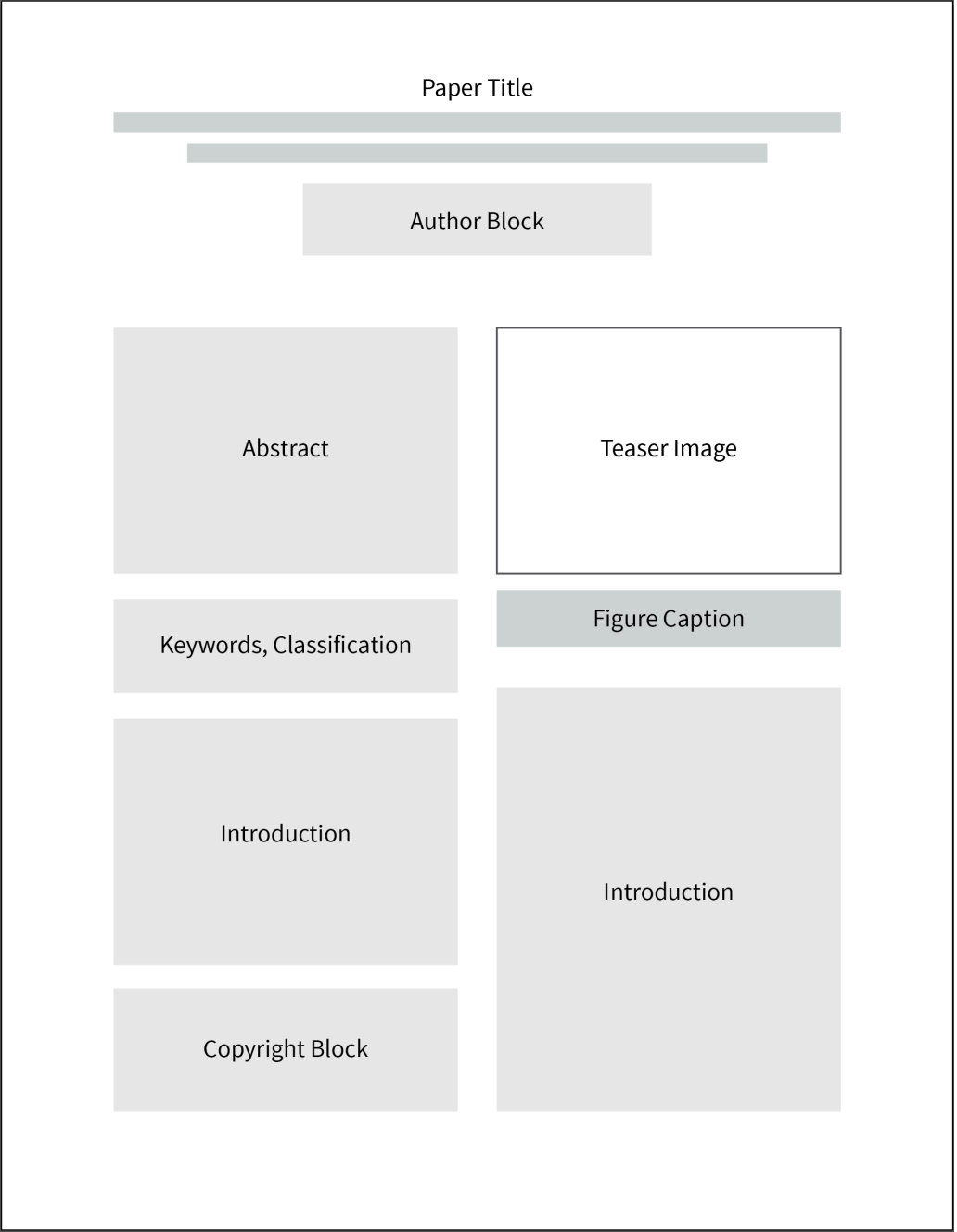
Aspect	Norm
Paper structure	APA (loosely)
Results of statistical analyses	APA (strictly)
Tables, figures	APA (very loosely)
Citations	Depends on the publisher (ACM, IEEE, etc.)
Formulas	AMS (loosely)
Style	APA (loosely), generally high standards in writing

APA Publication Manual: Print, Web; AMS Style Guide: Web¹



¹Sources: Left, Right

What does an HCI paper look like?



How is an HCI paper structured?

HCI papers commonly follow the structure below:

- » Abstract
- » Introduction
- » Related Work/Background
- » *Hypotheses (quant. empirical)*
- » *System/Design (design-based)*
- » Method
- » Results
- » Discussion
- » Conclusion
- » Acknowledgements
- » References
- » Appendices

*What is an abstract?*²

The abstract provides a brief but comprehensive summary of the contents of the paper. It gives readers an overview of the paper and helps them decide whether to read the full text. Usually *150 words* max.

The abstract usually includes (1–2 sentences each):

- | | |
|--------------------------------|-----------------|
| » Summary of literature review | » Methods used |
| » Problem investigated/RQs | » Study results |
| » Hypotheses | » Implications |

²APA

How do I choose a title?

There is no formula or requirement, but a few things to consider:

- » It should be as short as it can be, but not too broad.
 - » E.g., *Bodystorming Human-Robot Interactions*
- » A common format in HCI:
 - » Catchy headline/System name: Technical title
 - » E.g., *Pay attention!: Designing adaptive agents that monitor and improve user engagement*
 - » E.g., *Reading socially: Transforming the in-home reading experience with a learning-companion robot*

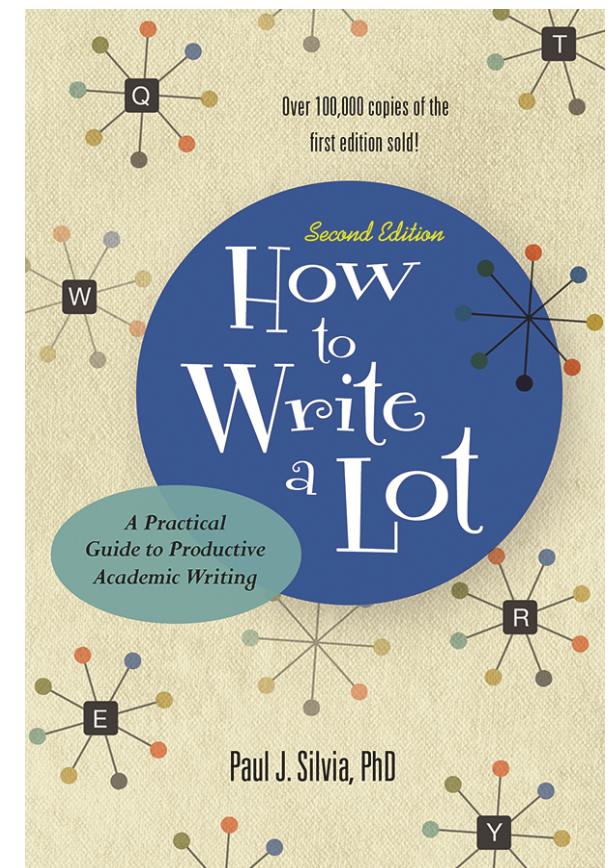
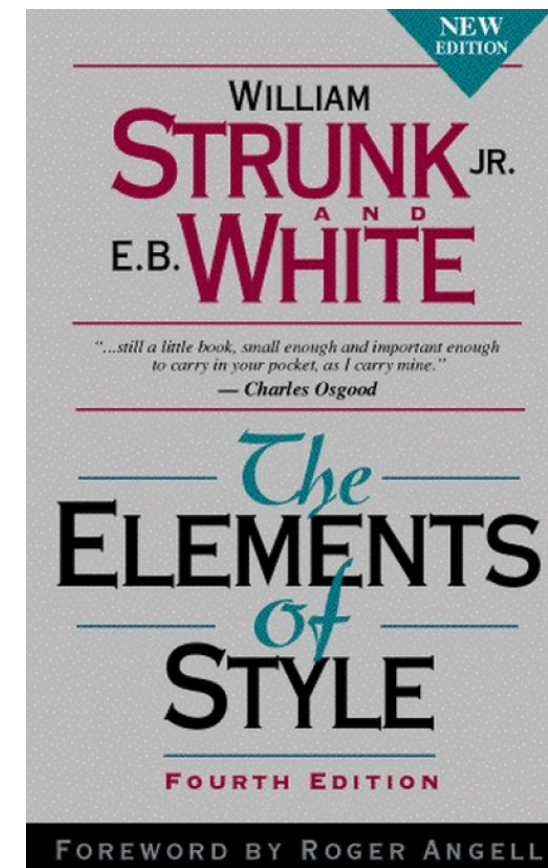
What are other things I should pay attention to?

1. Writing
2. Formatting
3. Presentation

Writing³

The HCI community pays more attention to writing than most other CS communities, so writing is very important, in particular:

1. Reporting as *storytelling*
2. Flow among parts
3. "Cut deadwood"
4. Avoid any deviation from rules (syntax, grammar, punctuation, etc.)



³Image sources: [Left](#), [Right](#)

Formatting⁴

For good *typography*, become familiar with *leading*, *tracking*, *kerning*, *widows*, *orphans*, *runts*, *rags*, *rivers*.



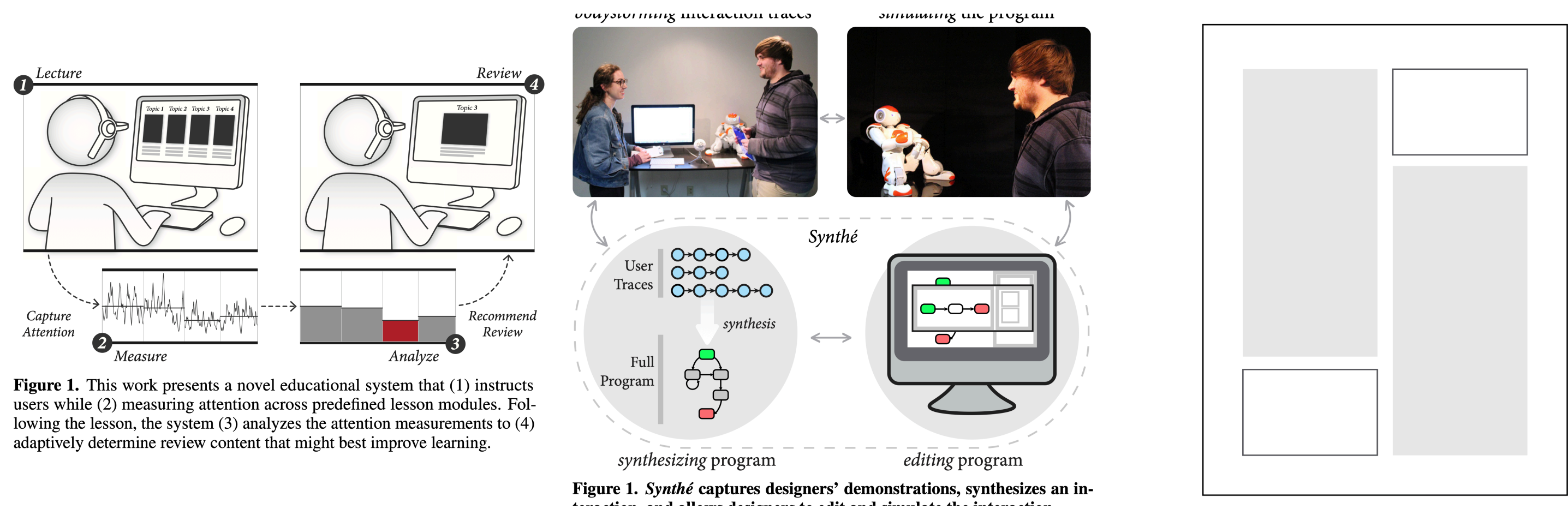
⁴ Image source: [Left](#), [Right](#)

kerning
tracking
leading
point size
typeface
justification
line width

what is typography?

Presentation⁵

The overall organization and visual appearance, using informative figures (e.g., a "teaser"), will improve accessibility and appeal.



⁵ **Left:** Szafir & Mutlu, 2014; **Center:** Porfirio et al., 2019

How do we report statistics?

Descriptive statistics: Distribution characteristics using summary statistics in text, tables, or graphs.

Inferential statistics: Test parameters and results in text or tables and highlighting of significance in graphs.

In *text*, APA guidelines are strictly followed; in *graphs*, you can be creative.

Descriptive statistics⁶

```
> describeBy(data$Guesses, list(data$Leakage,data$TBI))

Descriptive statistics by group
: Leakage
: HC
  vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
X1     1 291 3.87 1.91      4    3.68 1.48   1  13   12 1.08     1.95 0.11
-----
: No Leakage
: HC
  vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
X1     1 367 4.02 1.85      4    3.86 1.48   1  11   10 0.82     0.83 0.1
-----
: Leakage
: TBI
  vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
X1     1 282 3.92 2.24      4    3.63 1.48   1  17   16 2.11     7.83 0.13
-----
: No Leakage
: TBI
  vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
X1     1 353 4.37 2.46      4    4.05 1.48   1  19   18 1.55     4.24 0.13
```

The healthy controls guessed the item that the robot picked in 3.97 guesses (*SD*=1.91) when the robot gazed toward the item and in 4.02 guesses (*SD*=1.85) when the robot did not gaze toward it. Participants with TBI guessed the robot's pick in 3.92 guesses (*SD*=2.24) when the robot gazed toward it and in 4.37 guesses (*SD*=2.46) when the robot did not.

⁶Data from Mutlu et al., 2018, Social-cue perception

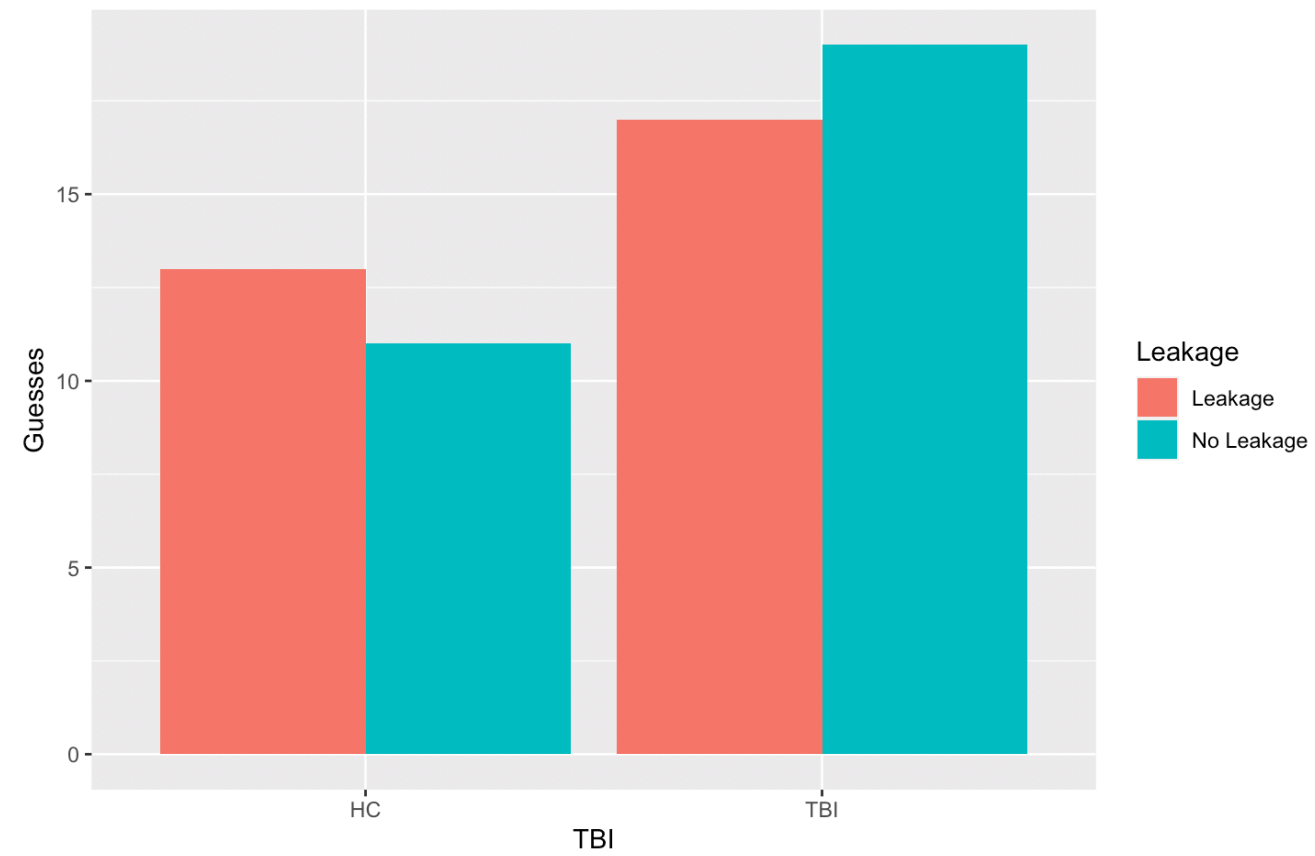
How do we deal with decimals?⁷

For numbers...	Round to...	SPSS	Report
Greater than 100	Whole number	1034.963	1035
10 - 100	1 decimal place	11.4378	11.4
0.10 - 10	2 decimal places	4.3682	4.37
0.001 - 0.10	3 decimal places	0.0352	0.035
Less than 0.001	As many digits as needed for non-zero	0.00038	0.0004

⁷Source

Descriptive statistics (visual)⁸

```
library(ggplot2)
ggplot(data, aes(fill=Leakage, y=Guesses, x=TBI)) +
  geom_bar(position="dodge", stat="identity")
```



⁸More information on using ggplot2

Inferential statistics⁹

```
> summary(aov(Guesses~(TBI*Leakage)+Error(ID/Leakage)+TBI,data=data))

Error: ID
      Df Sum Sq Mean Sq F value Pr(>F)
TBI      1  15.2  15.236   2.360  0.127
Leakage   1   4.0   4.012   0.621  0.432
TBI:Leakage 1   7.5   7.467   1.157  0.284
Residuals 142  916.6   6.455

Error: ID:Leakage
      Df Sum Sq Mean Sq F value Pr(>F)
Leakage  1  27.3  27.268   6.680 0.0107 *
TBI:Leakage 1   7.1   7.131   1.747 0.1884
Residuals 144  587.8   4.082

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: Within
      Df Sum Sq Mean Sq F value Pr(>F)
Residuals 1001  4325   4.321
```

A mixed-model analysis of variance (ANOVA) revealed a significant effect of the leakage cue, $F(1,144) = 6.68, p = .011$.

Participants correctly identified the robot’s pick on an average of 3.89 questions ($SD = 2.08$) when the robot displayed the gaze cue and 4.19 ($SD = 2.17$) when it did not.

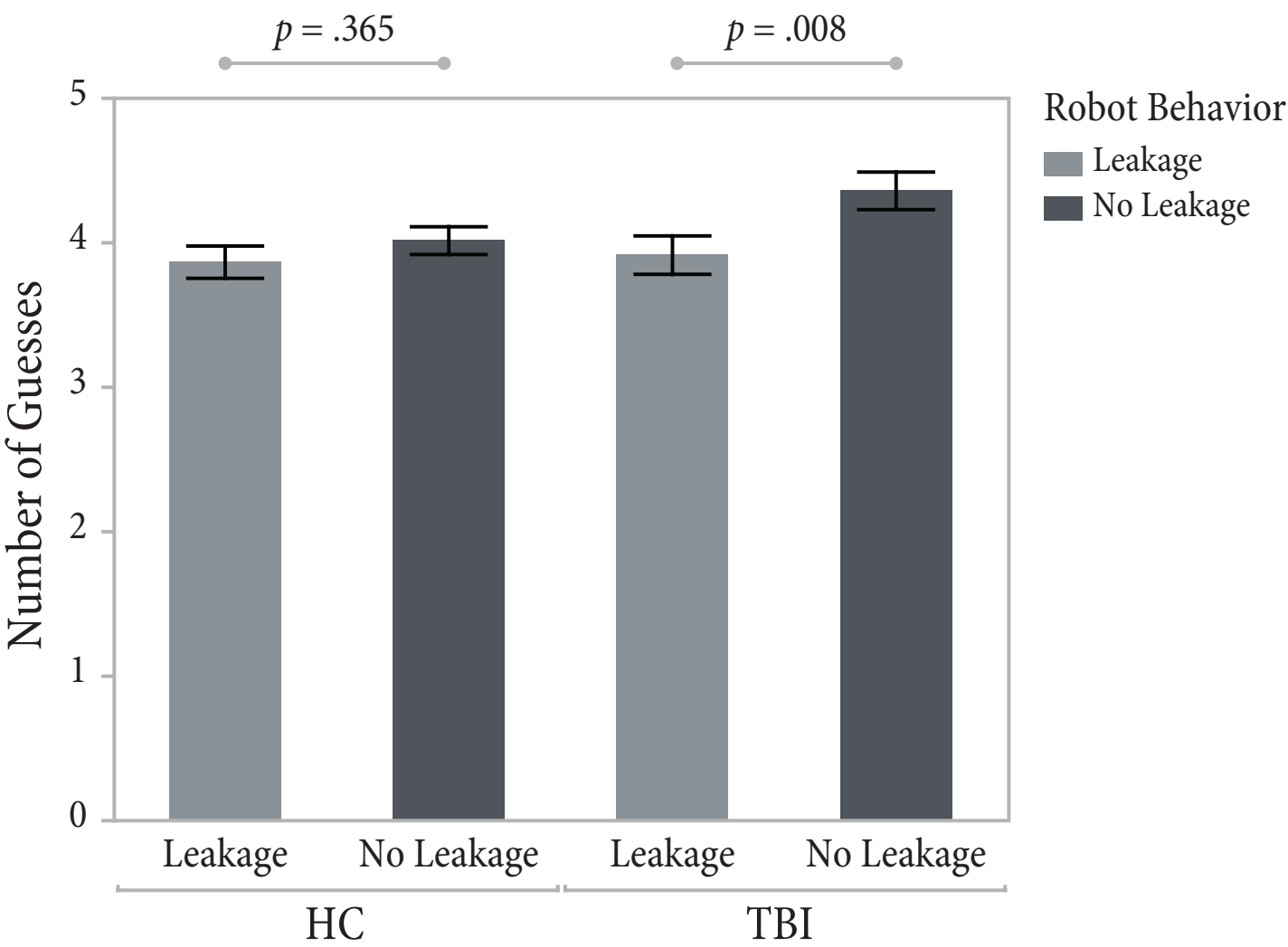
⁹Shown is a simplified model using data from Mutlu et al., 2018

How do I report different tests?⁷

Statistic	Example
Mean and standard deviation	$M = 3.45, SD = 1.21$
Mann-Whitney	$U = 67.5, p = .034, r = .38$
Wilcoxon signed-ranks	$Z = 4.21, p < .001$
Sign test	$Z = 3.47, p = .001$
t-test	$t(19) = 2.45, p = .031, d = 0.54$
ANOVA	$F(2, 1279) = 6.15, p = .002, \eta_p^2 = 0.010$
Pearson's correlation	$r(1282) = .13, p < .001$

⁷Source

Test results can also be mapped on graphs either manually (e.g., using Adobe Illustrator) or automatically using advanced scripting (e.g., `ggplot2`, `matplotlib`).



Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
    stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers. Add one geom function per layer.

geom(aes(x = cty, y = hwy, data = mpg, geom = "point")) Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5 x 5 file named "plot.png" in working directory. Matches file type to file extension.



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES
a = ggplot(economics, aes(date, unemploy))
b = ggplot(seals, aes(long, y = lat))

a = geom_blank() (useful for expanding limits)

b = geom_curve(aes(yend = lat + 1, xend = long + 1, curvature = 1), aes(x, yend, alpha, angle, color, curvature, linetype, size, linelength, size, vjust)

a = geom_path(aes(linetype = "butt", linejoin = "round", linelength = 1), aes(x, y, alpha, color, group, linetype, size, stroke)

a = geom_polygon(aes(group = group), aes(x, y, alpha, color, fill, group, linetype, size, stroke)

b = geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1), aes(x, y, alpha, color, fill, group, linetype, size, stroke)

a = geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900), aes(x, y, alpha, color, fill, group, linetype, size, stroke)

LINE SEGMENTS
common aesthetics: x, y, alpha, color, linetype, size

b = geom_abline(aes(intercept = 0, slope = 1))
b = geom_hline(aes(yintercept = lat))
b = geom_vline(aes(xintercept = long))

b = geom_segment(aes(yend = lat + 1, xend = long + 1))
b = geom_spoke(aes(angle = 1:155, radius = 1))

ONE VARIABLE continuous
c = ggplot(mpg, aes(hwy))
c2 = ggplot(mpg)

c = geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c = geom_density(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight

c = geom_dotplot()
x, y, alpha, color, fill

c = geom_freqpoly(x, y, alpha, color, group, linetype, size)

c = geom_histogram(bins = 10, x, y, alpha, color, fill, linetype, size, weight)

c2 = geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight

discrete
d = ggplot(mpg, aes(fit))

d = geom_bar()
x, y, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y
e = ggplot(mpg, aes(cty, hwy))

e = geom_label(aes(label = cty, nudges_x = 1, nudges_y = 1, check_overlap = TRUE), aes(x, y, label, alpha, angle, color, family, fontface, fontsize, linelength, size, vjust)

e = geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e = geom_point(x, y, alpha, color, fill, shape, size, stroke)

e = geom_quantile(x, y, alpha, color, group, linetype, size, weight)

e = geom_rug(sides = "b", x, y, alpha, color, linetype, size)

e = geom_smooth(method = lm, x, y, alpha, color, fill, group, linetype, size, weight)

e = geom_text(aes(label = cty), nudges_x = 1, nudges_y = 1, check_overlap = TRUE, aes(x, y, label, alpha, angle, color, family, fontface, fontsize, linelength, size, vjust)

discrete x, continuous y
f = ggplot(mpg, aes(class, hwy))

f = geom_col(x, y, alpha, color, fill, group, linetype, size)

f = geom_boxplot(x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight)

f = geom_dotplot(binsize = "y", stackdir = "center", x, y, alpha, color, fill, group, linetype, size, weight)

f = geom_violin(scale = "area", x, y, alpha, color, fill, group, linetype, size, weight)

discrete x, discrete y
g = ggplot(diamonds, aes(carat, color))

g = geom_count(x, y, alpha, color, fill, shape, size, stroke)

THREE VARIABLES
seals = with(seals, sqrt(delta_long^2 + delta_lat^2))
h = ggplot(seals, aes(long, lat))

h = geom_contour(aes(z = 1))
x, y, z, alpha, color, group, linetype, size, weight

h = geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

h = geom_tile(aes(fill = z), x, y, alpha, color, fill, linetype, size, width)

continuous bivariate distribution
h = ggplot(diamonds, aes(carat, price))

h = geom_bin2d(binswidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h = geom_density2d()
x, y, alpha, color, group, linetype, size

h = geom_hex()
x, y, alpha, color, fill, size

continuous function
i = ggplot(economics, aes(date, unemploy))

i = geom_area()
x, y, alpha, color, fill, linetype, size

i = geom_line()
x, y, alpha, color, group, linetype, size

i = geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error
df = data.frame(g = c("A", "B"), fit = 4.5, se = 1.2)
j = ggplot(df, aes(g, fit))

j = geom_crossbar(batten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j = geom_errorbar(x, y, ymax, ymin, alpha, color, group, linetype, size, width (also geom_errorbarh))

j = geom_linerange(x, y, ymax, ymin, alpha, color, group, linetype, size)

j = geom_pointrange(x, y, ymax, ymin, alpha, color, fill, group, linetype, shape, size)

maps
data = data.frame(murder = USArrests\$Murder, state = factor(1:nrow(USArrests)))
map = map_data("state")
k = ggplot(data, aes(fill = murder))

k = geom_map(aes(map_id = state), map = map)

expand_limits(x = map\$long, y = map\$lat)
map_id, alpha, color, fill, linetype, size

l = geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

l = geom_tile(aes(fill = z), x, y, alpha, color, fill, linetype, size, width)

Stats

An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, **geom_bar(stat = "count")** or by using a stat function, **stat_count(geom = "bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **__name__** syntax to map stat variables to aesthetics.

geom to use stat function geommap plays variable created by stat

c = stat_bin(binswidth = 1, origin = 10)
x, y | count, ...count, density, ...density

c = stat_count(width = 1) x, y | count, ...prop

c = stat_density2d(kernel = "gaussian")
x, y | count, ...density, ...scaled

e = stat_bin_2d(bins = 30, drop = T)
x, y, fill | count, ...density

e = stat_bin_hex(bins = 30) x, y, fill | count, ...density

e = stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | level

e = stat_ellipse(level = 0.95, segments = 51, type = "r")

l = stat_contour(aes(z = 1), x, y, x, order | level

l = stat_summary_hex(aes(x = z), bins = 30, fun = max)
x, y, fill | value

l = stat_summary_2d(aes(x = z), bins = 30, fun = mean)
x, y, fill | value

f = stat_boxplot(coef = 1.5) x, y | lower, middle, upper, width, ymin, ymax

f = stat_ydensity(kernel = "gaussian", scale = "area") x, y | density, ...count, ...scaled, ...width

e = stat_ecdf(n = 40) x, y | x, y

e = stat_quantile(quantiles = c(0.1, 0.5), formula = y ~ log(x), method = "rq") x, y | quantile

e = stat_smooth(method = "lm", formula = y ~ x, se = 1, level = 0.95) x, y | fit, ...se, ...ymin, ...ymax

ggplot() = stat_function(aes(x = 1:3), n = 99, fun = dnorm, args = list(sd = 0.5)) x, y | x, y

e = stat_identity(na.rm = TRUE)

ggplot() = stat_qq(aes(sample = 1:100), dist = qt, qnorm = FALSE) sample, x, y | sample, ...theoretical

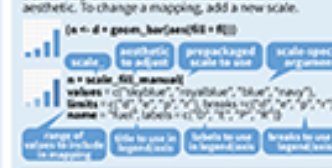
e = stat_sum(x, y, size | ...prop

h = stat_summary_bin(fun.data = "mean_cl_boot")

e = stat_unique()

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



GENERAL PURPOSE SCALES
Use with most aesthetics

scale_ "continuous" - map cont values to visual ones

scale_ "discrete" - map discrete values to visual ones

scale_ "identity" - use data values as visual ones

scale_ "manual" values = c() - map discrete values to manually chosen visual ones

scale_ "date" date, labels = "mm/dd", date_breaks = "2 weeks", ...

scale_ "datetime" - treat data x values as date times. Use same arguments as scale_x_date(). See http://time for label formats.

X & Y LOCATION SCALES
Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale

scale_x_reverse() - Reverse direction of x axis

scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

n = d = geom_bar(aes(fill = fit))

n = scale_fill_brewer(palette = "Blues")
For palette choices: RColorBrewer::display.brewer.all()

n = scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

COLOR AND FILL SCALES (CONTINUOUS)

e = c = geom_dotplot(aes(fill = x...))

e = scale_fill_distiller(palette = "Blues")

e = scale_fill_gradient(low = "red", high = "yellow")

e = scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)

e = scale_fill_gradientn(colors = tops.colors(10))
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p = e = geom_point(aes(shape = fit, size = cty))

p = scale_shape() = scale_size()

p = scale_shape_manual(values = c(3, 7))
x, y, size, shape = 1:6, 1:6, 1:6, 1:6, 1:6, 1:6
p = scale_radius(range = c(1, 6))
p = scale_size_area(max_size = 6)

Coordinate Systems

r = d = geom_bar()

r = coord_cartesian(ylim = c(0, 5))
xlim, ylim

r = coord_fixed(ratio = 1.0)
xlim, ylim

r = coord_flip()
xlim, ylim

r = coord_polar(theta = "x", direction = 1)
theta, start, direction

r = coord_trans(xlim = "right")
x, y, alpha, color, fill, group, linetype, size, weight

r = coord_quickmap()
x, y, alpha, color, fill, group, linetype, size, weight

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

r = coord_map(projection = "ortho", crs = "+proj=ortho", crs_args = c("+proj=ortho", "+datum=WGS84", "+units=m", "+no_defs"))

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t = ggplot(mpg, aes(cty, hwy)) + geom_point()

t = facet_grid(~ fit)

t = facet_grid(year ~ .)

t = facet_grid(year ~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)

t = facet_wrap(~ fit)



Open Q&A

1. What courses should I take next to build stronger HCI skills?
2. How can I turn my class project into a publishable research paper?
3. What are good ways to get involved in ongoing HCI research on campus?
4. How can I integrate HCI methods into my own research area or discipline?
5. What skills from this course should I develop further to work effectively in HCI?
6. If I want to pursue HCI for my thesis or dissertation, what should I plan for next?
7. ...