

Descriptive Statistics and Data Visualization

EPSY 630 - Statistics II

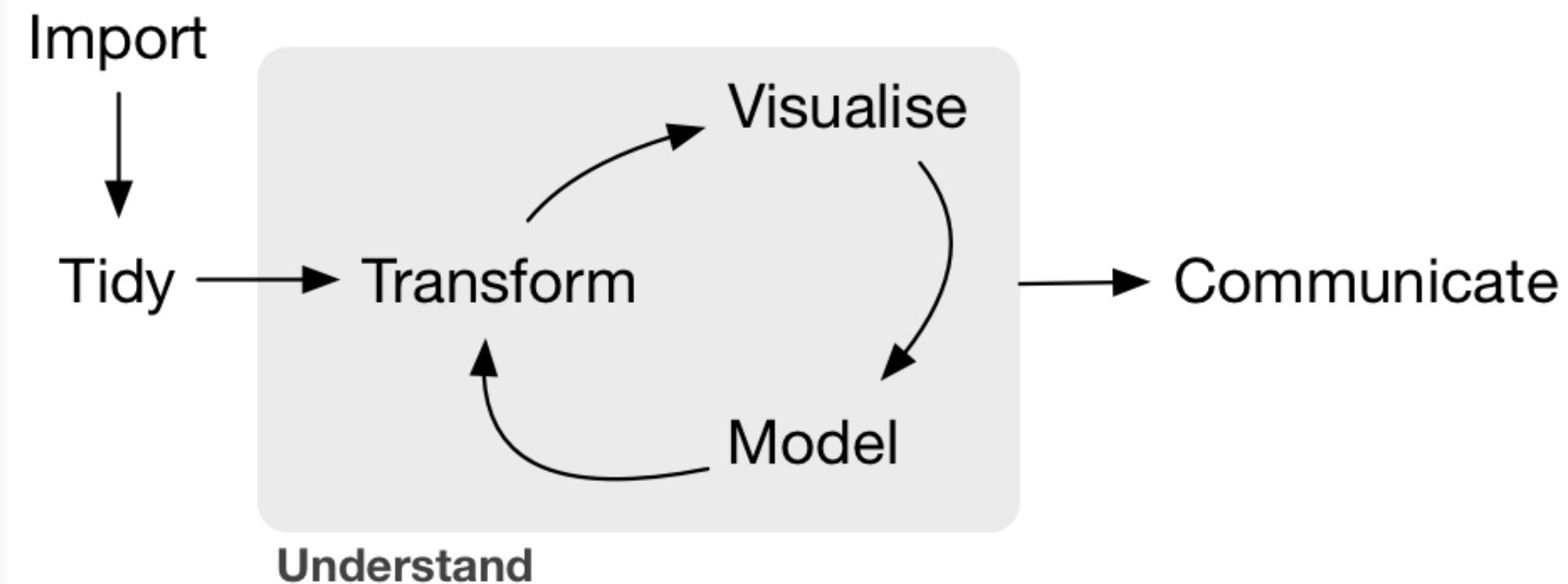
Jason Bryer, Ph.D.

February 9, 2021

Agenda

- Data wrangling
 - Data types
 - Descriptive statistics
- Data visualization
 - Grammar of graphics
 - Types of graphics

Workflow



Source: Wickham & Grolemund, 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

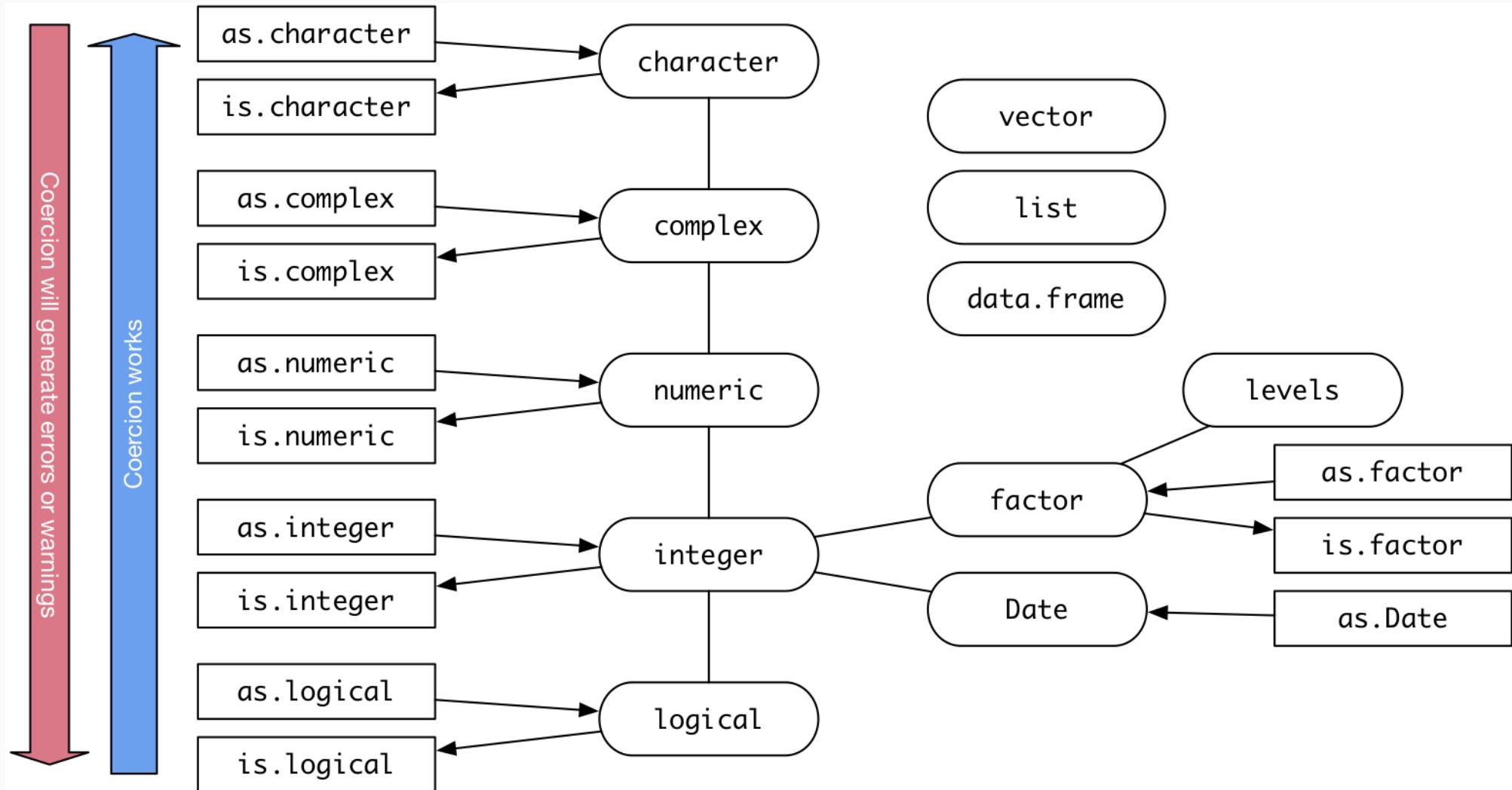
Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

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	mosiac plot, bar plot
Quantitative & Qualitative	grouped summaries, ANOVA, t-test	box plot



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"                  "theme"
## [5] "themeGroup"             "subtheme"               "category"               "released"
## [9] "pieces"                 "minifigs"               "bricksetURL"            "rating"
## [13] "reviewCount"            "packagingType"          "availability"           "agerange_min"
## [17] "US_retailPrice"          "US_dateFirstAvailable" "US_dateLastAvailable" "UK_retailPrice"
## [21] "UK_dateFirstAvailable"  "UK_dateLastAvailable"  "CA_retailPrice"         "CA_dateFirstAvailable"
## [25] "CA_dateLastAvailable"   "DE_retailPrice"          "DE_dateFirstAvailable" "DE_dateLastAvailable"
```

Structure

```
str(legosets)
```

```
## 'data.frame': 16355 obs. of 34 variables:
## $ setID      : int 7693 7695 7697 7698 25534 7418 7419 6020 22704 7421 ...
## $ 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 ...
## $ 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 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 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://images.brickset.com/sets/small/2-8.jpg" "https://images.brickset.com/sets/small/3-6.jpg" "https://images.brickset.com/sets/small/4-4.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" "https://images.brickset.com/sets/images/4-4.jpg" ...
```



RStudio Environment tab can help



Screenshot of the RStudio Environment tab showing the 'legosets' dataset. The dataset has 16355 observations and 34 variables. The variables include:

Variable	Type	Description
\$ setID	int	7693 7695 7697 7698 25534 7418 7419 6020 22704 7421 ...
\$ 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 TRUE ...
\$ pieces	int	67 109 158 233 NA 1 1 60 65 NA ...
\$ minifigs	int	NA ...
\$ bricksetURL	chr	"https://brickset.com/sets/1-8" "https://brickset.com/sets..."
\$ rating	num	0 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 ...
\$ UK_dateFirstAvailable	Date	format: NA NA NA NA ...
\$ UK_dateLastAvailable	Date	format: NA NA NA NA ...
\$ CA_retailPrice	num	NA ...
\$ CA_dateFirstAvailable	Date	format: NA NA NA NA ...
\$ CA_dateLastAvailable	Date	format: NA NA NA NA ...
\$ DE_retailPrice	num	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 NA 5.08 NA ...
\$ weight	num	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	8472	Fountain of Youth	2011	Pirates of the Caribbean	Licensed	Normal	19.99	128	4	3.8
2	878	Kitty Cat's Building Set	1995	Duplo	Pre-school	Normal		36		0
3	6486	Volkswagen Beetle	2008	Creator Expert	Model making	Normal	119.99	1626		4.4
4	30726	Digi Kai	2020	Ninjago	Action/Adventure	Other		3	1	0
5	6265	Chameleon Hunter	2008	Exo-Force	Action/Adventure	Normal	14.99	188	1	3.8
6	7646	LEGO Minifigures Series 1 {Random bag}	2010	Collectable Minifigures	Miscellaneous	Random	1.99			4.7
7	13622	Bonus/Value Pack	2012	HERO Factory	Construction	Collection		39		0
8	7672	Shoot 'n' Score (Zidane Edition)	2000	Sports	Modern day	Normal		22	2	0
9	26474	Battle Suit Lance	2017	Nexo Knights	Action/Adventure	Normal	9.99	83	1	3.8
10	28273	Brick Backpack Orange	2018	Gear	Miscellaneous	Gear	49.99			0

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:



Each **variable** is in its own **column**



Each **observation**, or **case**, is in its own **row**



`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()

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

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)` `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).

- **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)`

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 %>%



The pipe operator (`%>%`) introduced with the `magrittr` R package allows for the chaining of R operations. It takes the output from the left-hand side and passes it as the first parameter to the function on the right-hand side. In base R, to get the output of a proportional table, you need to first call `table` then `prop.table`.

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()
```

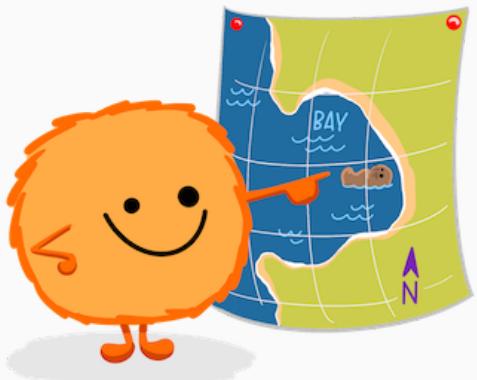


dplyr::filter()

KEEP ROWS THAT
satisfy
your CONDITIONS

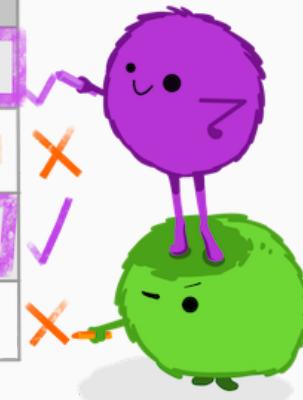
keep rows from... this data... ONLY IF... type MATCHES "otter" AND site MATCHES "bay"

```
filter(df, type == "otter" & site == "bay")
```



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

@alison_lherst



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] 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!



@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



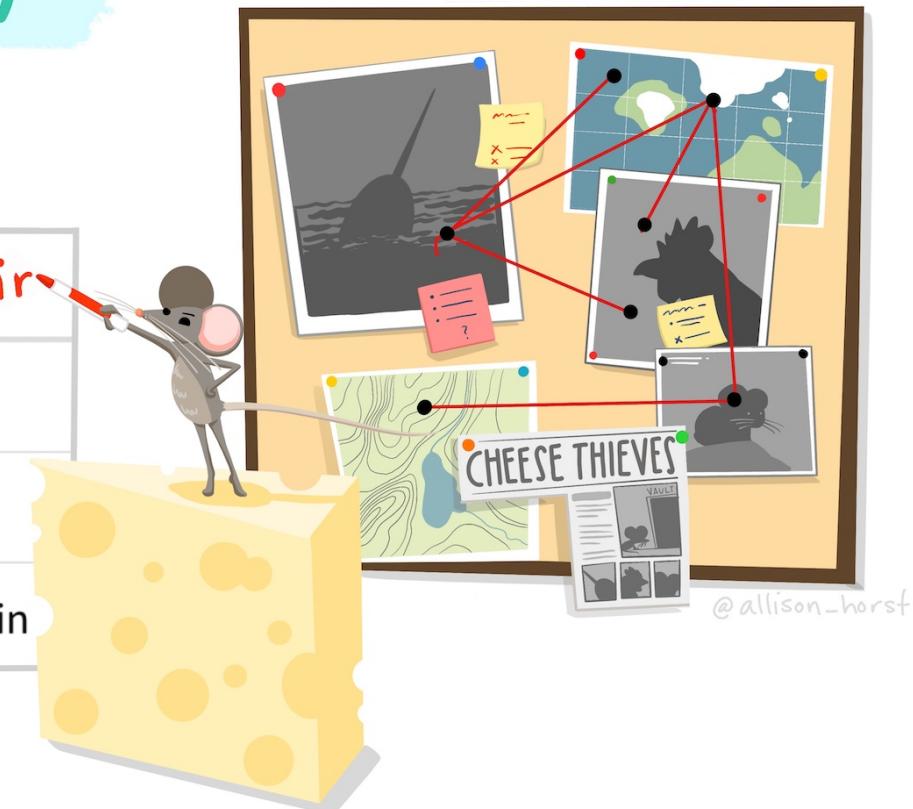
dplyr::rename()

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
```

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)
```





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 x 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      10.0     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 14.99  14.99  20.14 14.83    0 799.99 799.99  5.62    58.91 0.42
```

```
describe.by(legosets$US_retailPrice, group = legosets$availability, mat = TRUE, skew = FALSE)
```

```
##      item           group1 vars     n   mean       sd   min     max   range skew kurtosis     se
## X11     1 {Not specified}  1 3197 24.24484 36.282072 0.60 789.99 789.39 0.641683
## X12     2 Educational     1    9 140.95000 86.358265 14.95 244.95 230.00 28.786088
## X13     3 LEGO exclusive  1 1066 28.79797 70.954538 0.00 799.99 799.99 2.173209
## X14     4 LEGOLAND exclusive  1    7 12.70429  6.447591 4.99 19.99 15.00 2.436960
## 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 1.831456
## X17     7 Promotional (Airline) 1   11 15.79455  6.614819 5.00 28.00 23.00 1.994111
## X18     8 Retail            1 4824 29.82030 33.270049 1.95 399.99 398.04 0.479011
```




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_bin2d"          "geom_contour"         "geom_count"          "geom_crossbar"        "geom_curve"          "geom_density"         "geom_density2d"       "geom_dotplot"         "geom_errorbar"        "geom_hex"             "geom_label"          "geom_linerange"       "geom_line"            "geom_map"             "geom_point"          "geom_polygon"         "geom_rect"            "geom_sf"              "geom_sf_label"        "geom_smooth"          "geom_spoke"           "geom_tile"            "geom_violin"          "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.



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

required
Not required, sensible defaults supplied

ggplot(x = 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

qplot(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, unemployed))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
# (useful for expanding limits)

b + geom_curve(aes(yend = lat + 1,
xend = long + curvevature * z), x, yend, 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 = unempoly - 900,
ymax = unempoly + 900)) -> x, ymax, ymin,
alpha, color, fill, group, linetype, size
```

LINE SEGMENTS

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

```
b + geom_abline(aes(intercept0 = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

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

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

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

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

discrete

```
d <- ggplot(mpg, aes(f))
d + geom_bar()
x, alpha, color, fill, linetype, size, weight
```

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, alpha, colour, group, 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_quartile() x, y, alpha, color, group,
linetype, size, weight

e + geom_rect(sides = "bl") x, y, alpha, color,
linetype, size
```

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,
ymin, ymax, 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
```

THREE VARIABLES

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

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
```



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_pc
```





Scatterplot (cont.)

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





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)
```



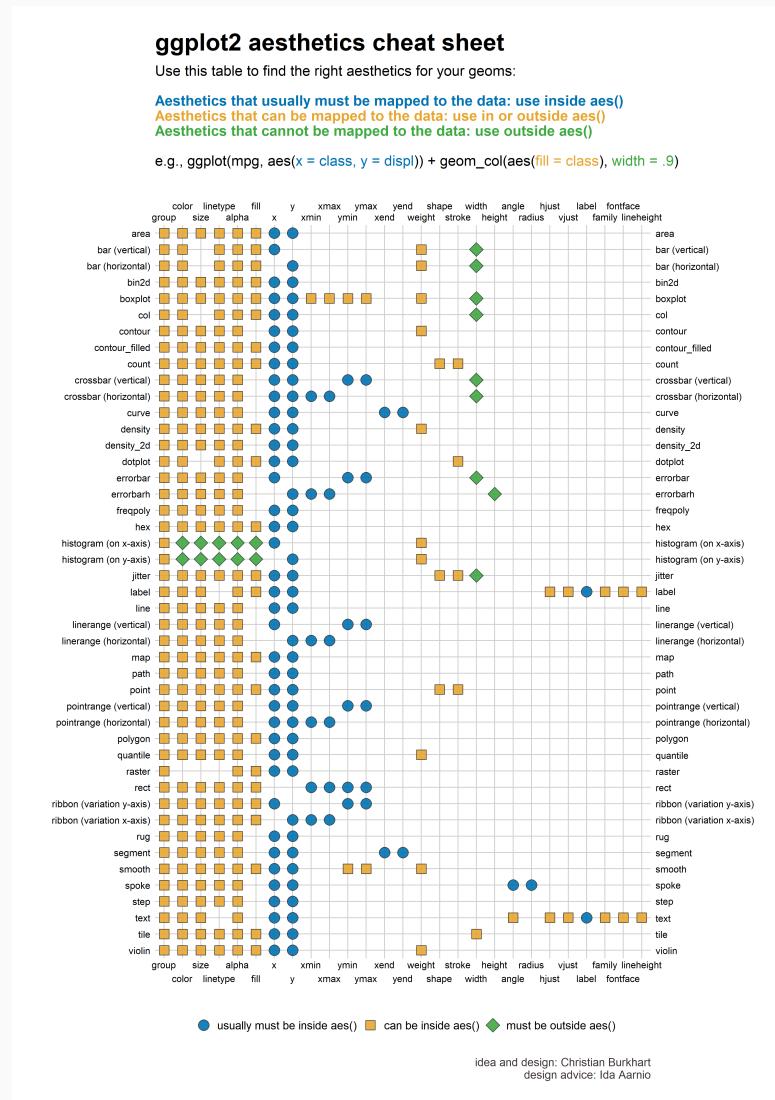


Density Plots

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



ggplot2 aesthetics





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.",
```



likert R Package

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

		Item	low	neutral	high	r
##						
## 10	I like to express my opinions about books I have read.	41.07516	0	58.92484	2.604	
## 5	I feel happy if I receive a book as a present.	46.93475	0	53.06525	2.466	
## 8	I read only to get information that I need.	50.39874	0	49.60126	2.484	
## 7	I enjoy going to a bookstore or a library.	51.21231	0	48.78769	2.428	
## 3	I like talking about books with other people.	54.99129	0	45.00871	2.328	
## 11	I like to exchange books with my friends.	55.54115	0	44.45885	2.343	
## 2	Reading is one of my favorite hobbies.	56.64470	0	43.35530	2.344	
## 1	I read only if I have to.	58.72868	0	41.27132	2.291	
## 4	I find it hard to finish books.	65.35125	0	34.64875	2.178	
## 9	I cannot sit still and read for more than a few minutes.	76.24524	0	23.75476	1.974	
## 6	For me, reading is a waste of time.	82.88729	0	17.11271	1.810	



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.

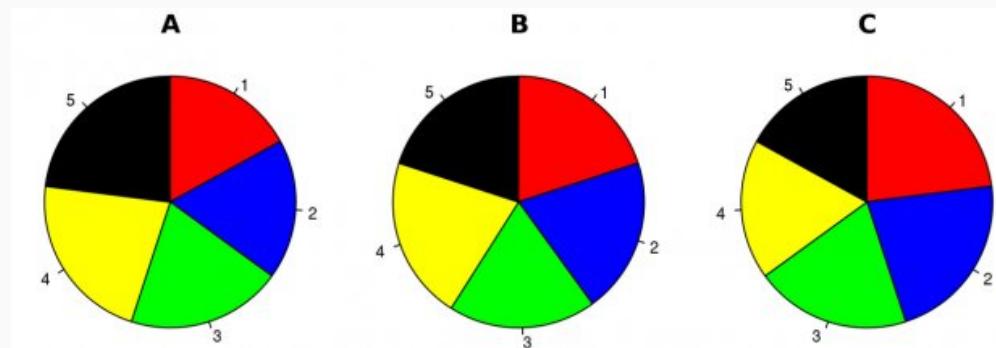
```
DATA606::shiny_demo('DualScales', package='DATA606')
```

My advise:



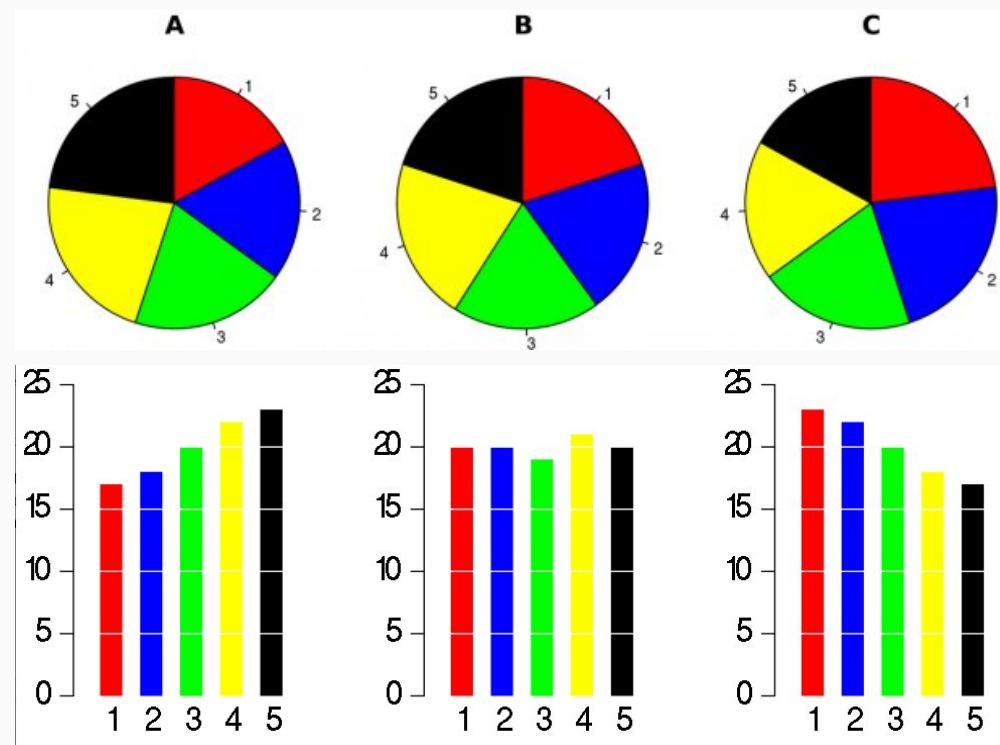
Pie Charts

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/yB3ds6MYE89Z1pURA>

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