

Human-Computer Interaction

# Reporting & Writing HCI Papers

Professor Bilge Mutlu

# Today's Agenda

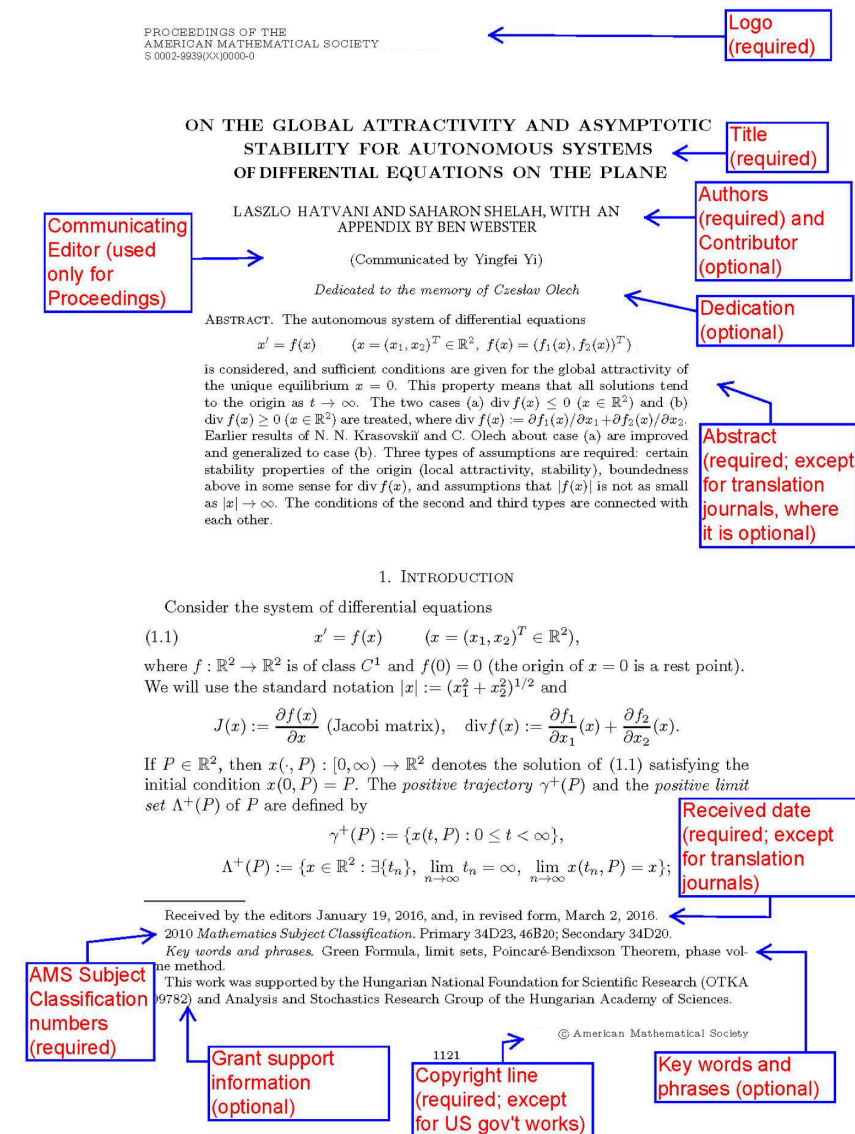
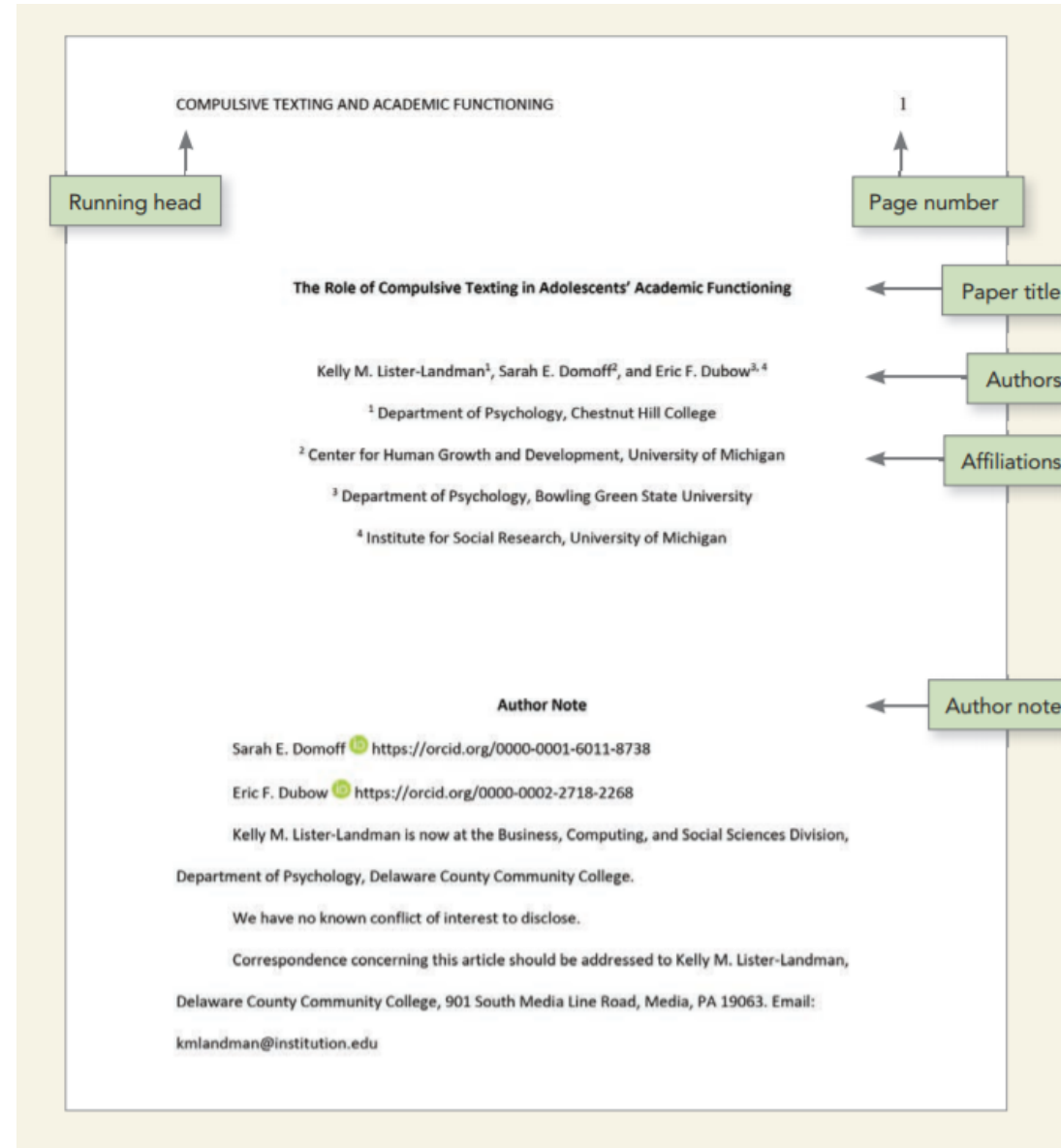
» Overview: *Reporting Statistics, Writing*

## What are reporting norms in HCI research?

Because HCI is a rather eclectic field, the reporting norms are adopted from different fields, roughly as follows:

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

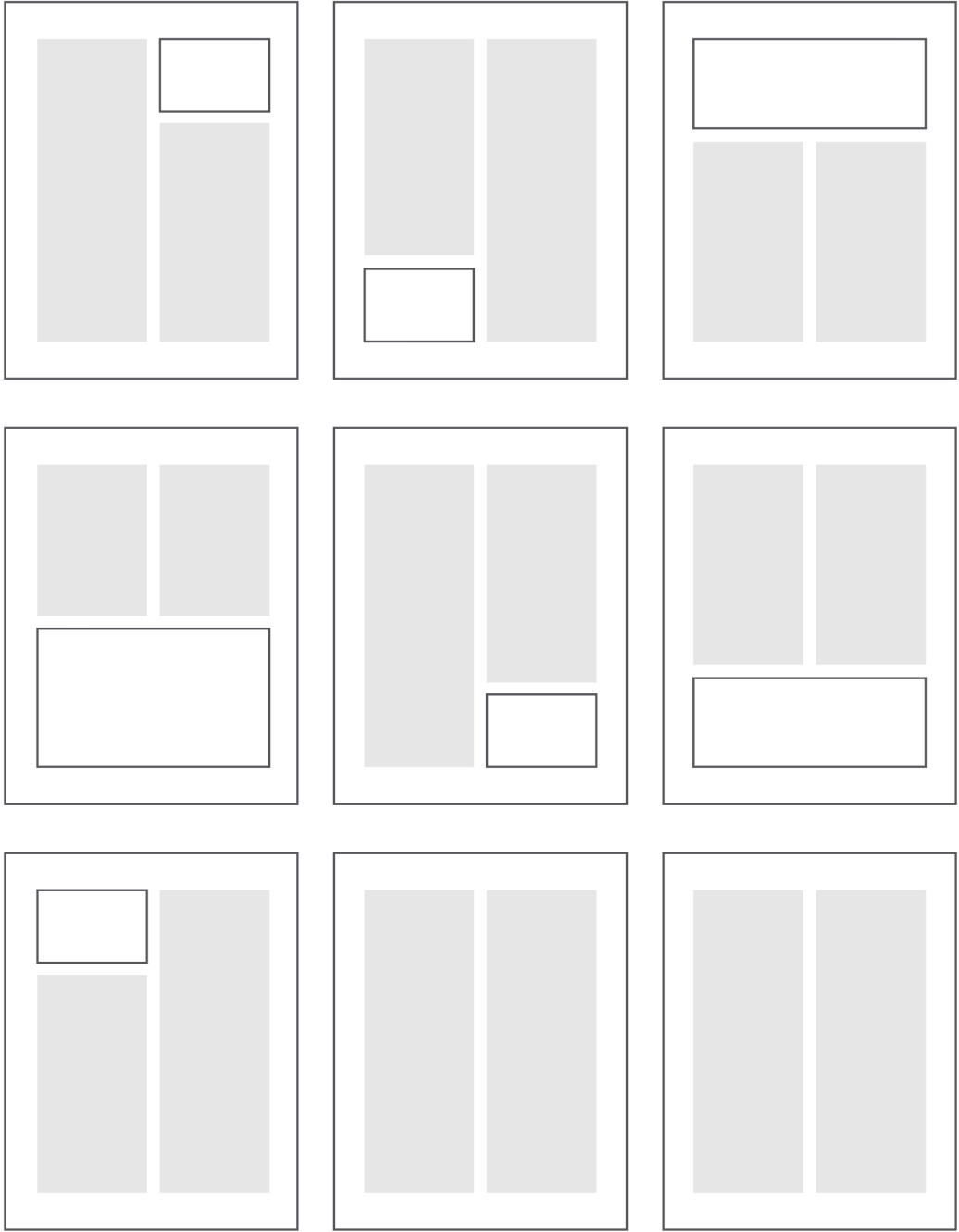
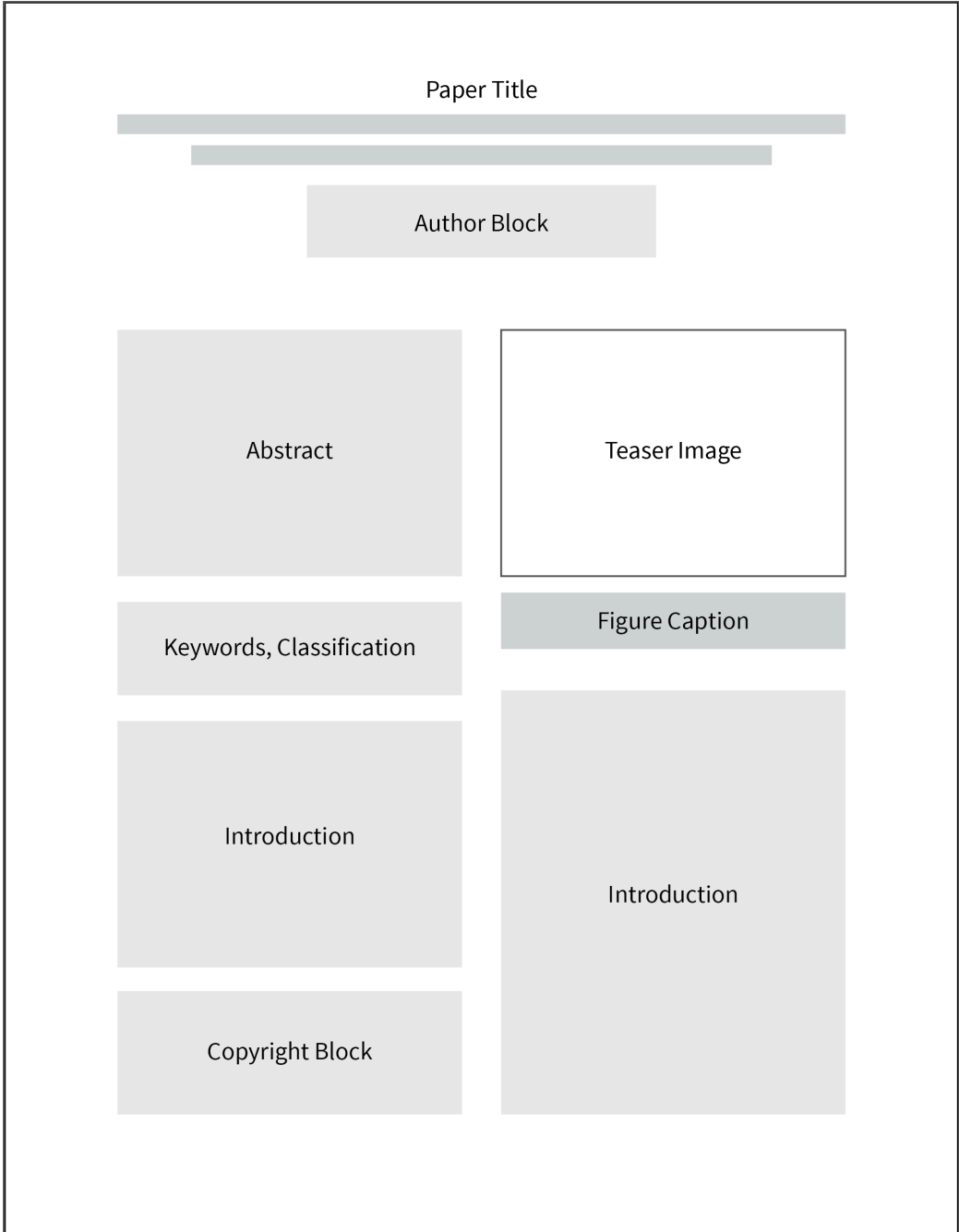
# APA Publication Manual: Print, Web; AMS Style Guide: Web<sup>1</sup>



<sup>1</sup>Sources: 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?*<sup>2</sup>

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
- » Problem investigated/RQs
- » Hypotheses
- » Methods used
- » Study results
- » Implications

---

<sup>2</sup>APA

## *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*

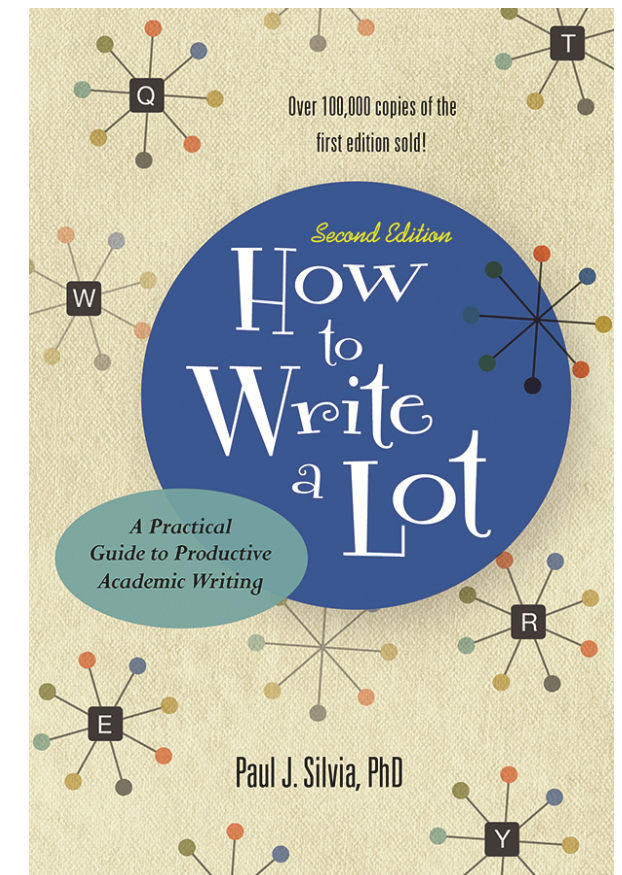
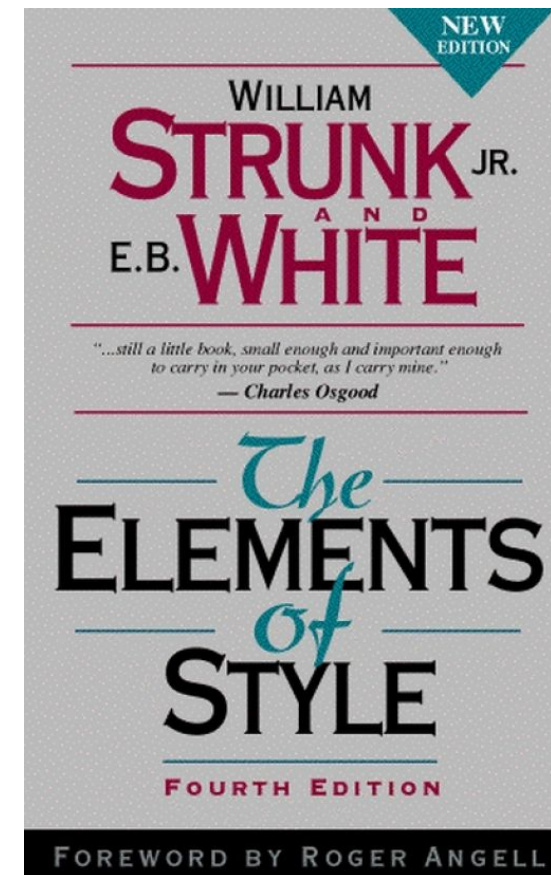
*What are other things I should pay attention to?*

1. Writing
2. Formatting
3. Presentation

## Writing<sup>3</sup>

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.)



<sup>3</sup>Image sources: [Left](#), [Right](#)



## Formatting<sup>4</sup>

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



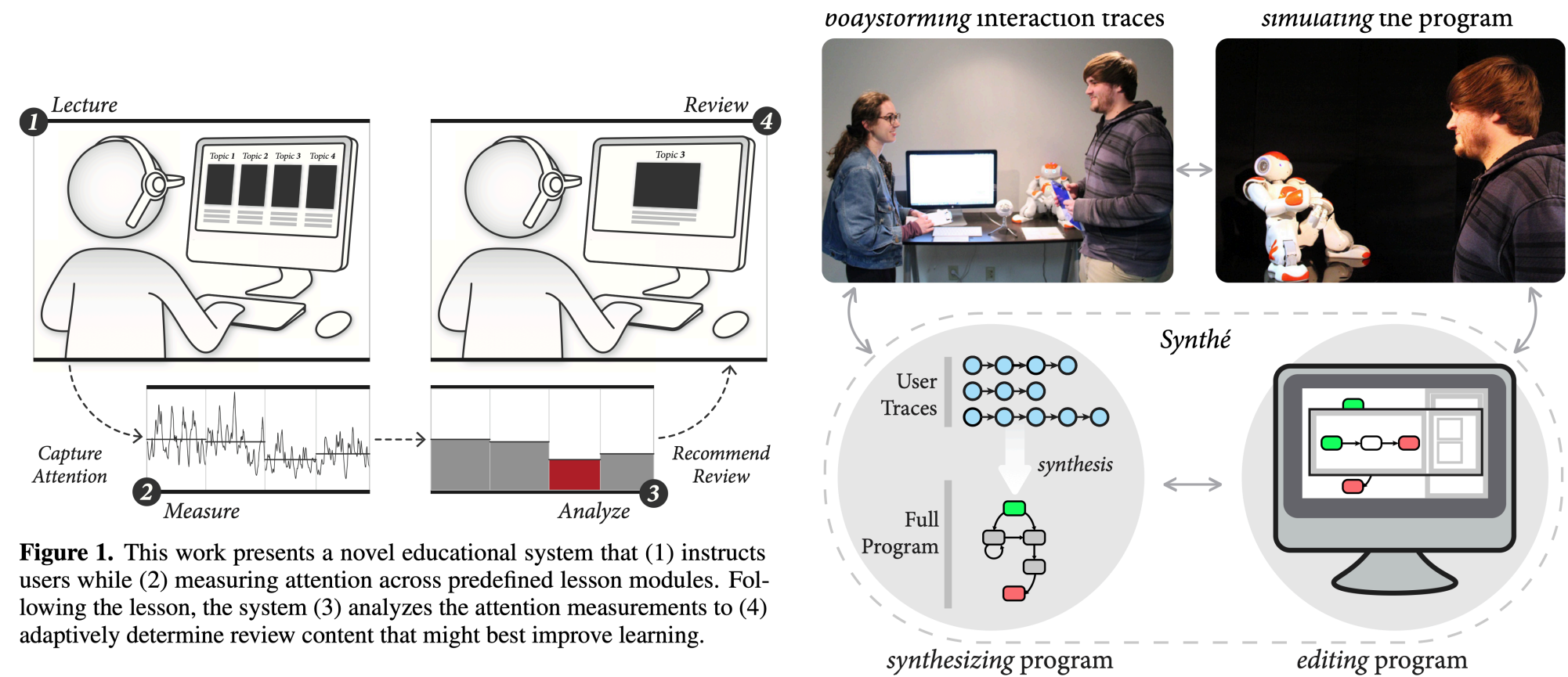
<sup>4</sup>Image source: [Left](#), [Right](#)

kerning  
tracking  
leading  
point size  
typeface  
justification  
line width

what is typography?

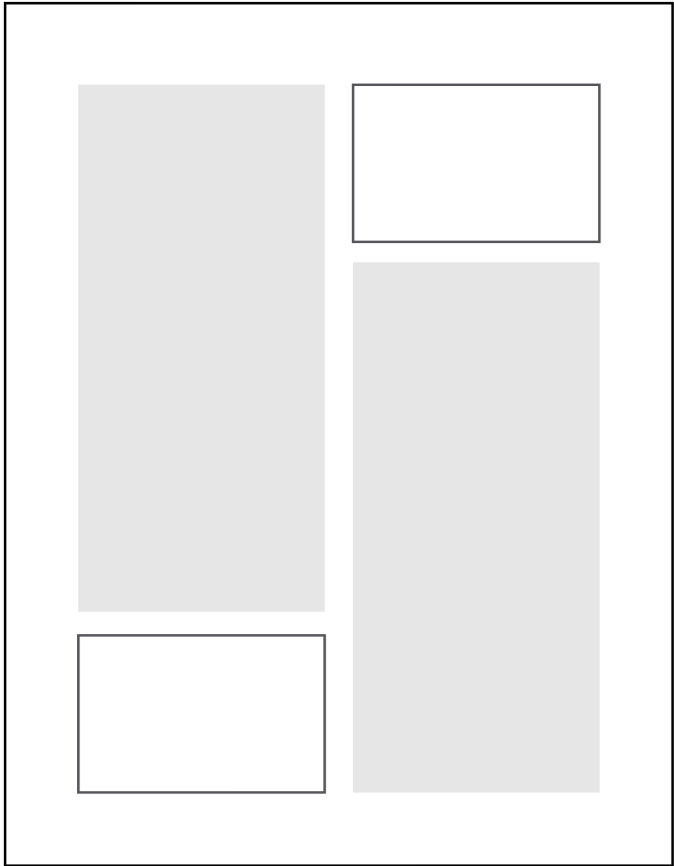
# Presentation<sup>5</sup>

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



**Figure 1.** This work presents a novel educational system that (1) instructs users while (2) measuring attention across predefined lesson modules. Following the lesson, the system (3) analyzes the attention measurements to (4) adaptively determine review content that might best improve learning.

**Figure 1.** Synthé captures designers' demonstrations, synthesizes an interaction and allows designers to edit and simulate the interaction



<sup>5</sup> **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 statistics<sup>6</sup>

```
> 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.

---

<sup>6</sup>Data from [Mutlu et al., 2018, Social-cue perception](#)

*How do we deal with decimals?<sup>7</sup>*

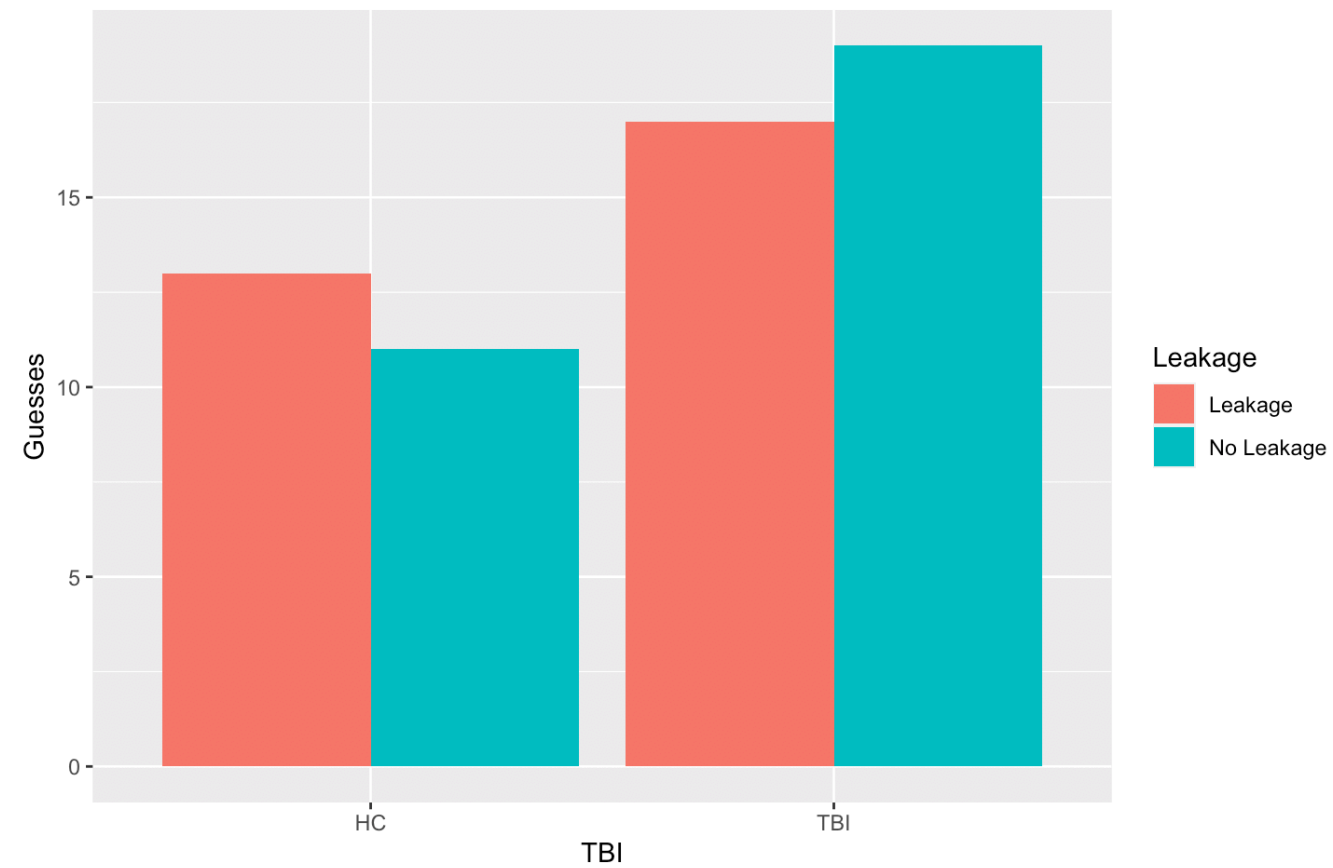
<b>For numbers...</b>	<b>Round to...</b>	<b>SPSS</b>	<b>Report</b>
Greater than 100	Whole number	1034.963	1035
10 - 100	1 decimal place	11.4378	11.4
0.10 - 10	2 decimal places	4.3682	4.37
0.001 - 0.10	3 decimal places	0.0352	0.035
Less than 0.001	As many digits as needed for non-zero	0.00038	0.0004

---

<sup>7</sup>Source

## Descriptive statistics (visual)<sup>8</sup>

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



<sup>8</sup>[More information on using ggplot2](#)

## Inferential statistics<sup>9</sup>

```
> 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.

---

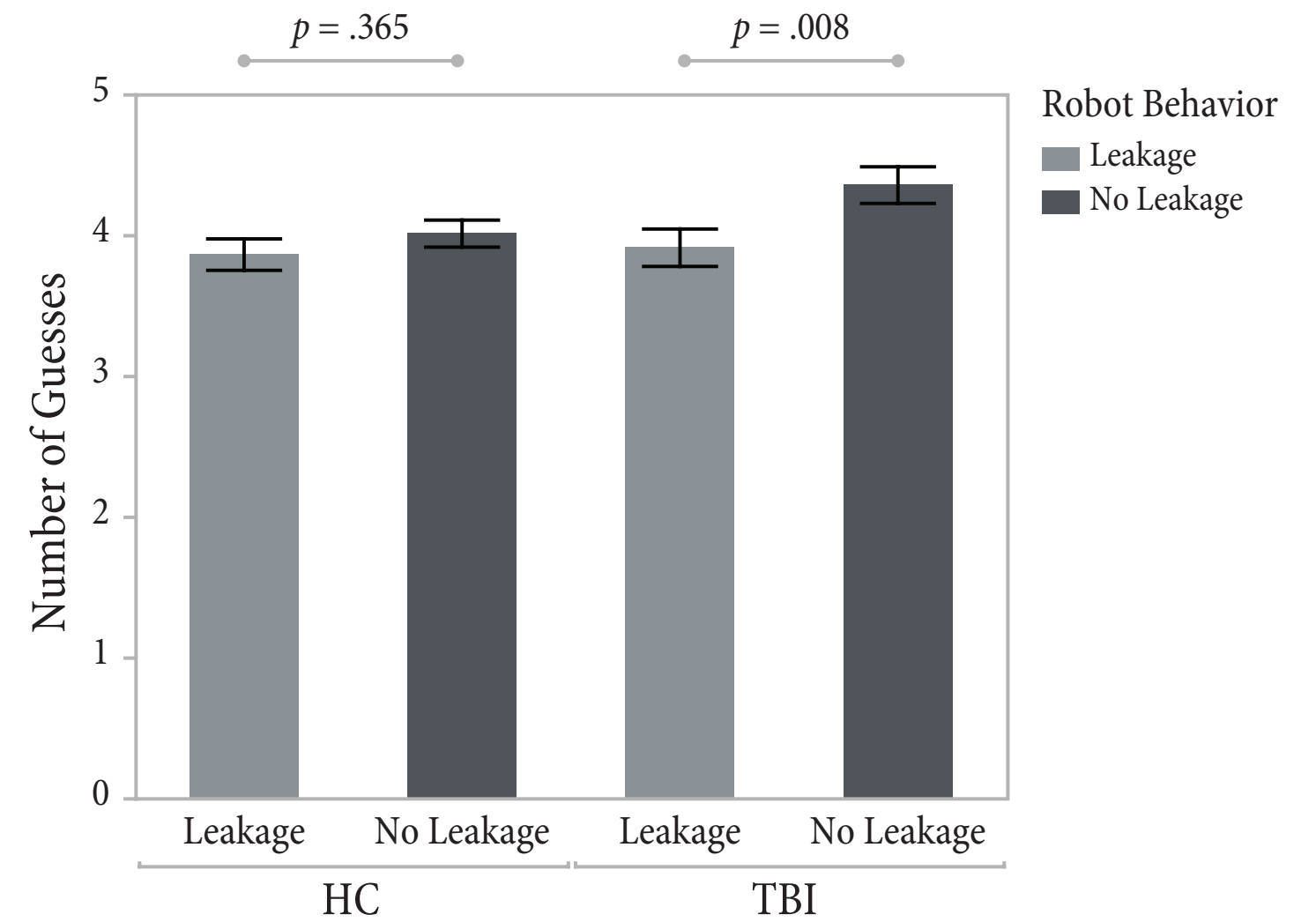
<sup>9</sup> Shown is a simplified model using data from Mutlu et al., 2018

How do I report different tests?<sup>7</sup>

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	$r(1282) = .13, p < .001$

<sup>7</sup>Source

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(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(mpg, aes(x, hwy))

a + geom_blank()
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(linetype = "butt", linewidth = "round",
  linemitre = 1)
b + geom_polygon(aes(group = group),
  aes(x, y, alpha, color, fill, group, linetype, size))

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

a + geom_ribbon(aes(xmin = unemploy - 900,
  xmax = unemploy + 900), aes(xmin, xmax,
  alpha, color, fill, group, linetype, size))
```

```
LINE SEGMENTS
common aesthetics: x, y, alpha, color, linetype, size
b = geom_abline(aes(intercept = 0, slope = 1))
b = geom_hline(aes(intercept = lat))
b = geom_vline(aes(xintercept = long))

b = geom_segment(aes(yend = lat + 1, xend = long + 1))
b = geom_spoke(aes(angle = 1.1155, radius = 1))
```

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

c + geom_area(stat = "bin")
c + geom_density(kernel = "gaussian")
c + geom_dotplot()
c + geom_freqpoly()
c + geom_histogram(binwidth = 1)
c2 + geom_qq(aes(sample = hwy))
```

```
discrete
d = ggplot(mpg, aes(f))
d + geom_bar()
```

```
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)
e + geom_jitter(height = 2, width = 2)
e + geom_point()
e + geom_quantile()
e + geom_rug(sides = "tl")
e + geom_smooth(method = lm)
e + geom_text(aes(label = cty), nudges_x = 1,
  nudges_y = 1, check_overlap = TRUE)
```

```
discrete x, discrete y
f = ggplot(mpg, aes(f, hwy))
f + geom_boxplot()
f + geom_violin(scale = "area")
f + geom_violin(scale = "width")
```

```
THREE VARIABLES
g = ggplot(diamonds, aes(carat, price))
g + geom_raster(aes(fill = z))
g + geom_tile(aes(fill = z))
```

```
continuous bivariate distribution
h = ggplot(diamonds, aes(carat, price))
h + geom_bin2d(binwidth = c(0.25, 100))
h + geom_density2d()
h + geom_hex()
```

```
continuous function
i = ggplot(economics, aes(date, unemploy))
i + geom_area()
i + geom_line()
i + geom_step(direction = "hv")
```

```
visualizing error
j = data.frame(ggp = c("A", "B"), fit = 4.5, se = 1.2)
j + geom_crossbar(latten = 2)
j + geom_errorbar()
j + geom_linerange()
j + geom_pointrange()
```

```
maps
k = ggplot(murder, aes(state, murder))
k + geom_map(map_id = "state", map = map)
k + geom_sf(aes(long, y = murder))
```

```
maps
l = ggplot(murder, aes(state, murder))
l + geom_sf(aes(long, y = murder))
```

```
maps
m = ggplot(murder, aes(state, murder))
m + geom_sf(aes(long, y = murder))
```

```
maps
n = ggplot(murder, aes(state, murder))
n + geom_sf(aes(long, y = murder))
```

```
maps
o = ggplot(murder, aes(state, murder))
o + geom_sf(aes(long, y = murder))
```

```
maps
p = ggplot(murder, aes(state, murder))
p + geom_sf(aes(long, y = murder))
```

```
continuous bivariate distribution
h = ggplot(diamonds, aes(carat, price))
h + geom_bin2d(binwidth = c(0.25, 100))
h + geom_density2d()
h + geom_hex()
```

```
continuous function
i = ggplot(economics, aes(date, unemploy))
i + geom_area()
i + geom_line()
i + geom_step(direction = "hv")
```

```
visualizing error
j = data.frame(ggp = c("A", "B"), fit = 4.5, se = 1.2)
j + geom_crossbar(latten = 2)
j + geom_errorbar()
j + geom_linerange()
j + geom_pointrange()
```

```
maps
k = ggplot(murder, aes(state, murder))
k + geom_map(map_id = "state", map = map)
k + geom_sf(aes(long, y = murder))
```

```
maps
l = ggplot(murder, aes(state, murder))
l + geom_sf(aes(long, y = murder))
```

```
maps
m = ggplot(murder, aes(state, murder))
m + geom_sf(aes(long, y = murder))
```

```
maps
n = ggplot(murder, aes(state, murder))
n + geom_sf(aes(long, y = murder))
```

```
maps
o = ggplot(murder, aes(state, murder))
o + geom_sf(aes(long, y = murder))
```

```
maps
p = ggplot(murder, aes(state, murder))
p + geom_sf(aes(long, y = murder))
```

## 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.

```
c = stat_bin(binwidth = 1, origin = 0)
x, y, ..count.., ..ncount.., ..density..
e = stat_bin_hex(binwidth = 1, origin = 0)
x, y, ..count.., ..ncount.., ..density..
e = stat_density2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
e = stat_bin_2d(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_bin_hex(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_density_2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
e = stat_bin_2d(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_bin_hex(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_density_2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
```

```
c = stat_bin(binwidth = 1, origin = 0)
x, y, ..count.., ..ncount.., ..density..
e = stat_bin_hex(binwidth = 1, origin = 0)
x, y, ..count.., ..ncount.., ..density..
e = stat_density2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
e = stat_bin_2d(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_bin_hex(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_density_2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
e = stat_bin_2d(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_bin_hex(binwidth = 1, drop = T)
x, y, ..count.., ..density..
e = stat_density_2d(kernel = "gaussian")
x, y, ..count.., ..density.., ..scaled..
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```

```
ggplot() + stat_function(aes(x = 1:3, y = 1:9), fun =
  dnorm, args = list(mean = 0, sd = 1))
ggplot() + stat_qq(aes(sample = 1:100), dist = qt,
  quantiles = seq(0.05, 0.95, by = 0.05))
ggplot() + stat_smooth(method = "lm", formula = y ~ x, se =
  level = 0.95)
ggplot() + stat_summary(fun.data = "mean_cl_boot")
ggplot() + stat_summary_bin(fun.y = "mean", geom = "bar")
ggplot() + stat_unique()
```



## Scales

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

```
(a = d + geom_bar(aes(fill = fill)))
scale_fill_manual(values = c("red", "blue", "green"))
scale_fill_discrete(values = c("red", "blue", "green"))
scale_fill_continuous(values = c("red", "blue", "green"))
scale_fill_gradient(low = "red", high = "blue")
scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 2)
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```

```
scale_x_log10()
scale_x_reverse()
scale_x_sqrt()
scale_y_log10()
scale_y_reverse()
scale_y_sqrt()
```



## Coordinate Systems

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

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```

```
r = d + geom_bar()
r + coord_cartesian(clim = c(0, 50))
r + coord_fixed(ratio = 1)
r + coord_flip()
r + coord_map(proj = "merc", xlim = c(0, 100), ylim = c(0, 100))
r + coord_polar(theta = "x", direction = 1)
r + coord_spatial4()
r + coord_quickmap()
```



## Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables. 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(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

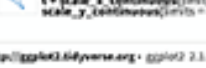
```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```

```
t = ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(~ f)
t + facet_grid(year ~ .)
t + facet_grid(year ~ f)
t + facet_wrap(~ f)
t + coord_quickmap()
```





*Questions?*