EPSY 887: Computation Statistics

Reshaping & Merging Data

Jason Bryer

http://github.com/jbryer/CompStats
jason@bryer.org

Week 3
February 11, 2013

Agenda

1. Sorting Data
2. Merging data
3. Reshaping Data
4. ggplot2: A Grammar of Graphics

Outline

Sorting Data

The `order` function will return a vector with the position of the elements ordered.

```
> test <- c("s","t","a","t","i","s","t","i","c","s")
> test
 [1] "s" "t" "a" "t" "i" "s" "t" "i" "c" "s"
> order(test)
 [1]  3  9  5  8  1  6 10  2  4  7
> test[order(test)]
 [1] "a" "c" "i" "i" "s" "s" "s" "t" "t" "t"
```

Sorting a Data Frame

Using the `mtcars` dataset, first we'll sort by mpg

```
> mtcars.mpg <- mtcars[order(mtcars$mpg),]
> head(mtcars.mpg)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Cadillac Fleetwood	10	8	472	205	2.9	5.2	18	0	0	3	4
Lincoln Continental	10	8	460	215	3.0	5.4	18	0	0	3	4
Camaro Z28	13	8	350	245	3.7	3.8	15	0	0	3	4
Duster 360	14	8	360	245	3.2	3.6	16	0	0	3	4
Chrysler Imperial	15	8	440	230	3.2	5.3	17	0	0	3	4
Maserati Bora	15	8	301	335	3.5	3.6	15	0	1	5	8

Sorting a Data Frame

By default the `order` function returns a list in ascending order. To return in descending order specify the `decreasing=TRUE` parameter.

```
> mtcars.mpg.desc <- mtcars[order(mtcars$mpg, decreasing=TRUE),]
> head(mtcars.mpg.desc)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Toyota Corolla	34	4	71	65	4.2	1.8	20	1	1	4	1
Fiat 128	32	4	79	66	4.1	2.2	19	1	1	4	1
Honda Civic	30	4	76	52	4.9	1.6	19	1	1	4	2
Lotus Europa	30	4	95	113	3.8	1.5	17	1	1	5	2
Fiat X1-9	27	4	79	66	4.1	1.9	19	1	1	4	1
Porsche 914-2	26	4	120	91	4.4	2.1	17	0	1	5	2

Outline

Merging Data

The `merge` function will combine two data frames by one or more common key variables. For example, consider two data frames, one listing authors and another listing books. We may wish to combine these two data frames into a single data frame. We can link them by the author's name.

Merging Data

```
> authors
```

	surname	nationality	deceased
1	Tukey	US	yes
2	Venables	Australia	no
3	Tierney	US	no
4	Ripley	UK	no
5	McNeil	Australia	no

```
> books
```

	name	title	other.author
1	Tukey	Exploratory Data Analysis	<NA>
2	Venables	Modern Applied Statistics ...	Ripley
3	Tierney	LISP-STAT	<NA>
4	Ripley	Spatial Statistics	<NA>
5	Ripley	Stochastic Simulation	<NA>
6	McNeil	Interactive Data Analysis	<NA>
7	R Core	An Introduction to R	Venables & Smith

Merging Data

```
> merge(authors, books, by.x = "surname", by.y = "name")
   surname nationality deceased                          title
1   McNeil   Australia       no      Interactive Data Analysis
2   Ripley          UK       no             Spatial Statistics
3   Ripley          UK       no          Stochastic Simulation
4  Tierney          US       no                       LISP-STAT
5    Tukey          US      yes      Exploratory Data Analysis
6 Venables   Australia       no Modern Applied Statistics ...
  other.author
1         <NA>
2         <NA>
3         <NA>
4         <NA>
5         <NA>
6       Ripley
```

Merging Data

```
> merge(books, authors, by.x = "name", by.y = "surname")
```

	name	title	other.author	nationality	deceased
1	McNeil	Interactive Data Analysis	<NA>	Australia	no
2	Ripley	Spatial Statistics	<NA>	UK	no
3	Ripley	Stochastic Simulation	<NA>	UK	no
4	Tierney	LISP-STAT	<NA>	US	no
5	Tukey	Exploratory Data Analysis	<NA>	US	yes
6	Venables	Modern Applied Statistics ...	Ripley	Australia	no

Merging Data

```
> merge(authors, books, by.x = "surname", by.y = "name", all=TRUE)
   surname nationality deceased                                 title
1   McNeil   Australia       no          Interactive Data Analysis
2   R Core        <NA>     <NA>               An Introduction to R
3   Ripley          UK       no                 Spatial Statistics
4   Ripley          UK       no              Stochastic Simulation
5  Tierney          US       no                           LISP-STAT
6    Tukey          US      yes          Exploratory Data Analysis
7 Venables   Australia       no Modern Applied Statistics ...
      other.author
1             <NA>
2 Venables & Smith
3             <NA>
4             <NA>
5             <NA>
6             <NA>
7           Ripley
```

Outline

Transpose

The t function will transpose a matrix or data frame. That is, rows become columns and columns become rows.

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.6	16	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.9	17	0	1	4	4
Datsun 710	23	4	108	93	3.9	2.3	19	1	1	4	1
Hornet 4 Drive	21	6	258	110	3.1	3.2	19	1	0	3	1
Hornet Sportabout	19	8	360	175	3.1	3.4	17	0	0	3	2
Valiant	18	6	225	105	2.8	3.5	20	1	0	3	1

```
> head(t(mtcars))
```

	Mazda RX4	Mazda RX4 Wag	Datsun 710	Hornet 4 Drive	Hornet Sportabout	Valiant	Duster 360	Merc 240D	Merc 230	Merc 28
mpg	21.0	21.0	22.8	21.4						
cyl	6.0	6.0	4.0	6.0						
disp	160.0	160.0	108.0	258.0						
hp	110.0	110.0	93.0	110.0						
drat	3.9	3.9	3.9	3.1						
wt	2.6	2.9	2.3	3.2						

Reshape

The `reshape` and `reshape2` packages provide a robust set of functions to melt and cast data frames.

```
> mydata <- data.frame(id=c(1,1,2,2), time=c(1,2,1,2), x1=c(5,3,6,2)
> mydata
```

```
  id time x1 x2
1  1    1  5  6
2  1    2  3  5
3  2    1  6  1
4  2    2  2  4
```

```
> require(reshape2)
```

Melting

```
> mydata.melted <- melt(mydata, id=c("id","time"))
> mydata.melted
```

	id	time	variable	value
1	1	1	x1	5
2	1	2	x1	3
3	2	1	x1	6
4	2	2	x1	2
5	1	1	x2	6
6	1	2	x2	5
7	2	1	x2	1
8	2	2	x2	4

Casting

The `reshape2` package has two functions for casting depending on the desired output. Thedcast function will return a data frame and the acast function will return an array or matrix.

```
> dcast(mydata.melted, id ~ variable, mean)
```

```
  id x1  x2
1  1  4 5.5
2  2  4 2.5
```

```
> dcast(mydata.melted, time ~ variable, mean)
```

```
  time  x1  x2
1    1 5.5 3.5
2    2 2.5 4.5
```

Outline

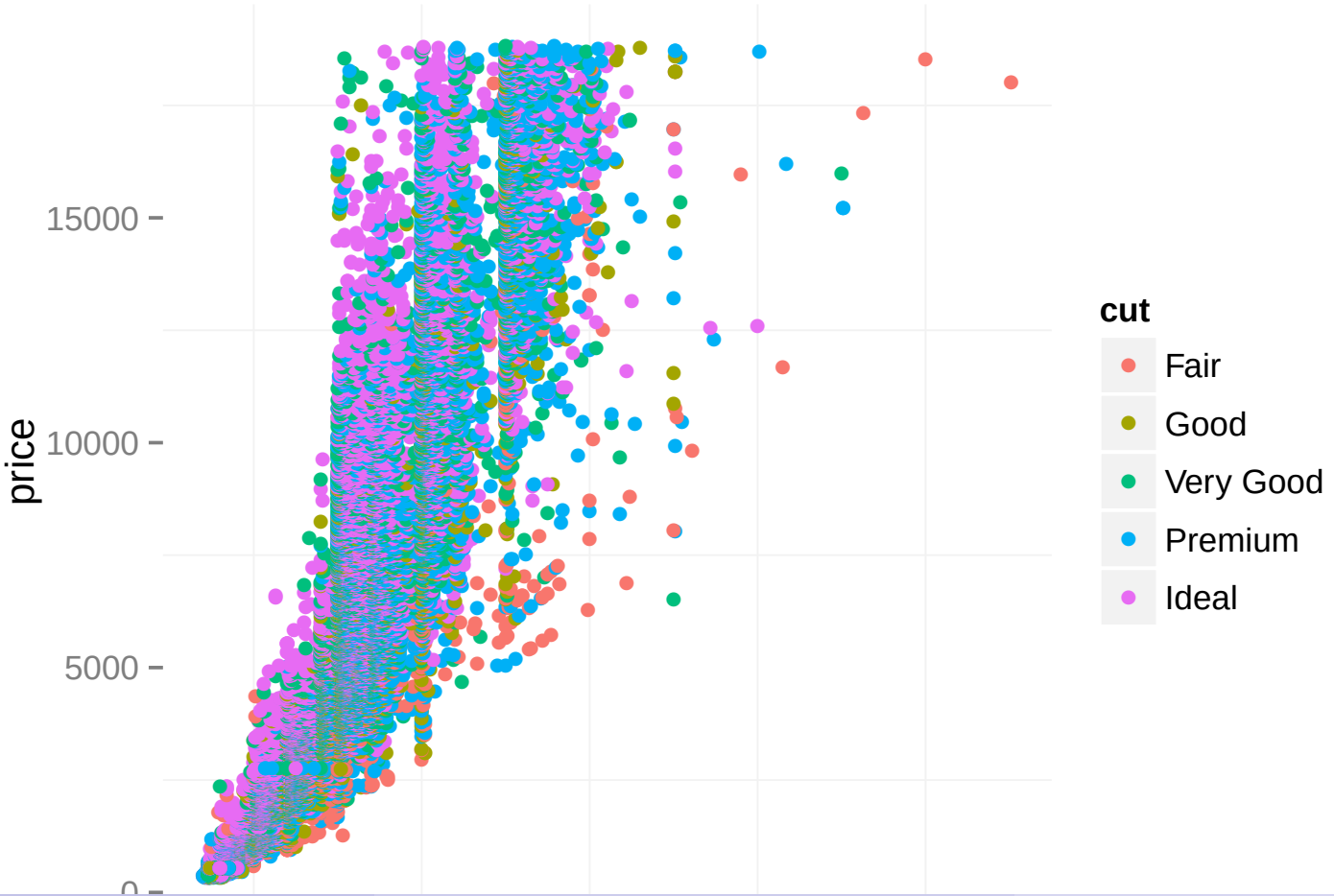
ggplot2: A Grammar of Graphics

- 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.

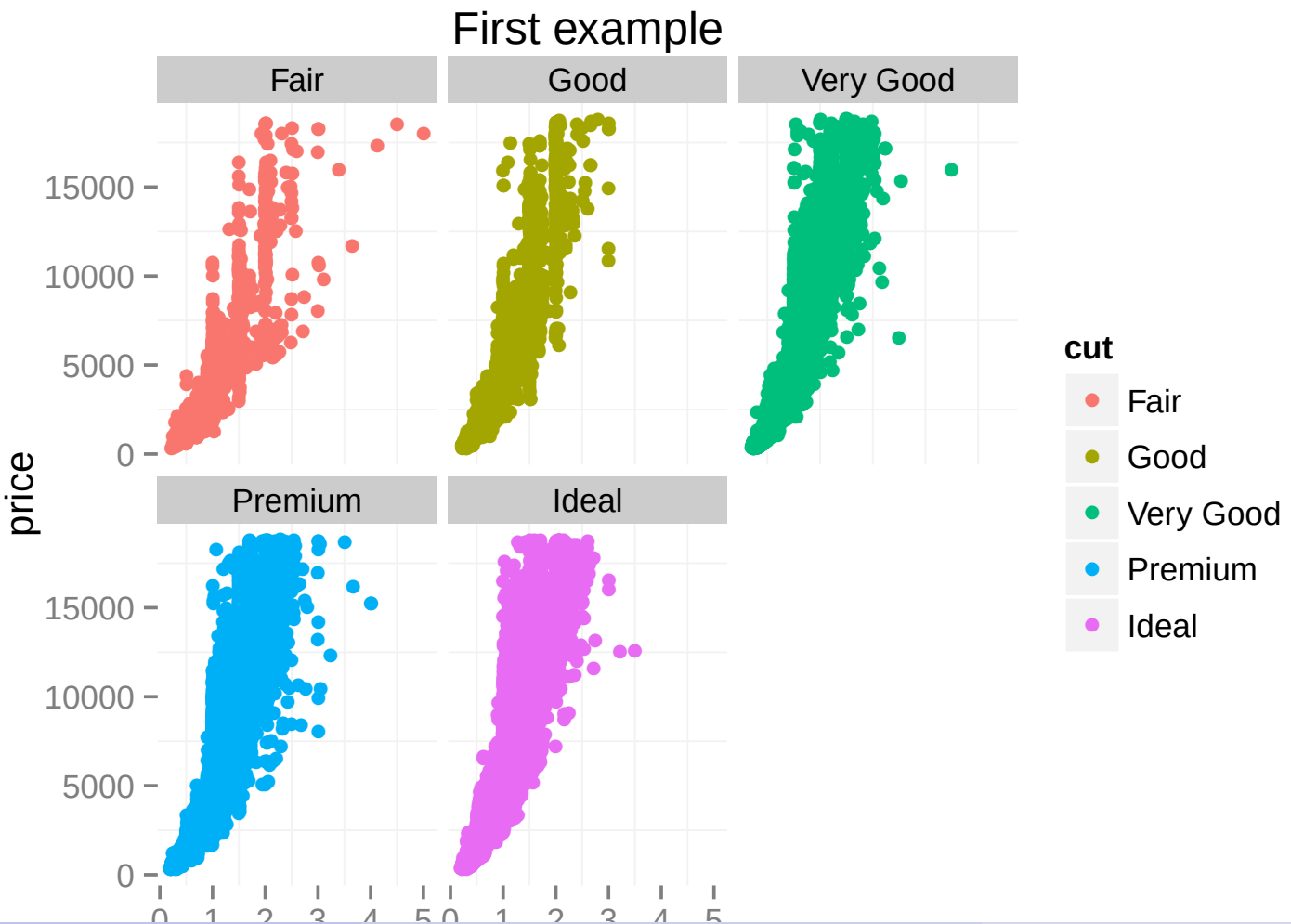
First Example

```
> data(diamonds)
> p <- ggplot(diamonds, aes(x=carat,y=price,colour=cut)) +
    geom_point()
> print(p)
```



First Example

```
> p <- p + facet_wrap(~cut) +
    ggtitle("First example")
> print(p)
```



Parts of a ggplot2 statement

- Data

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

Parts of a ggplot2 statement

- Data

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

- Layers

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

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)`

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()`

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

```
geom_abline
geom_jitter
geom_area
geom_line
geom_bar
geom_linerange
geom_bin2d
geom_path
geom_blank
geom_point
geom_boxplot
geom_pointrange
geom_contour
geom_polygon
geom_crossbar
geom_quantile
```

```
geom_density
geom_rect
geom_density2d
geom_ribbon
geom_errorbar
geom_rug
geom_errorbarh
geom_segment
geom_freqpoly
geom_smooth
geom_hex
geom_step
geom_histogram
geom_text
geom_hline
geom_tile
geom_vline
```