Summarizing Data

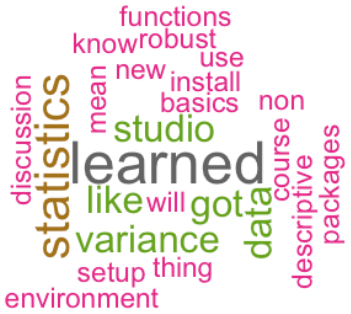
Computational Mathematics and Statistics

Jason Bryer, Ph.D.

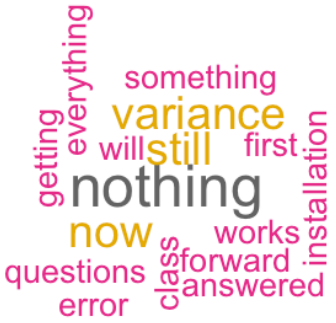
September 3, 2024

One Minute Paper Results

What was the most important thing you learned during this class?

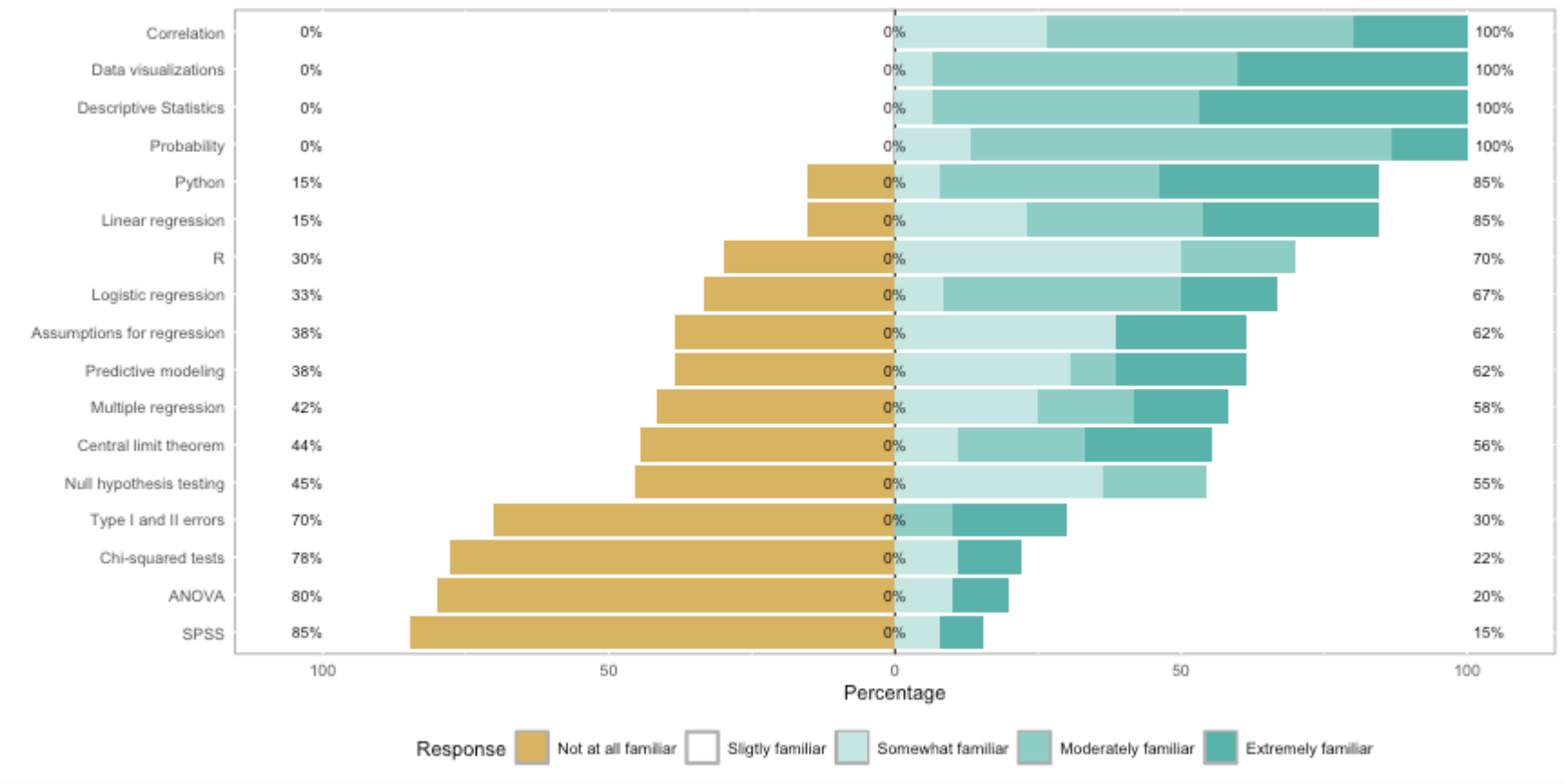


What important question remains unanswered for you?



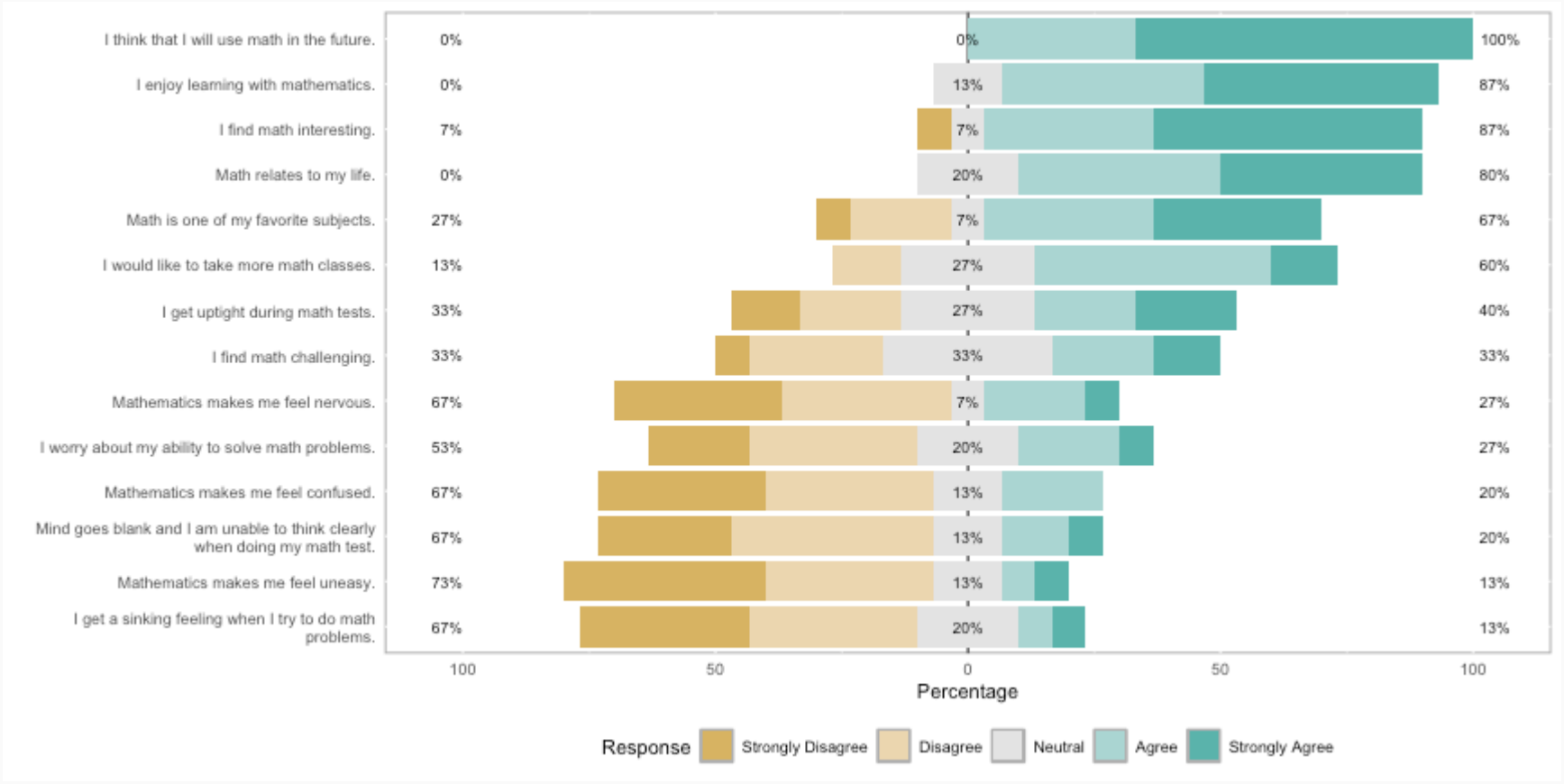
Familiarity with Statistical Topics

```
likert(stats.results) %>% plot(center = 2.5)
```



Math Anxiety Survey Scale

```
likert(mass.results) %>% plot()
```



About `legosets`

To install the `brickset` package:

```
remotes::install_github('jbryer/brickset')
```

To load the load the `legosets` dataset.

```
data('legosets', package = 'brickset')
```

The `legosets` data has 19409 observations of 36 variables.

```
names(legosets)
```

```
##  [1] "setID"                "number"               "numberVariant"
##  [4] "name"                 "year"                 "theme"
##  [7] "themeGroup"           "subtheme"             "category"
## [10] "released"             "pieces"               "minifigs"
## [13] "bricksetURL"          "rating"               "reviewCount"
## [16] "packagingType"        "availability"         "agerange_min"
## [19] "thumbnailURL"         "imageURL"             "US_retailPrice"
## [22] "US_dateFirstAvailable" "US_dateLastAvailable" "UK_retailPrice"
## [25] "UK_dateFirstAvailable" "UK_dateLastAvailable" "CA_retailPrice"
## [28] "CA_dateFirstAvailable" "CA_dateLastAvailable" "DE_retailPrice"
## [31] "DE_dateFirstAvailable" "DE_dateLastAvailable" "height"
## [34] "width"                "depth"                "weight"
```

Structure (`str`)

```
str(legosets)
```

```
## 'data.frame':    19409 obs. of  36 variables:
##  $ setID                : int  7693 7695 7697 7698 25534 7418 7419 6020 22704 7421 ...
##  $ number               : chr  "1" "2" "3" "4" ...
##  $ numberVariant        : int  8 8 6 4 6 1 1 1 3 4 ...
##  $ name                 : chr  "Small house set" "Medium house set" "Medium house set" "Large house set" ...
##  $ year                 : int  1970 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
##  $ theme                : chr  "Minitalia" "Minitalia" "Minitalia" "Minitalia" ...
##  $ themeGroup           : chr  "Vintage" "Vintage" "Vintage" "Vintage" ...
##  $ subtheme             : chr  NA NA NA NA ...
##  $ category             : chr  "Normal" "Normal" "Normal" "Normal" ...
##  $ released             : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ pieces               : int  67 109 158 233 NA 1 1 60 65 NA ...
##  $ minifigs             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ bricksetURL          : chr  "https://brickset.com/sets/1-8" "https://brickset.com/sets/2-8" "https://brickset.com/sets/3-6" "https://brickset.com/sets/4-4" ...
##  $ rating               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ reviewCount          : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ packagingType        : chr  "{Not specified}" "{Not specified}" "{Not specified}" "{Not specified}" ...
##  $ availability         : chr  "{Not specified}" "{Not specified}" "{Not specified}" "{Not specified}" ...
##  $ agerange_min         : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ thumbnailURL         : chr  "https://images.brickset.com/sets/small/1-8.jpg" "https://images.brickset.com/sets/small/2-8.jpg" "https://images.brickset.com/sets/small/3-6.jpg" "https:
##  $ imageURL             : chr  "https://images.brickset.com/sets/images/1-8.jpg" "https://images.brickset.com/sets/images/2-8.jpg" "https://images.brickset.com/sets/images/3-6.jpg" "ht
##  $ US_retailPrice       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ US_dateFirstAvailable: Date, format: NA NA ...
##  $ US_dateLastAvailable : Date, format: NA NA ...
##  $ UK_retailPrice       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ UK_dateFirstAvailable: Date, format: NA NA ...
##  $ UK_dateLastAvailable : Date, format: NA NA ...
##  $ CA_retailPrice       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CA_dateFirstAvailable: Date, format: NA NA ...
##  $ CA_dateLastAvailable : Date, format: NA NA ...
##  $ DE_retailPrice       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ DE_dateFirstAvailable: Date, format: NA NA ...
##  $ DE_dateLastAvailable : Date, format: NA NA ...
##  $ height               : num  NA NA NA NA NA ...
##  $ width                : num  NA NA NA NA NA ...
```

RStudio Eenvironment tab can help

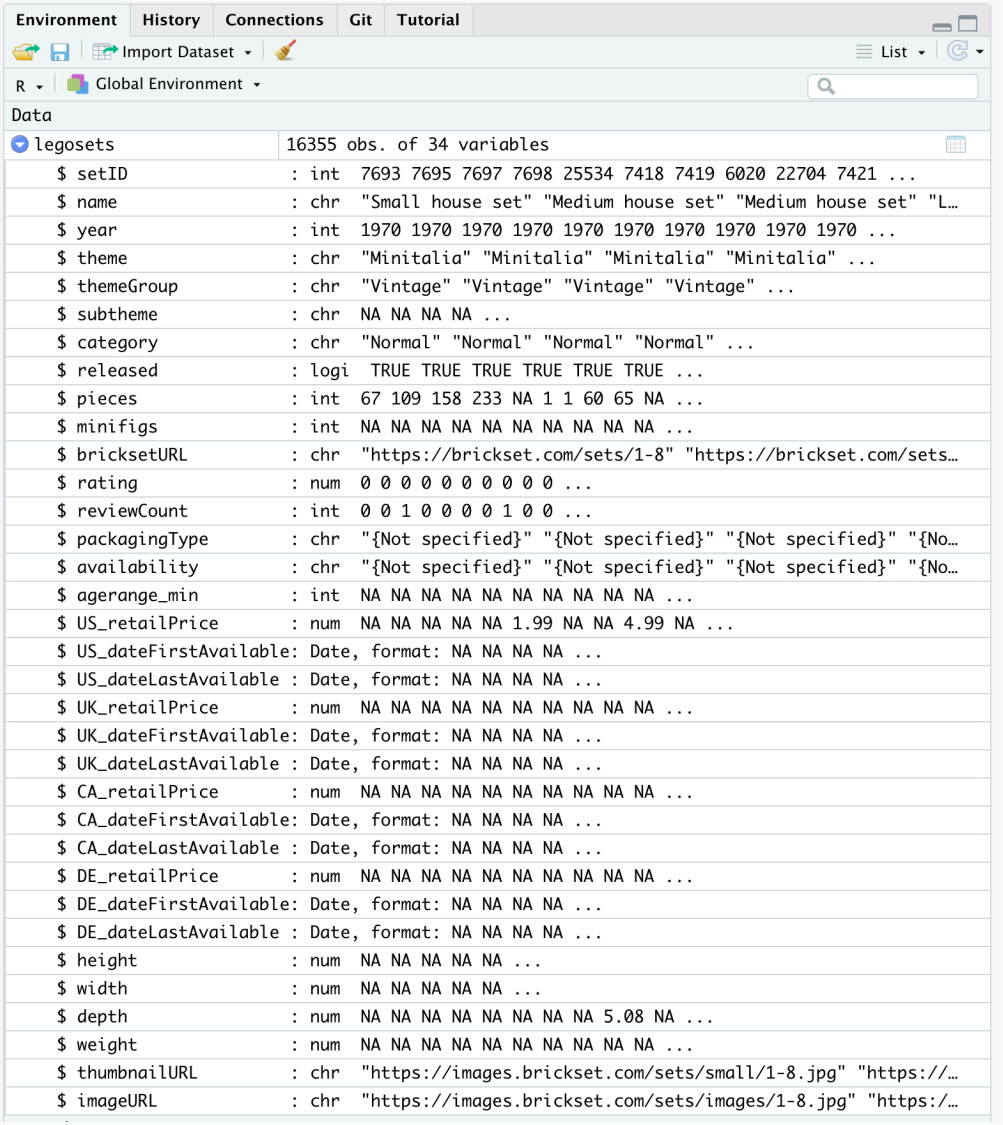


Table View

Show 10 entries

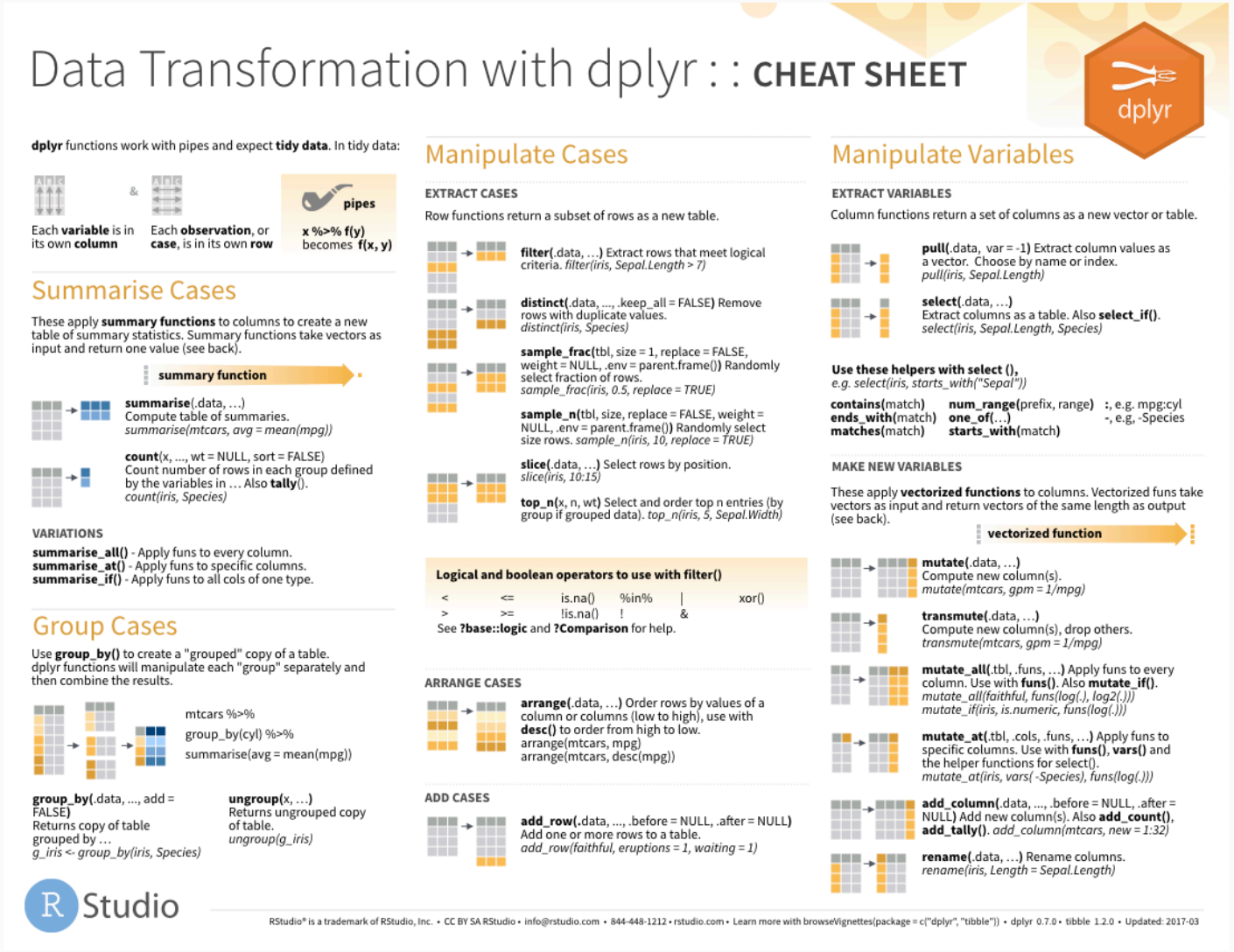
Search:

	setID	name	year	theme	themeGroup	category	US_retailPrice	pieces	minifigs	rating
1	1137	The Meadowsweet's' Home	1999	Duplo	Pre-school	Normal		41	4	0
2	31564	Spider-Man & Doctor Octopus Mech Battle	2021	Marvel Super Heroes	Licensed	Normal	19.99	305	2	3.4
3	6832	Garbage Truck	2009	Duplo	Pre-school	Normal	16.99	12	1	0
4	1654	NHL Championship Challenge	2004	Sports	Modern day	Normal		374	8	0
5	6383	LEGO Star Wars: The Video Game	2005	Gear	Miscellaneous	Gear				0
6	29916	Bricks and Animals	2020	Classic	Basic	Normal	59.99	1500		4
7	5571	Deep Sea Treasure Hunter	2007	Aqua Raiders	Action/Adventure	Normal	4.99	75	1	4
8	26320	Journal with White Band	2016	Gear	Miscellaneous	Gear	19.99			0
9	4795	Hans Christian Andersen Bucket	2005	Creator	Model making	Normal		500		0
10	9247	Racing Car	2012	City	Modern day	Normal		35	1	3

Showing 1 to 10 of 100 entries

Previous 1 2 3 4 5 ... 10 Next

Data Wrangling Cheat Sheet



Data Transformation with dplyr : : CHEAT SHEET

dplyr

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(…)
starts_with(match)
:, e.g. mpg:cyl
-, e.g, -Species

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

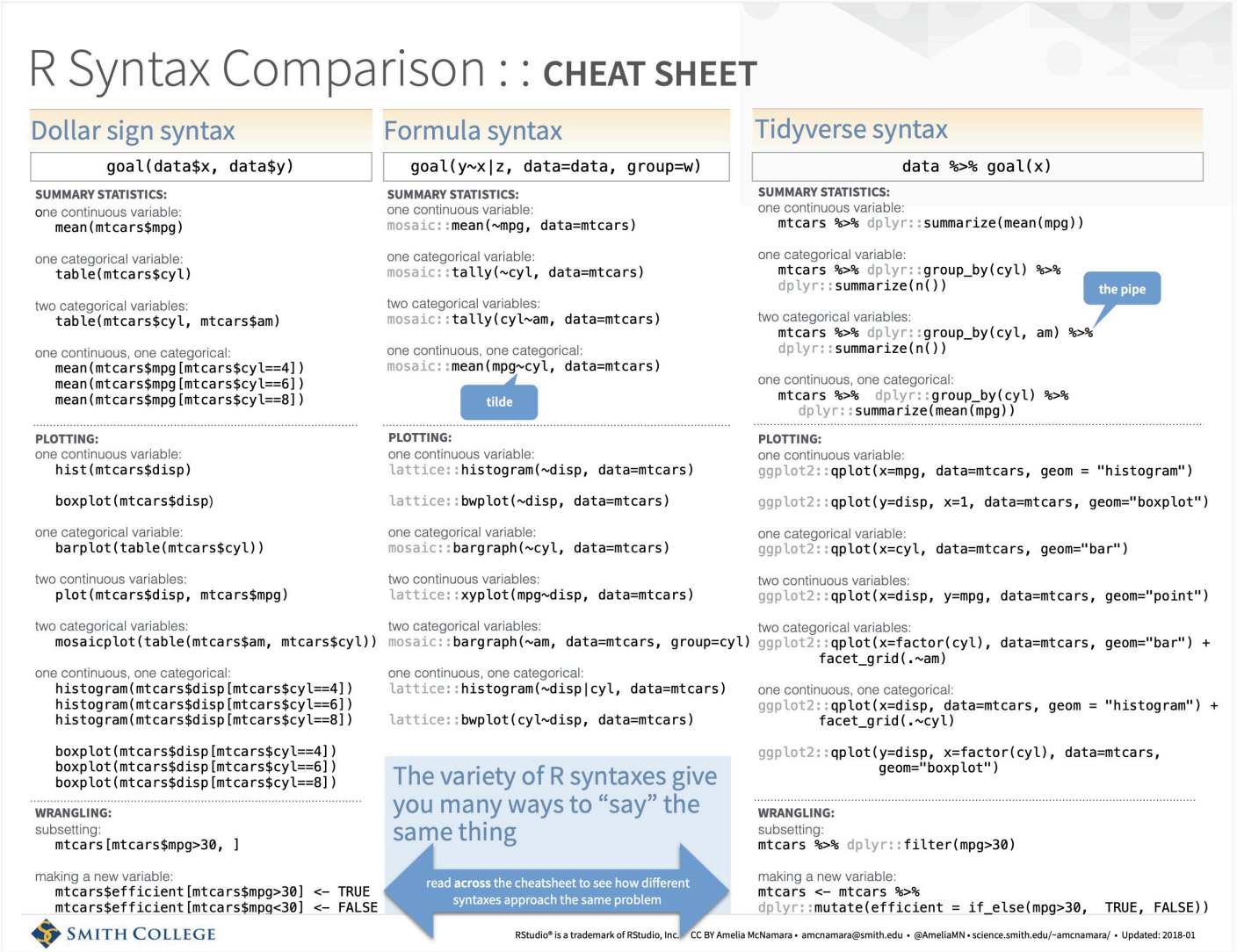
mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

RStudio

Tidyverse vs Base R



R Syntax Comparison : : CHEAT SHEET

Dollar sign syntax

`goal(data$x, data$y)`

SUMMARY STATISTICS:

one continuous variable:
```
mean(mtcars$mpg)
```

one categorical variable:
```
table(mtcars$cyl)
```

two categorical variables:
```
table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:
```
mean(mtcars$mpg[mtcars$cyl==4])
mean(mtcars$mpg[mtcars$cyl==6])
mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:
```
hist(mtcars$disp)

boxplot(mtcars$disp)
```

one categorical variable:
```
barplot(table(mtcars$cyl))
```

two continuous variables:
```
plot(mtcars$disp, mtcars$mpg)
```

two categorical variables:
```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:
```
histogram(mtcars$disp[mtcars$cyl==4])
histogram(mtcars$disp[mtcars$cyl==6])
histogram(mtcars$disp[mtcars$cyl==8])

boxplot(mtcars$disp[mtcars$cyl==4])
boxplot(mtcars$disp[mtcars$cyl==6])
boxplot(mtcars$disp[mtcars$cyl==8])
```

WRANGLING:

subsetting:
```
mtcars[mtcars$mpg>30, ]
```

making a new variable:
```
mtcars$efficient[mtcars$mpg>30] <- TRUE
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Formula syntax

`goal(y~x|z, data=data, group=w)`

SUMMARY STATISTICS:

one continuous variable:
```
mosaic::mean(~mpg, data=mtcars)
```

one categorical variable:
```
mosaic::tally(~cyl, data=mtcars)
```

two categorical variables:
```
mosaic::tally(cyl~am, data=mtcars)
```

one continuous, one categorical:
```
mosaic::mean(mpg~cyl, data=mtcars)
```

tilde

PLOTTING:

one continuous variable:
```
lattice::histogram(~disp, data=mtcars)

lattice::bwplot(~disp, data=mtcars)
```

one categorical variable:
```
mosaic::bargraph(~cyl, data=mtcars)
```

two continuous variables:
```
lattice::xyplot(mpg~disp, data=mtcars)
```

two categorical variables:
```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

one continuous, one categorical:
```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

The variety of R syntaxes give you many ways to "say" the same thing

read **across** the cheatsheet to see how different syntaxes approach the same problem

Tidyverse syntax

`data %>% goal(x)`

SUMMARY STATISTICS:

one continuous variable:
```
mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:
```
mtcars %>% dplyr::group_by(cyl) %>%
  dplyr::summarize(n())
```

the pipe

two categorical variables:
```
mtcars %>% dplyr::group_by(cyl, am) %>%
  dplyr::summarize(n())
```

one continuous, one categorical:
```
mtcars %>%  dplyr::group_by(cyl) %>%
   dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:
```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")

ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

one categorical variable:
```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:
```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

two categorical variables:
```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
        facet_grid(.~am)
```

one continuous, one categorical:
```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
        facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
               geom="boxplot")
```

WRANGLING:

subsetting:
```
mtcars %>% dplyr::filter(mpg>30)
```

making a new variable:
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30,  TRUE, FALSE))
```

SMITH COLLEGE

RStudio® is a trademark of RStudio, Inc. CC BY Amelia McNamara • amcnamara@smith.edu • @AmeliaMN • science.smith.edu/~amcnamara/ • Updated: 2018-01

Pipes %>% and |>

The pipe operator (`%>%`) introduced with the `magrittr` R package allows for the chaining of R operations. Base R has now added their own pipe operator (`|>`). They take the output from the left-hand side and passes it as the first parameter to the function on the right-hand side.

You can do this in two steps:

```
tab_out <- table(legosets$category)
prop.table(tab_out)
```

Or as nested function calls.

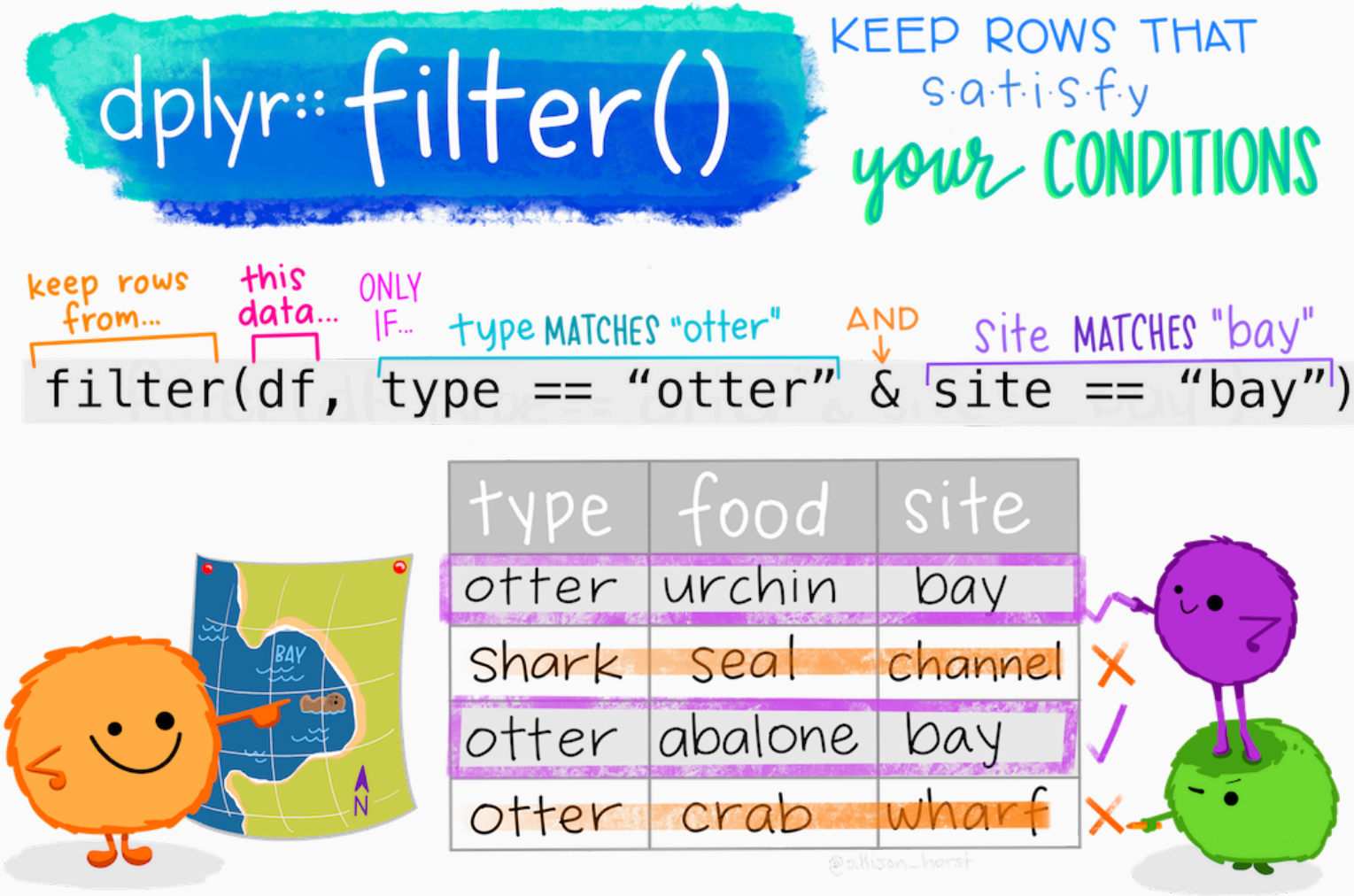
```
prop.table(table(legosets$category))
```

Using the pipe (`|>`) operator we can chain these calls in a what is arguably a more readable format:

```
table(legosets$category) |> prop.table()
```

```
##
##        Book  Collection    Extended        Gear      Normal       Other
## 0.034468546 0.031377196 0.028749549 0.154515946 0.684682364 0.062599825
```

Filter


dplyr::filter()
KEEP ROWS THAT satisfy your CONDITIONS
keep rows from...
this data...
ONLY IF...
type MATCHES "otter"
AND
site MATCHES "bay"
filter(df, type == "otter" & site == "bay")
BAY
N
type food site
otter urchin bay
shark seal channel
otter abalone bay
otter crab wharf
@allison_horst

Logical Operators

- `!a` - TRUE if a is FALSE
- `a == b` - TRUE if a and be are equal
- `a != b` - TRUE if a and b are not equal
- `a > b` - TRUE if a is larger than b, but not equal
- `a >= b` - TRUE if a is larger or equal to b
- `a < b` - TRUE if a is smaller than be, but not equal
- `a <= b` - TRUE if a is smaller or equal to b
- `a %in% b` - TRUE if a is in b where b is a vector

```
which( letters %in% c('a','e','i','o','u') )
```

```
## [1]  1  5  9 15 21
```

- `a | b` - TRUE if a *or* b are TRUE
- `a & b` - TRUE if a *and* b are TRUE
- `isTRUE(a)` - TRUE if a is TRUE

Filter

dplyr

```
mylego <- legosets %>% filter(themeGroup == 'Educational' & year > 2015)
```

Base R

```
mylego <- legosets[legosets$themeGroups == 'Educaitonal' & legosets$year > 2015,]
```

```
nrow(mylego)
```

```
## [1] 99
```

Select

dplyr

```
mylego <- mylego %>% select(setID, pieces, theme, availability, US_retailPrice, minifigs)
```

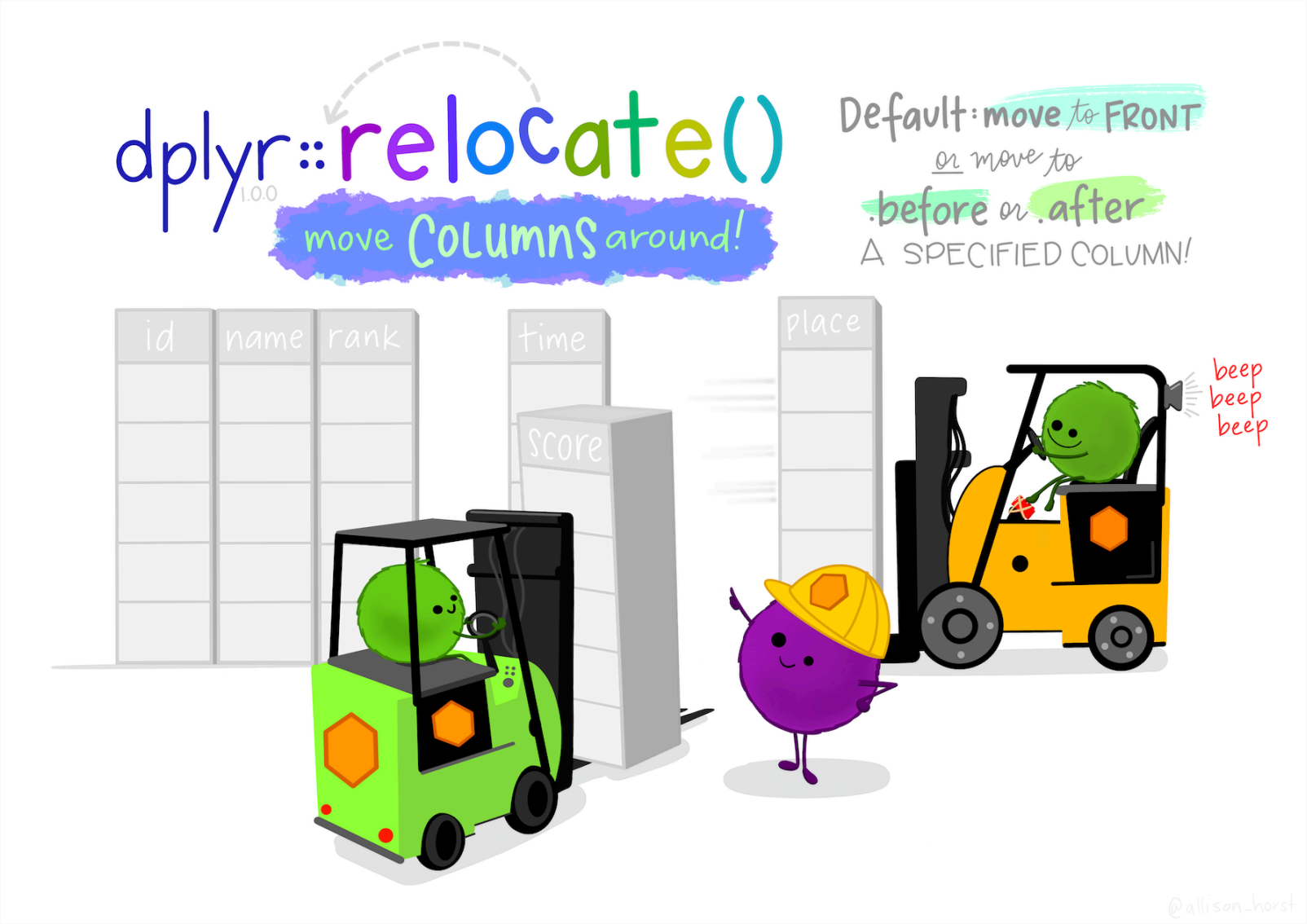
Base R

```
mylego <- mylego[,c('setID', 'pieces', 'theme', 'availability', 'US_retailPrice', 'minifigs')]
```

```
head(mylego, n = 4)
```

```
##   setID pieces     theme    availability US_retailPrice minifigs
## 1 26803    103 Education {Not specified}             NA        6
## 2 26689    142 Education {Not specified}             NA        4
## 3 26804     98 Education {Not specified}             NA        6
## 4 26277    188 Education     Educational          94.95       NA
```

Relocate


dplyr::relocate()
1.0.0
move columns around!
Default: move to FRONT
or move to
.before or .after
A SPECIFIED COLUMN!
id
name
rank
time
score
place
beep
beep
beep
@allison_horst

Relocate

dplyr

```
mylego %>% relocate(where(is.numeric), .after = where(is.character)) %>% head(n = 3)
```

```
##       theme    availability setID pieces US_retailPrice minifigs
## 1 Education {Not specified} 26803    103             NA        6
## 2 Education {Not specified} 26689    142             NA        4
## 3 Education {Not specified} 26804     98             NA        6
```

Base R

```
mylego2 <- mylego[,c('theme', 'availability', 'setID', 'pieces', 'US_retailPrice', 'minifigs')]
head(mylego2, n = 3)
```

```
##       theme    availability setID pieces US_retailPrice minifigs
## 1 Education {Not specified} 26803    103             NA        6
## 2 Education {Not specified} 26689    142             NA        4
## 3 Education {Not specified} 26804     98             NA        6
```

Rename

dplyr::rename()

RENAME COLUMNS*

```
df %>% rename(lair = site)
```

~~species~~ nemesis	status	~~site~~ lair
narwhal	unknown	ocean
chicken	active	coop
pika	active	mountain

*See **rename_with()** to rename using a function.

Rename

dplyr

```
mylego %>% dplyr::rename(USD = US_retailPrice) %>% head(n = 3)
```

```
##   setID pieces     theme    availability USD minifigs
## 1 26803    103 Education {Not specified}  NA        6
## 2 26689    142 Education {Not specified}  NA        4
## 3 26804     98 Education {Not specified}  NA        6
```

Base R

```
names(mylego2)[5] <- 'USD'
head(mylego2, n = 3)
```

```
##       theme    availability setID pieces USD minifigs
## 1 Education {Not specified} 26803    103  NA        6
## 2 Education {Not specified} 26689    142  NA        4
## 3 Education {Not specified} 26804     98  NA        6
```

Mutate


dplyr::mutate
add Column(s),
keep existing.
A
B
C
1
5
10
0
2
-3
4
9
6
3
2
14
9
8
THE PLAN:
A
B
C
+
=
HORST '18

Mutate

dplyr

```
mylego %>% filter(!is.na(pieces) & !is.na(US_retailPrice)) %>%
    mutate(Price_per_piece = US_retailPrice / pieces) %>% head(n = 3)
```

```
##   setID pieces     theme availability US_retailPrice minifigs Price_per_piece
## 1 26277    188 Education  Educational          94.95       NA       0.5050532
## 2 25949    280 Education  Educational         224.95       NA       0.8033929
## 3 25954      1 Education  Educational          14.95       NA      14.9500000
```

Base R

```
mylego2 <- mylego[!is.na(mylego$US_retailPrice) & !is.na(mylego$Price_per_piece),]
mylego2$Price_per_piece <- mylego2$Price_per_piece / mylego2$US_retailPrice
head(mylego2, n = 3)
```

```
## [1] setID           pieces          theme           availability
## [5] US_retailPrice  minifigs        Price_per_piece
## <0 rows> (or 0-length row.names)
```

Group By and Summarize


```
legosets %>% group_by(themeGroup) %>% summarize(mean_price = mean(US_retailPrice, na.rm = TRUE),
                                                sd_price = sd(US_retailPrice, na.rm = TRUE),
                                                median_price = median(US_retailPrice, na.rm = TRUE),
                                                n = n(),
                                                missing = sum(is.na(US_retailPrice)))
```

```
## # A tibble: 17 × 6
##    themeGroup       mean_price sd_price median_price     n missing
##    <chr>                 <dbl>    <dbl>        <dbl> <int>   <int>
##  1 Action/Adventure       40.2     38.9         30.0  1474     779
##  2 Art and crafts         34.9     47.7         17.5    97       9
##  3 Basic                  21.6     19.2         15.0   873     733
##  4 Constraction           16.4     12.4         13.0   502     284
##  5 Educational           182.     188.          130.    503     465
##  6 Girls                  35.8     24.0         23.0   240     227
##  7 Historical             34.2     32.4         20.0   473     400
##  8 Junior                 22.0     10.1         20.0   228     165
##  9 Licensed               53.3     71.7         30.0  2775    1066
## 10 Miscellaneous          20.7     29.2         13.0  6253    3961
## 11 Model making           74.3     92.1         40.0   771     384
## 12 Modern day             38.2     35.6         30.0  2469    1535
## 13 Pre-school             30.8     22.7         25.0  1562    1103
## 14 Racing                 26.8     26.5         15.0   270     176
```

Describe and Describe By

```
library(psych)
describe(legosets$US_retailPrice)
```

```
##    vars    n  mean   sd median trimmed   mad  min    max range skew kurtosis
## X1    1 7483 38.96 56.5  19.99    27.7 17.79 1.49 849.99 848.5 5.32    44.74
##      se
## X1 0.65
```

```
describeBy(legosets$US_retailPrice, group = legosets$availability, mat = TRUE, skew = FALSE)
```

```
##      item                group1 vars    n      mean        sd median   min    max range         se
## X11     1       {Not specified}    1 1831  26.84733  39.96747  19.99  1.49 789.99 788.5  0.9340335
## X12     2           Educational    1   12 212.86667 105.88283 222.45 14.95 399.95 385.0 30.5657410
## X13     3        LEGO exclusive    1 1039  57.21203 106.63125  12.99  1.99 849.99 848.0  3.3080857
## X14     4    LEGOLAND exclusive    1    2   4.99000   0.00000   4.99  4.99   4.99   0.0  0.0000000
## X15     5              Not sold    1    1  12.99000        NA  12.99 12.99  12.99   0.0         NA
## X16     6           Promotional    1    5   4.79000   0.83666   4.99  3.99   5.99   2.0  0.3741657
## X17     7 Promotional (Airline)    1    0       NaN        NA     NA   Inf   -Inf  -Inf         NA
## X18     8                Retail    1 4290  37.55889  38.44918  24.99  1.99 699.99 698.0  0.5870275
## X19     9      Retail - limited    1  302  63.54381  70.91908  39.99  2.49 449.99 447.5  4.0809343
## X110   10               Unknown    1    1   3.99000        NA   3.99  3.99   3.99   0.0         NA
```

Additional Resources

For data wrangling:

- `dplyr` website: https://dplyr.tidyverse.org
- R for Data Science book: https://r4ds.had.co.nz/wrangle-intro.html
- Wrangling penguins tutorial: https://allisonhorst.shinyapps.io/dplyr-learnr/#section-welcome
- Data transformation cheat sheet: https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf

Grammer of Graphics

Data Visualizations with ggplot2

- `ggplot2` is an R package that provides an alternative framework based upon Wilkinson's (2005) Grammar of Graphics.
- `ggplot2` is, in general, more flexible for creating "prettier" and complex plots.
- Works by creating layers of different types of objects/geometries (i.e. bars, points, lines, polygons, etc.) `ggplot2` has at least three ways of creating plots:

    1. `qplot`
    2. `ggplot(...) + geom_XXX(...) + ...`
    3. `ggplot(...) + layer(...)`

- We will focus only on the second.

Parts of a `ggplot2` Statement

- Data

  ```
  ggplot(myDataFrame, aes(x=x, y=y))
  ```

- Layers

  ```
  geom_point(), geom_histogram()
  ```

- Facets

  ```
  facet_wrap(~ cut), facet_grid(~ cut)
  ```

- Scales

  ```
  scale_y_log10()
  ```

- Other options

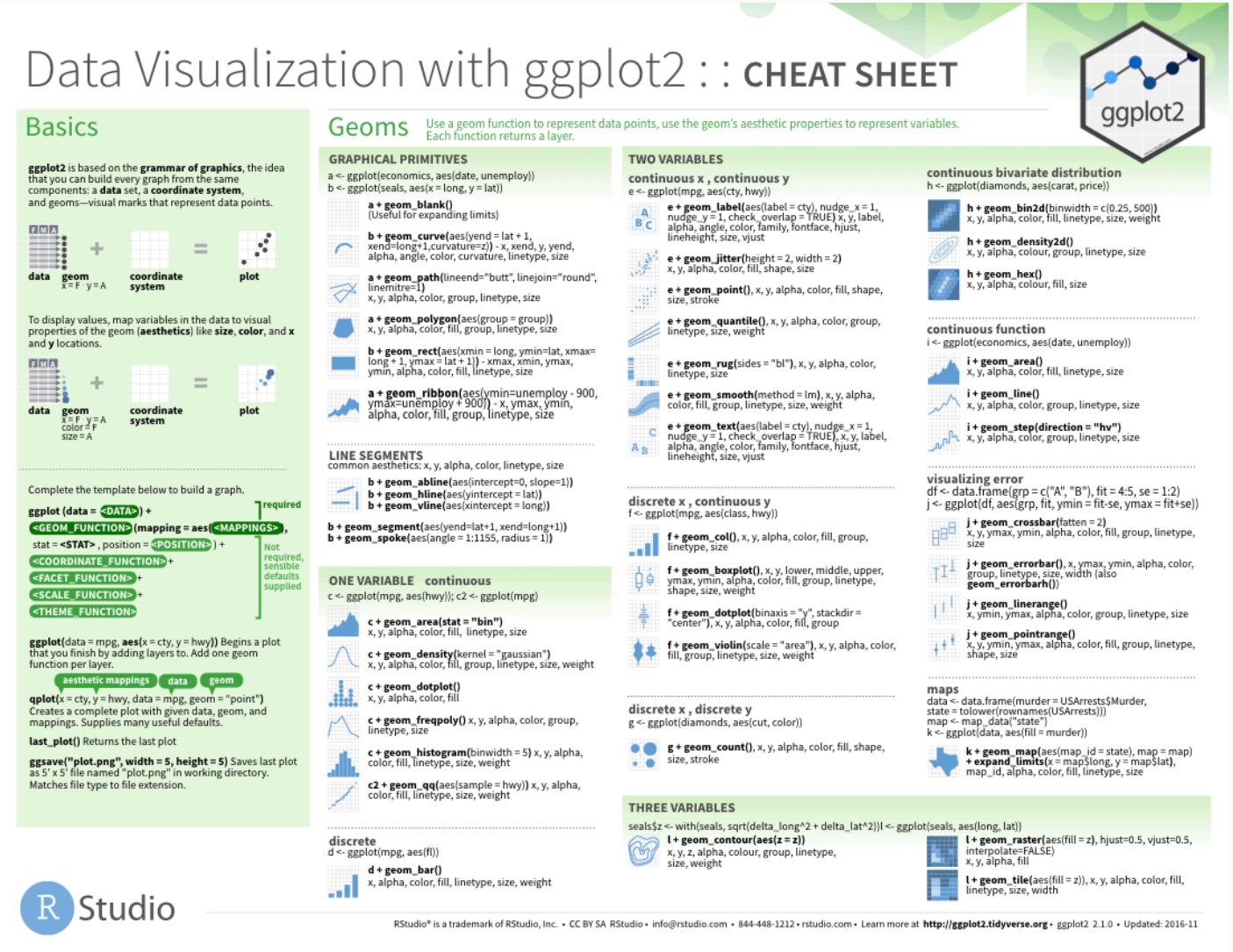
  ```
  ggtitle('my title'), ylim(c(0, 10000)), xlab('x-axis label')
  ```

Lots of geoms

```
ls('package:ggplot2')[grep('^geom_', ls('package:ggplot2'))]
```

```
##  [1] "geom_abline"           "geom_area"             "geom_bar"              "geom_bin_2d"
##  [5] "geom_bin2d"            "geom_blank"            "geom_boxplot"          "geom_col"
##  [9] "geom_contour"          "geom_contour_filled"   "geom_count"            "geom_crossbar"
## [13] "geom_curve"            "geom_density"          "geom_density_2d"       "geom_density_2d_filled"
## [17] "geom_density2d"        "geom_density2d_filled" "geom_dotplot"          "geom_errorbar"
## [21] "geom_errorbarh"        "geom_freqpoly"         "geom_function"         "geom_hex"
## [25] "geom_histogram"        "geom_hline"            "geom_jitter"           "geom_label"
## [29] "geom_line"             "geom_linerange"        "geom_map"              "geom_path"
## [33] "geom_point"            "geom_pointrange"       "geom_polygon"          "geom_qq"
## [37] "geom_qq_line"          "geom_quantile"         "geom_raster"           "geom_rect"
## [41] "geom_ribbon"           "geom_rug"              "geom_segment"          "geom_sf"
## [45] "geom_sf_label"         "geom_sf_text"          "geom_smooth"           "geom_spoke"
## [49] "geom_step"             "geom_text"             "geom_tile"             "geom_violin"
## [53] "geom_vline"
```

Data Visualization Cheat Sheet



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.

data | geom | coordinate system | plot

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

data | geom (x = F, y = A, color = F, size = A) | coordinate system | plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
  stat = <STAT> , position = <POSITION> ) +               Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

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

aesthetic mappings | data | geom

qplot(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(x = long, y = lat))

a + geom_blank() (Useful for expanding limits)

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

a + geom_path(lineend="butt", linejoin="round", linemitre=1) x, y, alpha, color, group, linetype, size

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

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

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

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

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

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

c + geom_histogram(binwidth = 5) 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(fl))

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

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, 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 = "bl"), 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), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, 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(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

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

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

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

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

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

h + geom_density2d() x, y, alpha, colour, group, linetype, size

h + geom_hex() x, y, alpha, colour, 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(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

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

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

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

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

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(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

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z)) x, y, z, alpha, colour, group, linetype, size, weight

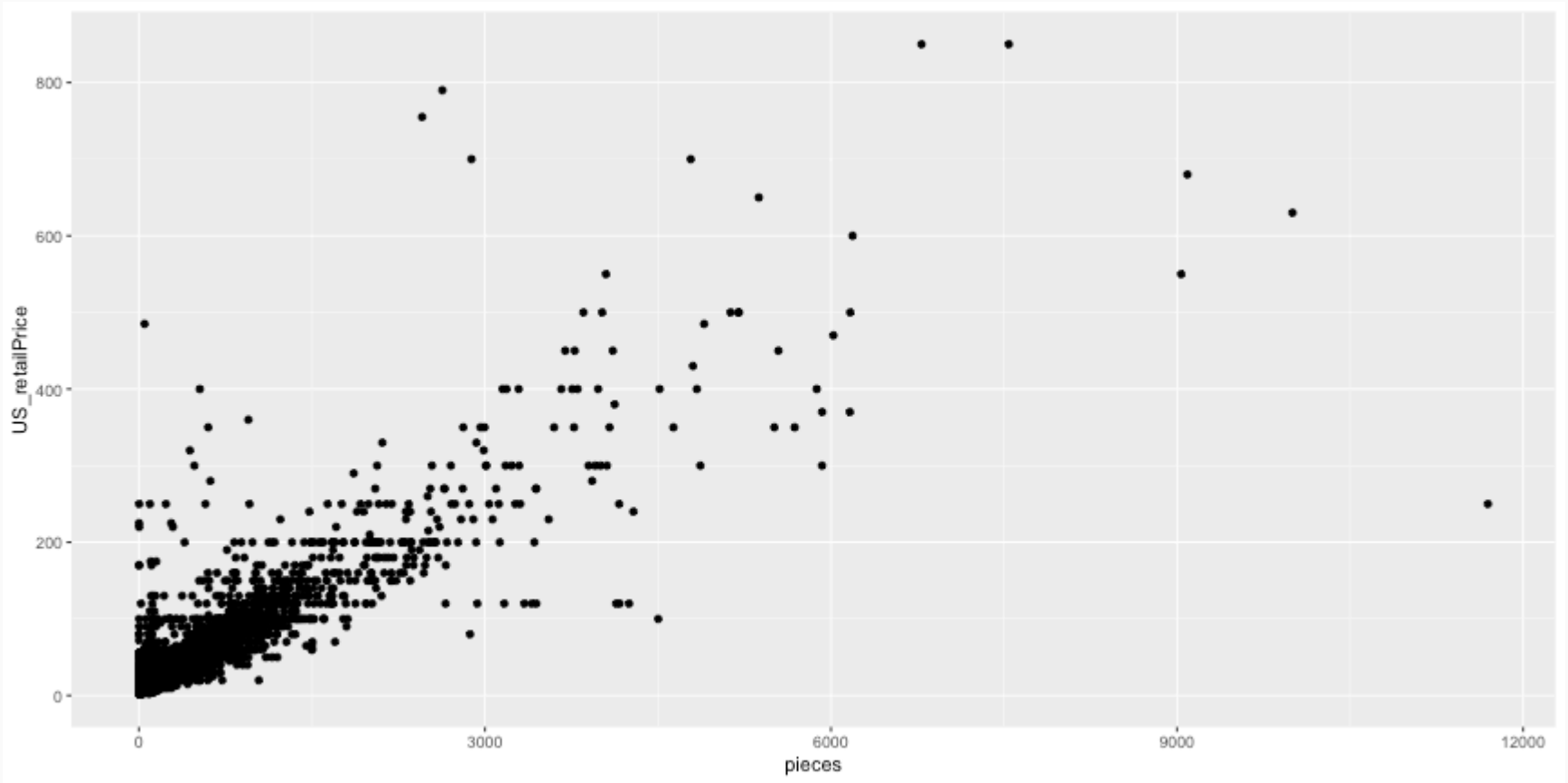
l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE) x, y, alpha, fill

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

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

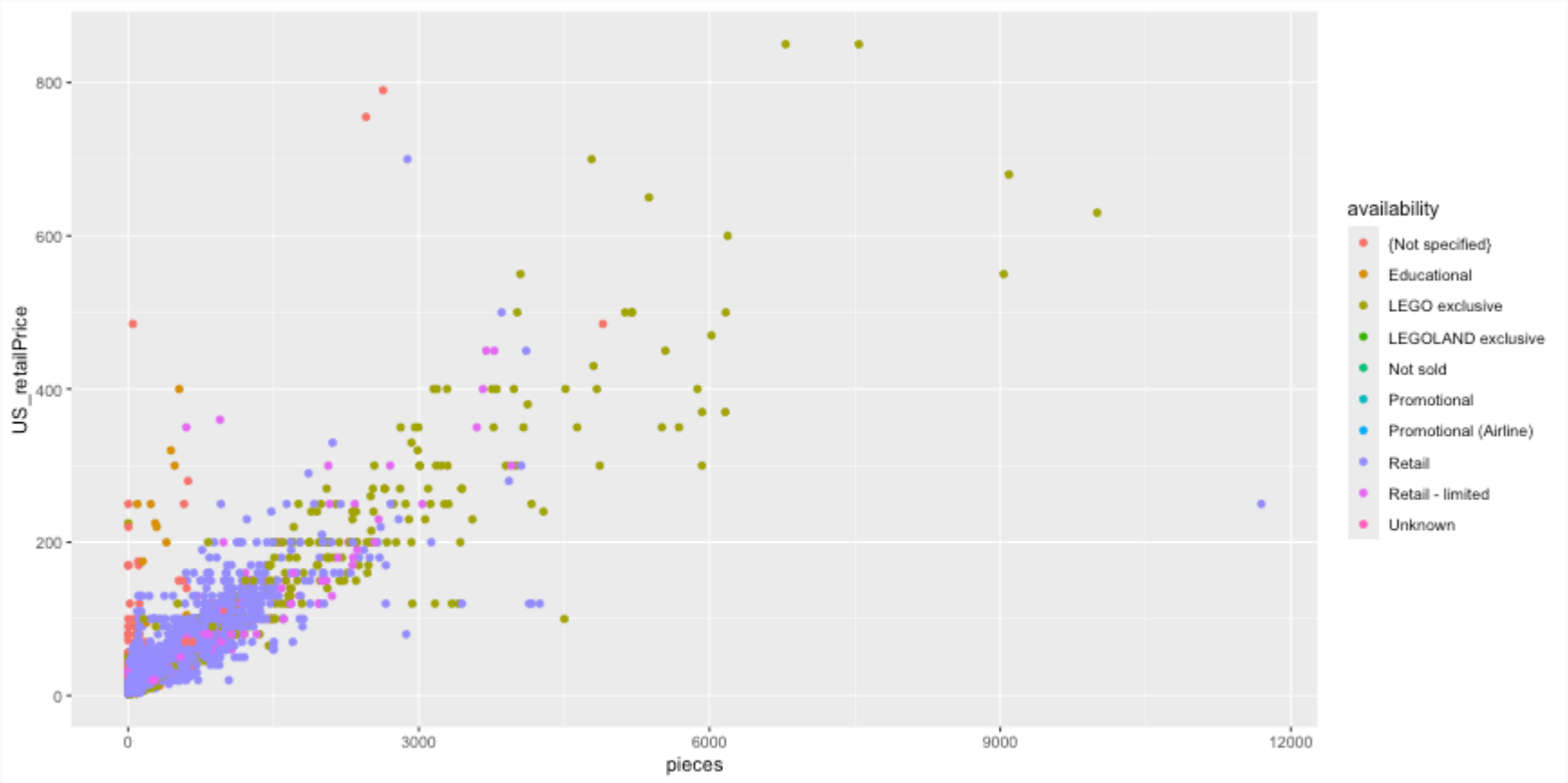
Scatterplot

```
ggplot(legosets, aes(x=pieces, y=US_retailPrice)) + geom_point()
```



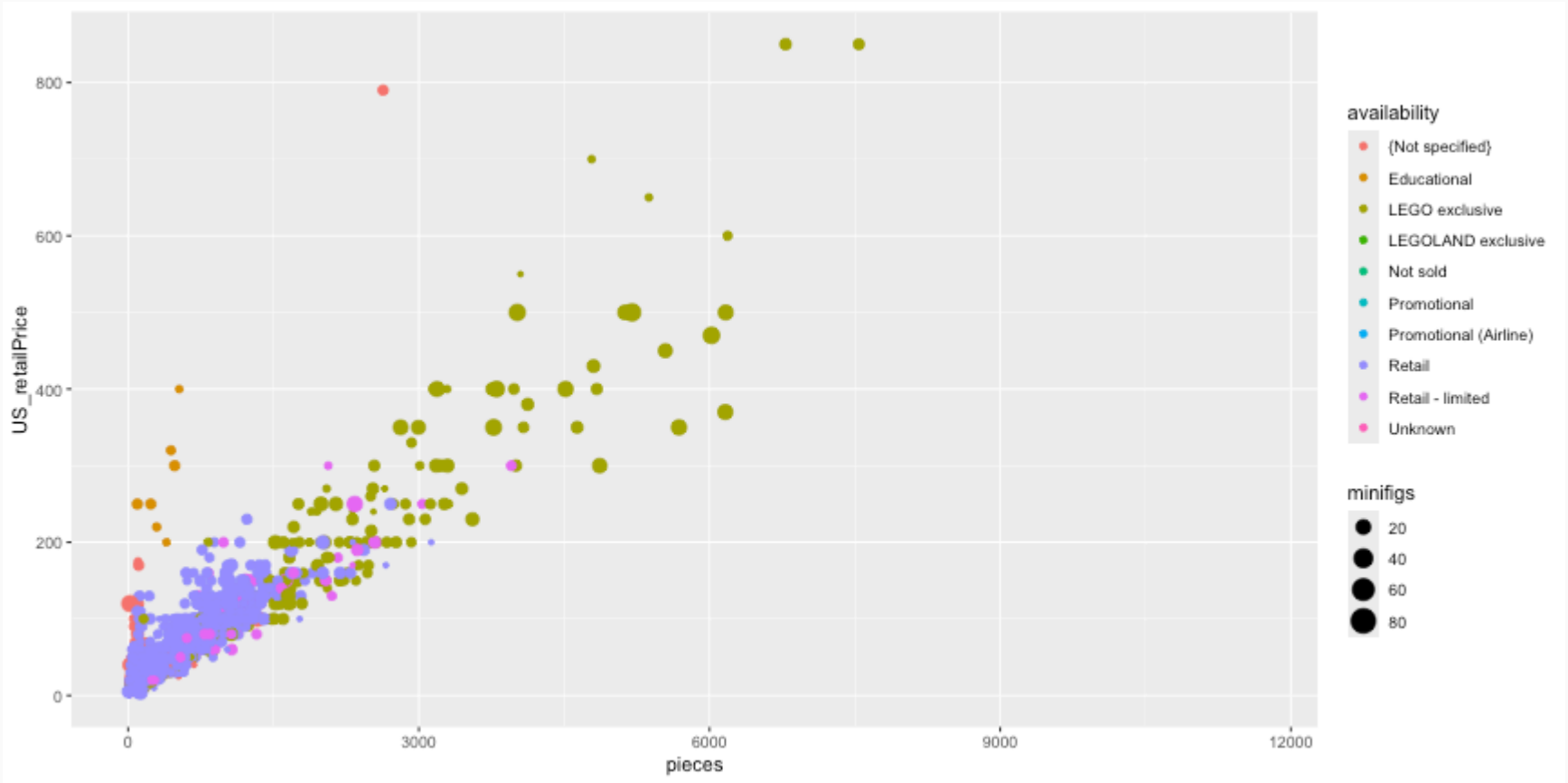
Scatterplot (cont.)

```
ggplot(legosets, aes(x=pieces, y=US_retailPrice, color=availability)) + geom_point()
```



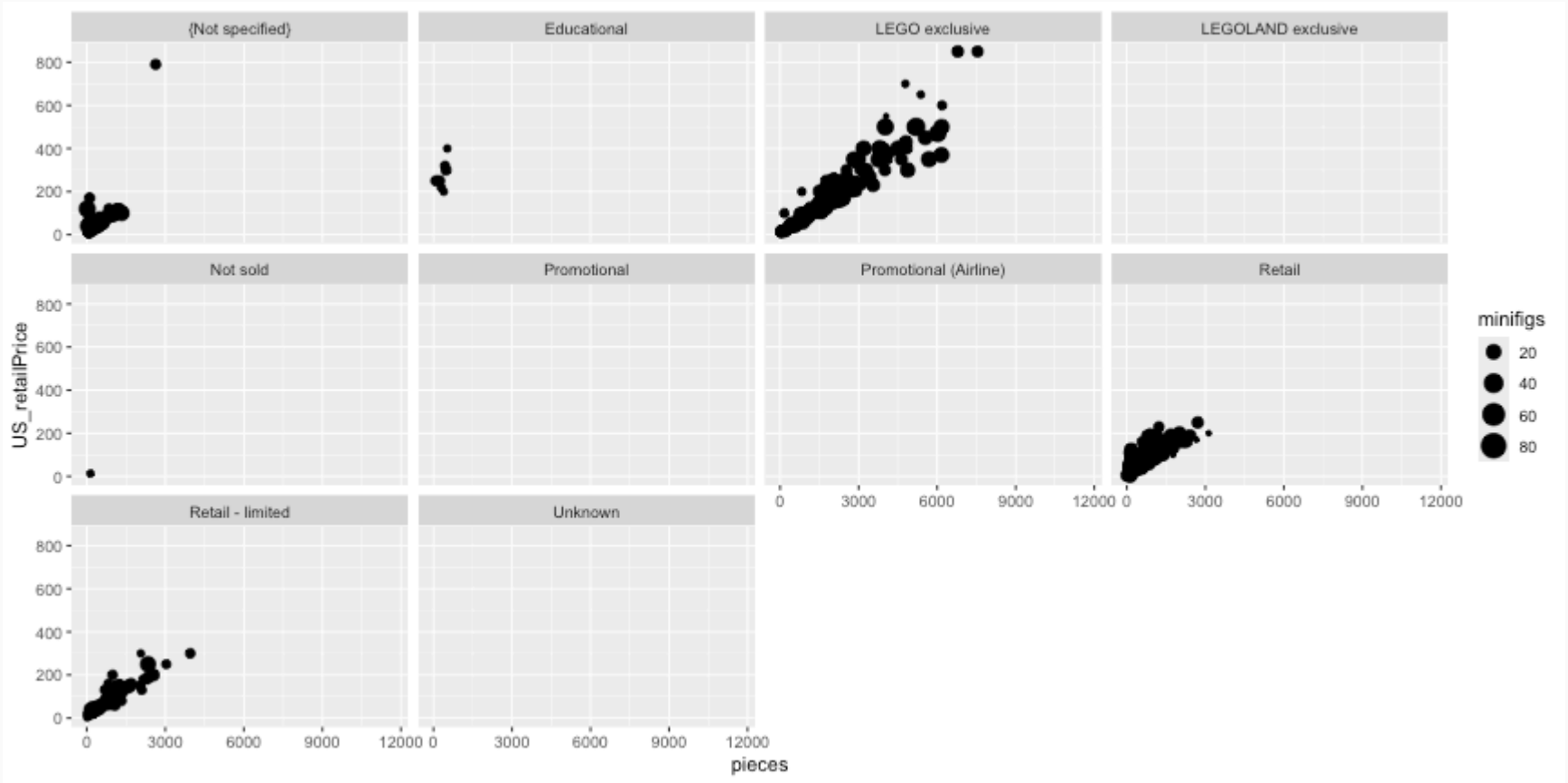
Scatterplot (cont.)

```
ggplot(legosets, aes(x=pieces, y=US_retailPrice, size=minifigs, color=availability)) + geom_point()
```



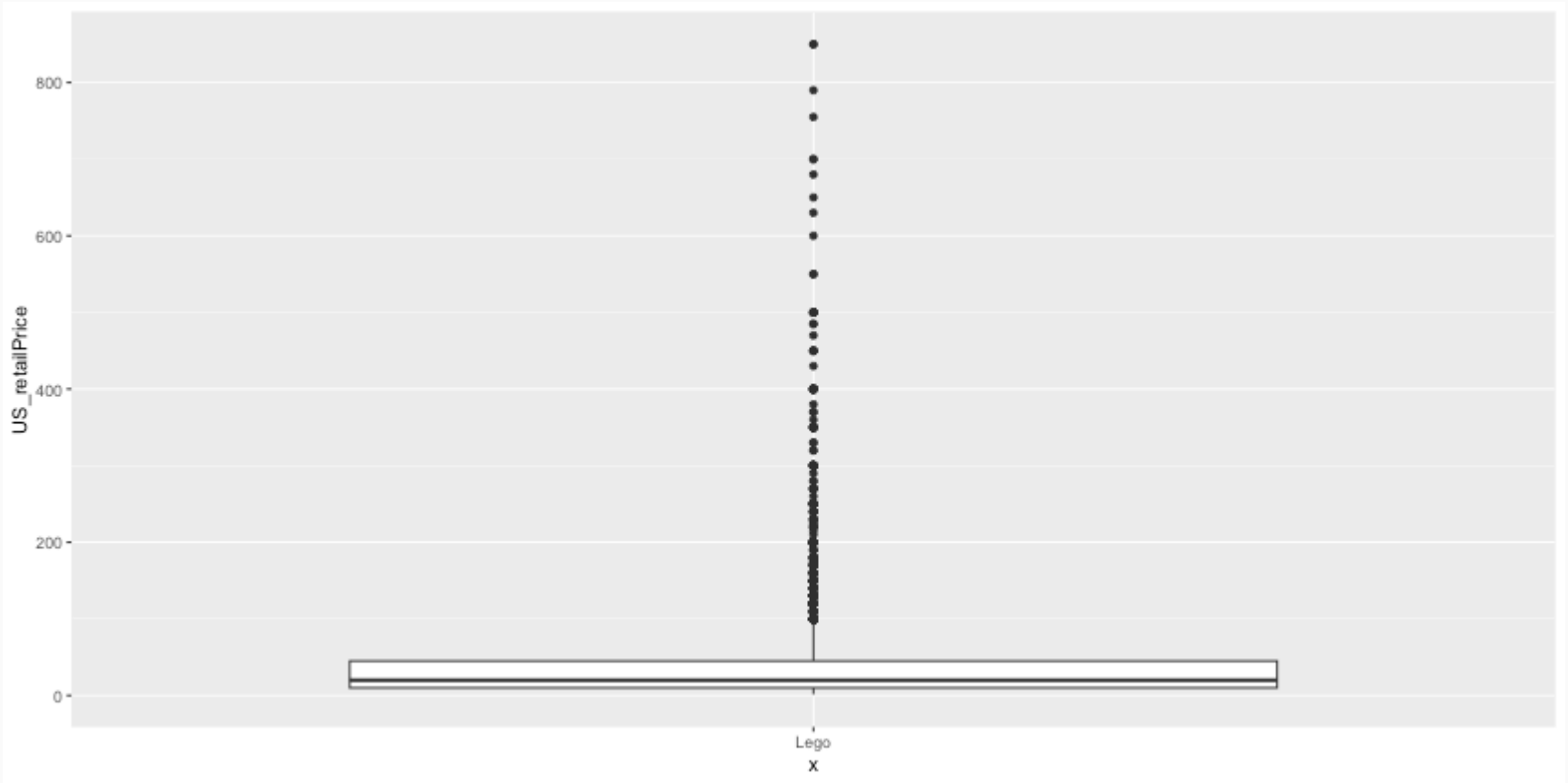
Scatterplot (cont.)

```
ggplot(legosets, aes(x=pieces, y=US_retailPrice, size=minifigs)) + geom_point() + facet_wrap(~ availability)
```



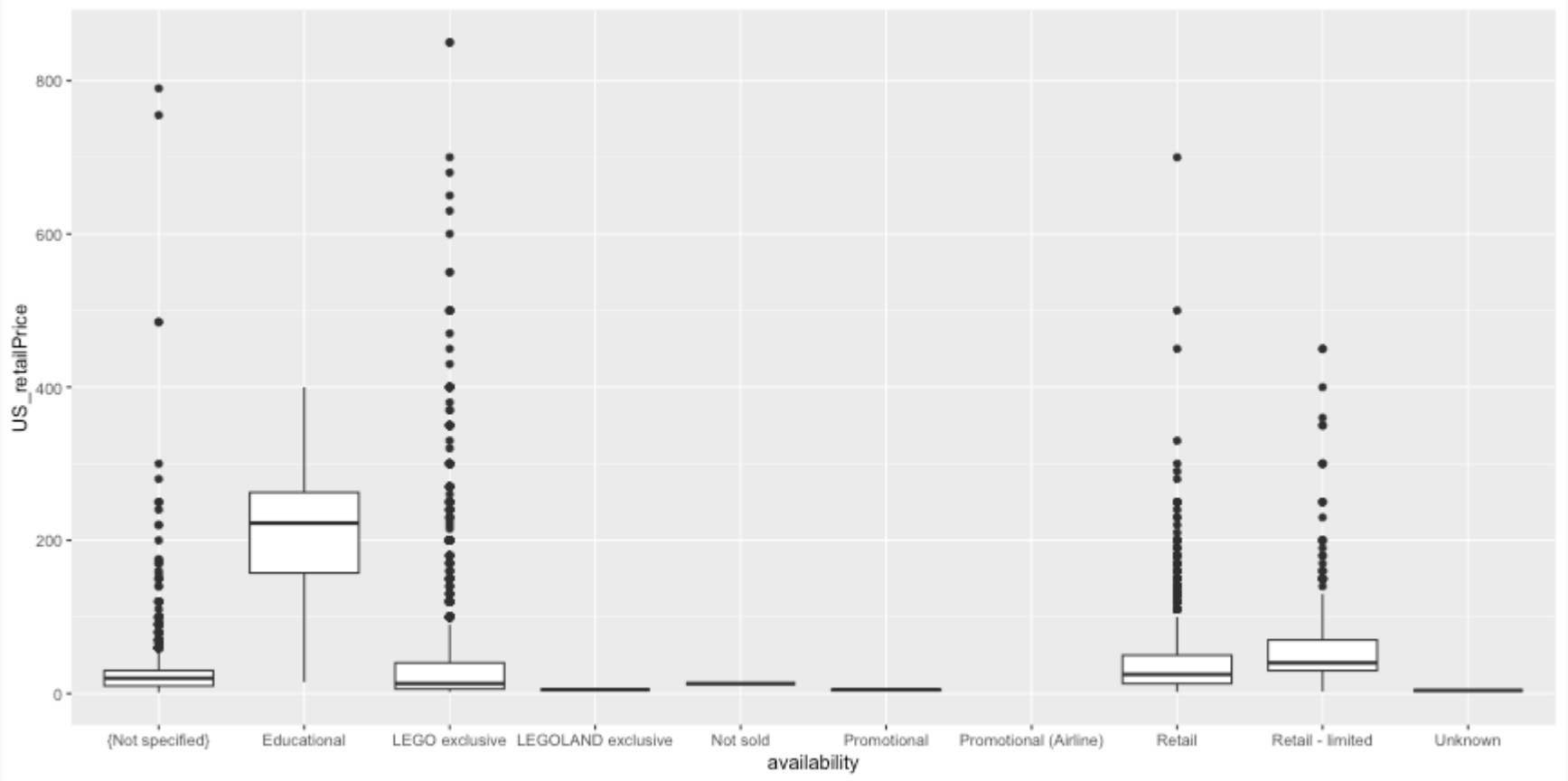
Boxplots

```
ggplot(legosets, aes(x='Lego', y=US_retailPrice)) + geom_boxplot()
```



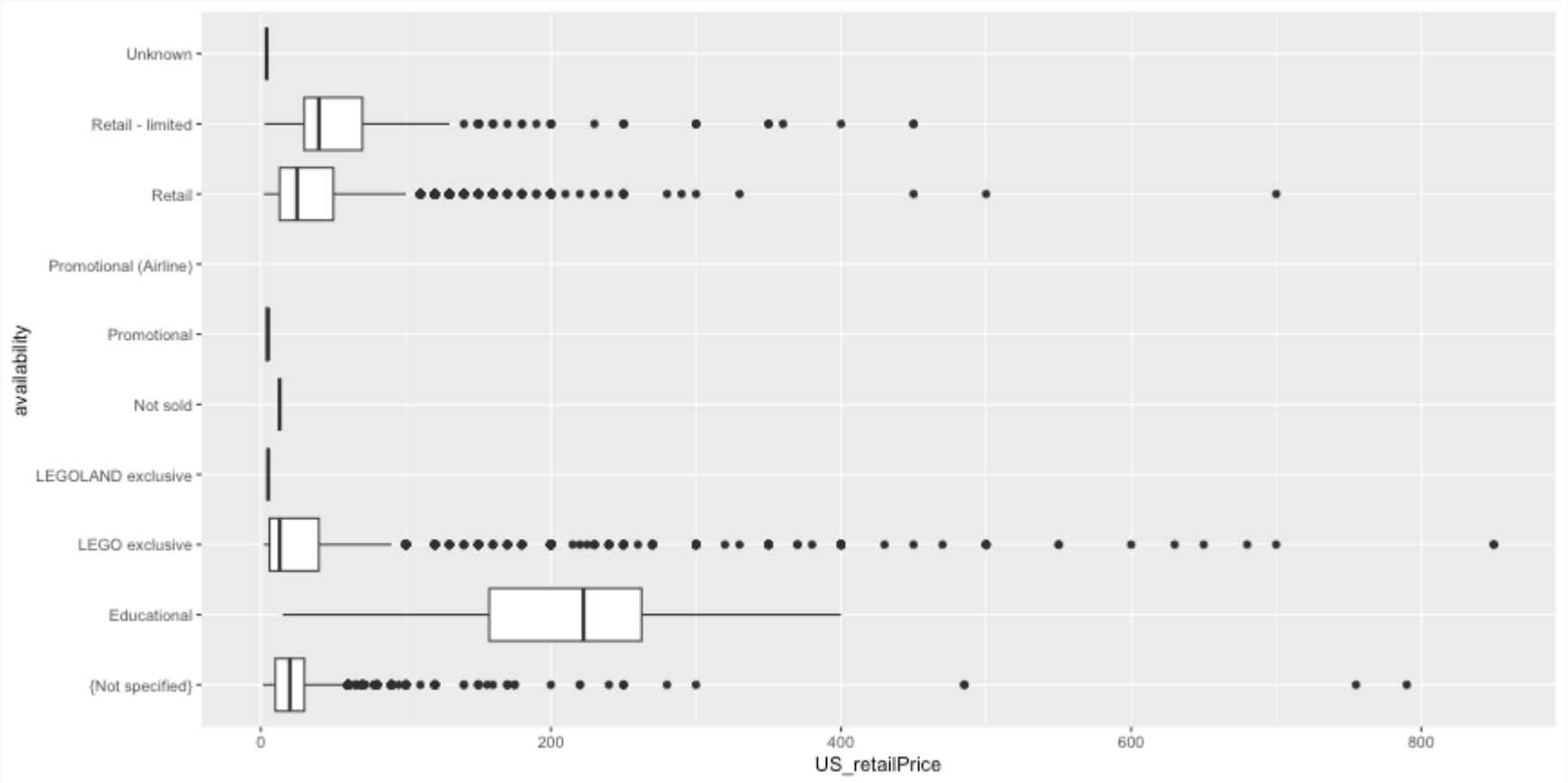
Boxplots (cont.)

```
ggplot(legosets, aes(x=availability, y=US_retailPrice)) + geom_boxplot()
```



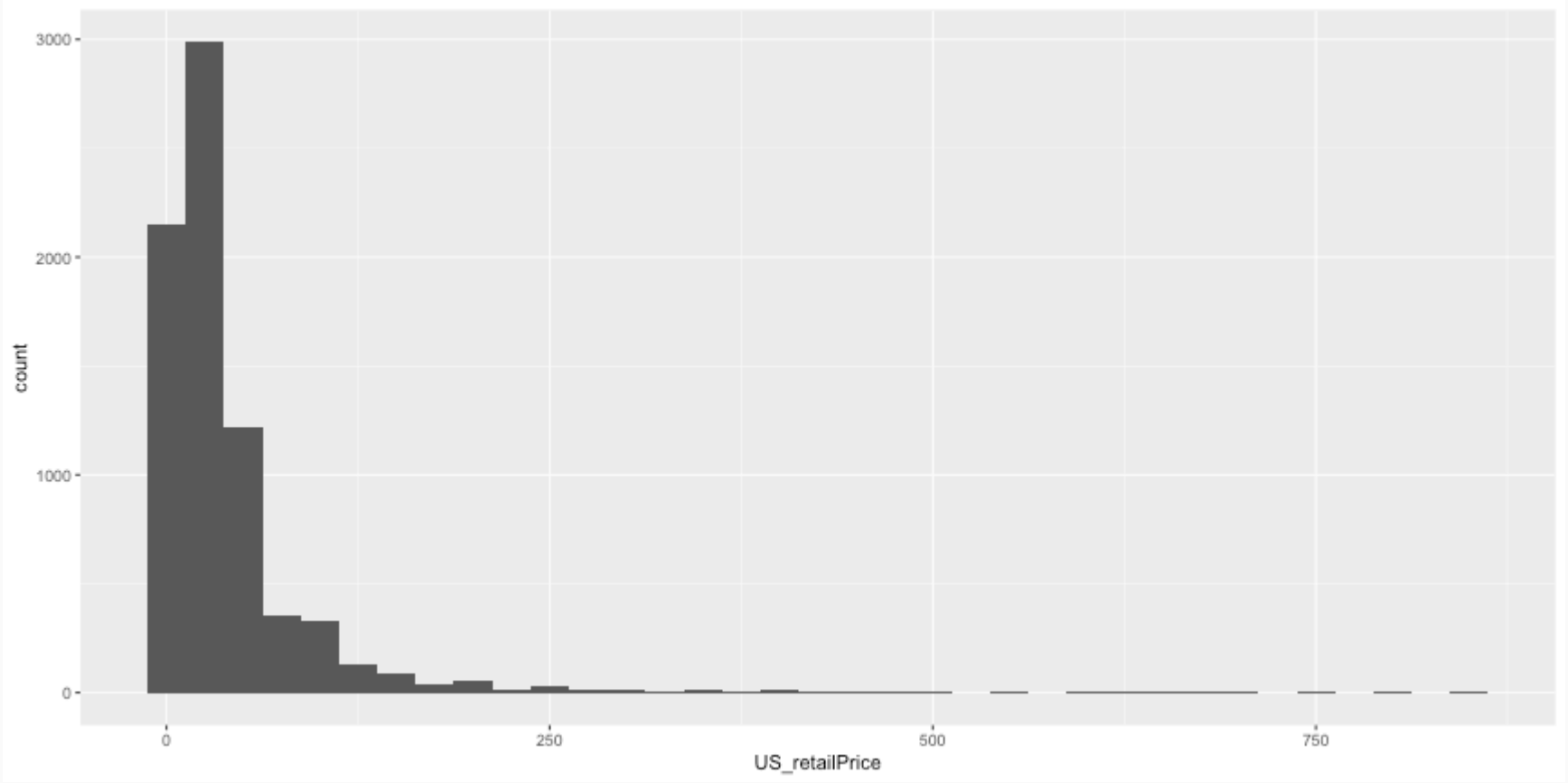
Boxplot (cont.)

```
ggplot(legosets, aes(x=availability, y=US_retailPrice)) + geom_boxplot() + coord_flip()
```



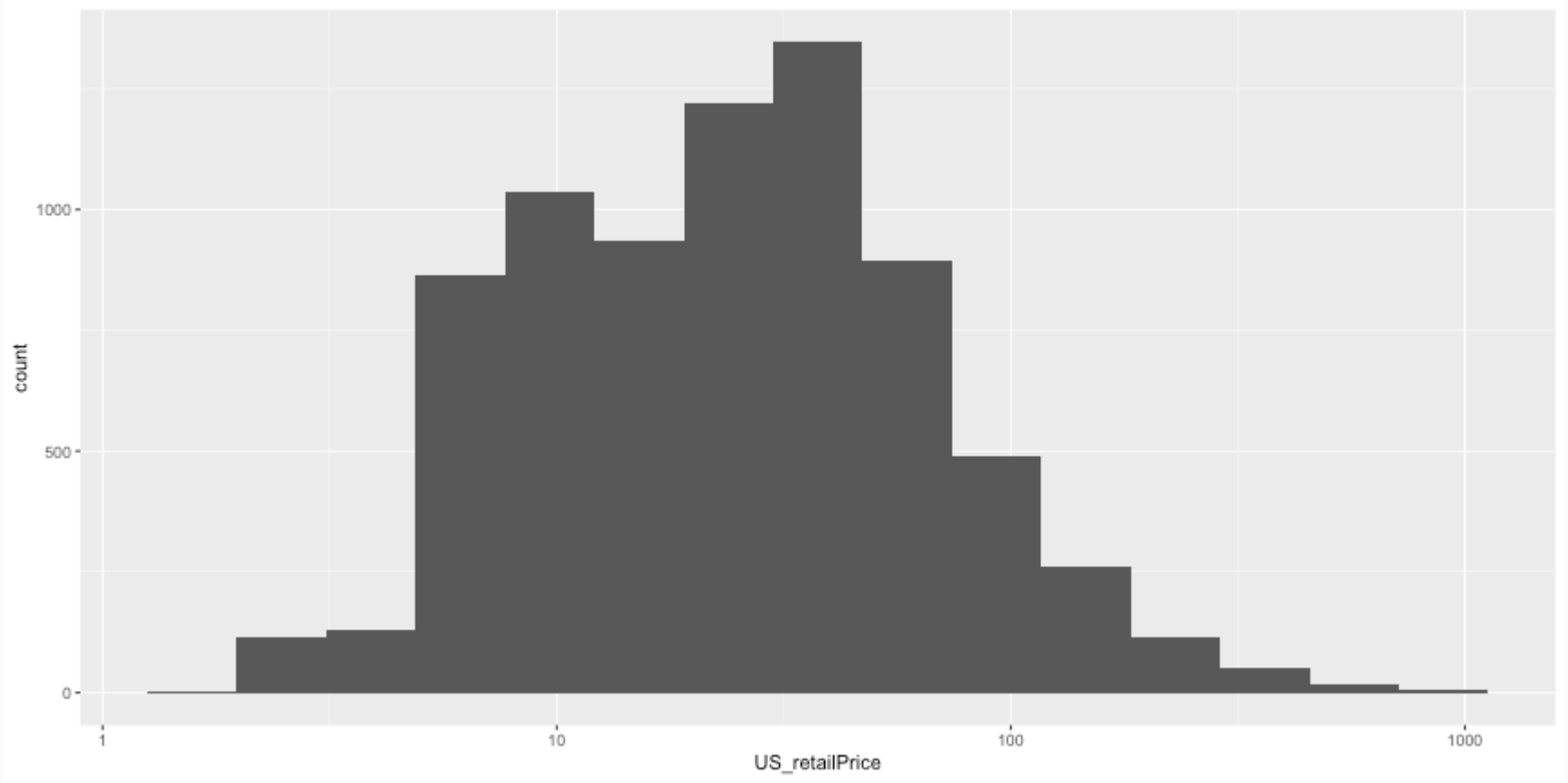
Histograms

```
ggplot(legosets, aes(x = US_retailPrice)) + geom_histogram(binwidth = 25)
```



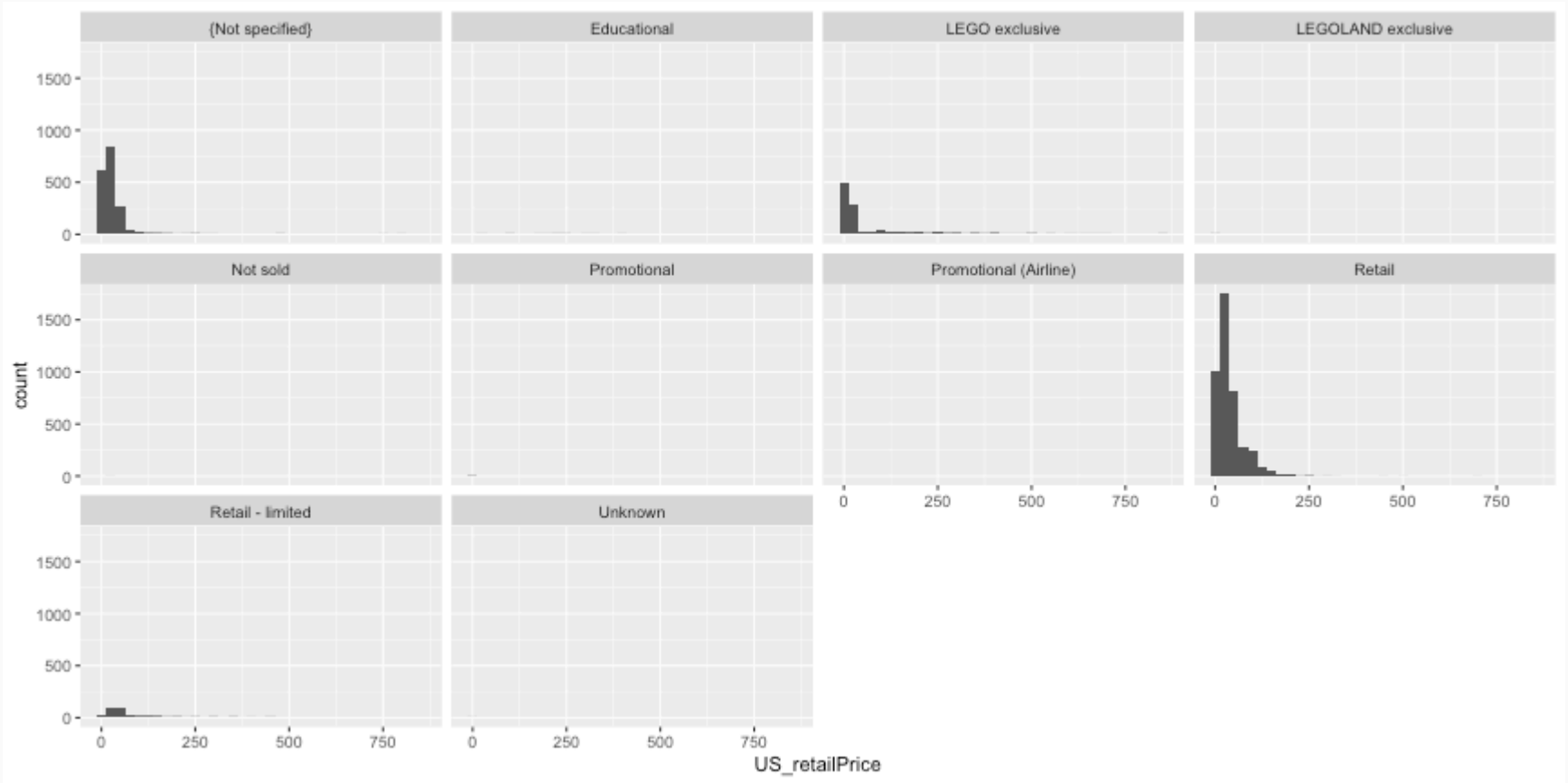
Histograms (cont.)

```
ggplot(legosets, aes(x = US_retailPrice)) + geom_histogram(bins = 15) + scale_x_log10()
```



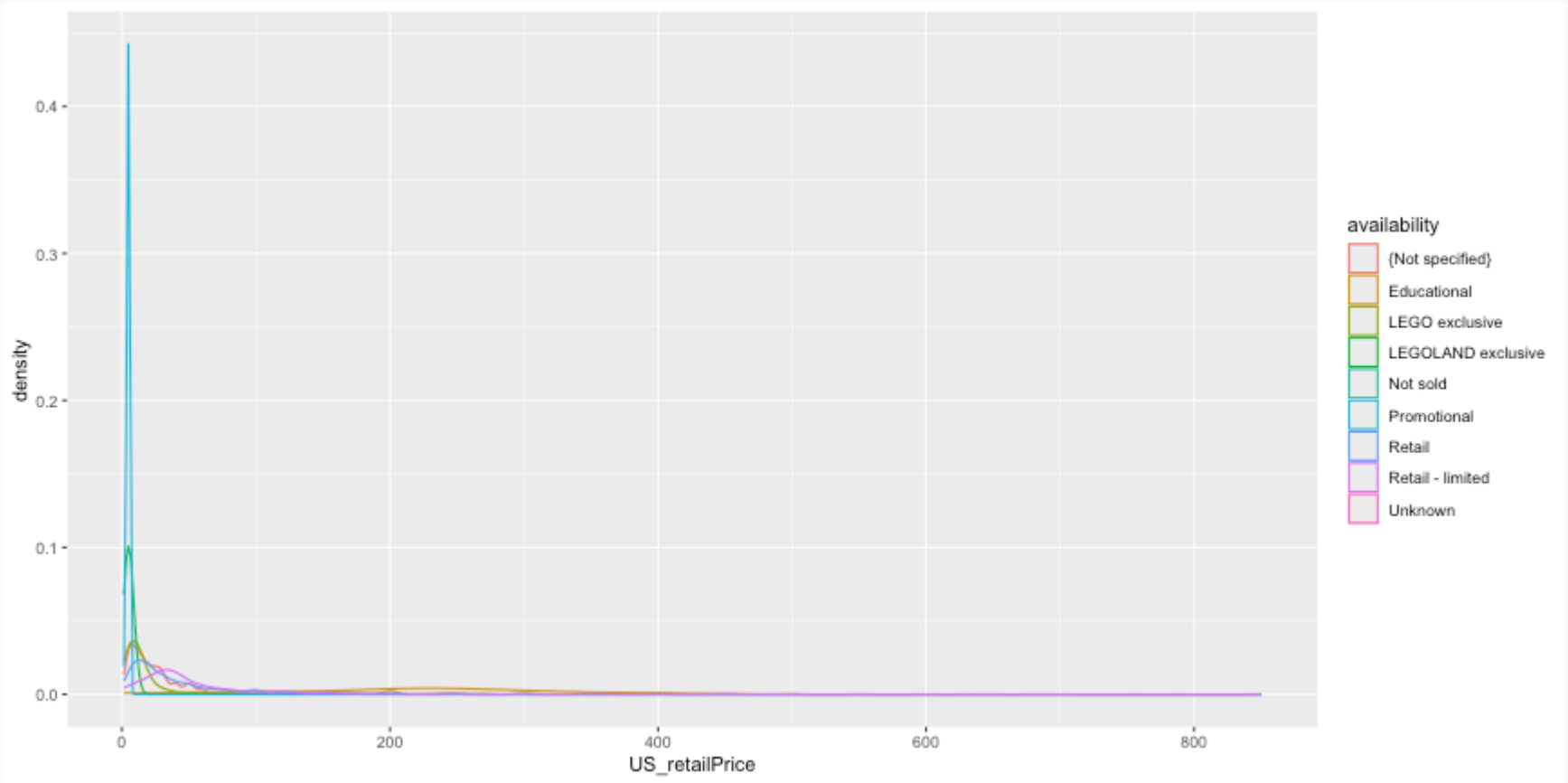
Histograms (cont.)

```
ggplot(legosets, aes(x = US_retailPrice)) + geom_histogram(binwidth = 25) + facet_wrap(~ availability)
```



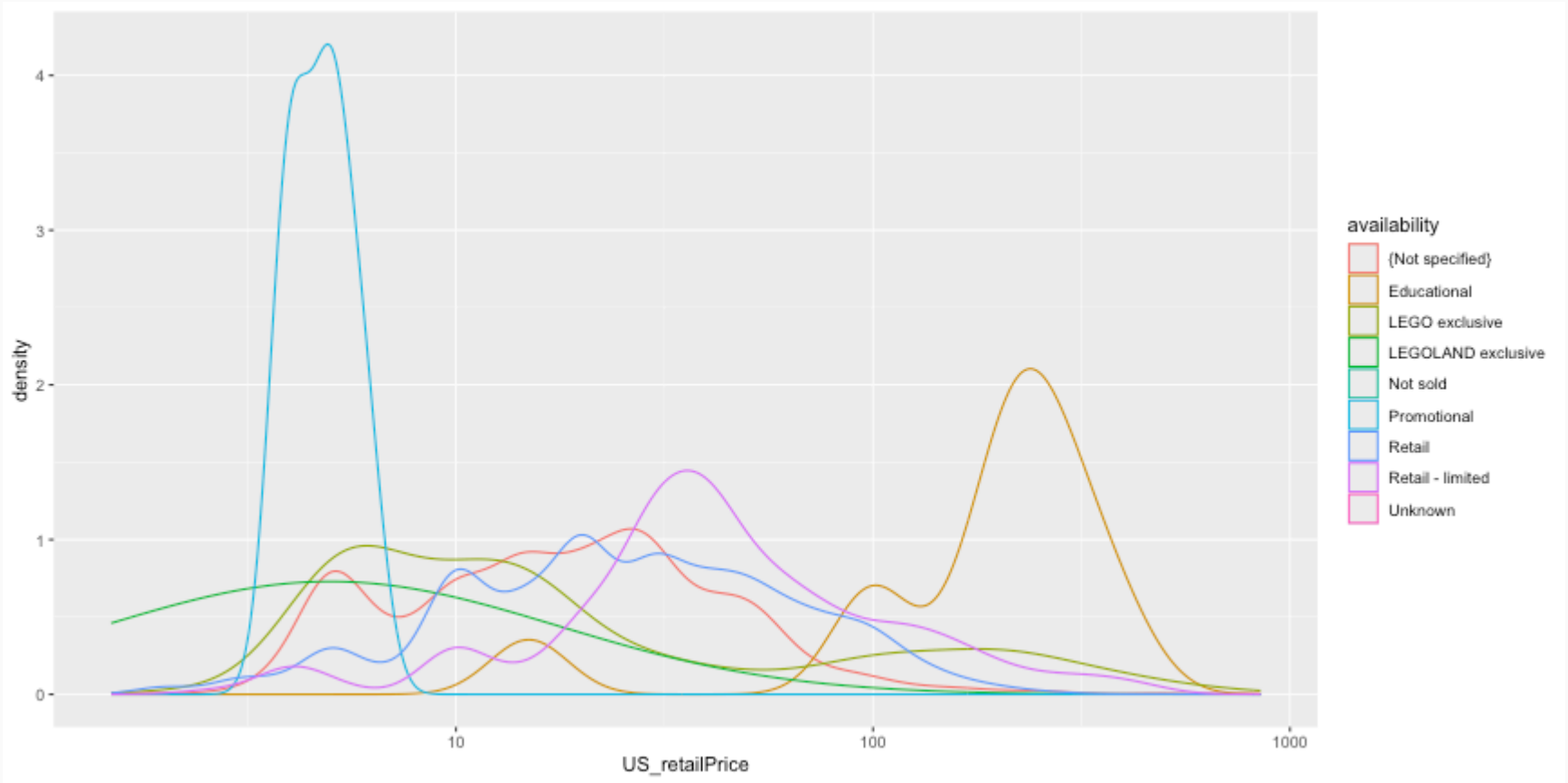
Density Plots

```
ggplot(legosets, aes(x = US_retailPrice, color = availability)) + geom_density()
```

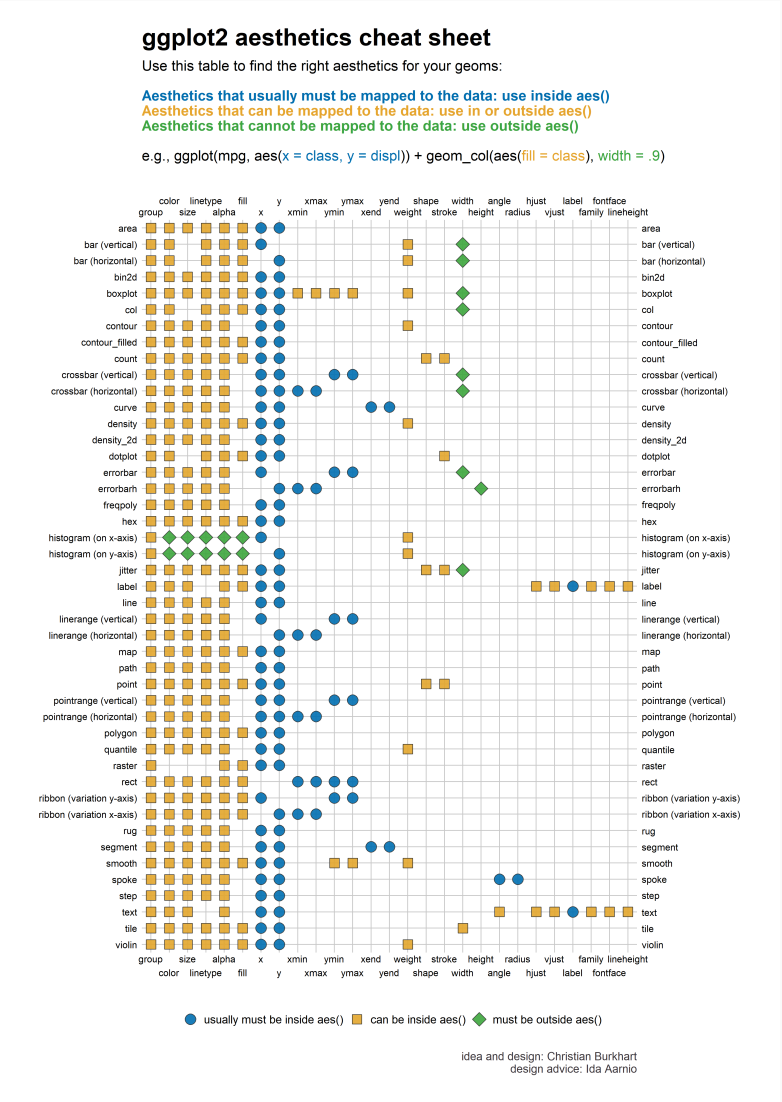


Density Plots (cont.)

```
ggplot(legosets, aes(x = US_retailPrice, color = availability)) + geom_density() + scale_x_log10()
```



ggplot2 aesthetics



Likert Scales

Likert scales are a type of questionnaire where respondents are asked to rate items on scales usually ranging from four to seven levels (e.g. strongly disagree to strongly agree).

```
library(likert)
library(reshape)
data(pisaitems)
items24 <- pisaitems[,substr(names(pisaitems), 1,5) == 'ST24Q']
items24 <- rename(items24, c(
            ST24Q01="I read only if I have to.",
            ST24Q02="Reading is one of my favorite hobbies.",
            ST24Q03="I like talking about books with other people.",
            ST24Q04="I find it hard to finish books.",
            ST24Q05="I feel happy if I receive a book as a present.",
            ST24Q06="For me, reading is a waste of time.",
            ST24Q07="I enjoy going to a bookstore or a library.",
            ST24Q08="I read only to get information that I need.",
            ST24Q09="I cannot sit still and read for more than a few minutes.",
            ST24Q10="I like to express my opinions about books I have read.",
            ST24Q11="I like to exchange books with my friends."))
```

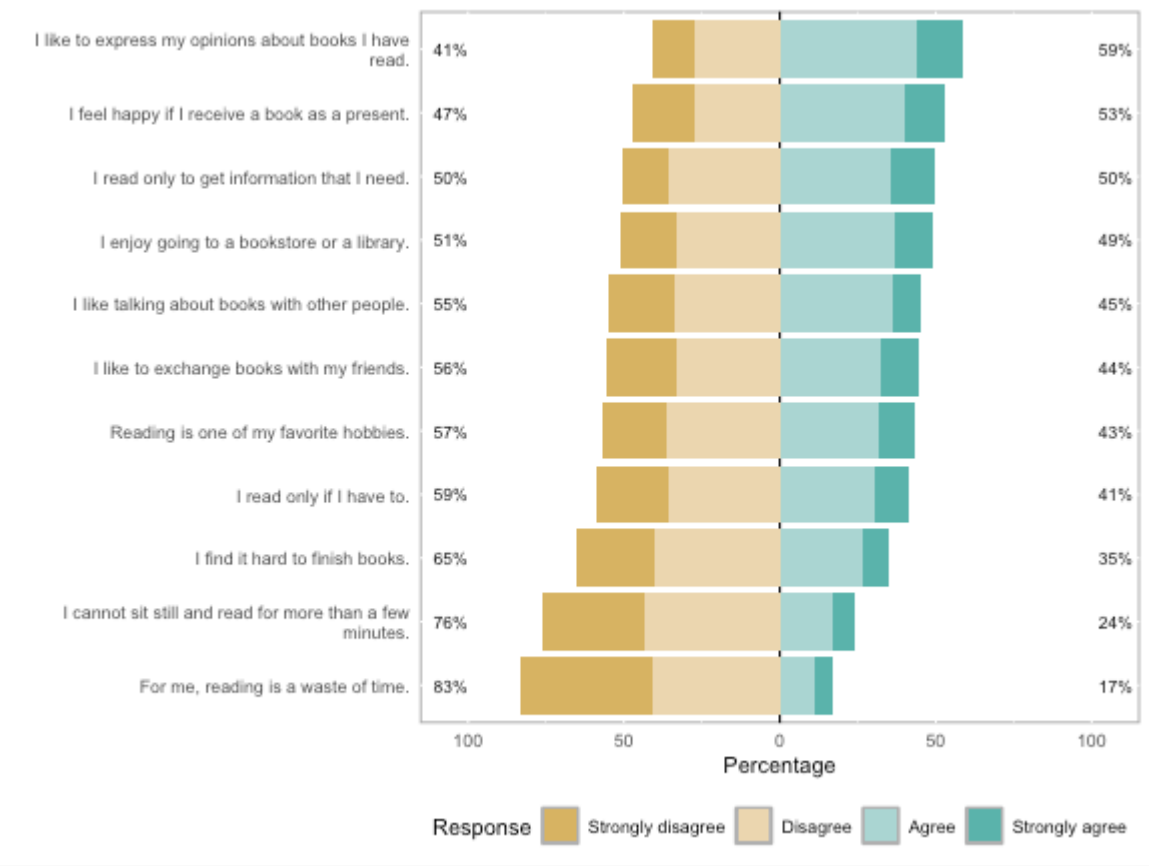
likert R Package

```
l24 <- likert(items24)
summary(l24)
```

```
##                                                      Item      low neutral     high     mean        sd
## 10   I like to express my opinions about books I have read. 41.07516       0 58.92484 2.604913 0.9009968
## 5            I feel happy if I receive a book as a present. 46.93475       0 53.06525 2.466751 0.9446590
## 8               I read only to get information that I need. 50.39874       0 49.60126 2.484616 0.9089688
## 7                I enjoy going to a bookstore or a library. 51.21231       0 48.78769 2.428508 0.9164136
## 3             I like talking about books with other people. 54.99129       0 45.00871 2.328049 0.9090326
## 11               I like to exchange books with my friends. 55.54115       0 44.45885 2.343193 0.9609234
## 2                   Reading is one of my favorite hobbies. 56.64470       0 43.35530 2.344530 0.9277495
## 1                                I read only if I have to. 58.72868       0 41.27132 2.291811 0.9369023
## 4                          I find it hard to finish books. 65.35125       0 34.64875 2.178299 0.8991628
## 9  I cannot sit still and read for more than a few minutes. 76.24524       0 23.75476 1.974736 0.8793028
## 6                       For me, reading is a waste of time. 82.88729       0 17.11271 1.810093 0.8611554
```

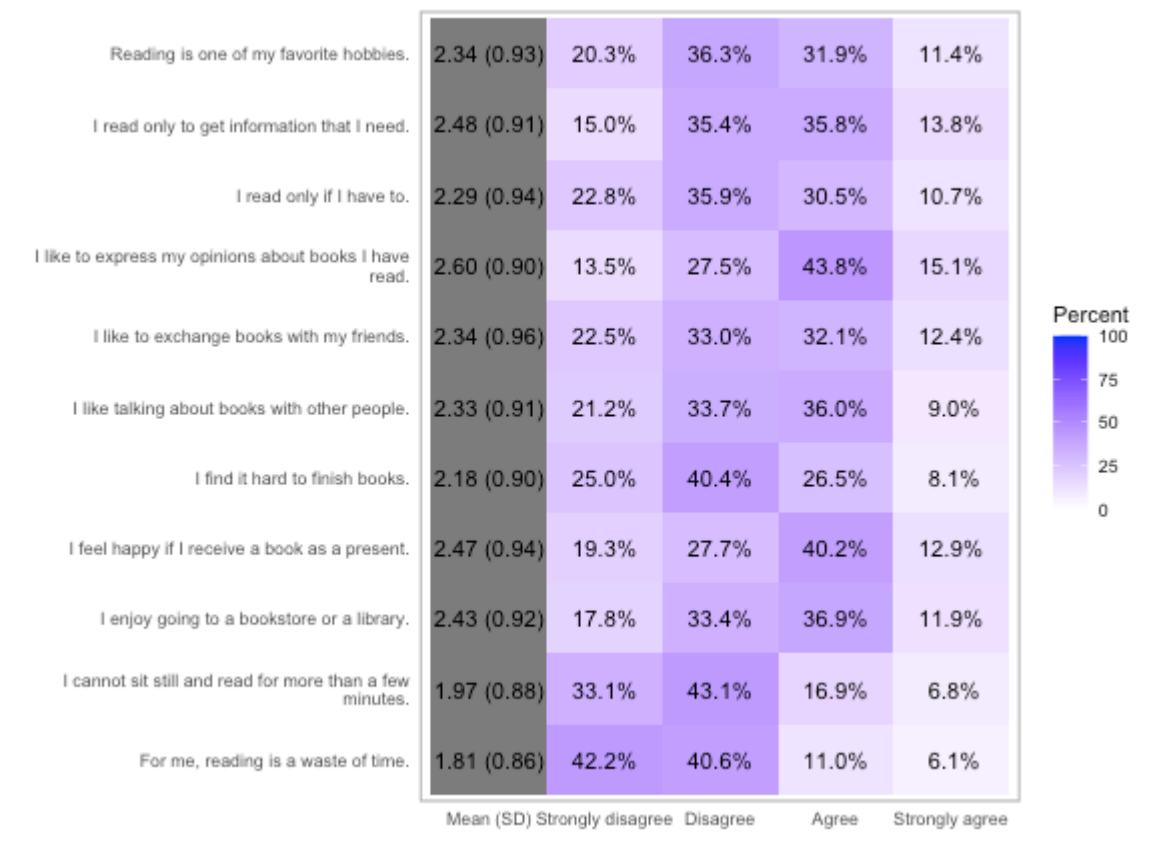
likert Plots

```
plot(l24)
```



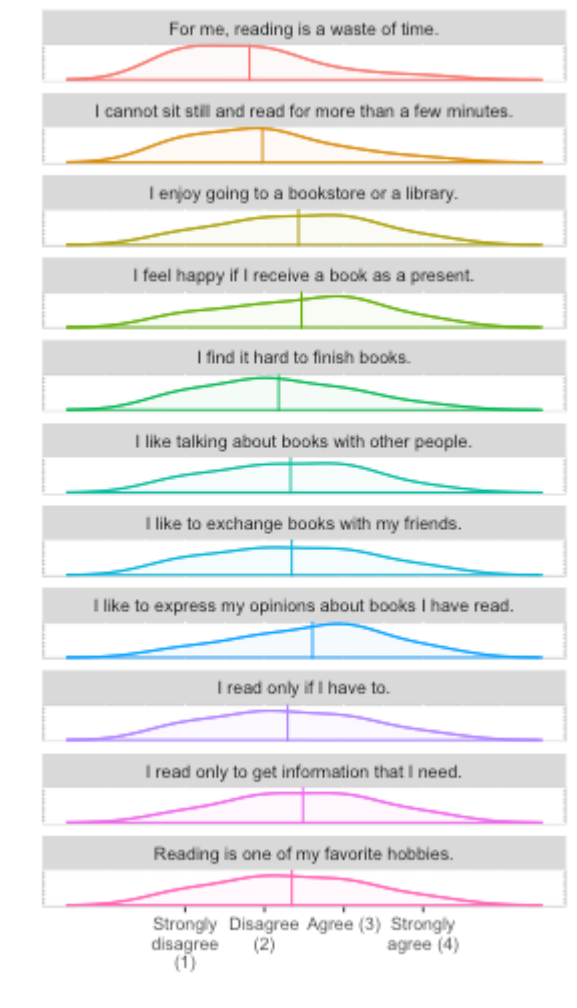
likert Plots

```
plot(l24, type='heat')
```



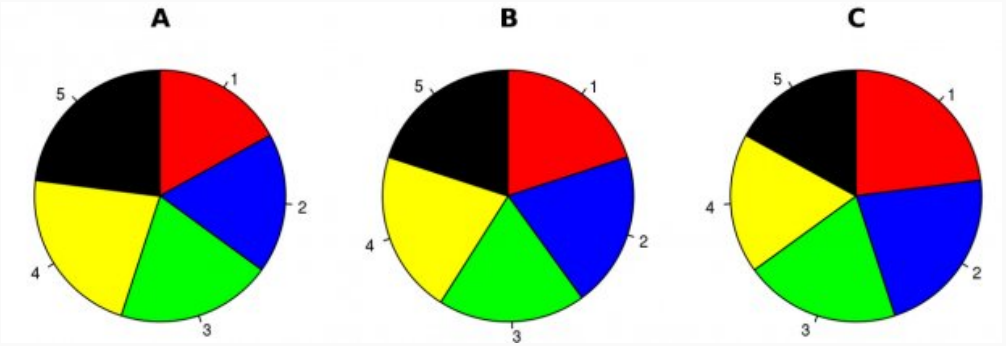
likert Plots

```
plot(l24, type='density')
```



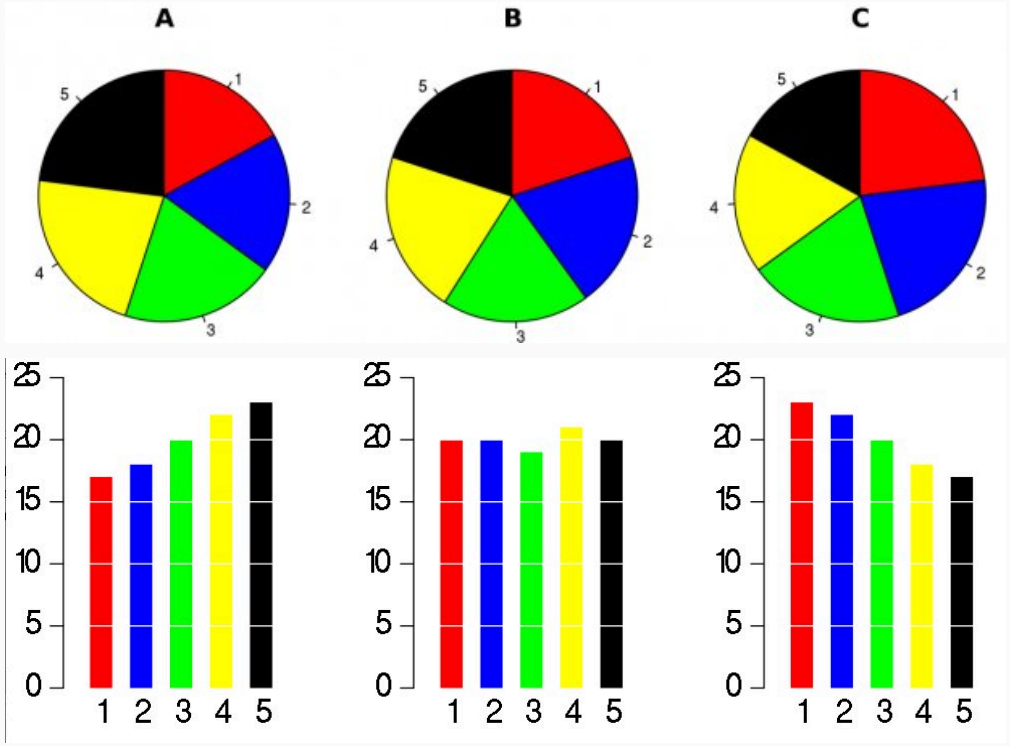
Pie Charts

There is only one pie chart in *OpenIntro Statistics* (Diez, Barr, & Çetinkaya-Rundel, 2015, p. 48). Consider the following three pie charts that represent the preference of five different colors. Is there a difference between the three pie charts? This is probably a difficult to answer.



Pie Charts

There is only one pie chart in *OpenIntro Statistics* (Diez, Barr, & Çetinkaya-Rundel, 2015, p. 48). Consider the following three pie charts that represent the preference of five different colors. Is there a difference between the three pie charts? This is probably a difficult to answer.



Source: https://en.wikipedia.org/wiki/Pie_chart.

"There is no data that can be displayed in a pie chart that cannot better be displayed in some other type of chart"

John Tukey

Additional Resources

For data wrangling:

- `dplyr` website: https://dplyr.tidyverse.org
- R for Data Science book: https://r4ds.had.co.nz/wrangle-intro.html
- Wrangling penguins tutorial: https://allisonhorst.shinyapps.io/dplyr-learnr/#section-welcome
- Data transformation cheat sheet: https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf

For data visualization:

- `ggplot2` website: https://ggplot2.tidyverse.org
- R for Data Science book: https://r4ds.had.co.nz/data-visualisation.html
- R Graphics Cookbook: https://r-graphics.org
- Data visualization cheat sheet: https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf

One Minute Paper

1. What was the most important thing you learned during this class?
2. What important question remains unanswered for you?

https://forms.gle/U4UXAosdjHorxY919