ggplot2 and Grammar of Graphics

Andrew Zieffler

Educational Psychology

University of Minnesota

Driven to Discover℠

```
# Load the vlss data
> vlss = read.csv("~/Documents/EPSY-8261/data/vlss.csv")

> head(vlss)

    id  expend urban           region
1  404  427.45     1 Red River Delta
2  409  876.63     1 Red River Delta
3  509  854.11     1 Red River Delta
4 1206 1366.94     1 Red River Delta
5 1528  518.45     1      South East
6 1808 1027.99     1      South East

> tail(vlss)

       id expend urban       region
245 37909 124.44     0 Mekong Delta
246 38002 122.26     0 Mekong Delta
247 38103 110.98     0 Mekong Delta
248 38223  93.11     0 Mekong Delta
249 38303 125.15     0 Mekong Delta
250 38515 322.59     0 Mekong Delta
```

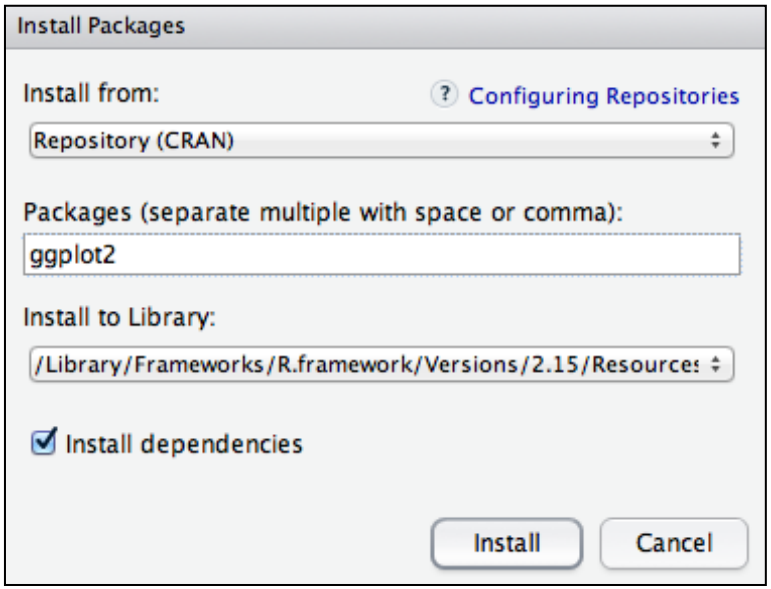
```
> summary(vlss)

       id            expend           urban                    region
 Min.   :  404   Min.   :  35.77   Min.   :0.000   Central Coast   :21
 1st Qu.: 9313   1st Qu.: 120.59   1st Qu.:0.000   Central Highland:11
 Median :17258   Median : 171.76   Median :0.000   Mekong Delta    :46
 Mean   :18511   Mean   : 240.61   Mean   :0.328   North Coast     :27
 3rd Qu.:28358   3rd Qu.: 277.18   3rd Qu.:1.000   Northern Uplands:39
 Max.   :38515   Max.   :1366.94   Max.   :1.000   Red River Delta :56
                                                   South East      :50
```

Install the **ggplot2** Package

Using the RStudio GUI...

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



...or directly from the R command line...

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

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 ggplot2 library
> library(ggplot2)
```

Plots are Built by Layering Components

Plots are built by layering graphical components. In the syntax, the layers are literally *summed* together to form the plot.

The first layer is always `ggplot()`. It contains reference to the **source data** (data frame) and ***global*** **aesthetic mappings**.

```
> ggplot(data = vlss, aes(x = region, y = expend)) +
```

The `data=` argument indicates the source data frame.

The `aes=` argument sets the aesthetic mapping(s).

Aesthetic Mappings and Geometric Objects

Aesthetic mappings define **how graphical elements are visually perceived**. They are used to define *x*-dimension (predictor), *y*-dimension (outcome), size, color, fill, groupings, etc.

- Each aesthetic can be mapped to a **variable** or to a **constant** value
  - If the aesthetic is to vary (e.g., is a variable) it is specified in the `aes()` function
  - If the aesthetic is constant it can be specified inside or outside the `aes()` function
- Aesthetics can be set **globally**—in `ggplot()` layer—or **locally** (only used in a specific layer)

Geometric objects, or *geoms*, are features that are actually drawn on plot (e.g., lines, points). They are specified using the prefix `geom_` and a suffix that names the feature to be plotted.

- **Points** specified with `geom_point()`
- **Jittered points** specified with `geom_jitter()`
- **Lines** specified with `geom_line()`
- **Boxplots** specified with `geom_boxplot()`

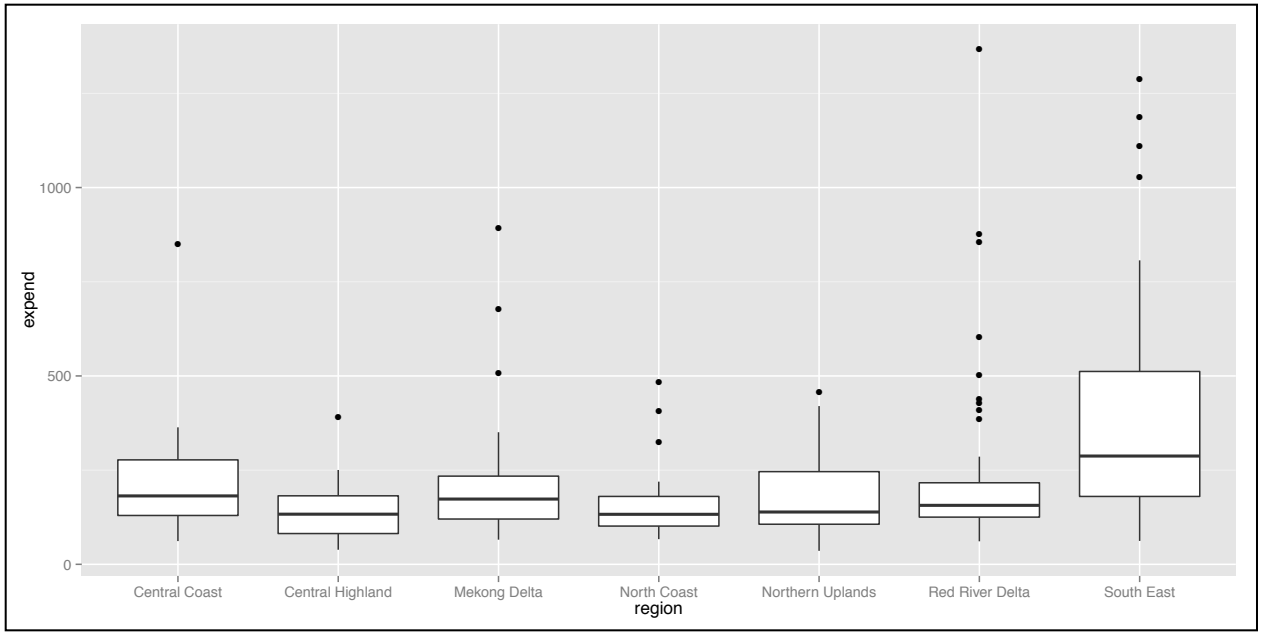
Understanding the Basic Syntax

Aesthetic mappings given in the `ggplot()` layer are applied to every subsequent layer

```
> ggplot(data = vlss, aes(x = region, y = expend)) +
    geom_boxplot()
```

The `geom_boxplot()` function adds the geometric object of boxplots using the global data and aesthetic mapping.

The `+` adds another layer.

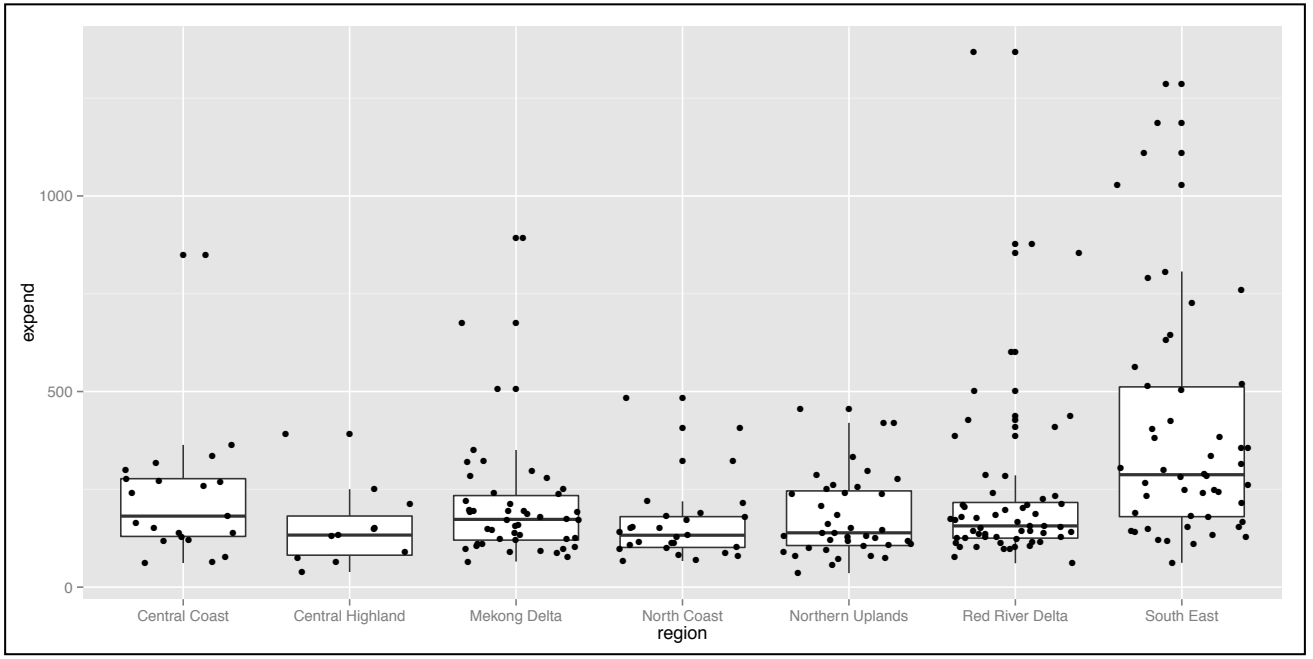


When layers are added they are "stacked" on top of previous layers. Imagine drawings on separate transparencies, and then those transparencies are stacked.

```
> ggplot(data = vlss, aes(x = region, y = expend)) +
    geom_boxplot() +
    geom_jitter()
```

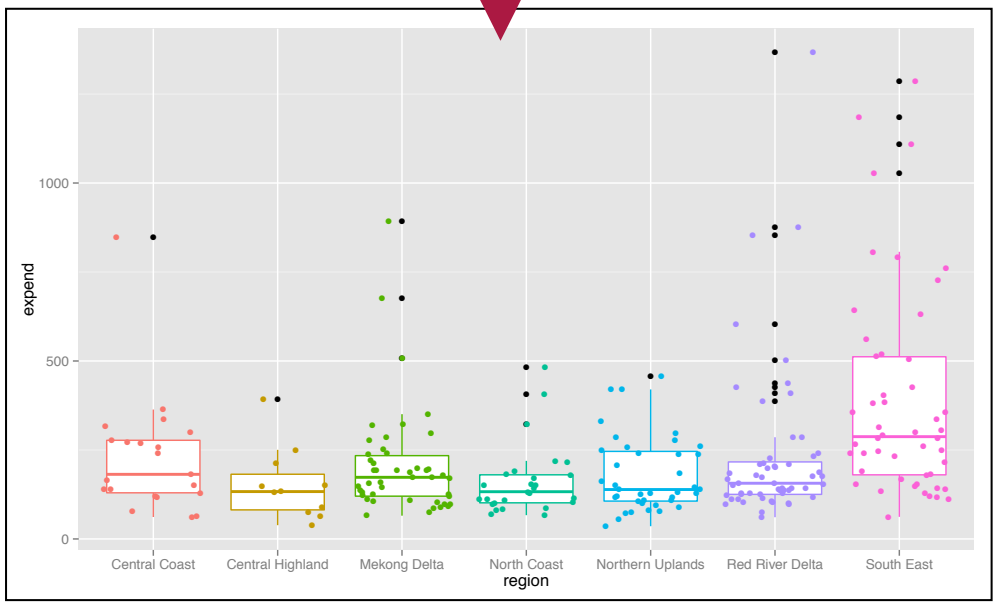
The `+` adds another layer.

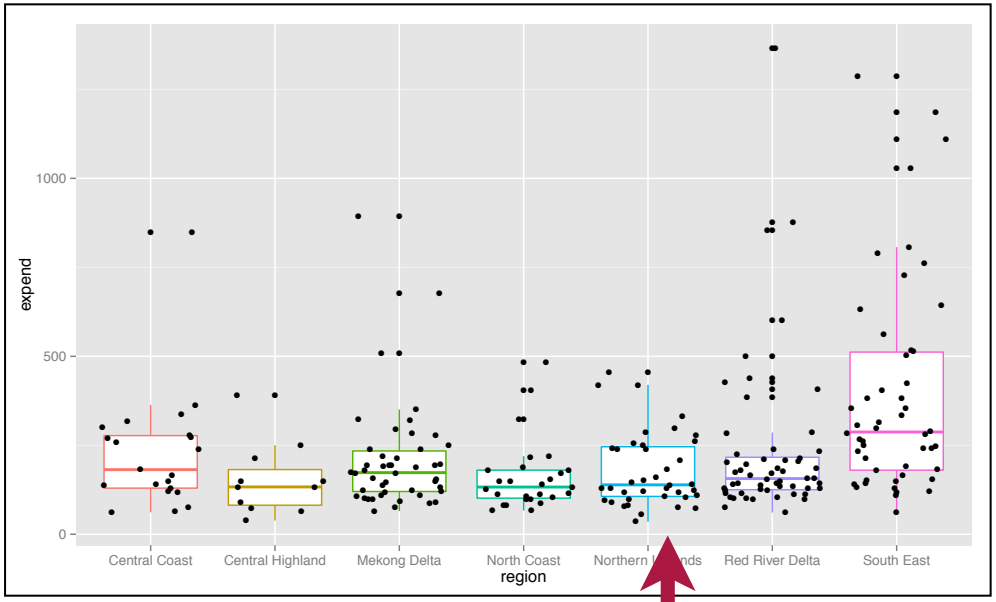
The `geom_jitter()` function adds the geometric object of jittered points using the global data and aesthetic mapping.



```
> ggplot(data = vlss, aes(x = region, y = expend, color = region)) +
    geom_boxplot() +
    geom_jitter()
```

Global aesthetic mappings are applied to *all* layers.






```
> ggplot(data = vlss, aes(x = region, y = expend)) +
    geom_boxplot(aes(color = region)) +
    geom_jitter()
```

Local aesthetic mappings (in a particular layer) are only applied to that layer.

Fixed vs Variable Aesthetics

```
> ggplot(data = vlss, aes(x = region, y = expend, color = region)) +
    geom_boxplot(color = "black", fill = "steelblue") +
    geom_jitter()
```

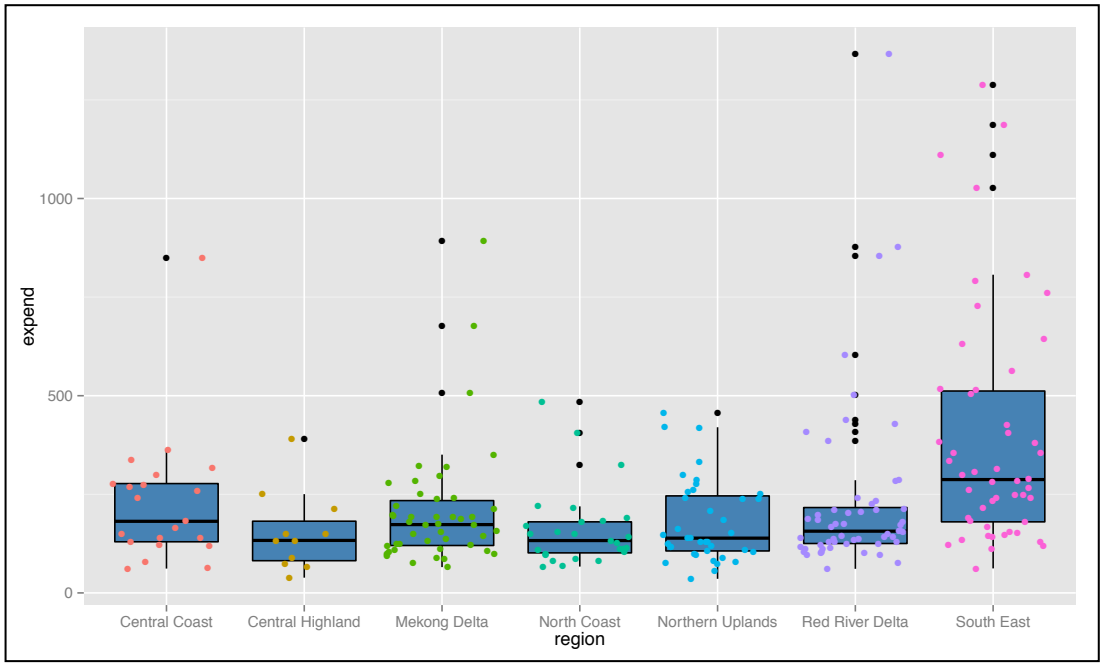
The `color=` argument sets the color for the outline in this layer.

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

Notice the quotation marks...color names are character 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 aesthetics override the global aesthetics



Statistical Transformations

Statistical transformations are used for plotting statistics/summaries (e.g., mean of the response at fixed levels of the predictor).

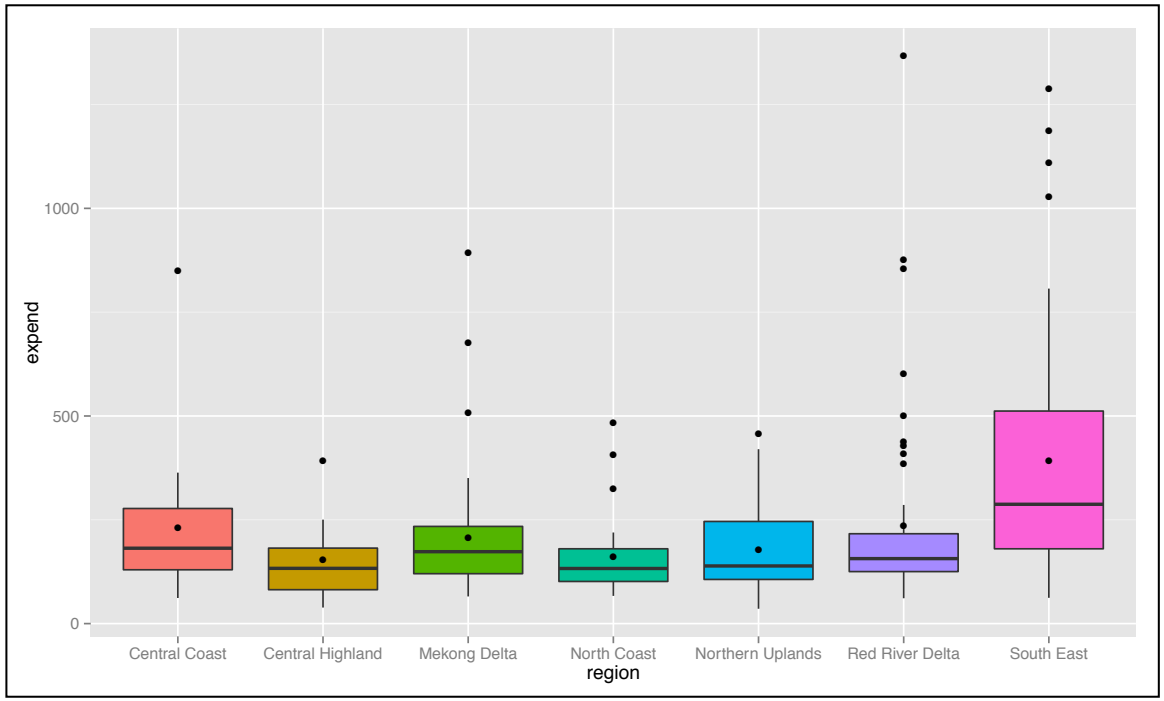
Statistical transformations are specified using the prefix `stat_` and a suffix that names the desired transformation

- Means, medians, and other summary statistics specified with `stat_summary()`
- Regression models specified with `stat_smooth()`

```
> ggplot(data = vlss, aes(x = region, y = expend, fill = region)) +
    geom_boxplot() +
    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.



Your Turn

How can we change the color, plotting character, or size (or a combination) of the mean points?

```
> ggplot(data = vlss, aes(x = region, y = expend, fill = region)) +
    geom_boxplot() +
    stat_summary(
        fun.y = mean,
        geom = "point",
        color = "black",
        pch = 19,
        size = 2
        )
```

The `pch=` argument sets the plotting character.

The `size=` argument sets the point size. (The default size is 4.)

```
> ggplot(data = vlss, aes(x = region, y = expend, fill = region)) +
    geom_boxplot() +
    stat_summary(
         fun.y = mean,
         geom = "point",
         color = "black",
         pch = 19,
         size = 2
         )
```

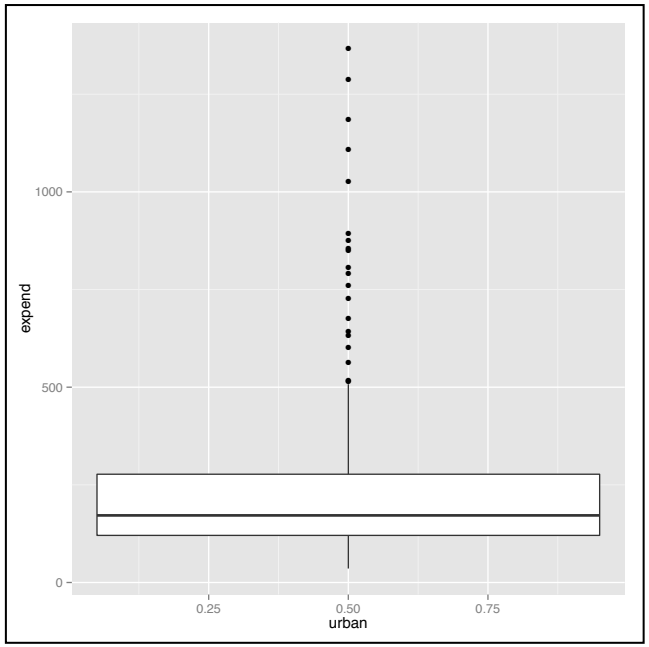
```
> ggplot(data = vlss, aes(x = region, y = expend, fill = region)) +
    stat_summary(
         fun.y = mean,
         geom = "point",
         color = "black",
         pch = 19,
         size = 2
         ) +
   geom_boxplot()
```

Order of the layers matters.

```
> ggplot(data = vlss, aes(x = region, y = expend, fill = region)) +
    geom_boxplot() +
    stat_summary(fun.y = mean, geom = "point", color = "black",
         pch = 19, 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

```
> ggplot(data = vlss, aes(x = urban, y = expend)) +
    geom_boxplot()
```



The plot is unexpected. Rather than showing the expenditures for the rural and urban households, there is only one boxplot!

The problem is that the urban variable is being treated as an integer.

```
> str(vlss)

'data.frame':  250 obs. of  4 variables:
 $ id    : int  404 409 509 1206 1528 1808 2104 2107 2218 2302 ...
 $ expend: num  427 877 854 1367 518 ...
 $ urban : int  1 1 1 1 1 1 1 1 1 1 ...
 $ region: Factor w/ 7 levels "Central Coast",..: 6 6 6 6 7 7 7 7 7 7 ...
```

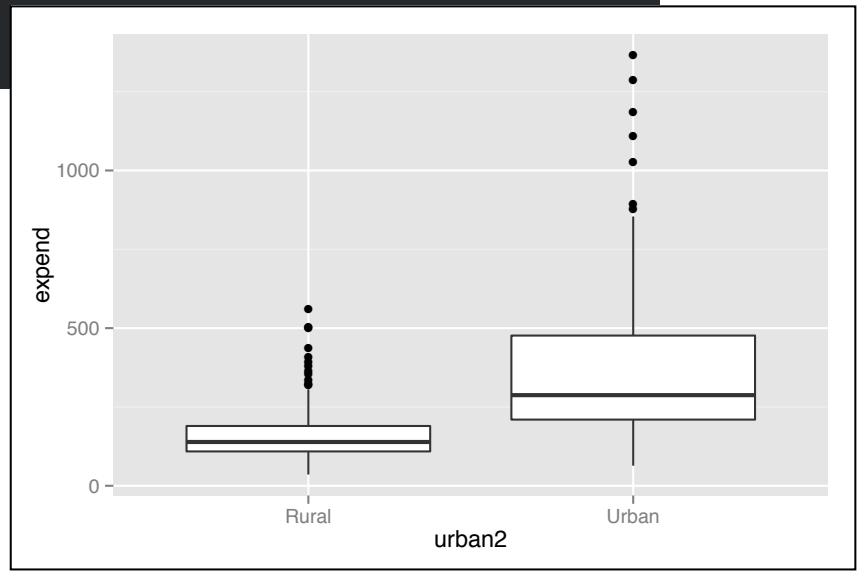
```
> vlss$urban2 = factor(vlss$urban,
    levels = c(0, 1),
    labels = c("Rural", "Urban")
    )

> head(vlss)

    id  expend urban          region urban2
1  404  427.45     1 Red River Delta  Urban
2  409  876.63     1 Red River Delta  Urban
3  509  854.11     1 Red River Delta  Urban
4 1206 1366.94     1 Red River Delta  Urban
5 1528  518.45     1      South East  Urban
6 1808 1027.99     1      South East  Urban
```

The solution is to coerce the `urban` variable into a factor using the `factor()` function.

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot()
```



Faceting

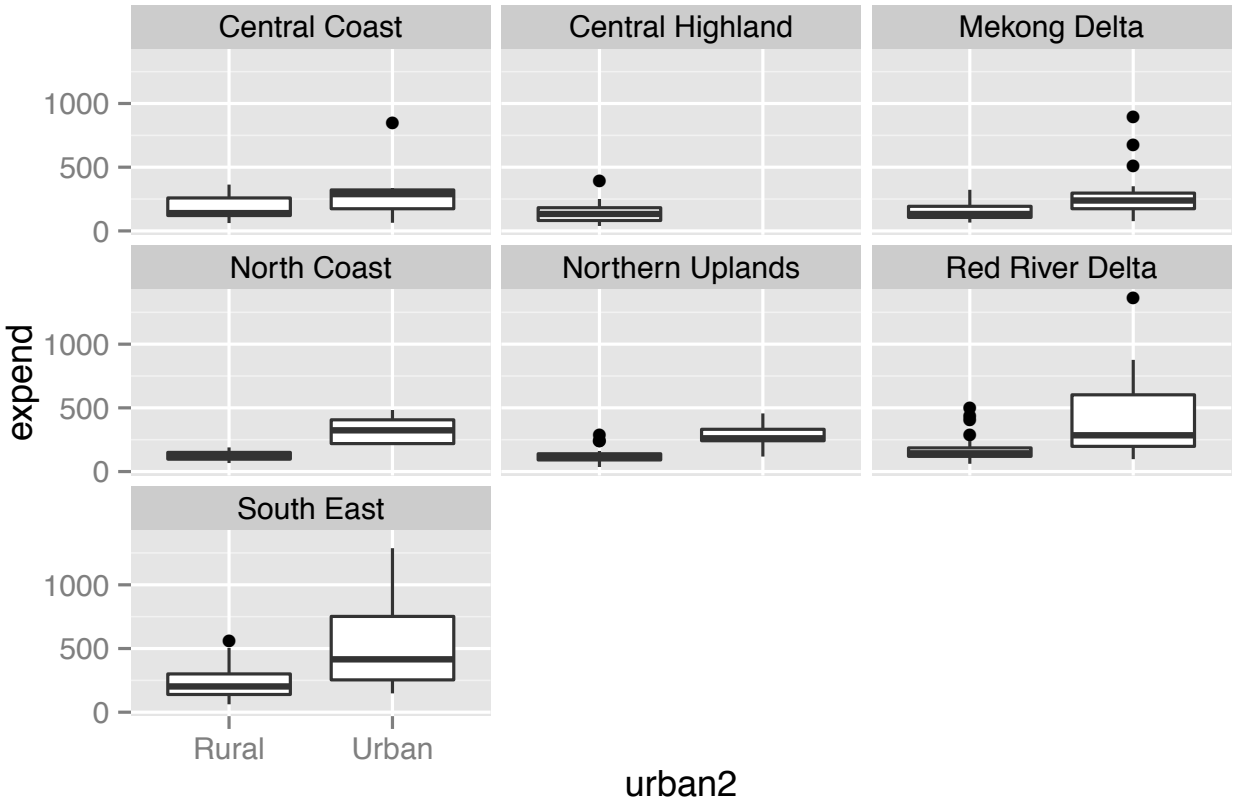
Faceting creates a separate plot for each subgroup declared

- `facet_wrap()` displays the plots conditioned on a **single predictor**
- `facet_grid()` displays the plots conditioned on **multiple predictors**

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot() +
    facet_wrap(~ region)
```

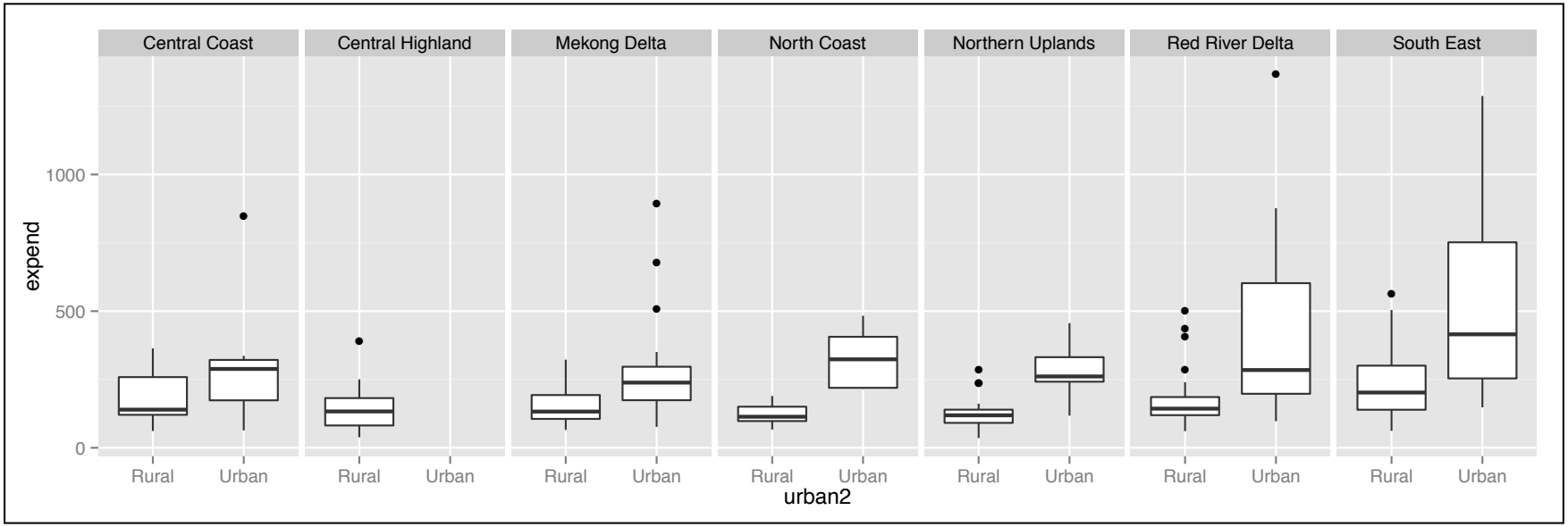
~ sets the predictor for conditioning

The boxplots compare the distributions of expenditures between rural and urban households conditioned on region.



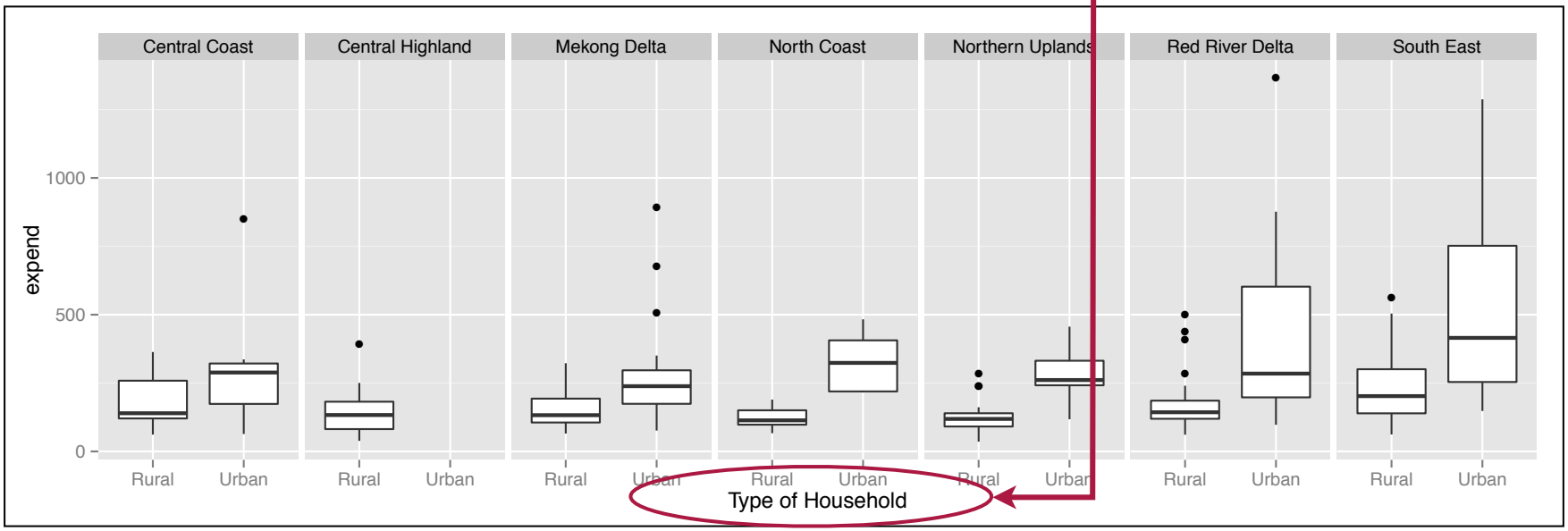
```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot() +
    facet_wrap(~ region, nrow = 1)
```

`nrow=` (and / or `ncol=`) sets the number of rows or columns in the plotting area



Changing the Axis Label

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot() +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household")
```



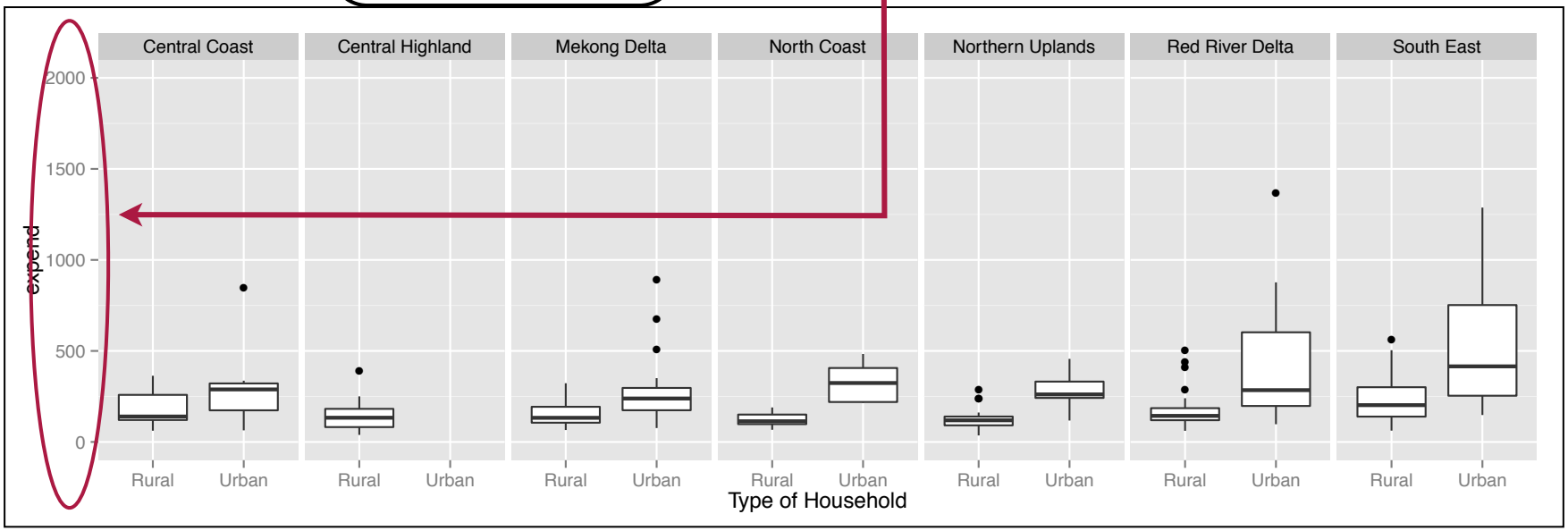
`xlab()` can be used to change the label on the x-axis, and `ylab()` is used to change the label on the y-axis.

Changing the Axis Limits

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot() +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylim(0, 2000)
```

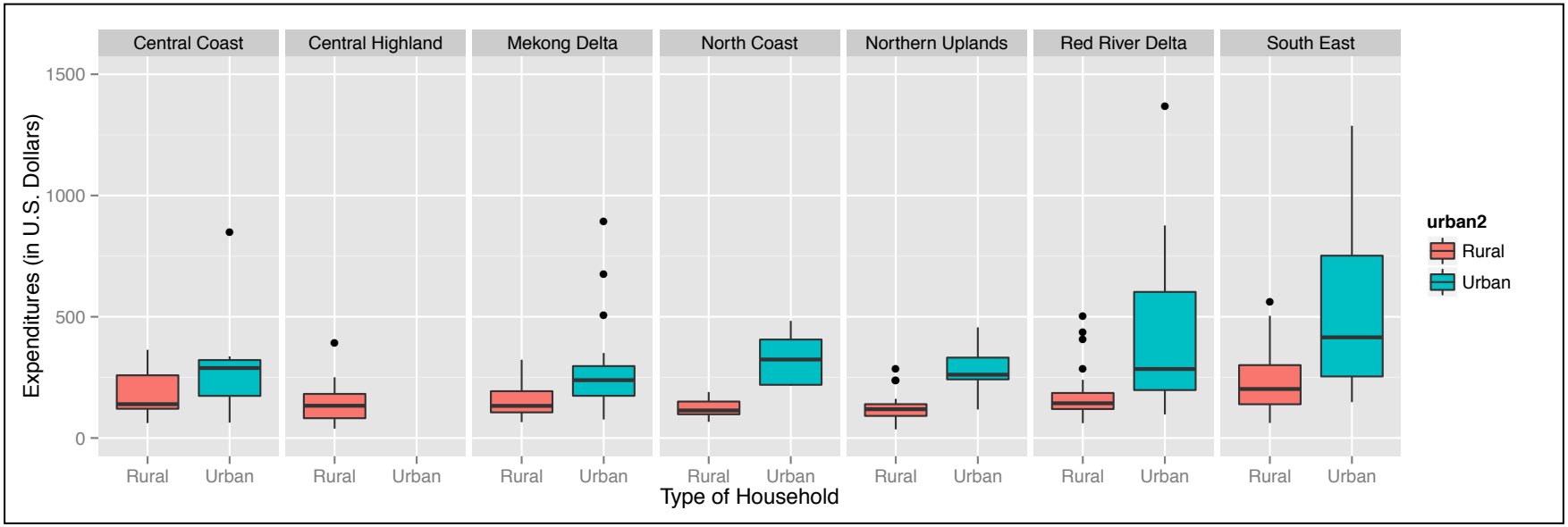
The first value is the minimum.

The second value is the maximum.



`xlim()` and `ylim()` are used to set the limits on the x-axis and y-axis respectively.

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000)
```



Fine-Tuning the Color

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000) +
    scale_fill_manual(
        values = c("#E69F00", "#56B4E9")
        )
```

`scale_fill_manual()` allows you to manually set the attributes associated with the fill aesthetic.

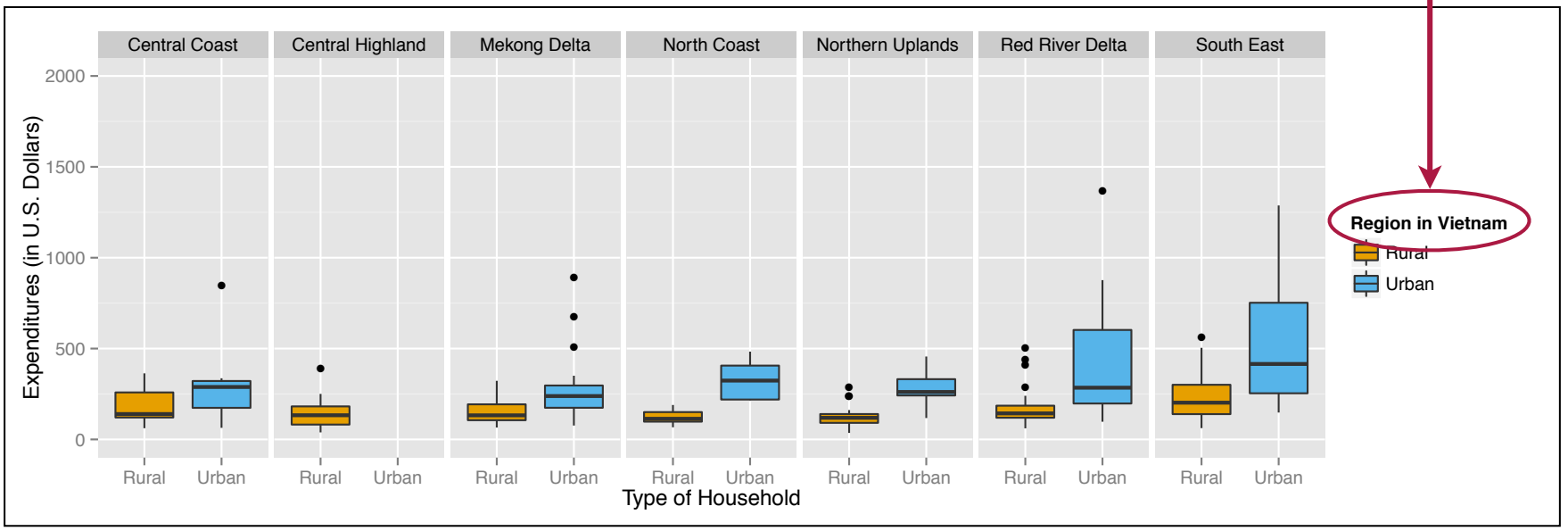
The `values=` argument sets the color values for each level of the factor.

RGB or hex values can be used in `values=` argument of `scale_fill_manual()` or `scale_color_manual()`.

`scale_color_manual()` can be used to manually set the colors when the `color=` argument is used.

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000) +
    scale_fill_manual(
        values = c("#E69F00", "#56B4E9"),
        name = "Region in Vietnam"
        )
```

The `name=` argument changes the title of the legend.



Choosing a Color Palette

`colors()` will provide a list of all the **named colors** available in R.

```
> colors()

  [1] "white"                "aliceblue"            "antiquewhite"
  [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
  [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
[10] "aquamarine2"           "aquamarine3"          "aquamarine4"
        ⋮                       ⋮                      ⋮
```

Most universities have official colors. The University of Minnesota's two official colors (for electronic display) are:

- `#ffcc33` (gold)
- `#7a0019` (maroon)

See more at: https://www.ur.umn.edu/brand/requirements-and-guidelines/color-and-type/

The U of M also has an entire palette of secondary colors available at:
https://www.ur.umn.edu/brand/assets/pdf/secondary_colors_rgb.pdf

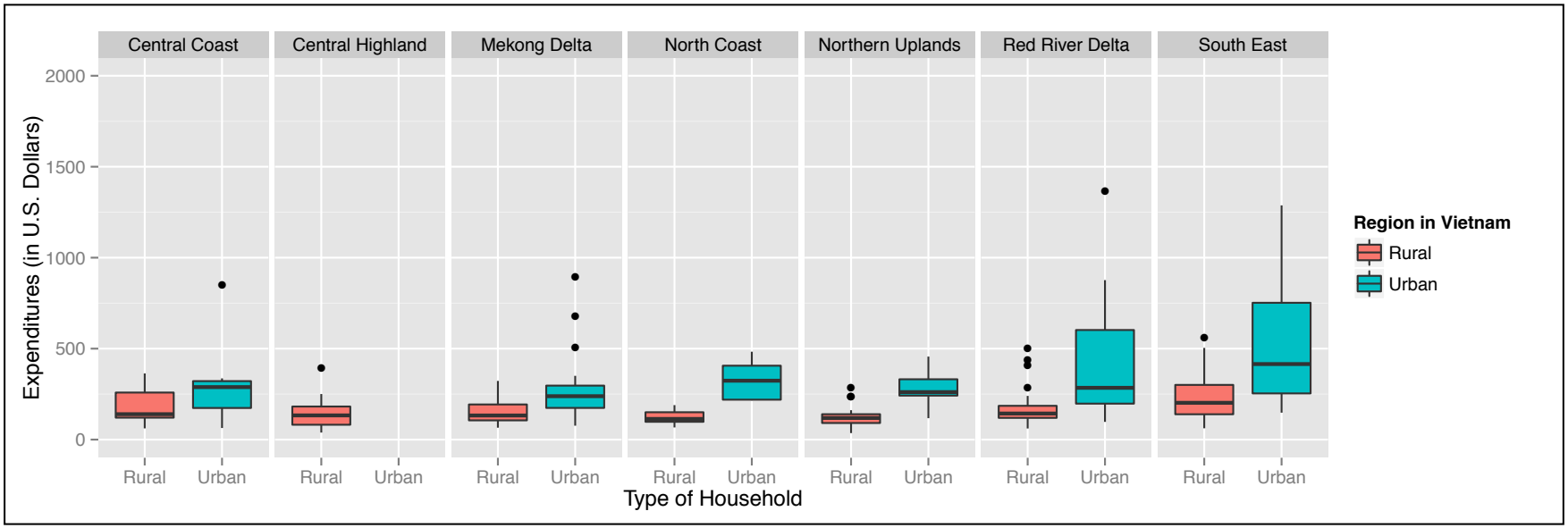
Pre-Selected Color Palettes

There are several "built-in" color palettes available for use in ggplot

Fill Scale	Color Scale	Description
`scale_fill_hue()`	`scale_color_hue()`	Colors evenly spaced around the color wheel
`scale_fill_grey()`	`scale_color_grey()`	Grey scale palette
`scale_fill_brewer()`	`scale_color_brewer()`	ColorBrewer palettes

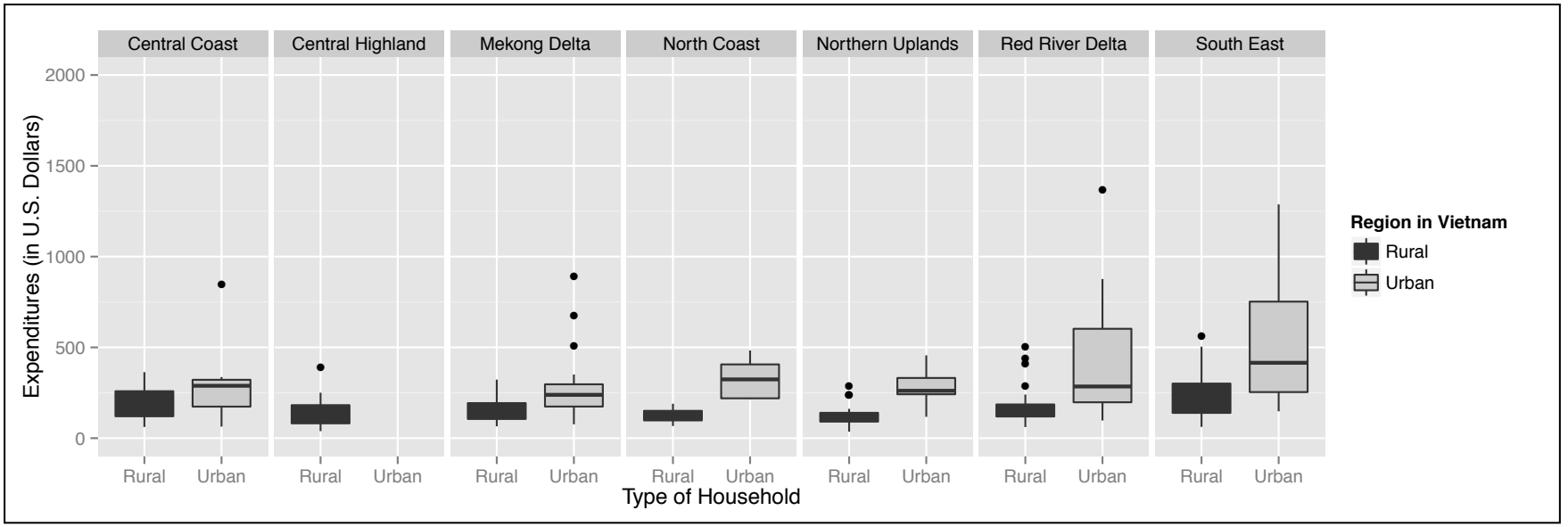
Default Color Palette

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000) +
    scale_fill_hue(name = "Region in Vietnam")
```



Grey Scale Color Palette

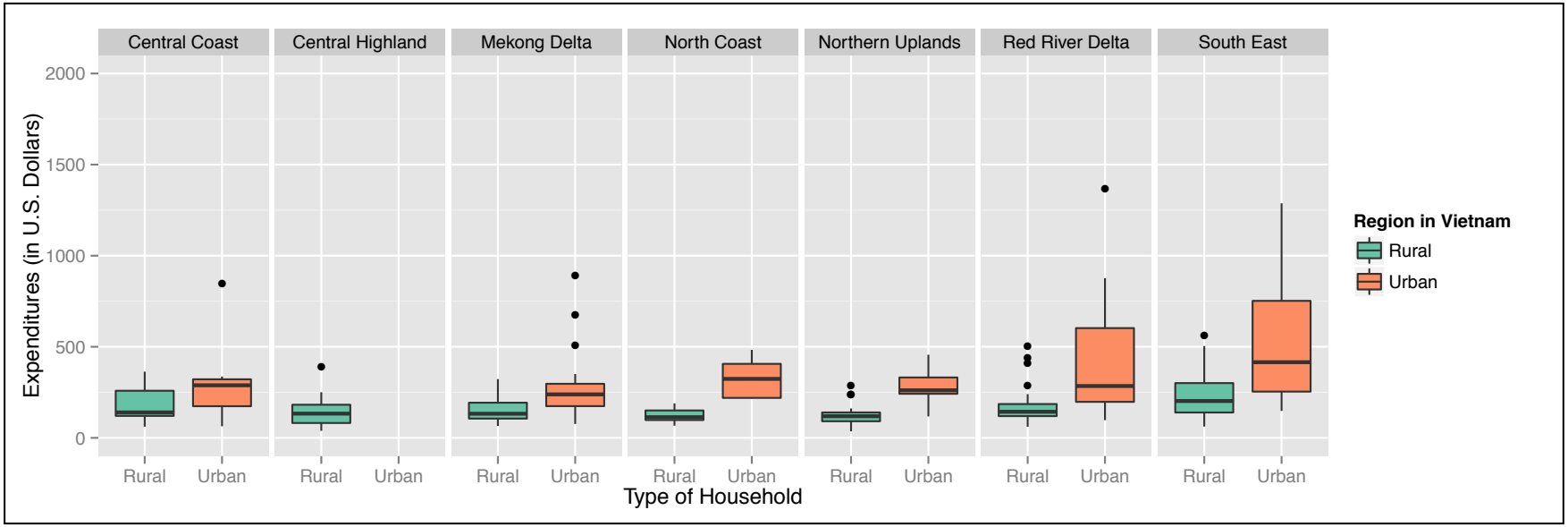
```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000) +
    scale_fill_grey(name = "Region in Vietnam")
```



Brewer Color Palette

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    ylim(0, 2000) +
    scale_fill_brewer(
        name = "Region in Vietnam",
        palette = "Set2"
        )
```

The `palette=` argument sets the pre-specified color palette.



Color Brewer

Cynthia Brewer chose color palettes that not only are aesthetically pleasing, but also based on how humans perceive the colors that are displayed.

http://www.colorbrewer2.org

She has palettes for three different types of data

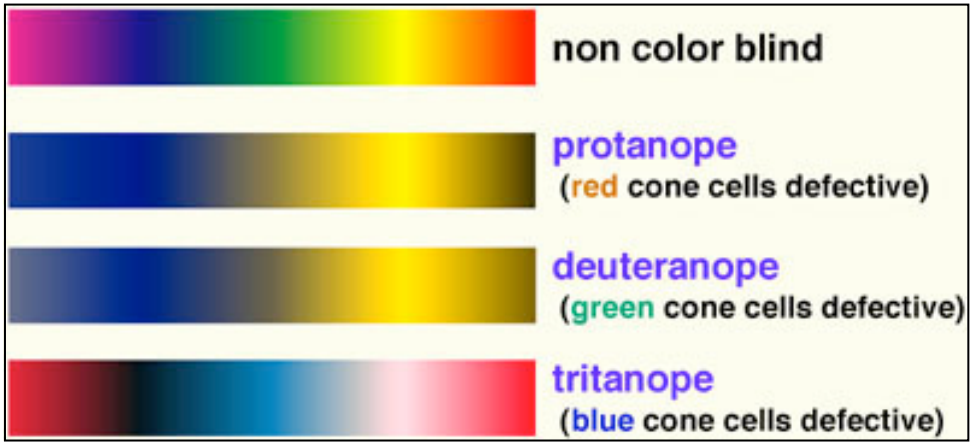
- **Qualitative/Categorical**—colors do not have a perceived order
- **Sequential**—colors have a *perceived order* and perceived difference between successive colors is uniform
- **Diverging**—two back-to-back sequential palettes starting from a common color (e.g., for Likert scale data)

There is a very readable introduction to color brewer palettes at http://mkweb.bcgsc.ca/brewer/

Palettes for Color-Blindness

About 8% of males and ½% of females have some form of color vision deficiency (good chance that someone in your audience will be one of these people)

Color *and* grey-scale palettes have been developed for use with people that have the more common forms of color-blindness



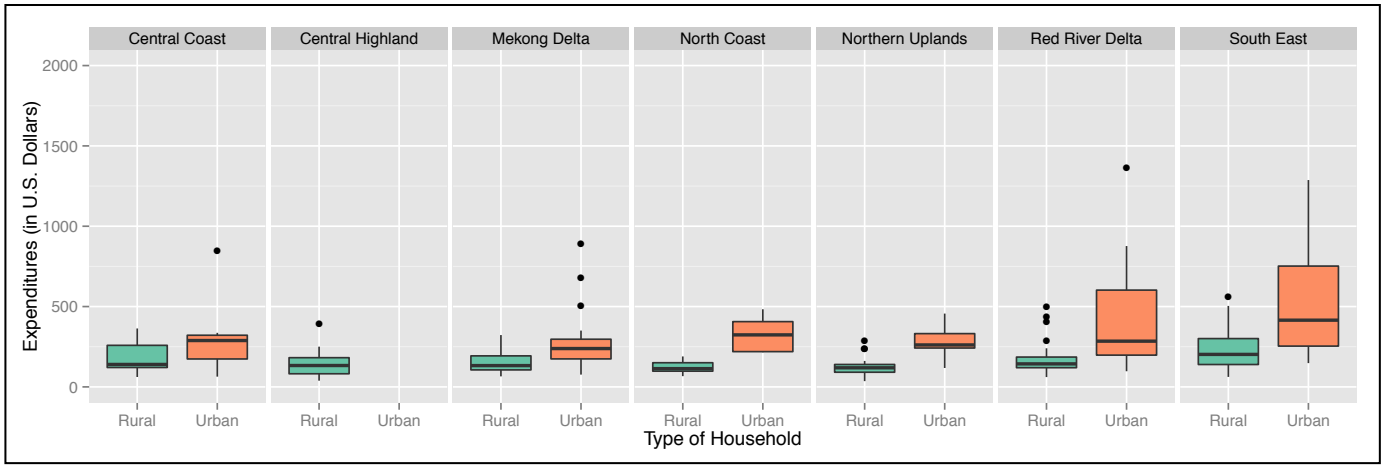
There is more information related to color-blindness and the creation of suitable color palettes for scientific figures at http://jfly.iam.u-tokyo.ac.jp/color/

There is a large body of research literature related to the creation of suitable color palettes for figures. As a starting point,

Lumley, T. (2006). Color-coding and color blindness in statistical graphics. *Statistical computing and graphics newsletter*. http://www.amstat-online.org/sections/graphics/newsletter/Volumes/v172.pdf

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    scale_fill_brewer(
         name = "Region in Vietnam",
         palette = "Set2"
         ) +
    theme(legend.position = "none")
```

The `legend.position=` argument can be used to move or remove the legend.

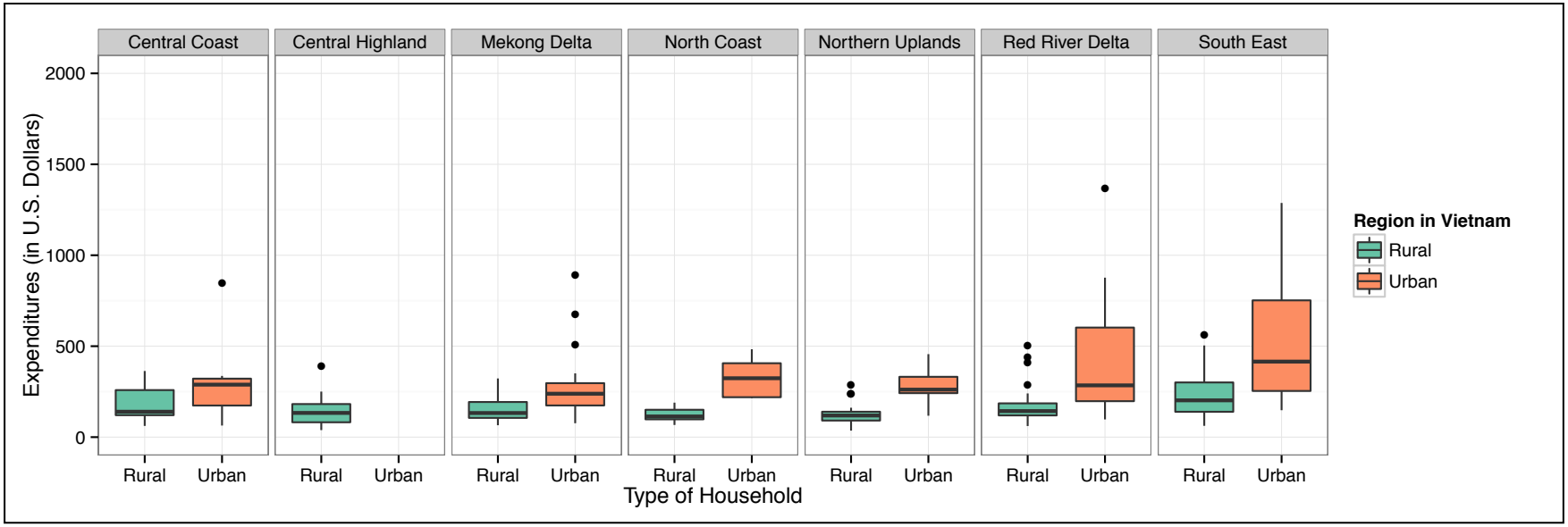


The `theme()` function can be used to change *every* element in the plot (e.g., grid lines, font, color, etc.). See http://docs.ggplot2.org/current/theme.html

Using "Built-In" Themes

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    scale_fill_brewer(
        name = "Region in Vietnam",
        palette = "Set2"
        ) +
    theme_bw()
```

The `theme_bw()` function is a "built-in" theme that uses a black-and-white background (rather than grey).

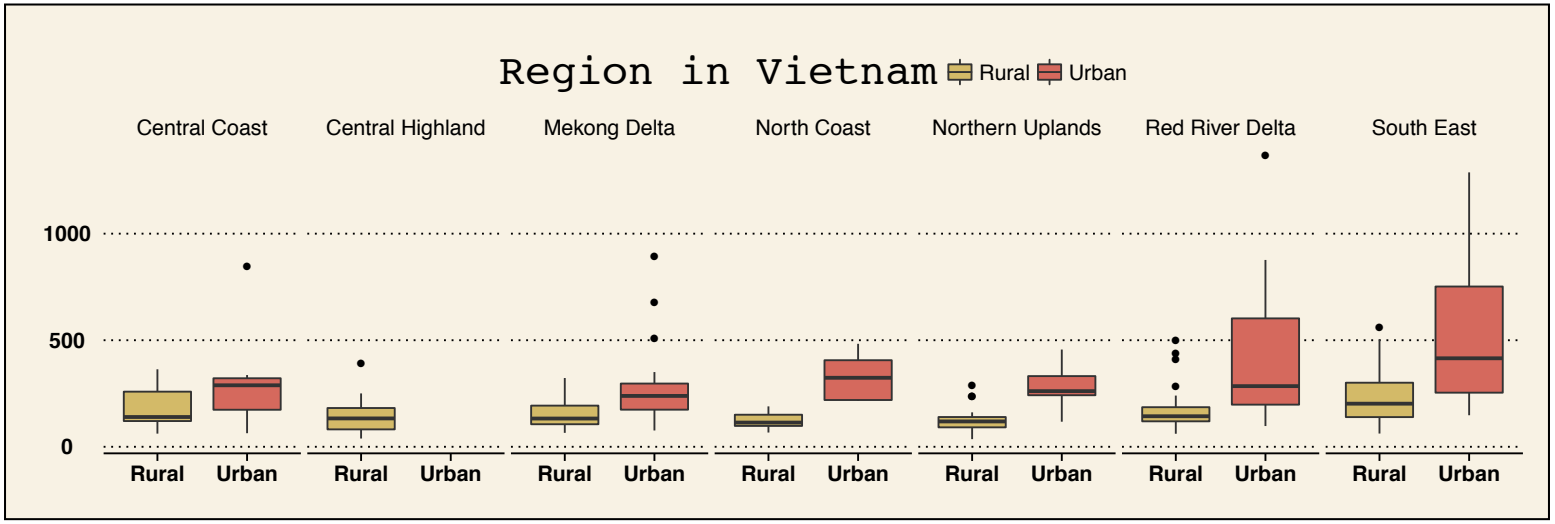


There are many other themes available

- http://drunks-and-lampposts.com/2012/10/02/clegg-vs-pleb-an-xkcd-esque-chart/
- https://github.com/jrnold/ggthemes

```
# Install the ggthemes library then load it
> library(ggthemes)

> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    theme_wsj() +
    scale_fill_wsj(name = "Region in Vietnam", palette = "rgby")
```



You can also build your own themes and use them.

Putting It All Together

```
> ggplot(data = vlss, aes(x = urban2, y = expend)) +
    geom_boxplot(aes(fill = urban2)) +
    facet_wrap(~ region, nrow = 1) +
    xlab("Type of Household") +
    ylab("Expenditures (in U.S. Dollars)") +
    scale_fill_brewer(name = "Region in Vietnam", palette = "Set2") +
    theme_bw() +
    theme(legend.position = "none")
```

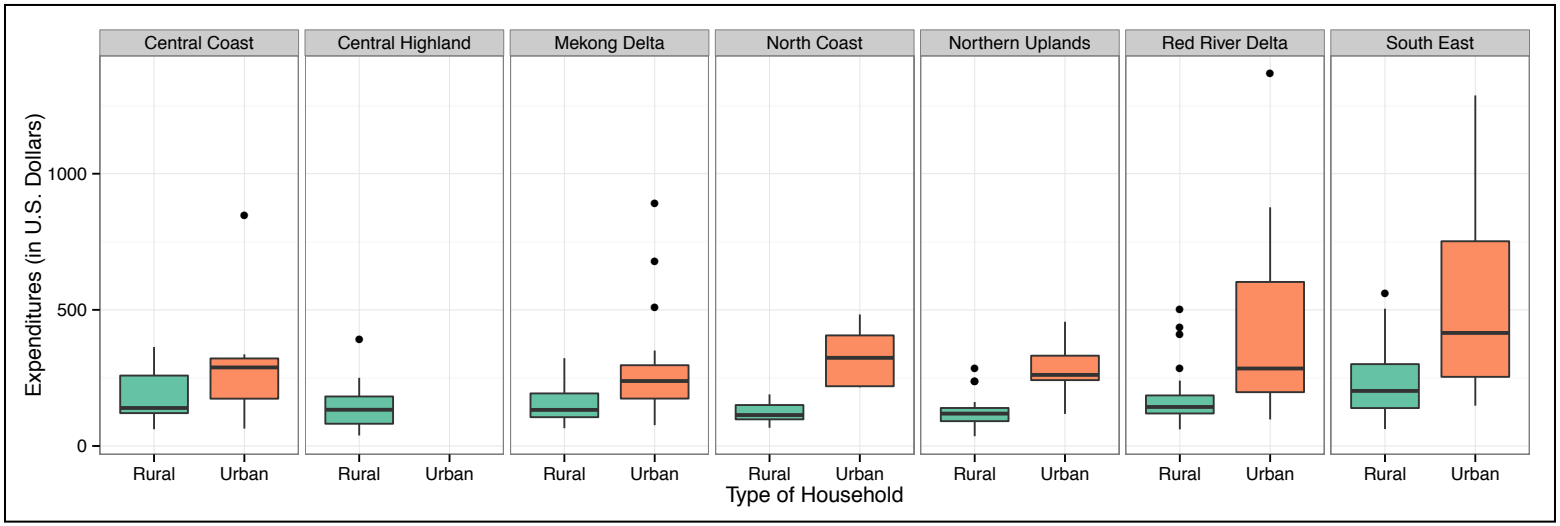


Figure 1. Expenditures (in U.S. dollars) for rural and urban households conditioned on region.

It is easier to use a word-processor (e.g., Word) to add the figure title and caption than to try and get it formatted correctly using R.