

Summarizing Data

DATA 606 - Statistics & Probability for Data Analytics

Jason Bryer, Ph.D. and Angela Lui, Ph.D.

September 7, 2022

Agenda

- Questions
- Data wrangling
 - Data types
 - Descriptive statistics
- Data visualization (next week)
 - Grammar of graphics
 - Types of graphics

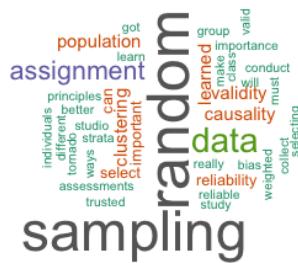


Image source: [@allison_horst](#)

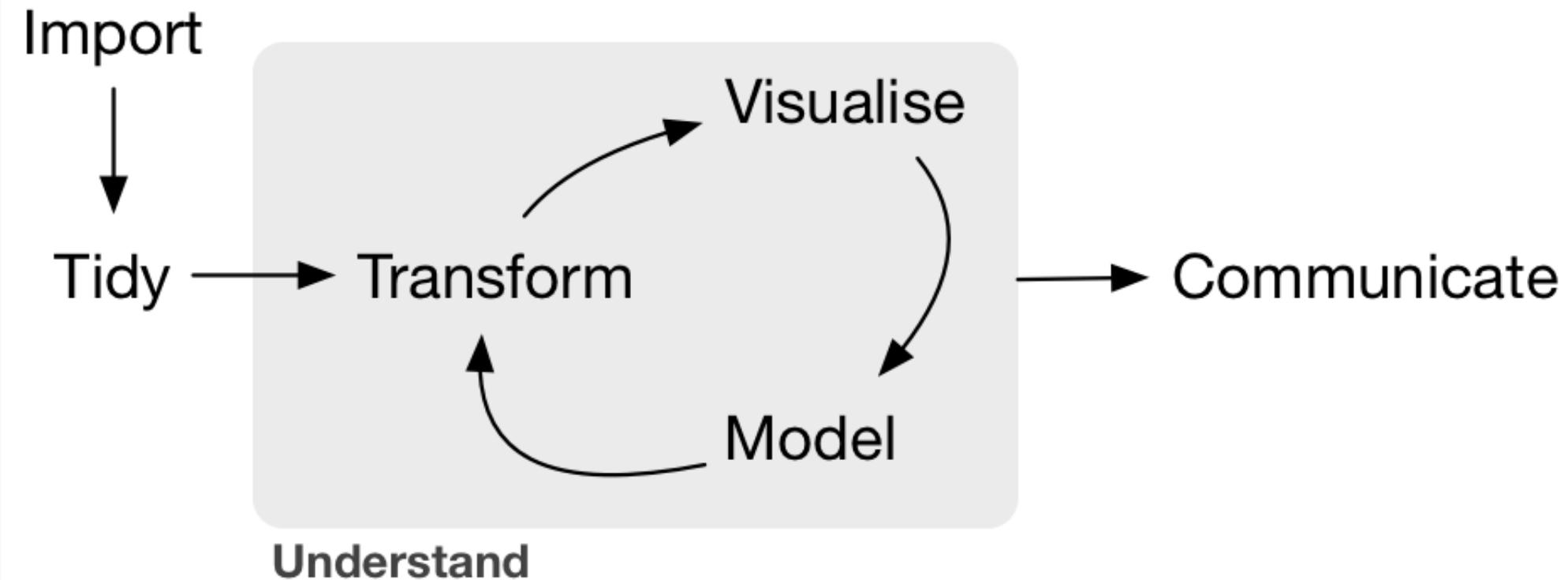
One Minute Paper Results

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

What important question remains unanswered for you?



Workflow



Source: Wickham & Gromelund, 2017



Tidy Data

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

See Wickham (2014) [Tidy data](#).

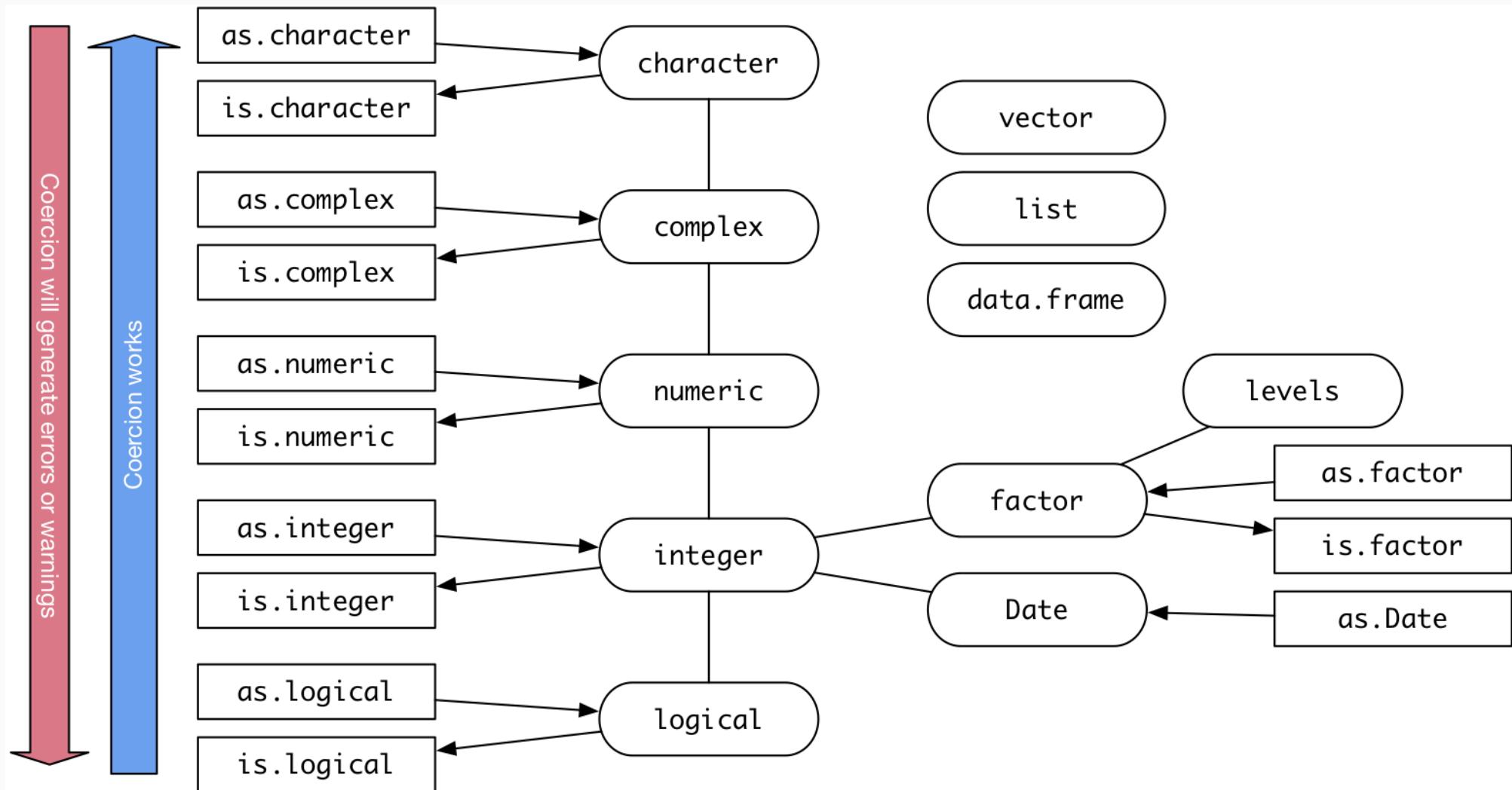


Types of Data

- Numerical (quantitative)
 - Continuous
 - Discrete
- Categorical (qualitative)
 - Regular categorical
 - Ordinal



Data Types in R



Data Types / Descriptives / Visualizations

Data Type	Descriptive Stats	Visualization
Continuous	mean, median, mode, standard deviation, IQR	histogram, density, box plot
Discrete	contingency table, proportional table, median	bar plot
Categorical	contingency table, proportional table	bar plot
Ordinal	contingency table, proportional table, median	bar plot
Two quantitative	correlation	scatter plot
Two qualitative	contingency table, chi-squared	mosaic plot, bar plot
Quantitative & Qualitative	grouped summaries, ANOVA, t-test	box plot



Variance

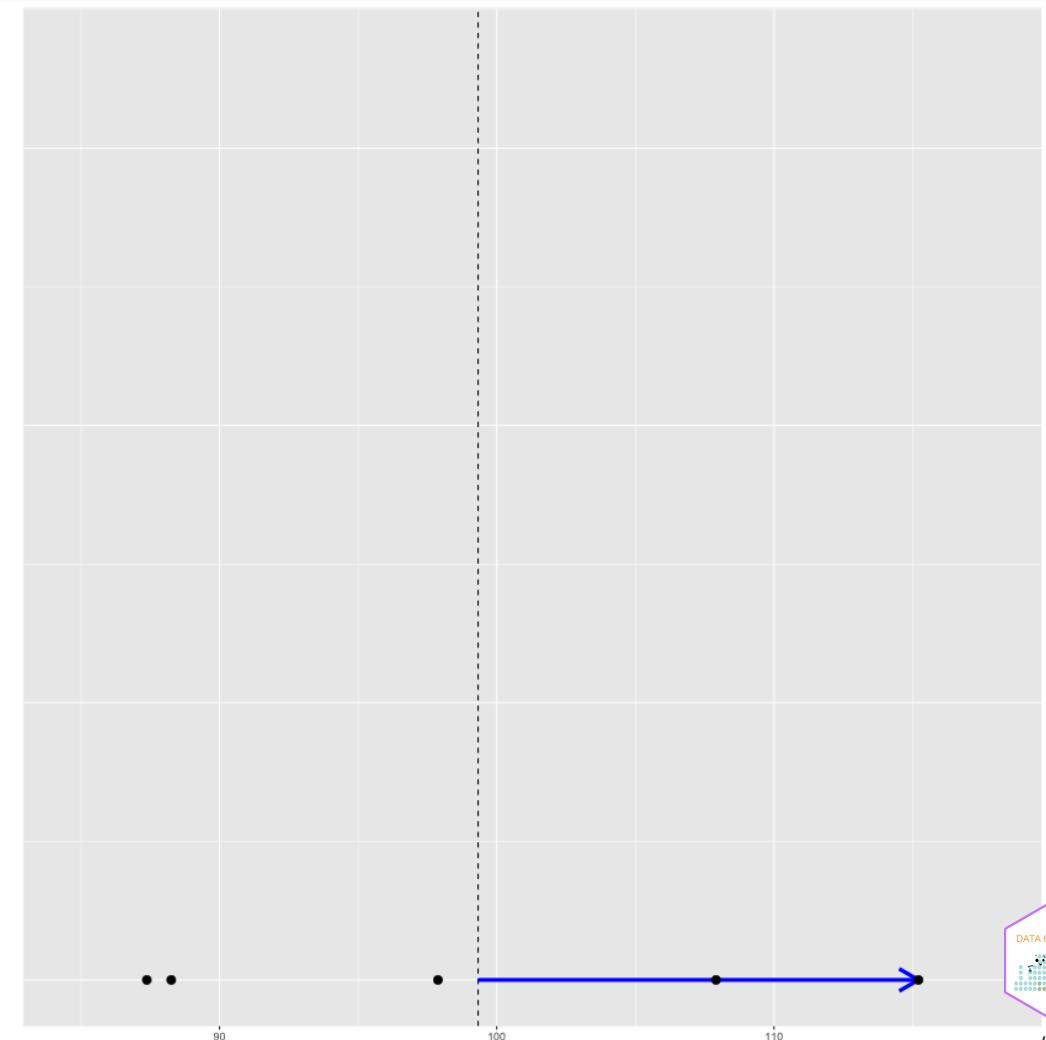
Population Variance:

$$S^2 = \frac{\Sigma(x_i - \bar{x})^2}{N}$$

Consider a dataset with five values (black points in the figure). For the largest value, the deviance is represented by the blue line ($x_i - \bar{x}$).

See also:

<https://shiny.rit.albany.edu/stat/visualizess/>
<https://github.com/jbryer/VisualStats/>

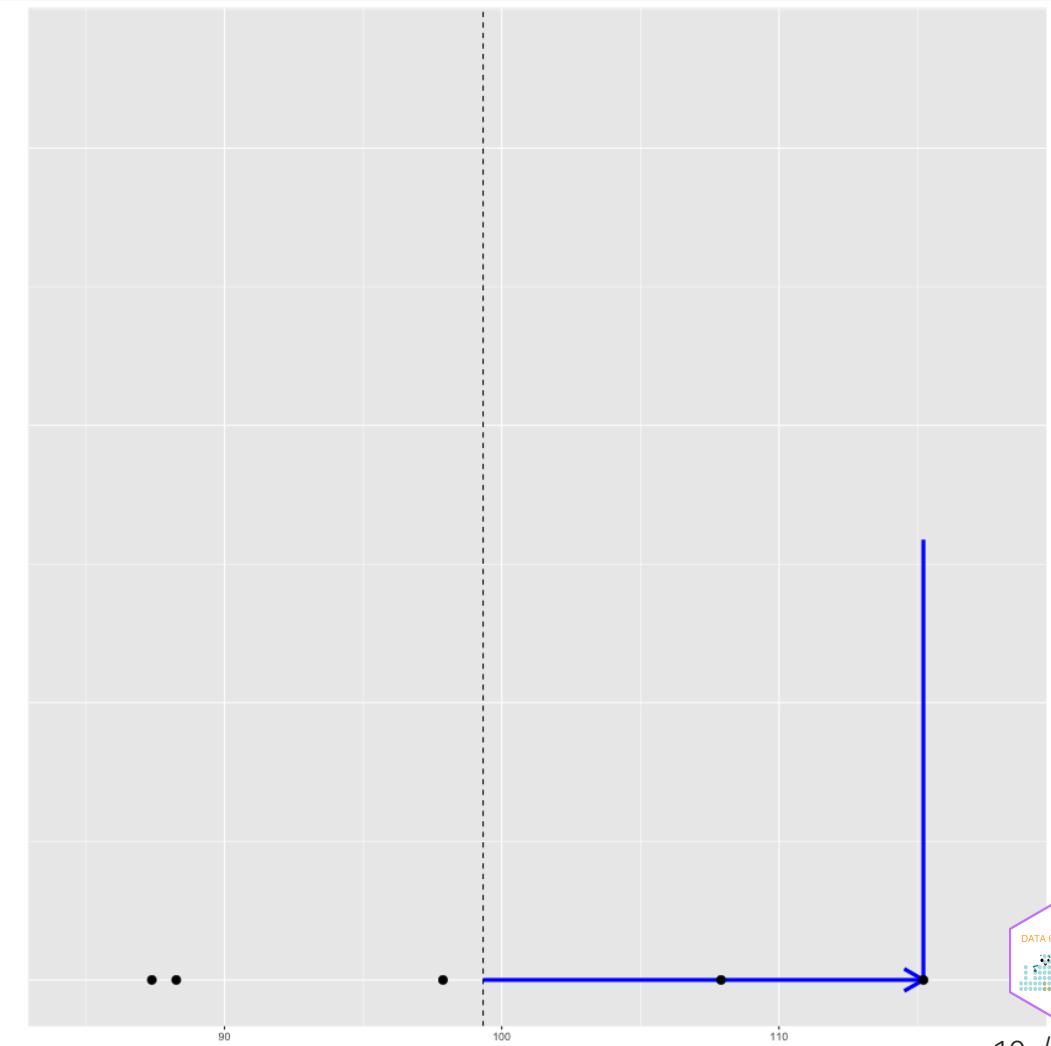


Variance (cont.)

Population Variance:

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{N}$$

In the numerator, we square each of these deviances. We can conceptualize this as a square. Here, we add the deviance in the y direction.

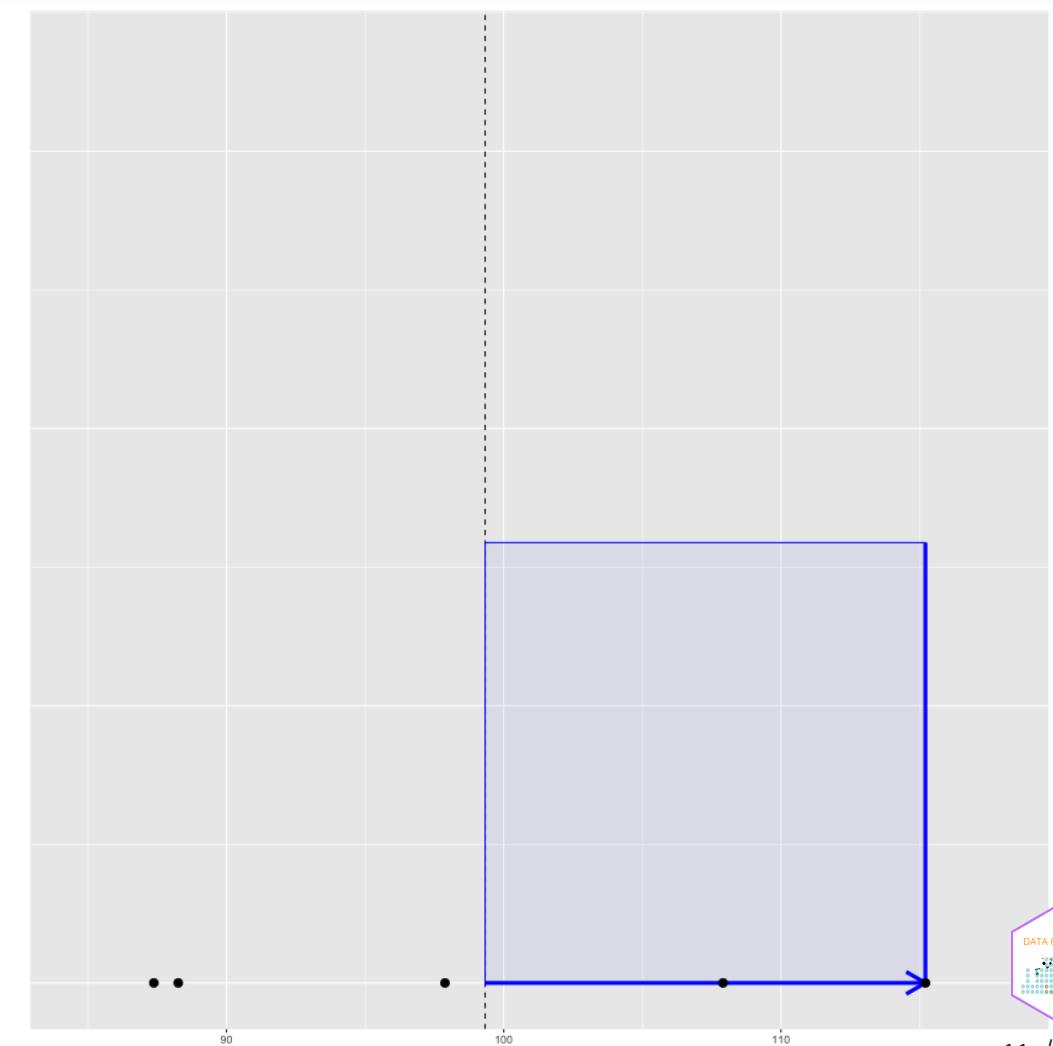


Variance (cont.)

Population Variance:

$$S^2 = \frac{\Sigma(x_i - \bar{x})^2}{N}$$

We end up with a square.

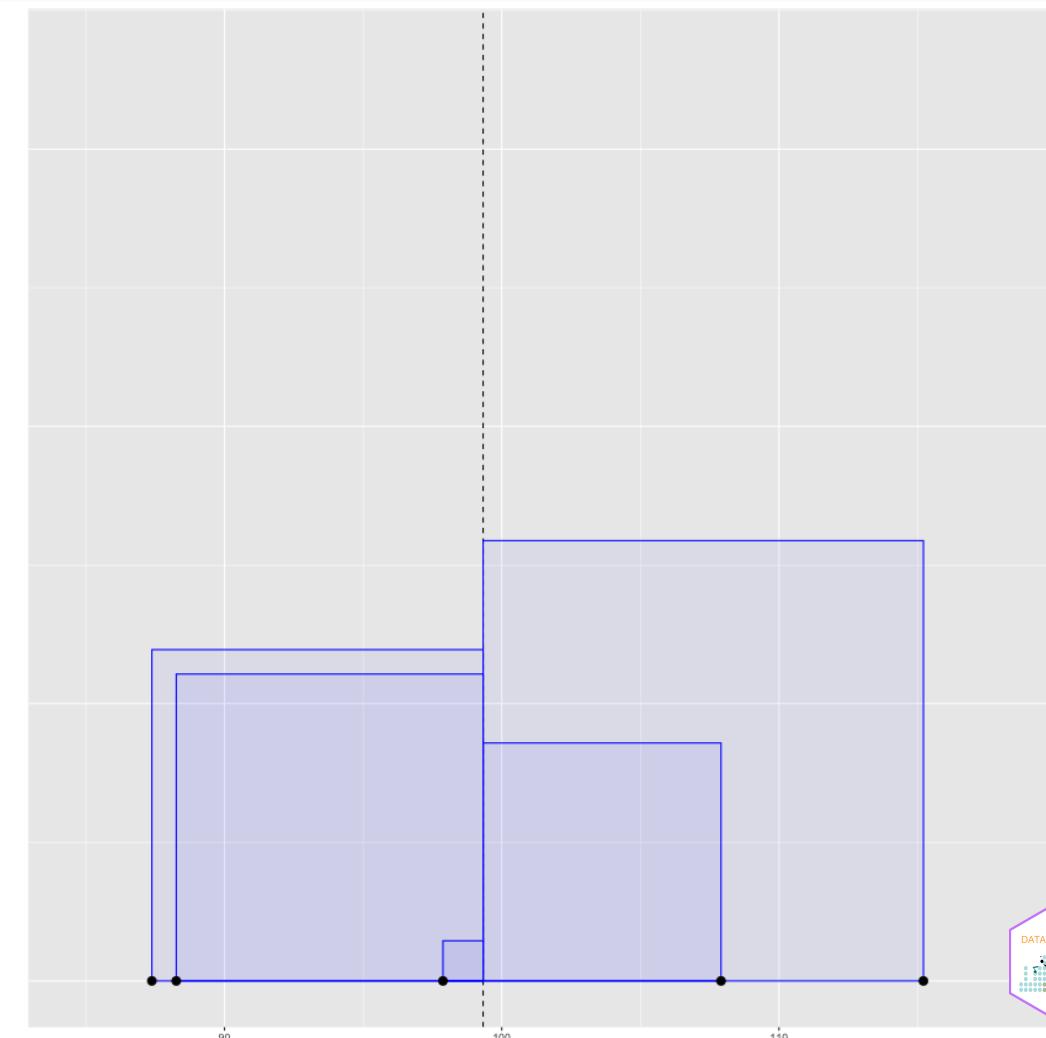


Variance (cont.)

Population Variance:

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{N}$$

We can plot the squared deviance for all the data points. That is, each component in the numerator is the area of each of these squares.

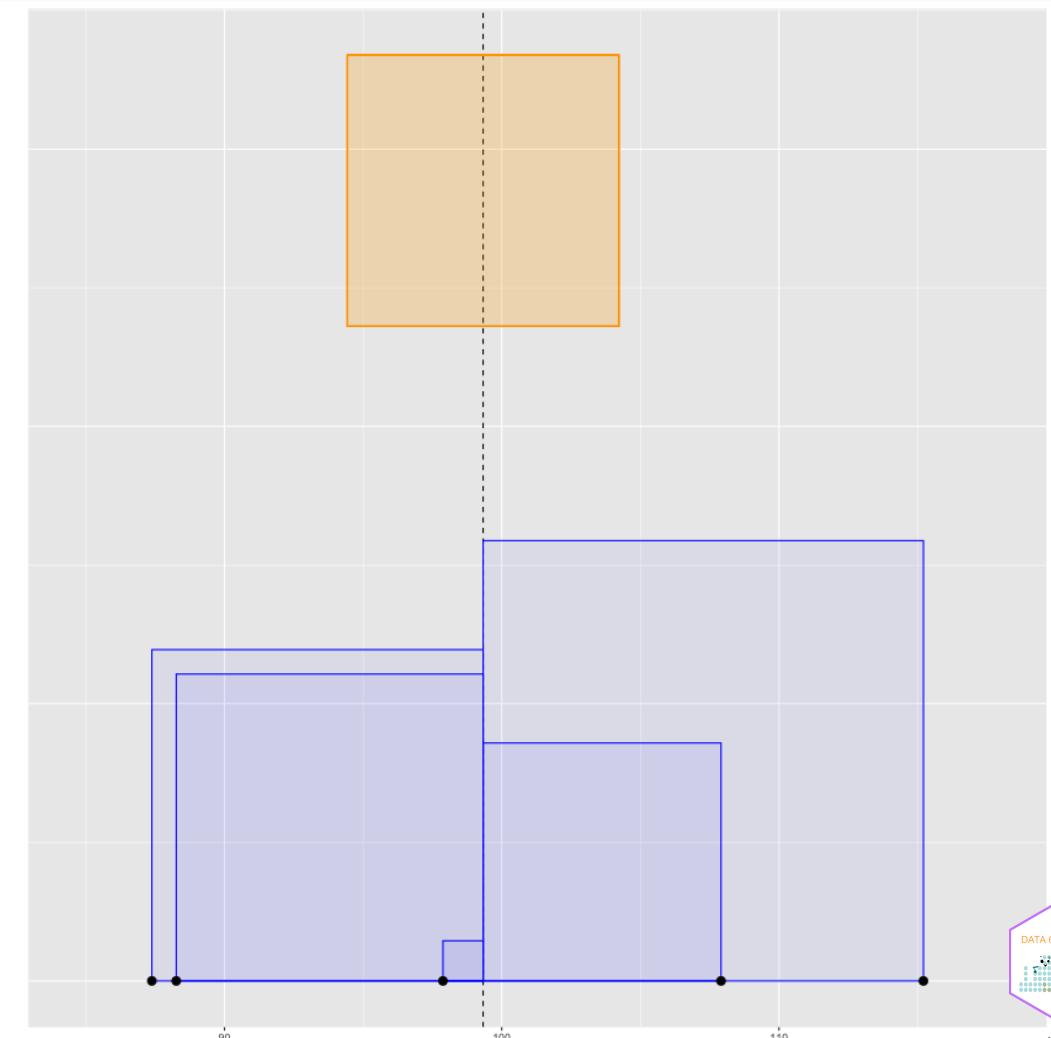


Variance (cont.)

Population Variance:

$$s^2 = \frac{\Sigma(x_i - \bar{x})^2}{N}$$

The variance is therefore the average of the area of all these squares, here represented by the orange square.



Population versus Sample Variance

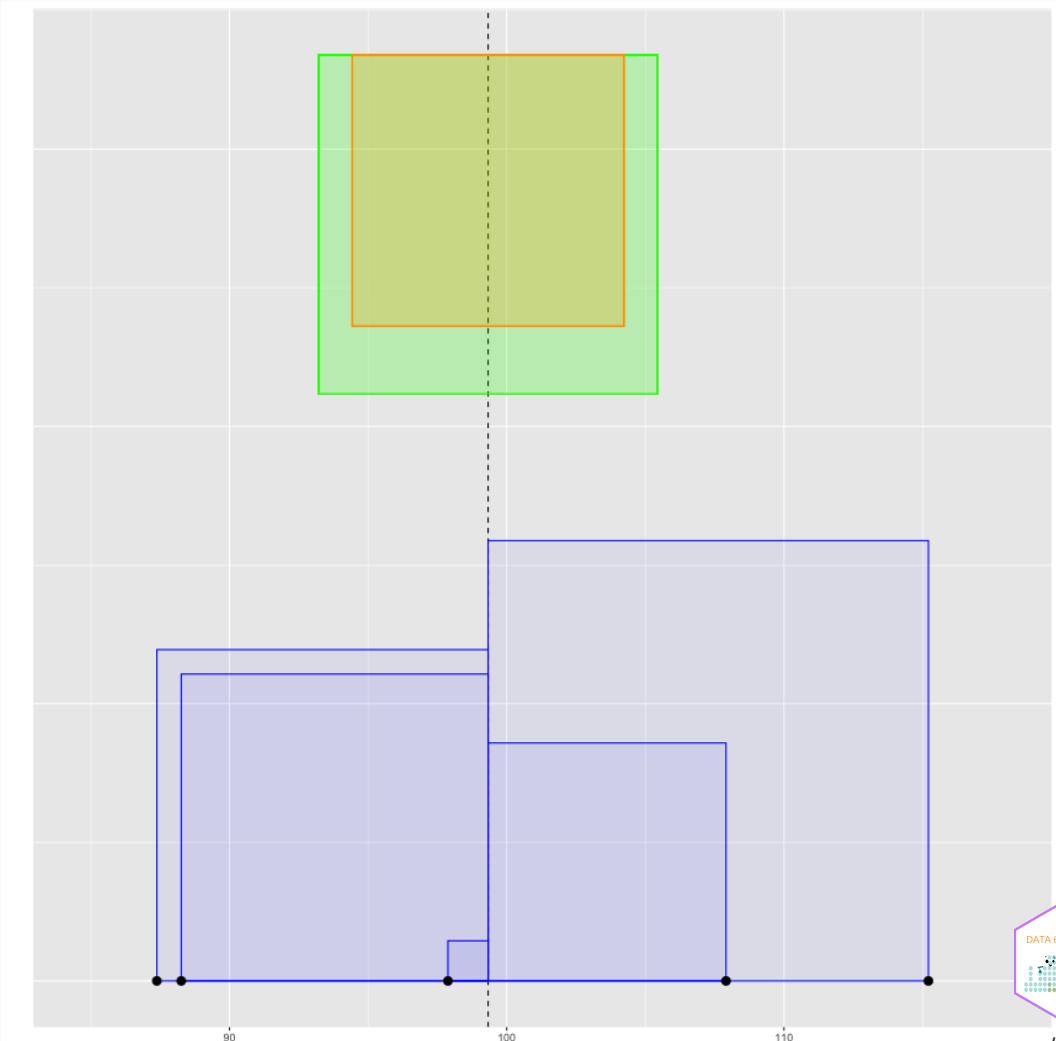
Typically we want the sample variance. The difference is we divide by $n - 1$ to calculate the sample variance. This results in a slightly larger area (variance) then if we divide by n .

Population Variance (yellow):

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{N}$$

Sample Variance (green):

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n - 1}$$



Robust Statistics

Consider the following data randomly selected from the normal distribution:

```
set.seed(41)
x <- rnorm(30, mean = 100, sd = 15)
mean(x); sd(x)
```

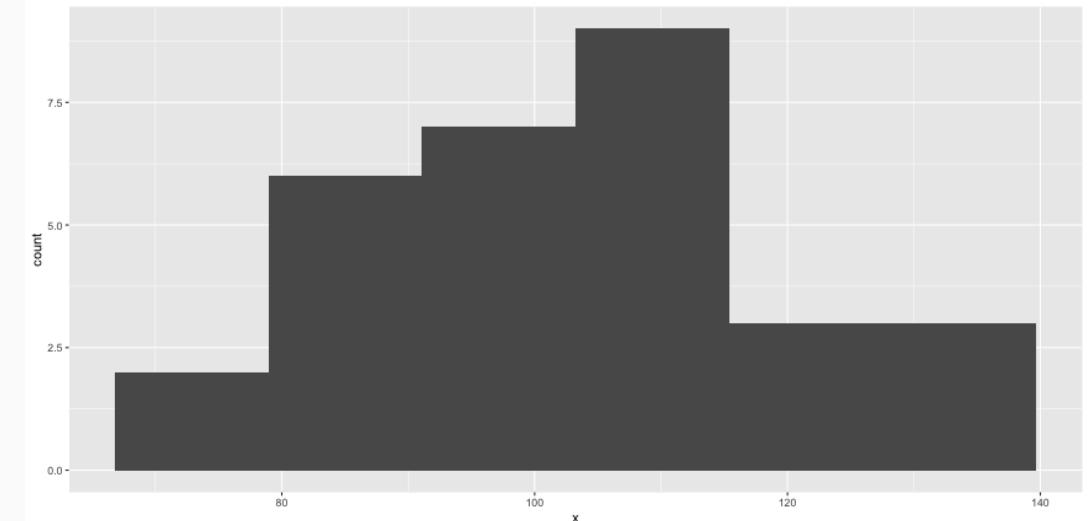
```
## [1] 103.1934
```

```
## [1] 16.8945
```

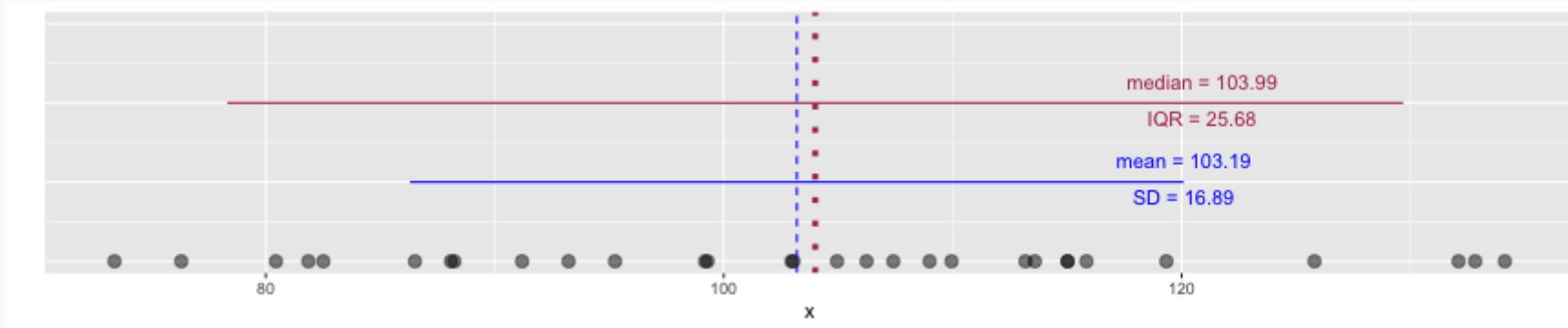
```
median(x); IQR(x)
```

```
## [1] 103.9947
```

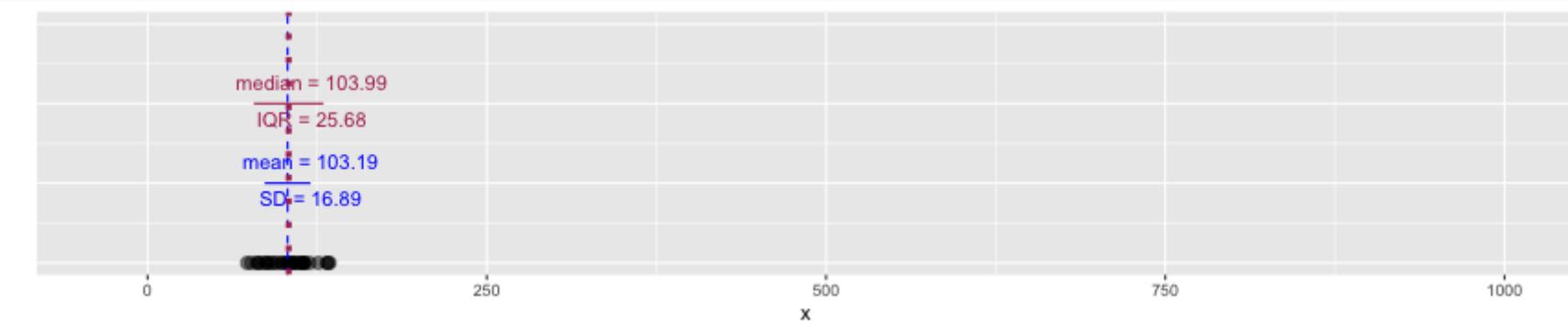
```
## [1] 25.68004
```



Robust Statistics

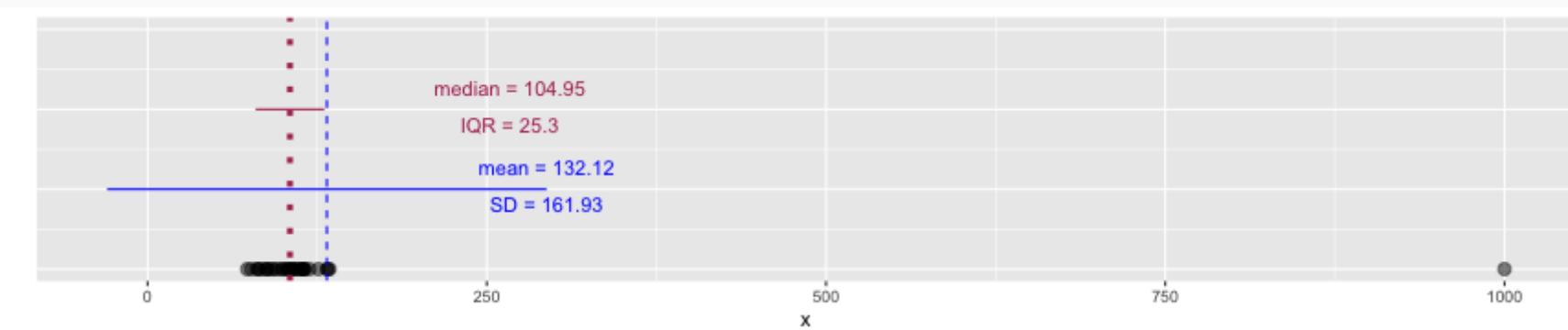


Robust Statistics



Let's add an extreme value:

```
x <- c(x, 1000)
```



Robust Statistics

Median and IQR are more robust to skewness and outliers than mean and SD. Therefore,

- for skewed distributions it is often more helpful to use median and IQR to describe the center and spread
- for symmetric distributions it is often more helpful to use the mean and SD to describe the center and spread





About legosets

To install the `brickset` package:

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

To load the `legosets` dataset.

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

The `legosets` data has 16355 observations of 34 variables.

```
names(legosets)
```

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



Structure (str)

str(legosets)

```
## 'data.frame': 16355 obs. of 34 variables:
## $ setID      : int 7693 7695 7697 7698 25534 ...
## $ 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 ...
## $ theme      : chr "Minitalia" "Minitalia" "Minitalia" "Minitalia" ...
## $ themeGroup : chr "Vintage" "Vintage" "Vintage" "Vintage" ...
## $ subtheme   : chr NA NA NA ...
## $ category   : chr "Normal" "Normal" "Normal" "Normal" ...
## $ released   : logi 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 ...
## $ 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 ...
## $ reviewCount: int 0 0 1 0 0 0 1 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 ...
## $ US_retailPrice: num NA NA NA NA NA 1.99 NA NA 4.99 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 ...
## $ UK_dateFirstAvailable: Date, format: NA NA ...
## $ UK_dateLastAvailable: Date, format: NA NA ...
## $ CA_retailPrice: num 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 ...
## $ 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 ...
## $ depth       : num NA NA NA NA NA NA NA NA 5.08 NA ...
## $ weight      : num 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" ...
## $ 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" ...
```

RStudio Environment tab can help



Environment History Connections Git Tutorial

Import Dataset | List |

R Global Environment |

Data

legosets	16355 obs. of 34 variables
\$ setID	: int 7693 7695 7697 7698 25534 ...
\$ name	: chr "Small house set" "Medium house set" "Medium house set" "L...
\$ 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 ...
\$ pieces	: int 67 109 158 233 NA 1 1 60 65 NA ...
\$ minifigs	: int NA NA NA NA NA NA NA NA NA ...
\$ bricksetURL	: chr "https://brickset.com/sets/1-8" "https://brickset.com/sets..."
\$ rating	: num 0 0 0 0 0 0 0 0 0 ...
\$ reviewCount	: int 0 0 1 0 0 0 0 1 0 0 ...
\$ packagingType	: chr "{Not specified}" "{Not specified}" "{No..."
\$ availability	: chr "{Not specified}" "{Not specified}" "{Not specified}" "{No..."
\$ agerange_min	: int NA NA NA NA NA NA NA NA NA ...
\$ US_retailPrice	: num NA NA NA NA NA 1.99 NA NA 4.99 NA ...
\$ US_dateFirstAvailable	: Date, format: NA NA NA NA ...
\$ US_dateLastAvailable	: Date, format: NA NA NA NA ...
\$ UK_retailPrice	: num NA NA NA NA NA NA NA NA NA ...
\$ UK_dateFirstAvailable	: Date, format: NA NA NA NA ...
\$ UK_dateLastAvailable	: Date, format: NA NA NA NA ...
\$ CA_retailPrice	: num NA NA NA NA NA NA NA NA NA ...
\$ CA_dateFirstAvailable	: Date, format: NA NA NA NA ...
\$ CA_dateLastAvailable	: Date, format: NA NA NA NA ...
\$ DE_retailPrice	: num NA NA NA NA NA NA NA NA NA ...
\$ DE_dateFirstAvailable	: Date, format: NA NA NA NA ...
\$ DE_dateLastAvailable	: Date, format: NA NA NA NA ...
\$ height	: num NA NA NA NA NA ...
\$ width	: num NA NA NA NA NA ...
\$ depth	: num NA NA NA NA NA NA NA 5.08 NA ...
\$ weight	: num NA NA NA NA NA NA NA NA NA ...
\$ thumbnailURL	: chr "https://images.brickset.com/sets/small/1-8.jpg" "https://..."
\$ imageURL	: chr "https://images.brickset.com/sets/images/1-8.jpg" "https://..."

Table View

Show **10** entries Search:

setID		name	year	theme	themeGroup	category	US_retailPrice	pieces	minifigs	rating
1	29512	Mummy Queen	2019	Collectable Minifigures	Miscellaneous	Normal	3.99	6	1	3.7
2	7650	Von Nebula	2010	HERO Factory	Construction	Normal	19.99	156		4.3
3	9937	Ninjago Kai ZX Kids' Watch	2012	Gear	Miscellaneous	Gear	24.99			0
4	6565	Aeroplane	2004	Creator	Model making	Normal				0
5	28932	Princess Leia Key Chain	2019	Gear	Miscellaneous	Gear	4.99			0
6	24433	Dressing table	2015	Friends	Girls	Other		22		0
7	2741	Crane and Digger Accessories	1998	Service Packs	Miscellaneous	Normal	4	14		0
8	23821	Azari and the Magical Bakery	2015	Elves	Action/Adventure	Normal	29.99	324	2	3.8
9	22536	Small Freestyle Bucket	1996	Freestyle	Basic	Normal				0
10	29301	Lady Liberty	2019	BrickHeadz	Licensed	Normal	9.99	153		4

Showing 1 to 10 of 100 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [10](#) Next



Data Wrangling Cheat Sheet

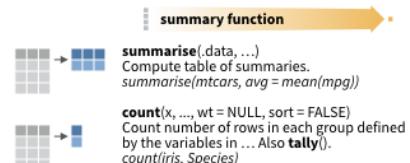
Data Transformation with dplyr :: CHEAT SHEET

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



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).

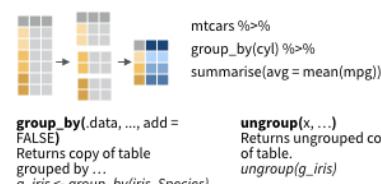


VARIATIONS

`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

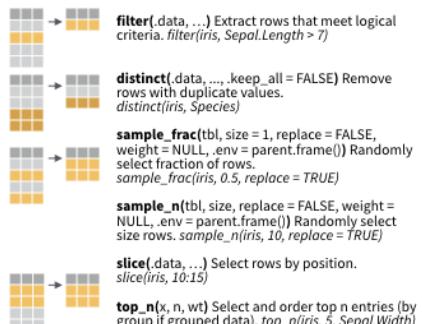
Use `group_by()` to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



Manipulate Cases

EXTRACT CASES

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

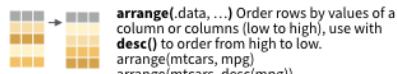


Logical and boolean operators to use with filter()

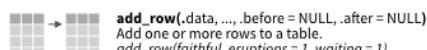
<	<=	is.na()	%in%		xor()
>	>=	!is.na()	!	&	

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

ARRANGE CASES



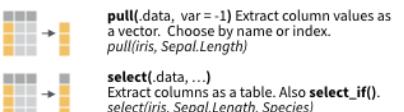
ADD CASES



Manipulate Variables

EXTRACT VARIABLES

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

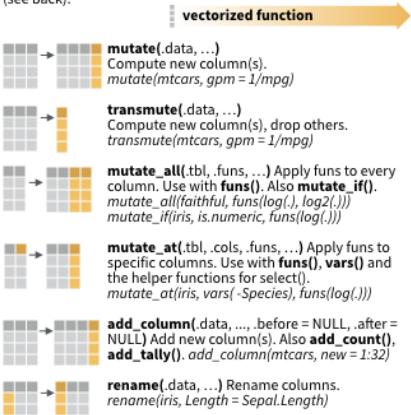


Use these helpers with `select()`,
e.g. `select(iris, starts_with("Sepal"))`

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

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).



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))`

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
## 0.028798533 0.032100275 0.025191073 0.143564659 0.713420972 0.054050749
```



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")



A cartoon illustration featuring three characters: an orange blob-like character on the left pointing at a map of a coastline with a bay labeled "BAY"; a purple character on the right holding a checkmark and an orange "X"; and a green character at the bottom holding an orange "X". Between them is a table with columns "type", "food", and "site". The first row is highlighted in pink and shows "otter", "urchin", and "bay". The second row shows "Shark", "seal", and "channel". The third row is highlighted in pink and shows "otter", "abalone", and "bay". The fourth row shows "otter", "crab", and "wharf". The green character is pointing at the "crab" entry.

type	food	site
otter	urchin	bay
Shark	seal	channel
otter	abalone	bay
otter	crab	wharf

@allisonhorst

Logical Operators

- `!a` - TRUE if a is FALSE
- `a == b` - TRUE if a and b 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 b, 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] 61
```

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        78.95       NA
```

Relocate



dplyr::**relocate()**
move COLUMNS around!

Default: move to FRONT
or move to
.before or .after
A SPECIFIED COLUMN!



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



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 78.95 NA 0.4199468  
## #> 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
```



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: 15 × 6  
##   themeGroup     mean_price    sd_price median_price      n missing  
##   <chr>          <dbl>       <dbl>        <dbl> <int>    <int>  
## 1 Action/Adventure  31.3        29.9       20.0   1280     462  
## 2 Basic            13.1        12.8       7.99    843     473  
## 3 Constraction    15.1        14.0       9.99    501     125  
## 4 Educational      89.0       107.       59.7    452     294  
## 5 Girls             23.4        22.6       15.0    677     225  
## 6 Historical       25.5        27.7       15.0    473     125  
## 7 Junior            18.6        13.2       17.8    228      93  
## 8 Licensed          42.9        58.3       25.0   2060     467  
## 9 Miscellaneous     14.3        20.8       6.99   4925    2117  
## 10 Model making     52.8        65.1       30.0    582     166  
## 11 Modern day        31.2        33.7       20.0   1723     763  
## 12 Pre-school        23.8        19.4       20.0   1487     699  
## 13 Racing             24.8        30.2       10     270      59  
## 14 Technical         60.8        68.1       40.0   550     137
```

Describe and Describe By

```
library(psych)
```

```
describe(legosets$US_retailPrice)
```

```
##      vars     n   mean    sd median trimmed    mad min     max   range skew kurtosis     se
## X1     1 9886 28.52 42 14.99   20.14 14.83    0 799.99 799.99 5.62    58.91 0.42
```

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

```
##      item           group1 vars     n   mean       sd   min     max   range     se
## X11     1 {Not specified}  1 3197 24.24484 36.282072 0.60 789.99 789.39 0.6416833
## X12     2 Educational     1    9 140.95000 86.358265 14.95 244.95 230.00 28.7860885
## X13     3 LEGO exclusive  1 1066 28.79797 70.954538 0.00 799.99 799.99 2.1732094
## X14     4 LEGOLAND exclusive 1    7 12.70429  6.447591 4.99 19.99 15.00 2.4369603
## X15     5 Not sold        1    1 12.99000       NA 12.99 12.99 0.00        NA
## X16     6 Promotional     1   167  9.19485 23.667555 0.00 249.99 249.99 1.8314504
## X17     7 Promotional (Airline) 1    11 15.79455  6.614819 5.00 28.00 23.00 1.9944429
## X18     8 Retail            1 4824 29.82030 33.270049 1.95 399.99 398.04 0.4790158
## X19     9 Retail - limited  1   600 44.64837 57.391438 0.40 379.99 379.59 2.3429956
## X110    10 Unknown          1     4  2.24750  1.253671 1.00  3.99  2.99 0.6268356
```



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



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.



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>) +
  <GEO FUNCTION> (mapping = aes(<POSITION>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE FUNCTION> +
  <FACET FUNCTION> +
  <SCALE FUNCTION> +
  <THEME FUNCTION>
```

required
Not required, supplied by defaults or supplied

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

aesthetic mappings data geom

plot(x = cyl, 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, y))

a + geom_blank()
a + geom_curve(aes(yend = lat + 1,
xend = long + curvevature * z), x, yend, y, end,
alpha, angle, color, curvature, linetype, hjust,
lineheight, size, vjust)
a + geom_path(linewidth = "butt", linejoin = "round",
x, y, alpha, color, group, linetype, size)
a + geom_polygon(aes(group = group))
a + geom_rect(aes(xmin = long, ymin = lat, xmax =
long + 1, ymax = lat + 1), x, xmin, ymax,
ymin, alpha, color, fill, linetype, size)
a + geom_ribbon(aes(ymin = unemploy - 900,
ymax = unemploy + 900), x, y, alpha, 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_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:115, 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 = 5)
c + geom_qq(aes(sample = hwy))
c2 + geom_qq(aes(sample = hwy))
```

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 = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
e + geom_hex()
e + geom_jitter(height = 2, width = 2)
e + geom_point(), x, y, alpha, color, fill, shape, size,
size, stroke
e + geom_quantile(), x, y, alpha, color, group,
linetype, size, weight
```

e + geom_rug(sides = "bl")
e + geom_smooth(method = lm)
e + geom_text(aes(label = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
f + geom_col()
f + geom_boxplot()
f + geom_dotplot(binaxis = "y", stackdir =
"center")
f + geom_violin(scale = "area")
```

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
g + geom_count(), x, y, alpha, color, fill, shape,
size, stroke
```

THREE VARIABLES

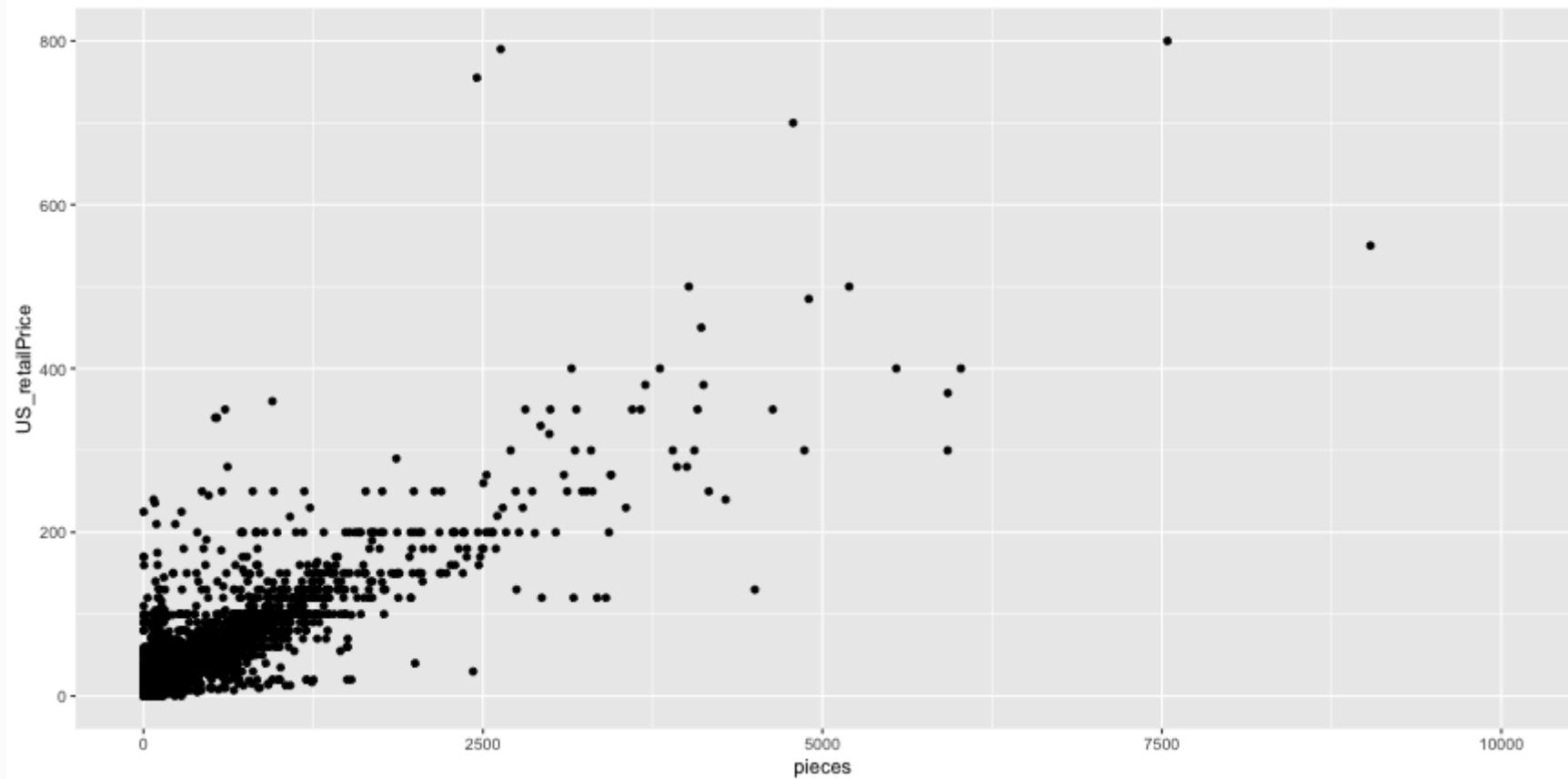
```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))
l + geom_contour(aes(z = z))
l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5,
interpolate = FALSE)
l + geom_tile(aes(fill = z)), x, y, alpha, color, fill,
linetype, size, weight
```

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



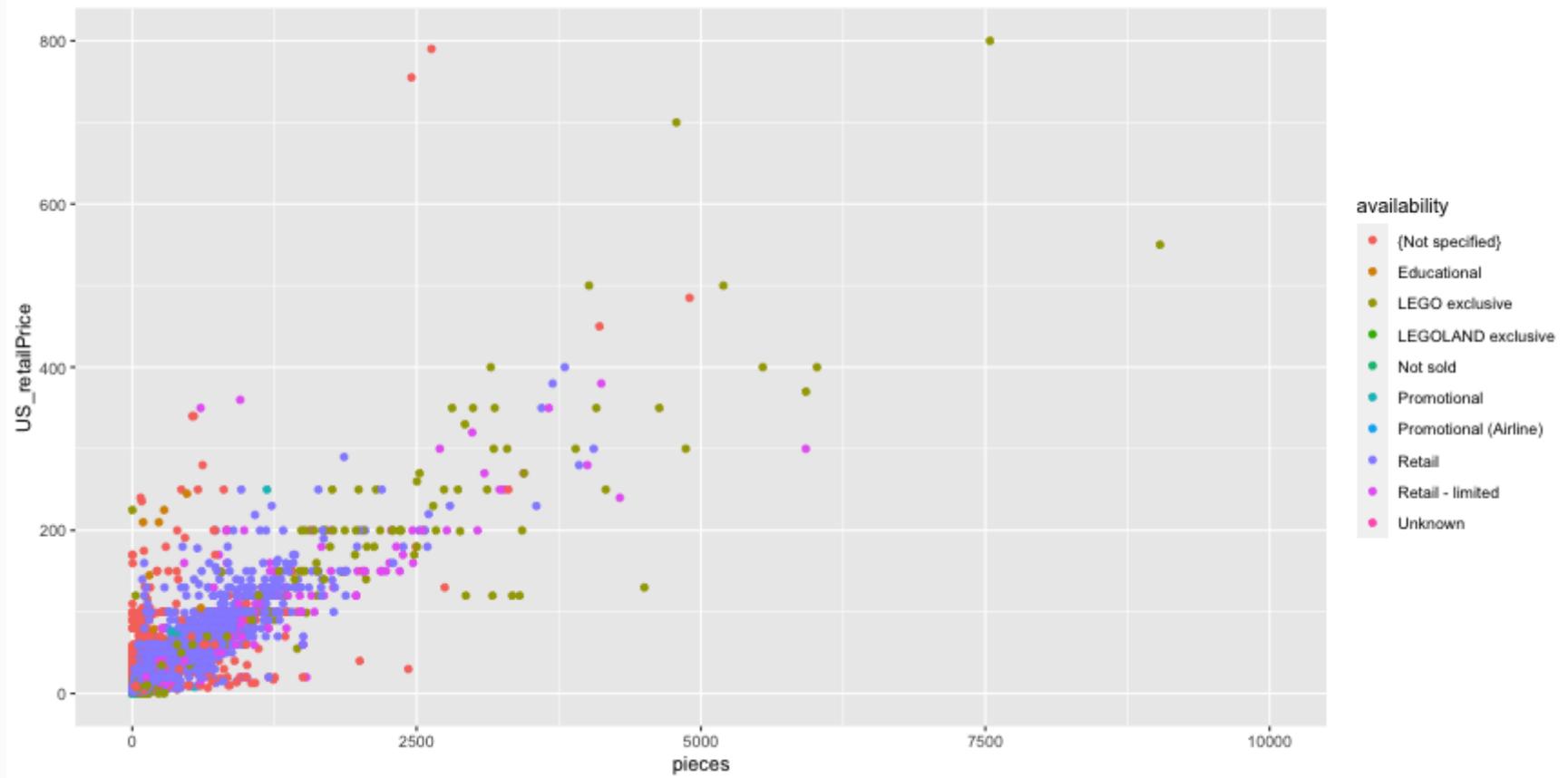
Scatterplot

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



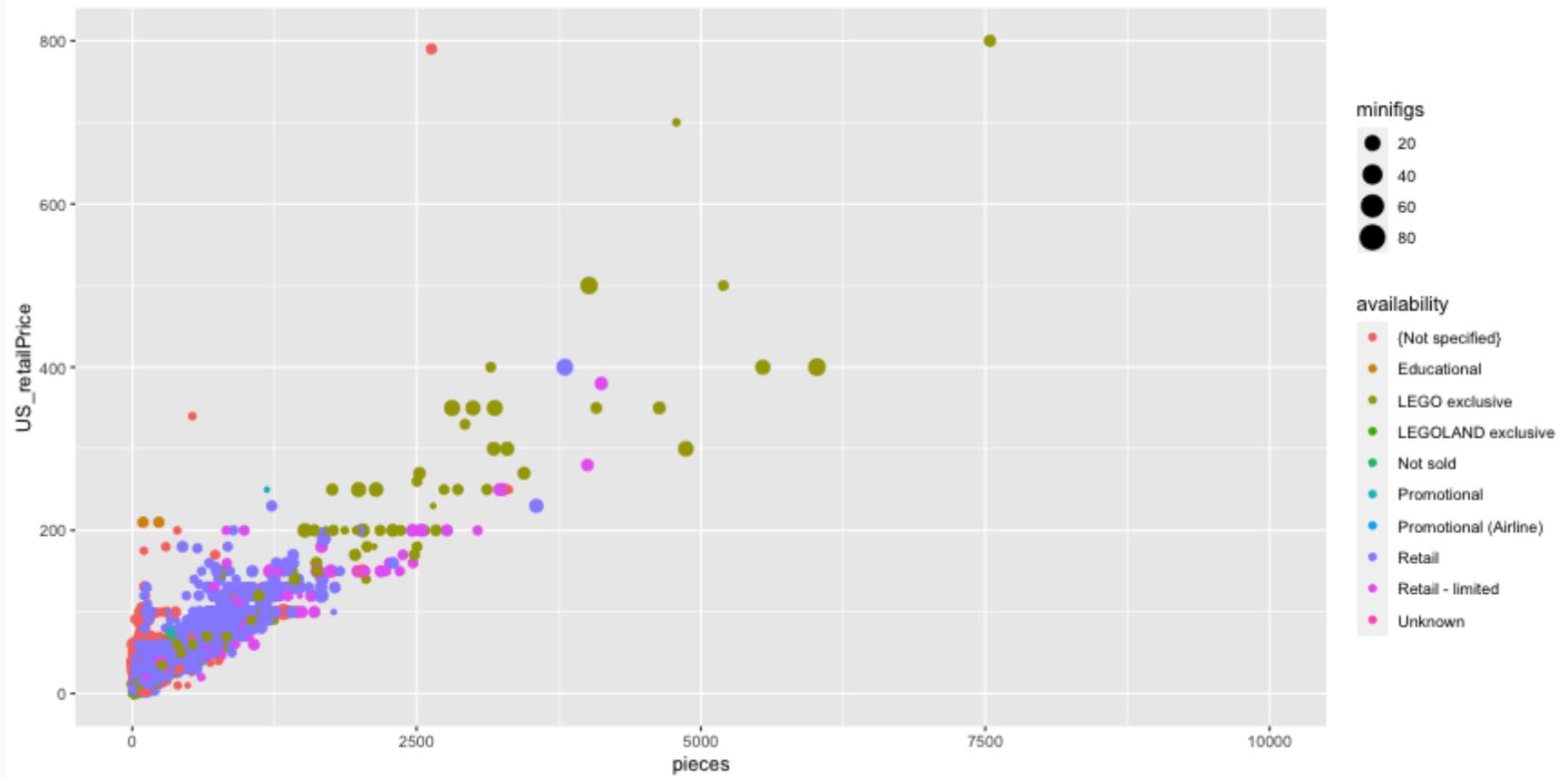
Scatterplot (cont.)

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



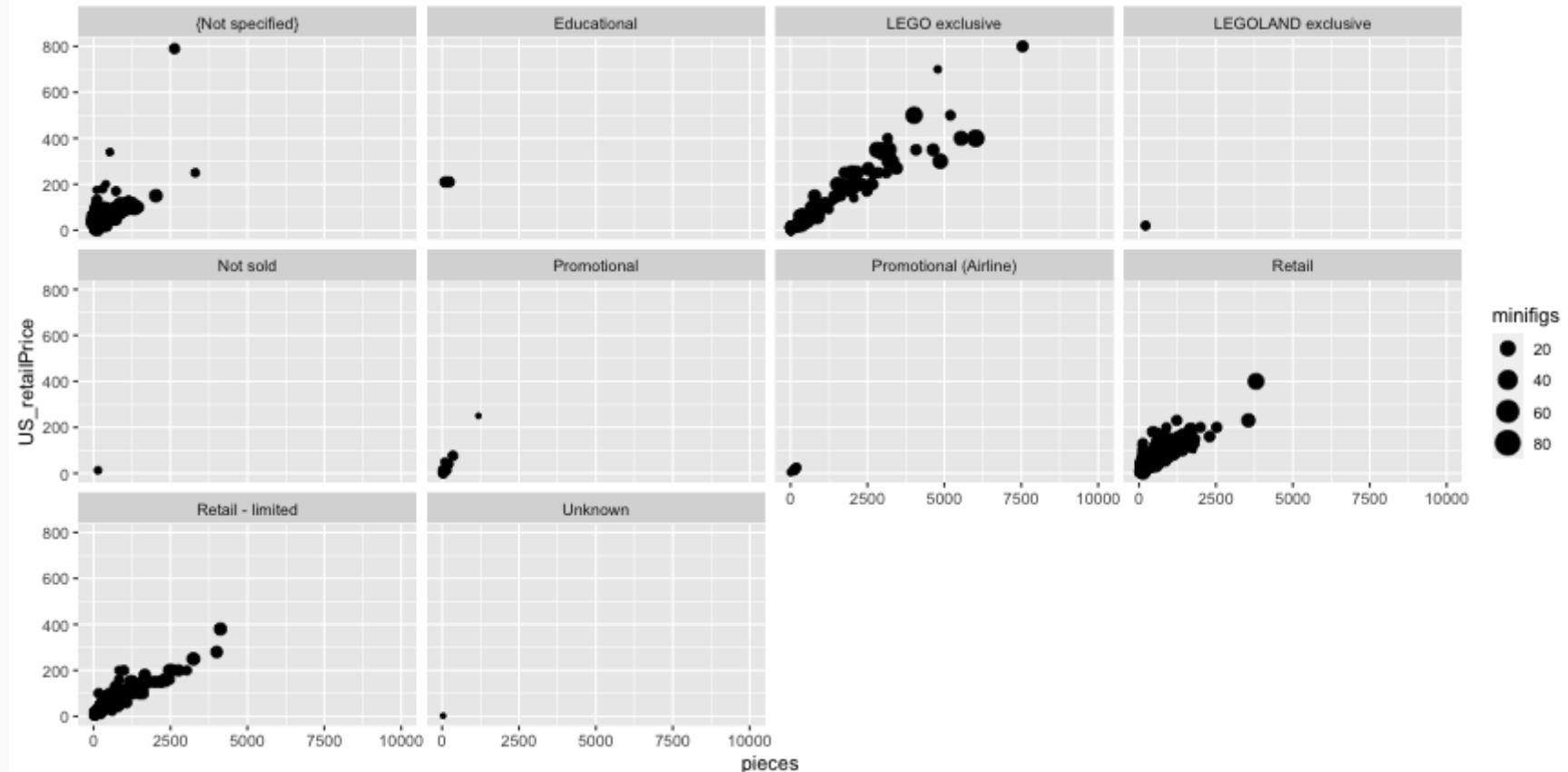
Scatterplot (cont.)

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



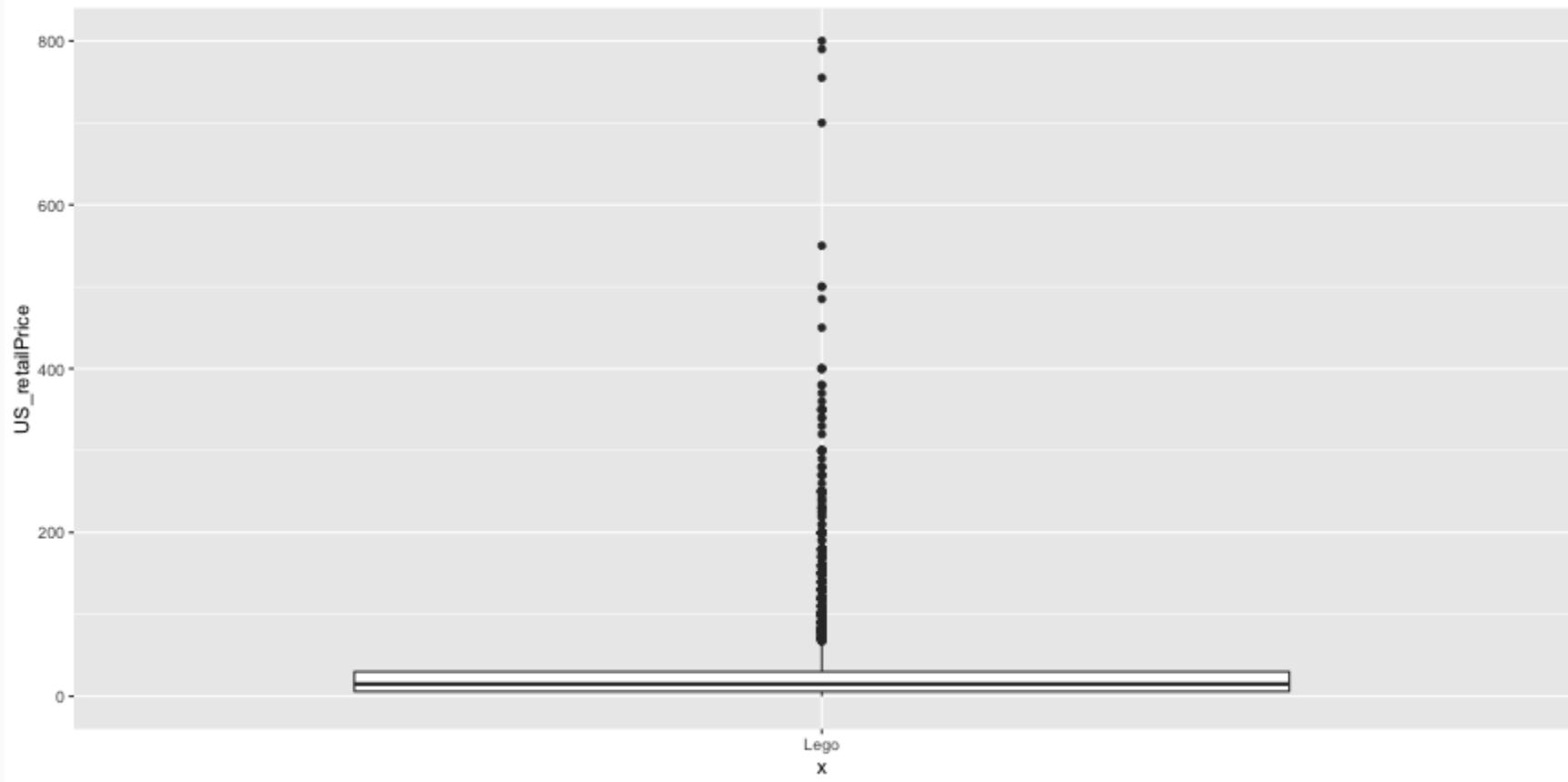
Scatterplot (cont.)

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



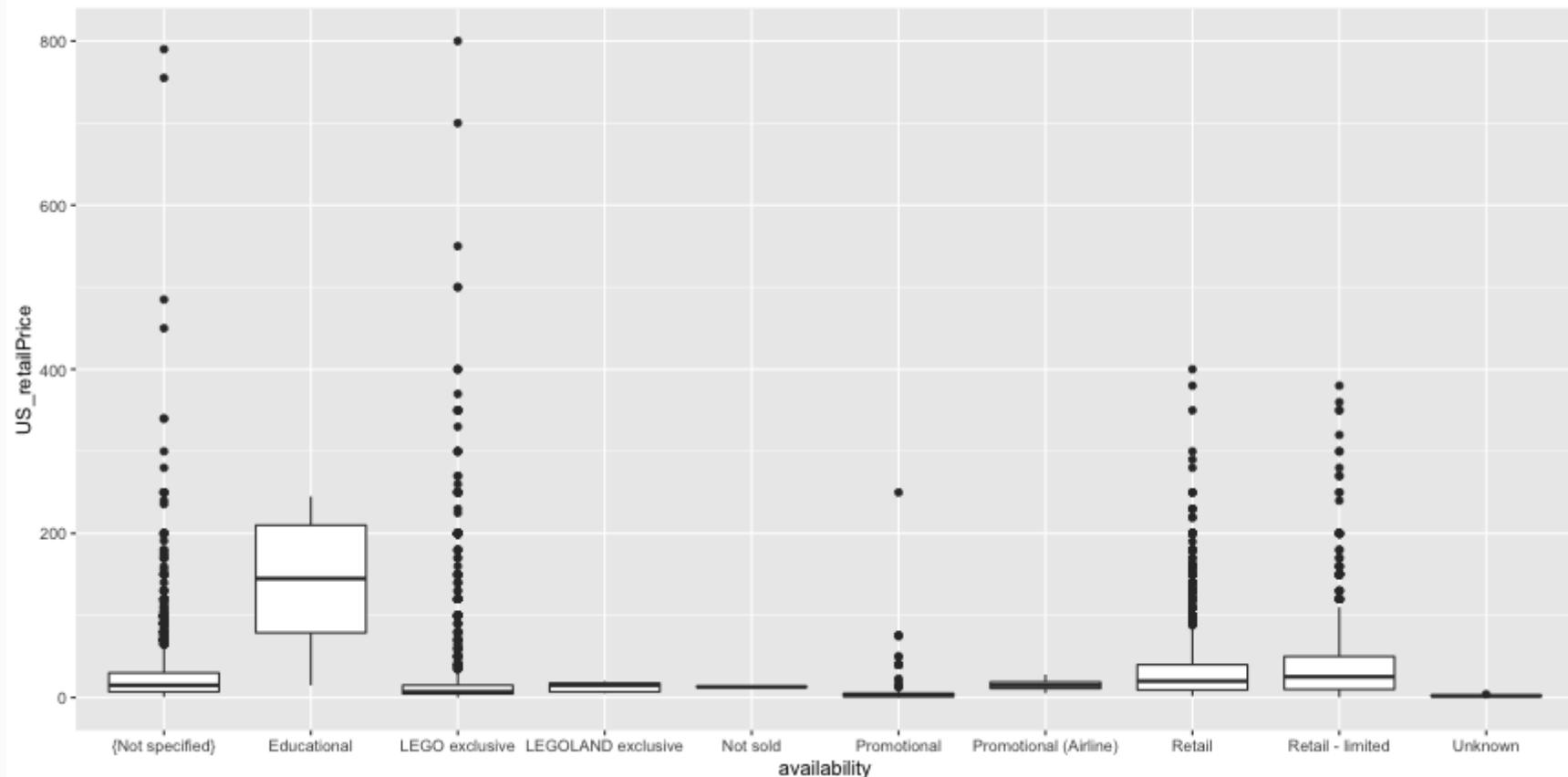
Boxplots

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



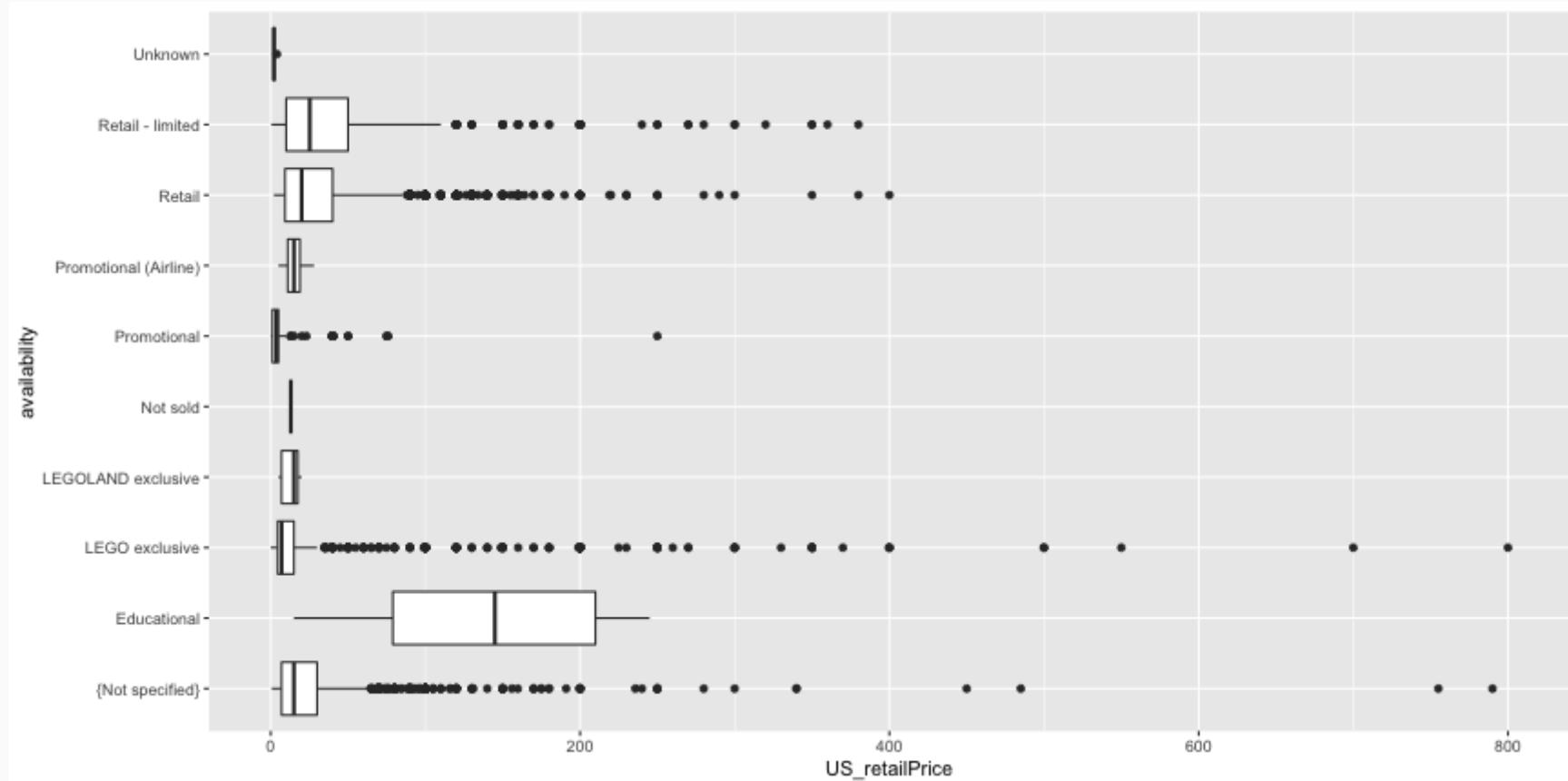
Boxplots (cont.)

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



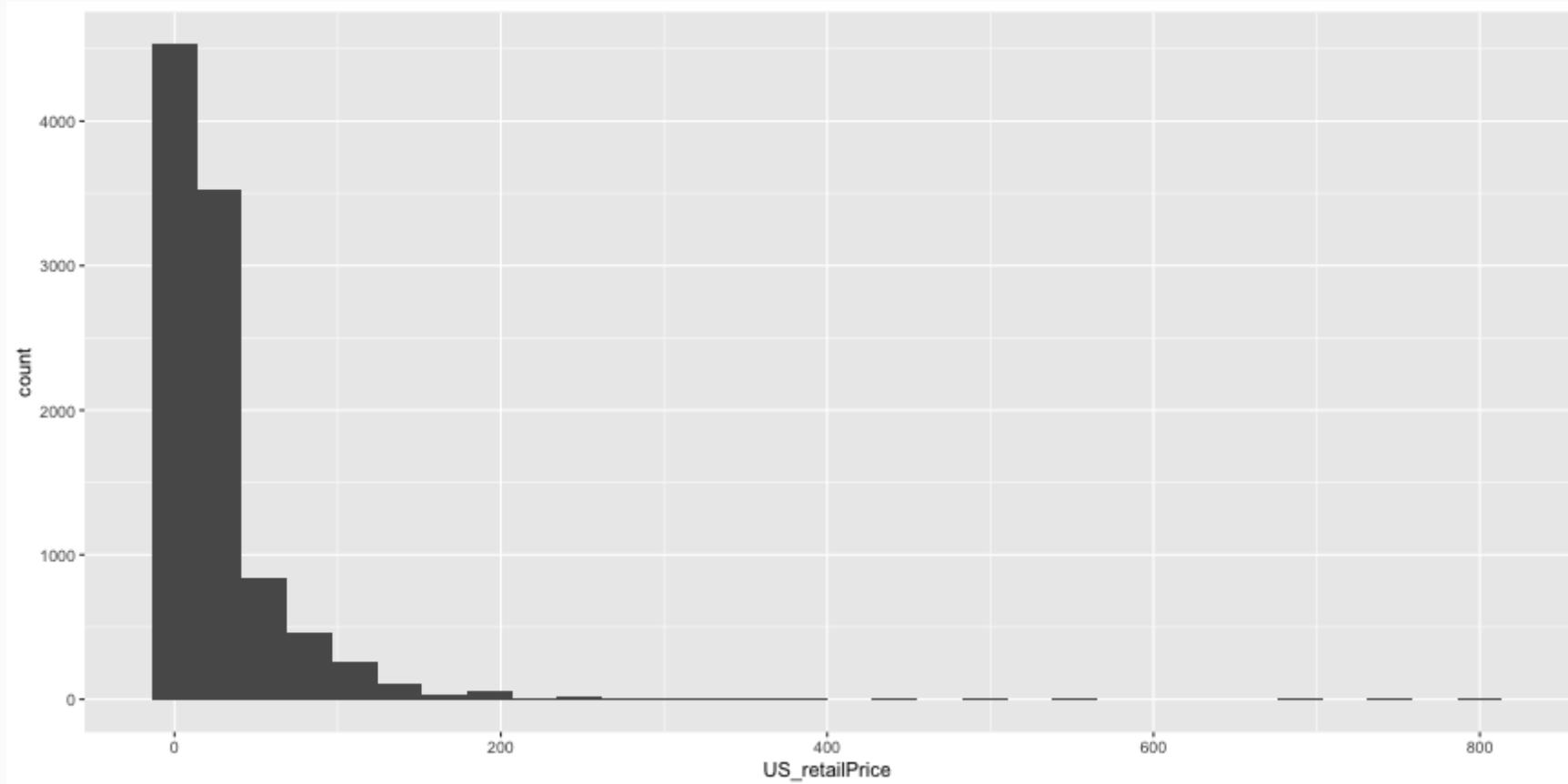
Boxplot (cont.)

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



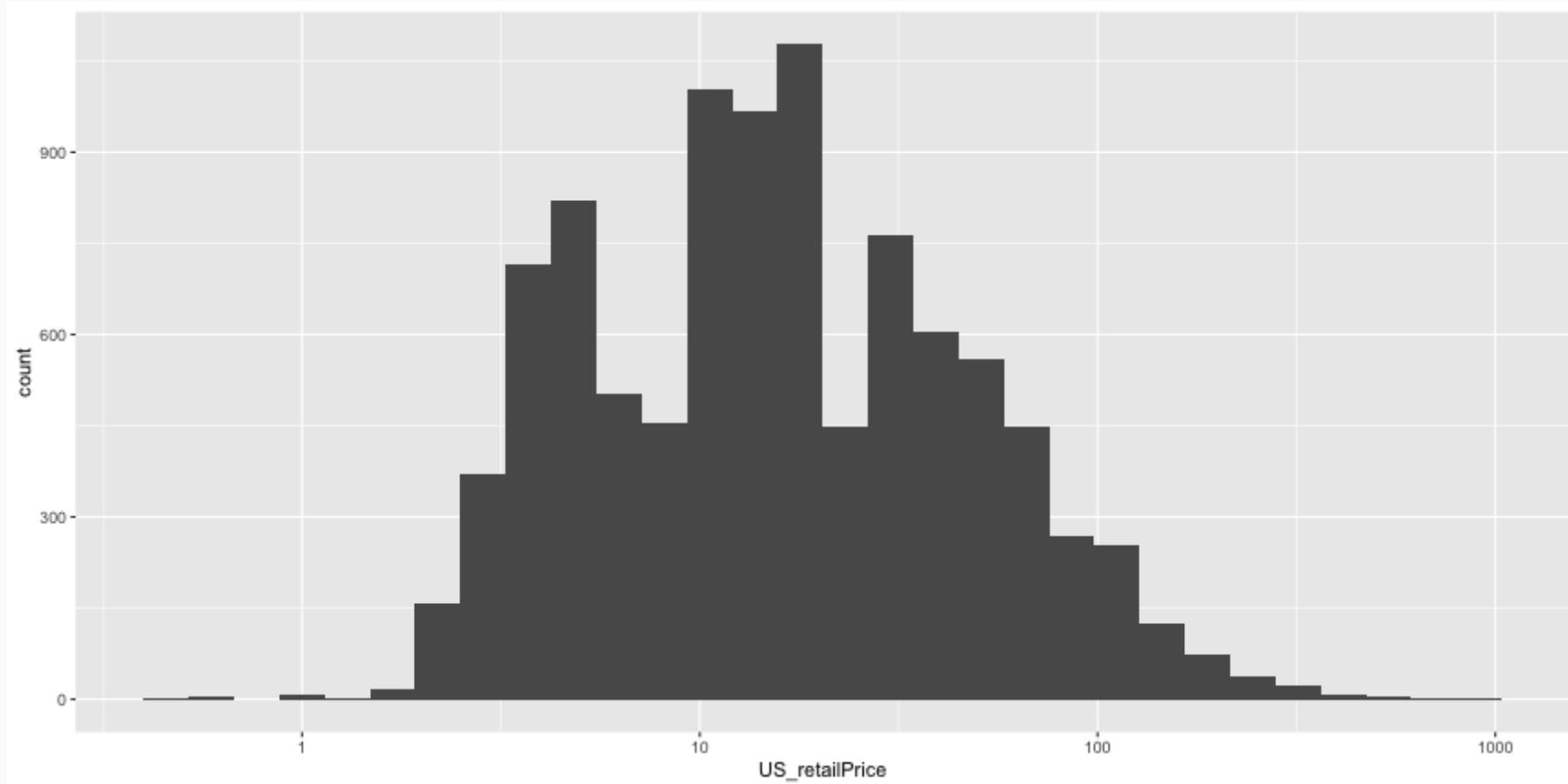
Histograms

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



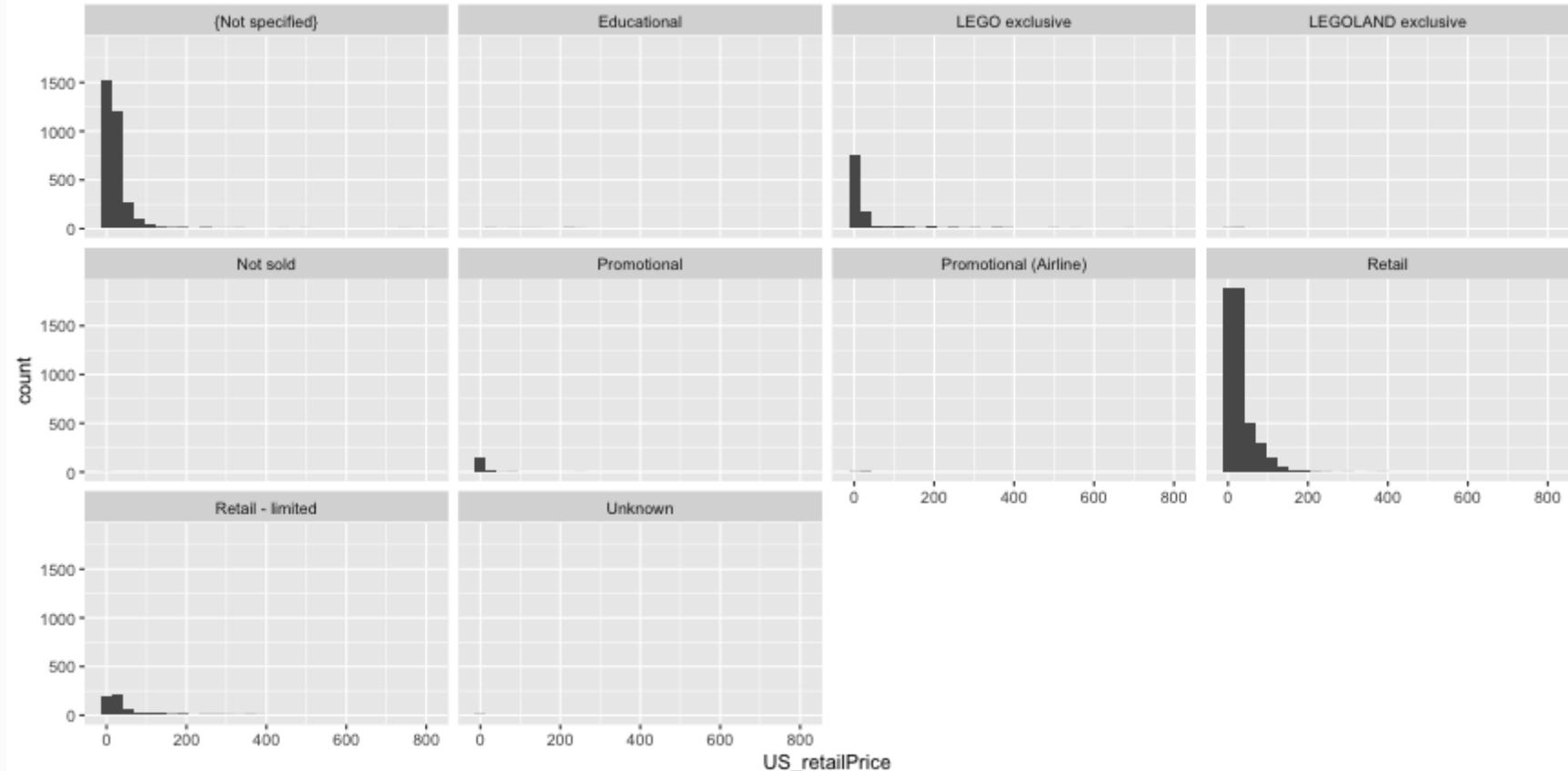
Histograms (cont.)

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



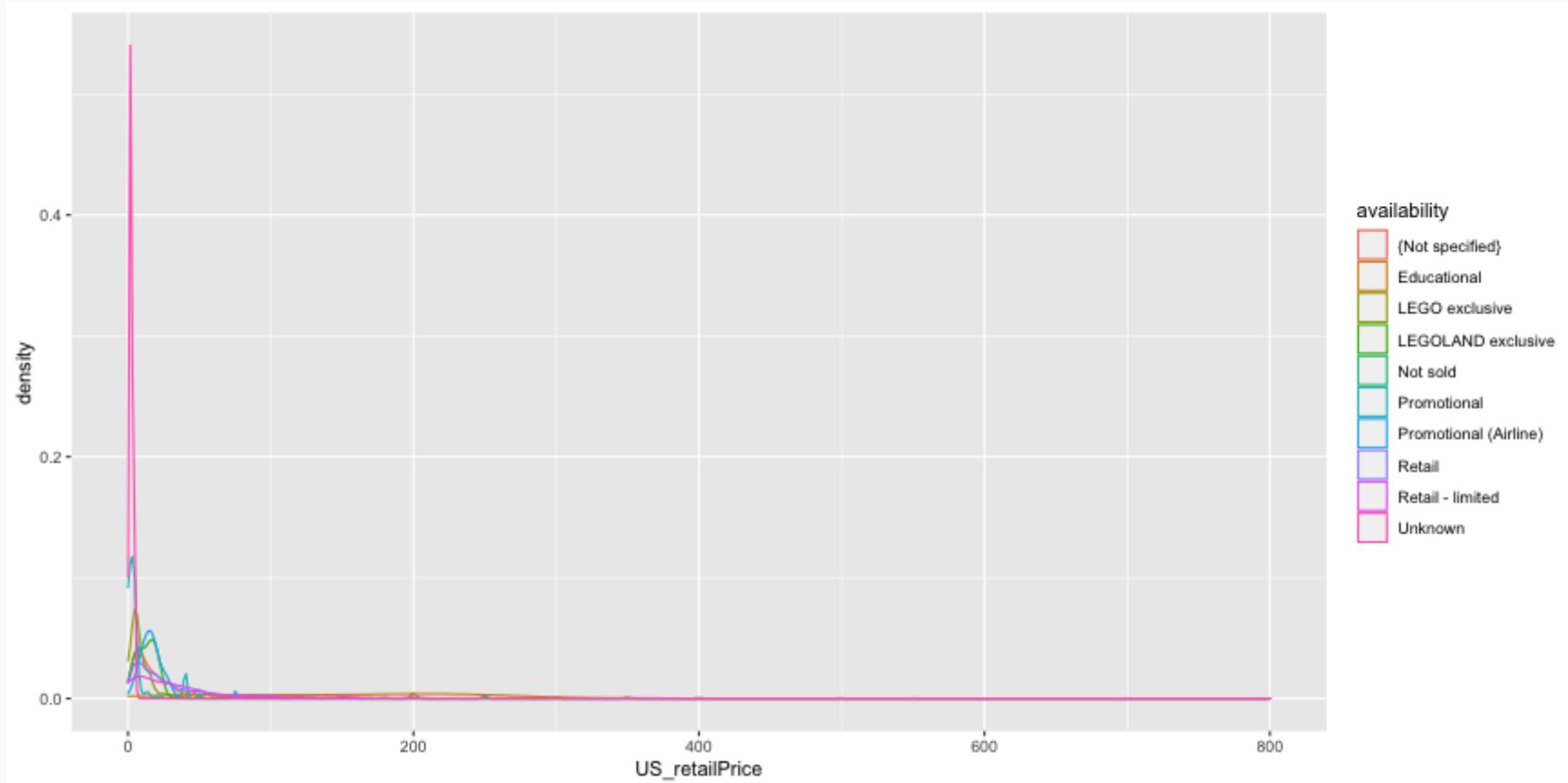
Histograms (cont.)

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



Density Plots

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



ggplot2 aesthetics



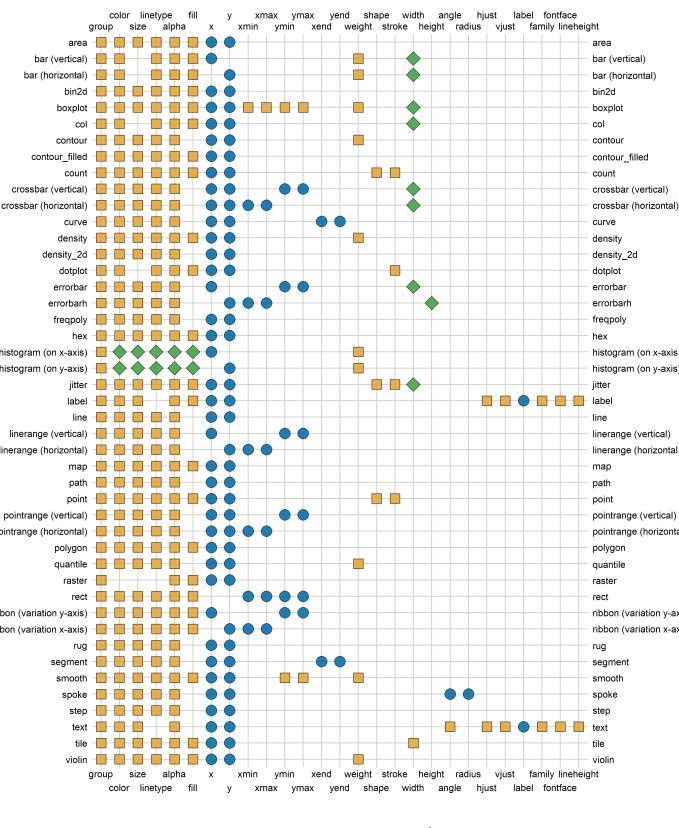
ggplot2 aesthetics cheat sheet

Use this table to find the right aesthetics for your geoms:

Aesthetics that usually must be mapped to the data: use inside aes()

Aesthetics that can be mapped to the data: use in or outside aes()

Aesthetics that cannot be mapped to the data: use `outside aes()`



idea and design: Christian Burkhart
design advice: Ida Aarnio



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)
```

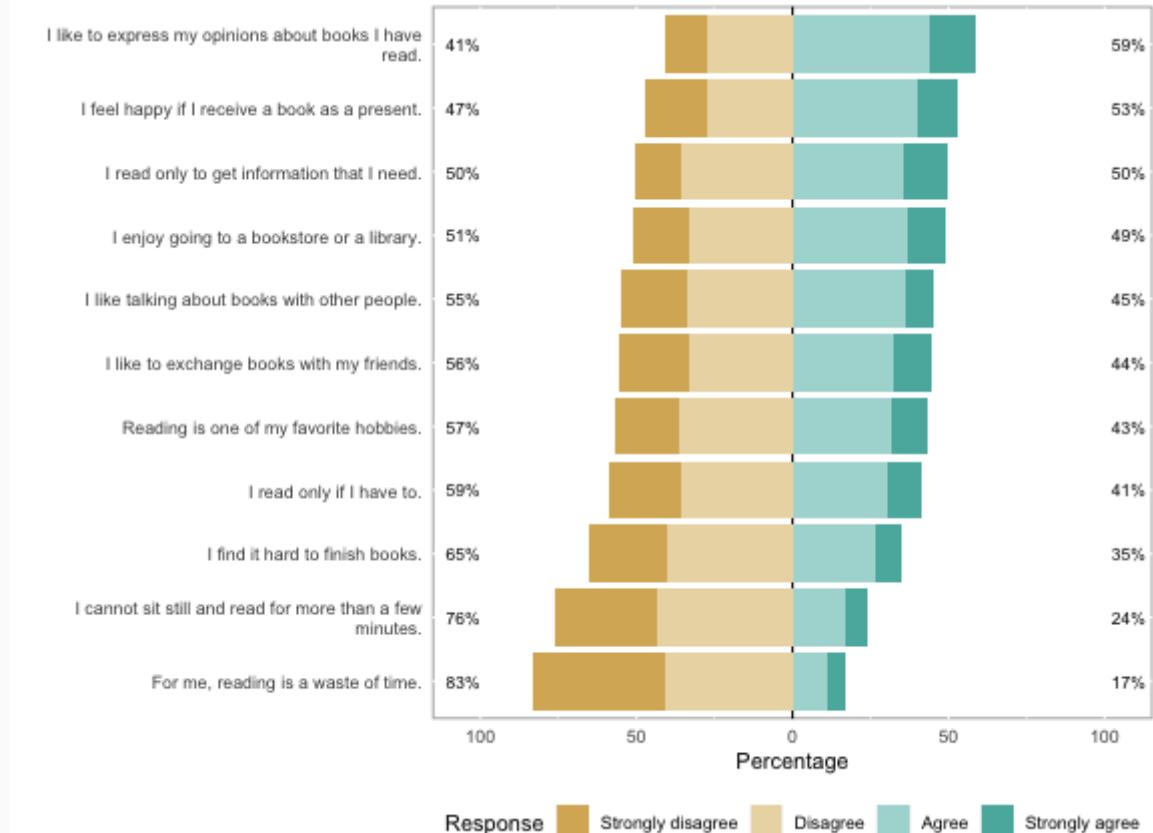
	Item	low	neutral	high	mean	sd
## 10	I like to express my opinions about books I have read.	41.07516	0	58.92484	2.604913	0.9009968
## 5	I feel happy if I receive a book as a present.	46.93475	0	53.06525	2.466751	0.9446590
## 8	I read only to get information that I need.	50.39874	0	49.60126	2.484616	0.9089688
## 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

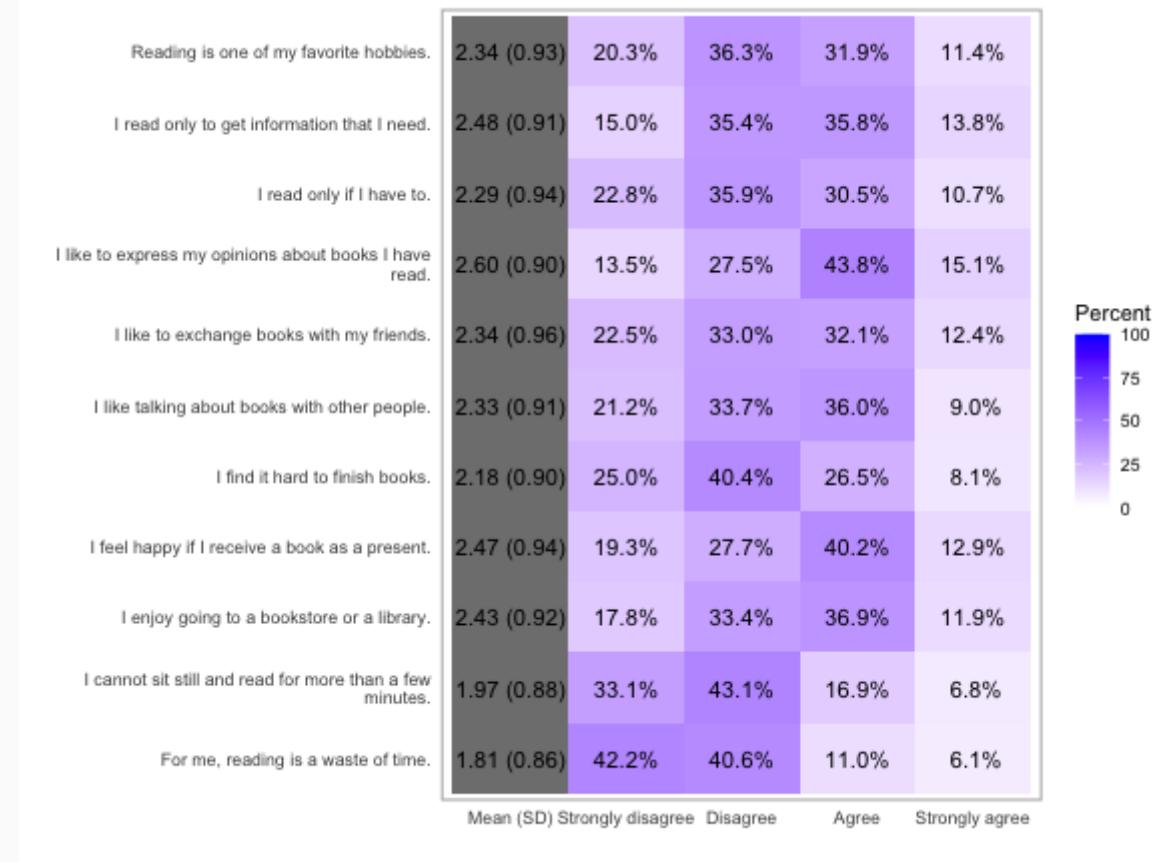
```
plot(l24)
```





likert Plots

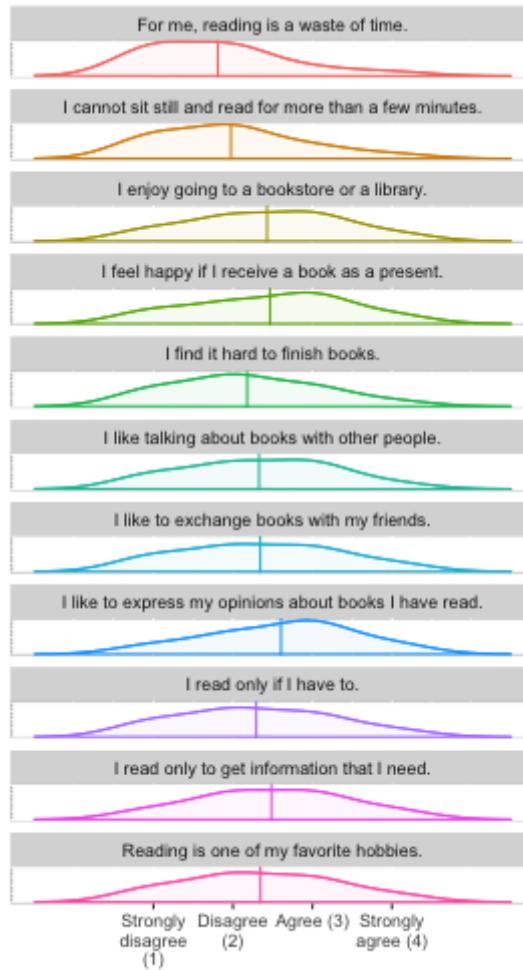
```
plot(l24, type='heat')
```





likert Plots

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



Dual Scales

Some problems¹:

- The designer has to make choices about scales and this can have a big impact on the viewer
- "Cross-over points" where one series cross another are results of the design choices, not intrinsic to the data, and viewers (particularly unsophisticated viewers)
- They make it easier to lazily associate correlation with causation, not taking into account autocorrelation and other time-series issues
- Because of the issues above, in malicious hands they make it possible to deliberately mislead

This example looks at the relationship between NZ dollar exchange rate and trade weighted index.

```
DATA606::shiny_demo('DualScales', package='DATA606')
```

My advise:

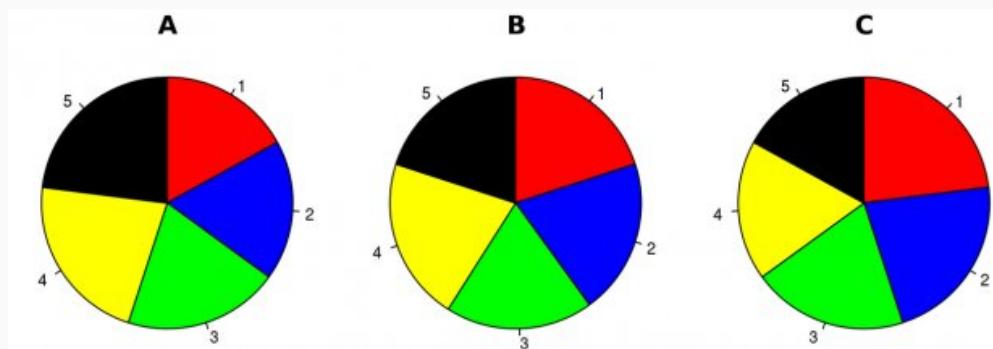
- Avoid using them. You can usually do better with other plot types.
- When necessary (or compelled) to use them, rescale (using z-scores, we'll discuss this in a few weeks)

¹ <http://blog.revolutionanalytics.com/2016/08/dual-axis-time-series.html>

² <http://ellisp.github.io/blog/2016/08/18/dualaxes>

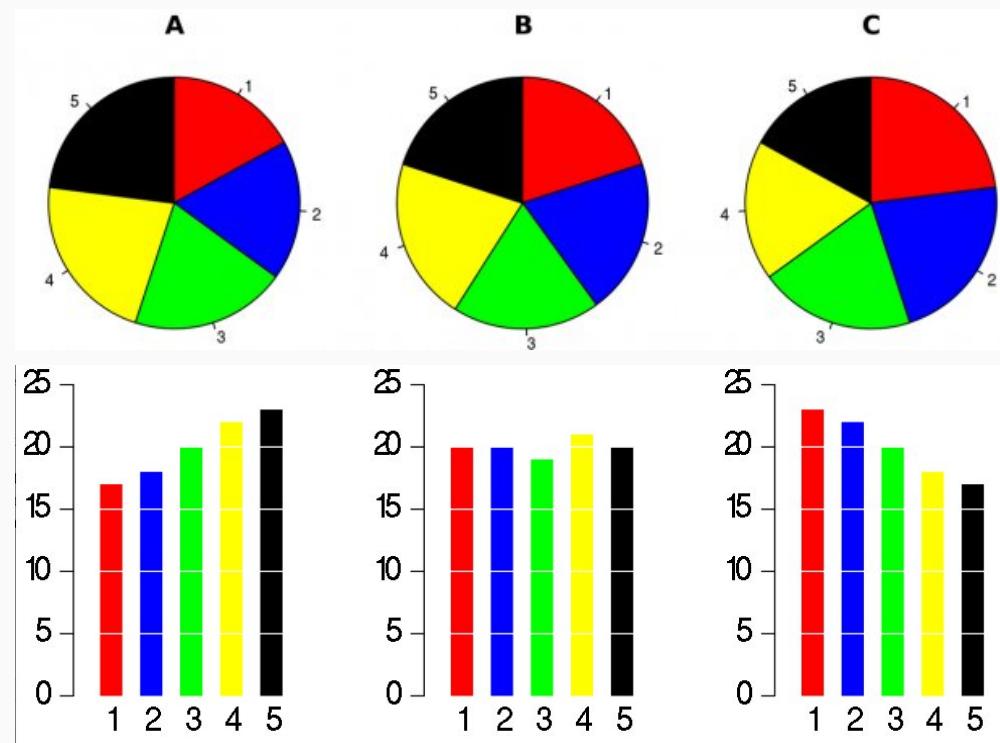
Pie Charts

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.



"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

Complete the one minute paper: <https://forms.gle/CA1dbnMtqQ7Zyj5Y8>

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

