Plotting Data

- Plots of individual data points and aggregates
- Serve different purposes in exploratory and confirmatory analysis
- Use **ggplot2** package
Household per capita expenditures for 7 regions of Vietnam
• How would you create this?
• **data**
  - *source* (“World Map”)

• **violin** (geometric object)
  - *position* \((x = \text{Region}, y = \text{Dollars})\)
  - *color* (Region)

• **median** (statistical summary)
  - *position* \((x = \text{Region}, y = \text{median(dollars)})\)
  - *geom* (point)
  - *color* (black)
  - *fill* (white)
  - *shape* (diamond)
  - *size* (small)
• **mean** (statistical summary)
  
  - `position ( x = Region, y = mean(dollars) )`
  
  - `geom ( point )`
  
  - `color ( black )`
  
  - `fill ( black )`
  
  - `shape( circle )`
  
  - `size( small )`
- Two guides that help reader understand graphic
  - **axis** representing one-dimensional x-position
    - *dimension* (x)
    - *label* ("Region")
  - **axis** representing one-dimensional y-position
    - *dimension* (y)
    - *label* ("Dollars")
Install ggplot2 Package

From the R command line...

```
> install.packages( "ggplot2", dependencies = TRUE )
```

...or using point-and-click

- Click the **Packages** tab.
- Click **Install Packages**.
- Enter **ggplot2** in the text box.
- Click **Install**.
# Load the ggplot2 library

```r
> library(ggplot2)
```

The `library()` function loads the package so that the functions in the package are accessible. Libraries need to be loaded every R session.

# Load the vlss data

```r
> vlss <- read.csv("http://www.tc.umn.edu/~zierf0002/Data/VLSS.csv")
```
Plots are Built by Layering Components

- Plots are built by layering graphical components
- The components are summed together to form the plot
- First component of first layer is `ggplot()`
  - Contains reference to the source data (data frame) and global aesthetic mappings
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region))

The **data** argument indicates the source data frame.

The **aes** argument sets the aesthetic mapping.
Aesthetic Mappings

- Aesthetic mappings define how graphical elements are visually perceived *when these elements vary*
  - Define $x$-dimension (predictor), $y$-dimension (response), size, color, fill, groupings, etc.
  - Each aesthetic can be mapped to a variable or set to a constant value
  - Aesthetics can be set globally (in `ggplot()` function) or locally (in a specific layer)
- Specified with `aes()`
Global Aesthetic Mappings

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region) ) +
  geom_violin() +
  geom_jitter(alpha = 0.3)
```

Aesthetic mappings inside the `ggplot()` layer are applied to every layer.
Local Aesthetic Mappings

Aesthetic mappings inside a particular layer are only applied to that layer.

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars)) +
  geom_violin(aes(color = Region)) +
  geom_jitter(alpha = 0.3)
```
Components of Plotting in ggplot2

- Geometric objects
- Statistical transformations
- Scales
- Coordinate systems
- Faceting
- Position adjustments
Geometric Objects

- Features drawn on plot (e.g., lines, points)
- Specified using prefix `geom_` and suffix that names feature to be plotted
  - **Points** specified with `geom_point()`
  - **Lines** specified with `geom_line()`
  - **Violins** specified with `geom_violin()`
```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin()
```

The `geom_violin()` function adds the geometric object of violins using the global data and aesthetic mapping.
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin(color = "black", fill = "steelblue")

The `color=` argument sets the color for the outline in this layer. The `fill=` argument sets the fill color for this layer.

Notice the quotation marks...color names are strings.

Aesthetic mappings that are fixed to a particular value (do not vary), rather, do not need to be enclosed in the `aes()` function.

Note also that the local commands override the global commands.
Statistical Transformations

- Used for plotting statistics/summaries
  - e.g., mean of the response at fixed levels of the predictor
- Specified using prefix `stat_` and suffix that names desired transformation
- Means, medians, and other summary statistics specified with `stat_summary()`
- Regression models specified with `stat_smooth()`
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point")

**fun.y** takes the function to be applied to the y-dimension for each value of x

**geom="point"** places a point at each computed summary.

How can we change the color, plotting character, or size (or a combination) of the mean points?
```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin() +
  stat_summary(  
    fun.y = mean,
    geom = "point",
    color = "black",
    fill = "black",
    pch = 21,
    size = 2
  )
```

The `pch=` argument sets the plotting character.

The `size=` argument sets the point size. The default size is 4.
Order of the layers matters.
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point", color = "black",
               fill = "black", pch = 21, size = 2) +
  stat_summary(fun.y = median, geom = "point", color = "black",
               fill = "white", pch = 23, size = 2)

Add the median household per capita expenditures as a small, white diamond
Faceting

- Creates separate graphic for each subgroup of subjects
  - `facet_wrap()` displays the graphics conditioned on a single predictor
  - `facet_grid()` displays the graphics conditioned on multiple predictors
The violin plots of Dollars by Region are conditioned on Urbanicity.
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point", color = "black",
               fill = "black", pch = 21, size = 2) +
  stat_summary(fun.y = median, geom = "point", color = "black",
               fill = "white", pch = 23, size = 2) +
  facet_wrap(~ Urban )
```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, color = Region)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point", color = "black",
               fill = "black", pch = 21, size = 2) +
  stat_summary(fun.y = median, geom = "point", color = "black",
               fill = "white", pch = 23, size = 2) +
  facet_wrap( ~ Urban, nrow = 2)
```

`nrow=` (and/or `ncol=`) sets the number of rows or columns in the plotting area.
Violin plots of **Dollars by Urbanicity** conditioned on **Region**.

```r
> ggplot(data = vlss, aes(x = Urban, y = Dollars, color = Urban)) + geom_violin() + 
  stat_summary(fun.y = mean, geom = "point", color = "black", 
              fill = "black", pch = 21, size = 2) + 
  stat_summary(fun.y = median, geom = "point", color = "black", 
              fill = "white", pch = 23, size = 2) + 
  facet_wrap(~ Region)
```
Urban is an integer and is being treated as such.
> vlss$Urban <- \textbf{factor}(vlss$Urban)

> ggplot(data = vlss, aes(x = Urban, y = Dollars, color = Urban)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point", color = "black",
               fill = "black", pch = 21, size = 2) +
  stat_summary(fun.y = median, geom = "point", color = "black",
               fill = "white", pch = 23, size = 2) +
  facet_wrap(~ Region)

\textbf{Coerce Urban into a factor.}
Better Labels

```r
> vlss$Urban <- factor(vlss$Urban, labels = c("Rural", "Urban"))

> ggplot(data = vlss, aes(x = Urban, y = Dollars, color = Urban)) +
  geom_violin() +
  stat_summary(fun.y = mean, geom = "point", color = "black",
               fill = "black", pch = 21, size = 2) +
  stat_summary(fun.y = median, geom = "point", color = "black",
               fill = "white", pch = 23, size = 2) +
  facet_wrap(~ Region)
```
Fine-Tuning the Color

> ggplot(data = vlss, aes(x = Urban, y = Dollars, color = Urban)) +
  geom_violin() +
  ... +
  scale_color_manual( values = c("#E69F00", "#56B4E9") )

scale_color_manual() allows you to manually set the attributes associated with the color aesthetic.

The `values=` argument sets the color values for each level of the factor.

RGB or hex values can be used in `values=` argument of `scale_fill_manual()` or `scale_color_manual()`.
scale_fill_manual() can be used to manually set the colors when the fill= argument is used.
Pre-selected Color Palettes

<table>
<thead>
<tr>
<th>Fill Scale</th>
<th>Color Scale</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale_fill_hue()</td>
<td>scale_color_hue()</td>
<td>Colors evenly spaced around the color wheel</td>
</tr>
<tr>
<td>scale_fill_grey()</td>
<td>scale_color_grey()</td>
<td>Grey scale palette</td>
</tr>
<tr>
<td>scale_fill_brewer()</td>
<td>scale_color_brewer()</td>
<td>ColorBrewer palettes</td>
</tr>
</tbody>
</table>
# Default palette

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = Urban)) +
  geom_boxplot() +
  scale_fill_hue(guide = FALSE)
```

The `guide=FALSE` argument removes the legend.
# Grey scale palette

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = Urban)) + geom_boxplot() + scale_fill_grey(guide = FALSE)
```
# ColorBrewer palette (http://colorbrewer2.org)

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = factor(Urban))) +
  geom_boxplot() +
  scale_fill_brewer(guide = FALSE, palette = "Set2")
```

The `palette=` sets the pre-specified color palette.
Qualitative Color Palettes

- Set3
- Set2
- Set1
- Pastel2
- Pastel1
- Paired
- Dark2
- Accent
Palettes for Color-Blindness

- About eight percent of males and one-half percent of females have some form of color vision deficiency (good chance that someone in your audience will be one of these people)
- Many different forms of color blindness
- Most common forms of color vision deficiency.
- Color and grey-scale palettes have been developed for many of these
Palettes for Color-Blindness

- http://jfly.iam.u-tokyo.ac.jp/color/

# Setting the labels on the x- and y-axis

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = Urban)) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Set2") +
  ylab("Expenditures per Capita (in U.S. dollars)")
```

Labels take strings as their input.

`xlab()` can be used to change the label on the x-axis, and `ylab()` is used to change the label on the y-axis.
The first value is the minimum.

The second value is the maximum.

`xlim()` and `ylim()` are used to set the limits on the x-axis and y-axis respectively.
Fine-Tuning Other Elements of the Plot

```r
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = Urban)) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Set2") +
  ylab("Expenditures per Capita (in U.S. dollars)") +
  theme(legend.position = "none")
```

Another way to remove the legend using `theme()`. Use the `theme()` function and indicate the elements and changes via the function's arguments. See [http://docs.ggplot2.org/0.9.2.1/theme.html](http://docs.ggplot2.org/0.9.2.1/theme.html)
> ggplot(data = vlss, aes(x = Region, y = Dollars, fill = Urban)) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Set2") +
  ylab("Expenditures per Capita (in U.S. dollars)") +
  theme_bw()

`theme_bw()` is a pre-built theme. It changes the colors of many of the elements (major and minor grid lines, background color, etc.) from the default settings.

---

You can build your own themes and use them.
- [https://github.com/jrnold/ggthemes](https://github.com/jrnold/ggthemes)