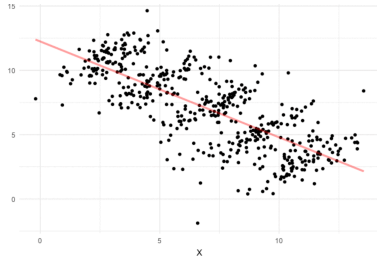
Multilevel Modeling

Introduction

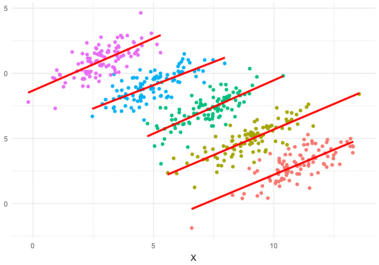
Exploratory Data Analysis *(including visualization)* is often the first step in business analysis. It is seldom the last, as it does not provide predictive values or business driver analysis *(we define business drivers to mean dimensins with causal, quantitative relationships to business outcome)*.

Unfortunately, the business world is flush with dashboarding technology marketed as panaceas, which encourages oversimplification and misapplications - which is an amplification of an old problem: **Simpson's Paradox** *(first described by Edward Simpson in 1951)*.

Simpsons Paradox involves cases where data groups *(e.g., products, offices, business units, locations,etc)* are aggregated, and the business drivers *(causes, correlations and effects)* change in magnitude and direction between levels. For example, the top aggregate level might show a negative correlation:



while correlations at the group levels are positive:



The result is often poor conclusions - maybe Simpson's paradox is named after Homer?

Competent business analysts must understand business drivers and outcomes at all levels and dimensions: Will an increase in demand for Dallas Cowboy branded apparel affect sales in Houston the same as Dallas? Will it affect sales in Womens Departments the same as Mens? What about a new Mens department in Atlanta? How do we plan inventory? To plan inventory, we need to quantify the effect of demand drivers on each of these locations and departments *(two levels in this case)*. This requires **multilevel, explanatory modeling**.

Multilevel Modeling Exercises

Load the following libraries

```
library(tidyverse)
library(lme4)
library(cowplot)
library(ggplot2)
library(lubridate)
library(ggridges)
library(kableExtra)
```

You've already worked with multilevel modeling. Recall that a regression equation typically takes the form:

$y = \beta_0 + \beta_n X + \epsilon$

and when you used a **varying intercepts model**, the equation became:

$y = \beta_0 + \alpha_m + \beta_x X + \epsilon$

where α_m was the variance of the group level intercept from the base level intercept. We used varying intercept models to minimize error within a fixed slope constraint. Recall:

```
Auto =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Github Staging/EllenwTerry/Archive/Data_Files/Automobi
Auto = select(Auto, Price = price, Make = make, Horsepower = horsepower) %>%
  filter(Make %in% c('bmw', 'honda'))
mod = lm(Price ~ Horsepower + Make, Auto)
coef(mod)
```

```
## (Intercept)  Horsepower   Makehonda
##  -4752.6435    222.2963  -4897.6644
```

The Intercept in lm related to the BMW group, and the Makehonda *(recall the model matrix from normal eqautions)* value is the **effect** of the Honda group, *sort of - it's in relation to the (bmw + intercept) value* - so, in this case, the Price of a honda will average $4,897 less than BMW for the same Horsepower. Again, we have a varying intercept here and a fixed slope.

Of course, a fixed slope constraint does not always produce the model we want, and we can expand the model to include intercepts and slopes. The equation then expands to:

$y = \beta_0 + \alpha_{m,n} + (\beta_1 + \gamma_{m,n})X + \epsilon$

The α and γ are often called "random" effects, but we'll just use the term **effect**. In frequentist terminology, the effect is **generally** determined by:

$\alpha_{m,n} = \frac{\sigma_b^2}{\sigma_b^2 + \sigma_m^2}$

$where \sigma_m^2$ is the variance within groups, and σ_b^2 is the variance between groups. This is a weighted, or **pooled** approach.

We're going to explore 3 different pooling approaches to modeling effects:

1. No Pooling *(seperate models for each data group)*
2. Full Pooling *(single model for all groups)*
3. Partial Pooling *(models that balance variance error across groups)*

Each of these has strengths and weaknesses. These exercises are intended to build the intuition before we move into more advanced modeling.

Load and transform data as follows *(this is sales data is for a chain of retail outlets with 10 merchandise groups across 10 locations)*:

```
SalesTrans =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Sales.csv")
Location =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Location.csv")
MerGroup =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/MerGroup.csv")
SalesTrans = SalesTrans %>% inner_join(Location, by = "LocationID")
SalesTrans = SalesTrans %>% inner_join(MerGroup, by = "MerGroup")
LocationID = as.factor(SalesTrans$LocationID)
SalesTrans$ProductID = as.factor(SalesTrans$ProductID)
SalesTrans$Description = as.factor(SalesTrans$Description)
SalesTrans$MerGroup = as.factor(SalesTrans$MerGroup)

# breaking out Q4 to simplify exercise

SalesTrans$Qtr = quarter(SalesTrans$Tdate)
SalesTrans = filter(SalesTrans, Qtr == 4)
```

For this exercise, we're interested in analyzing total sales by week *(The data are individual transactions)*. So let's summarize to get total sales by Wk:

```
SalesTransSummary = SalesTrans %>%
  group_by(Description, MerGroup, MfgPromo, Wk ) %>%
  summarise(Volume = n(), TotSales = sum(Amount) )
```

Exploring the data: Notice that not all locations have all MerGroups *(important point)*:

```
LocSum = SalesTransSummary %>% group_by(Description, MerGroup) %>%
  tally() %>%
  group_by(Description) %>% tally()

# tally() rolls up, so we count at the lowest level first

knitr::kable(LocSum) %>%
  kable_styling(full_width = F, bootstrap_options = "striped", font_size = 9)
```

Description	n
Arlington	5
Atlanta	8
Baltimore	8
Boston	9
Cleveland	8
Dallas	6
Los Angeles	8
San Francisco	10
San Jose	8
Seattle	8

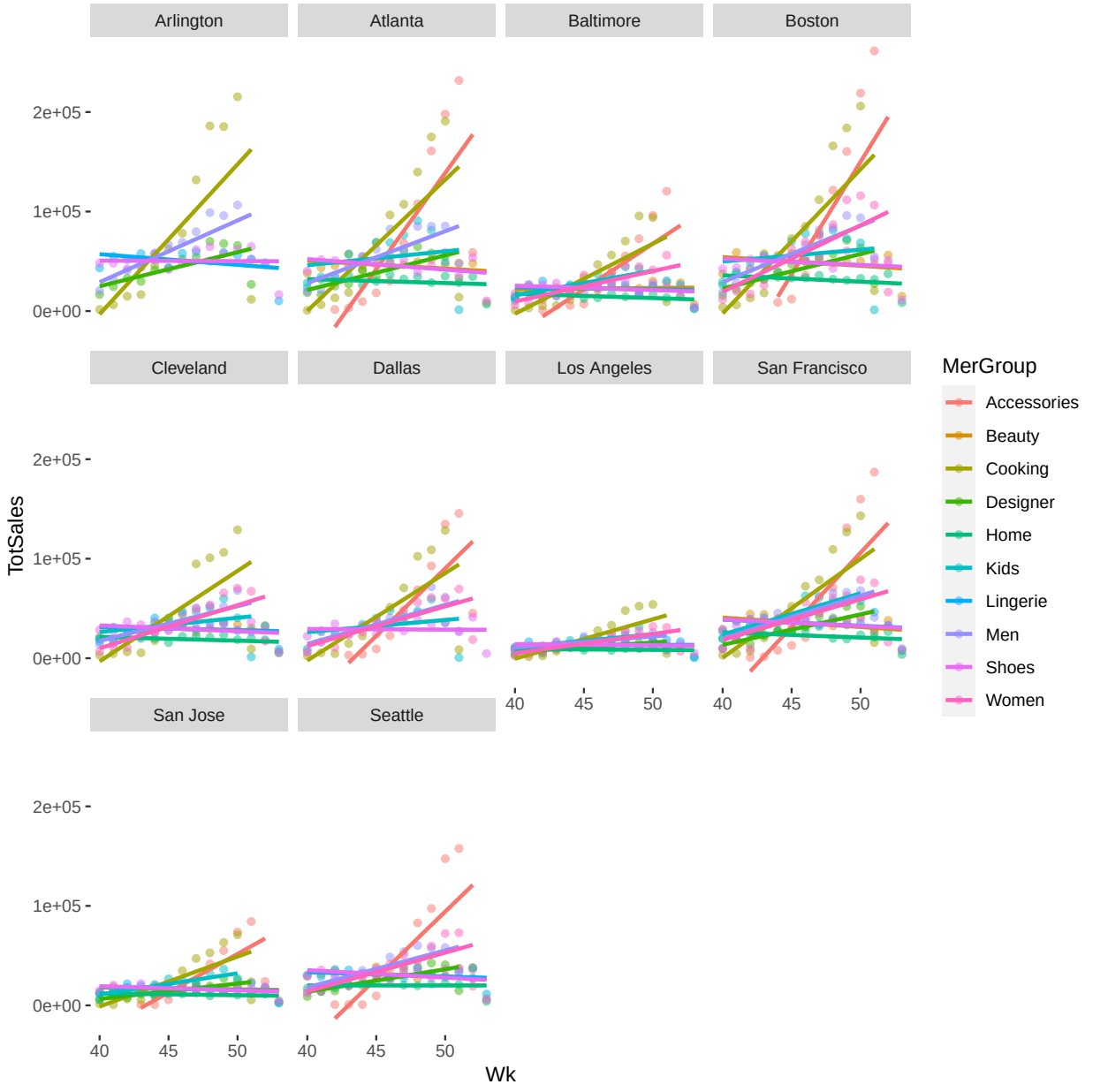
We're going to use the lmer function *(lme4 package)* to build models here, but we'll migrate this approach to Bayesian modeling later in the course.

No Pooling - Independent Models

Now, lets create our first series of multilevel models. We'll use ggplot and create groups for MerGroup *(defining color will force ggplot to create a group)*, and Locations *(Description - defining a facet will also*

force ggplot to create groups):

```
p =  ggplot(SalesTransSummary, aes(Wk, TotSales, color = MerGroup)) +
  geom_point(alpha = .5) +
  geom_smooth(method = "lm", se = F) +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```

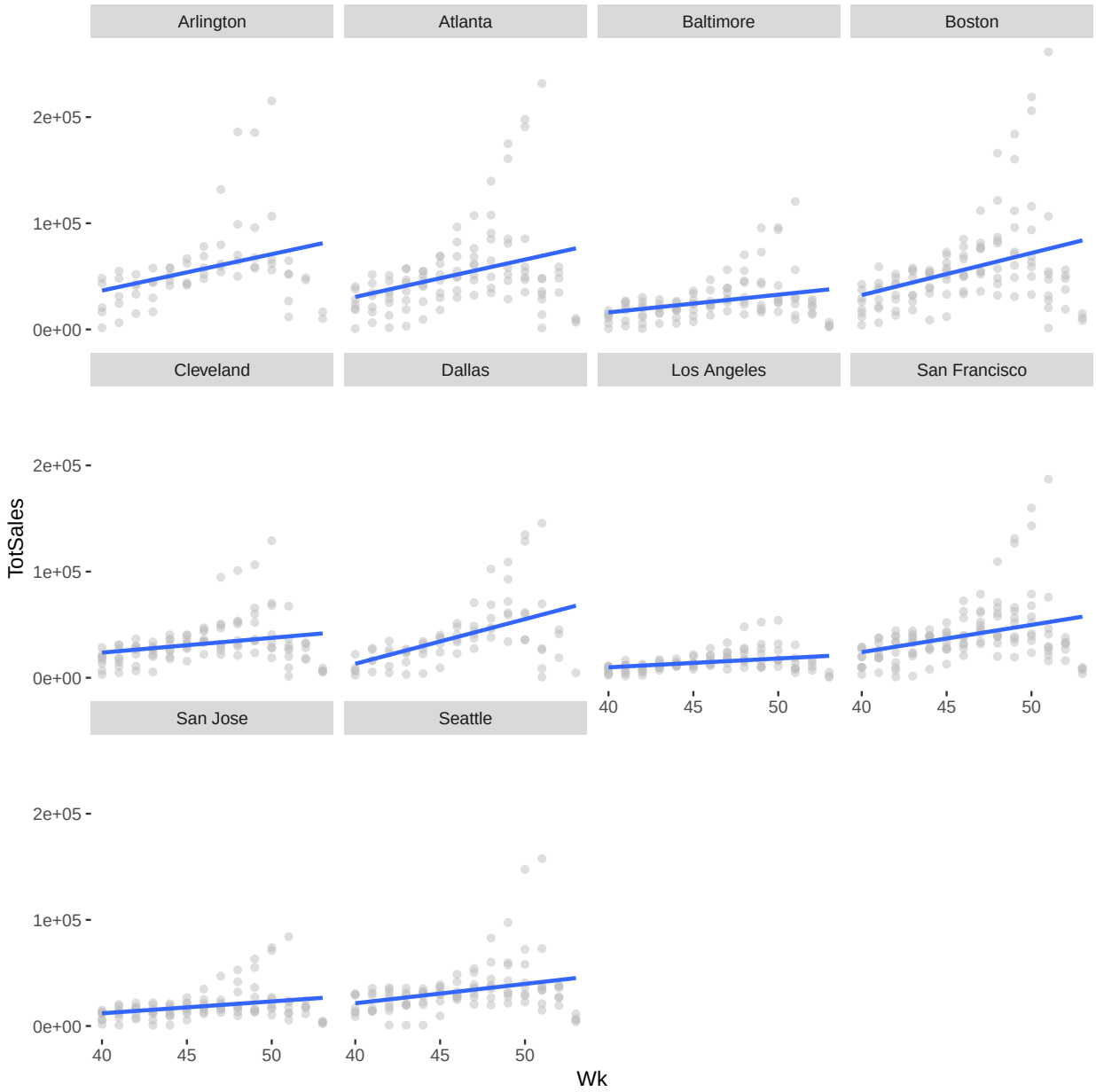


These models are truly independent *(not pooled)* - ggplot creates a separate model for each Description and MerGroup. So, in this case, 78 seperate models!

No Pooling - Single Level *(Group)* Models

We'll start with with ggplot models again - but this time with a single grouping level. let's generate a set of 10 models based on Description:

```
p =  ggplot(SalesTransSummary, aes(Wk, TotSales)) +
  geom_point(color = "gray", alpha = .5) +
  geom_smooth(method = "lm", se = F) +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```

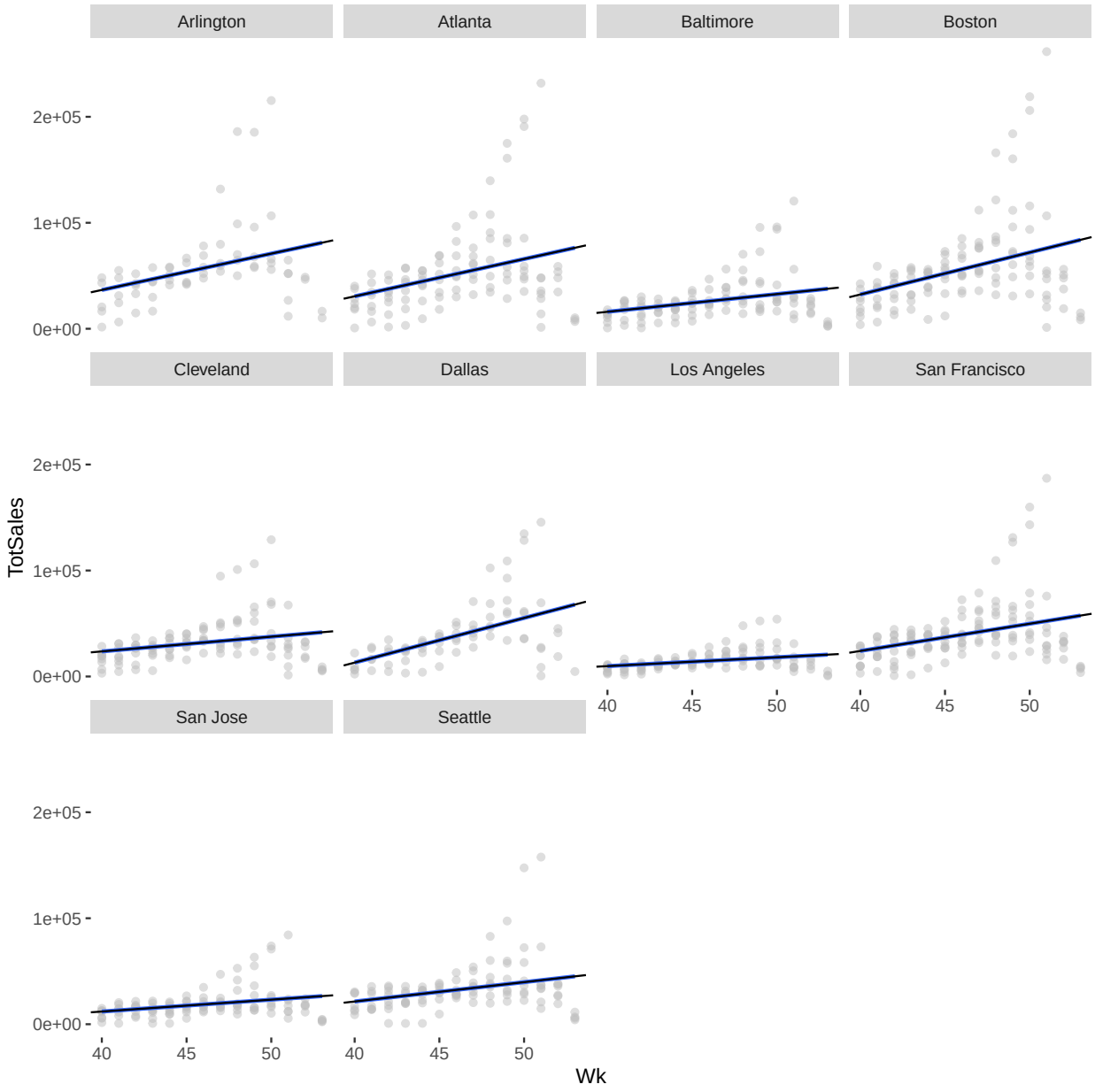


Notice again that these models still have different slopes and intercepts. And correlation is not shared *(no pooling)*.

At this point, we'll introduce a function of lme4: lmList. *(good reference here: https://cran.r-project.org/*

web/packages/lme4/vignettes/lmer.pdf). This will give you the coefficients without going through the entire modeling process. The code below will create a series of models based on one grouping variable. Then we can just pass the coefficients to abline and plot them out *(note the syntax)*:

```
NoPoolCoef = lmList(TotSales ~ Wk | Description, data = SalesTransSummary, RMEL = FALSE) %>%
  coef() %>%
  rownames_to_column("Description")

p = p + geom_abline(data = NoPoolCoef, aes(intercept = `(Intercept)`, slope =  Wk))
p
```

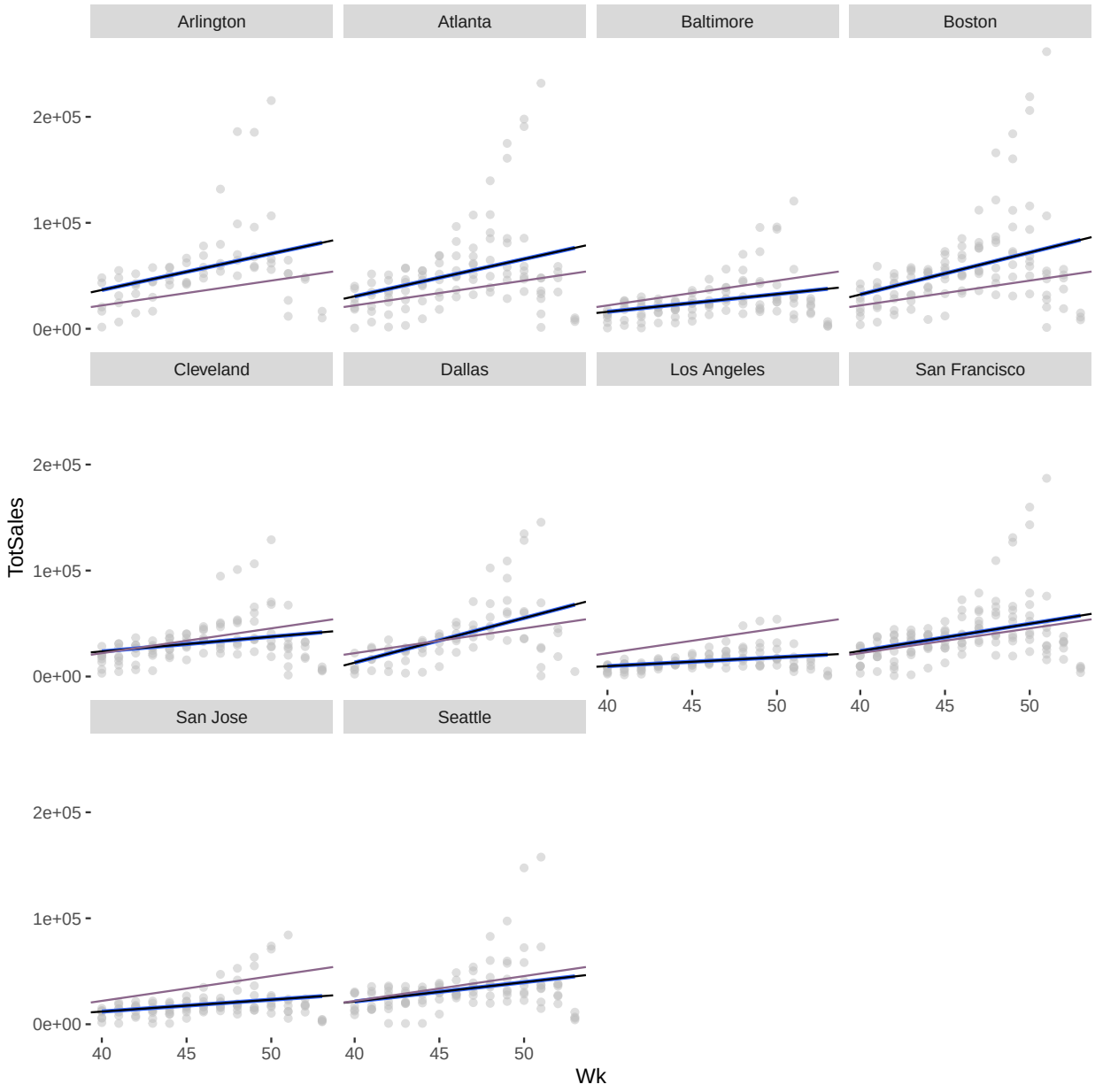


Notice that, in this case, lmer gets the same coefficients as lm. *(lmer grouping syntax at the end)*.

Full Pooling

Fully Pooled models don't change with groups - they're fit using all the data and ignore correlation between groups. The code below creates a fully pooled model:

```
fpLMod = lm(TotSales ~ Wk, SalesTransSummary )
fpLMCoef = coef(fpLMod)

p = p +
  geom_abline(aes(intercept = fpLMCoef[1],
                  slope = fpLMCoef[2]), color = "plum4")
p
```



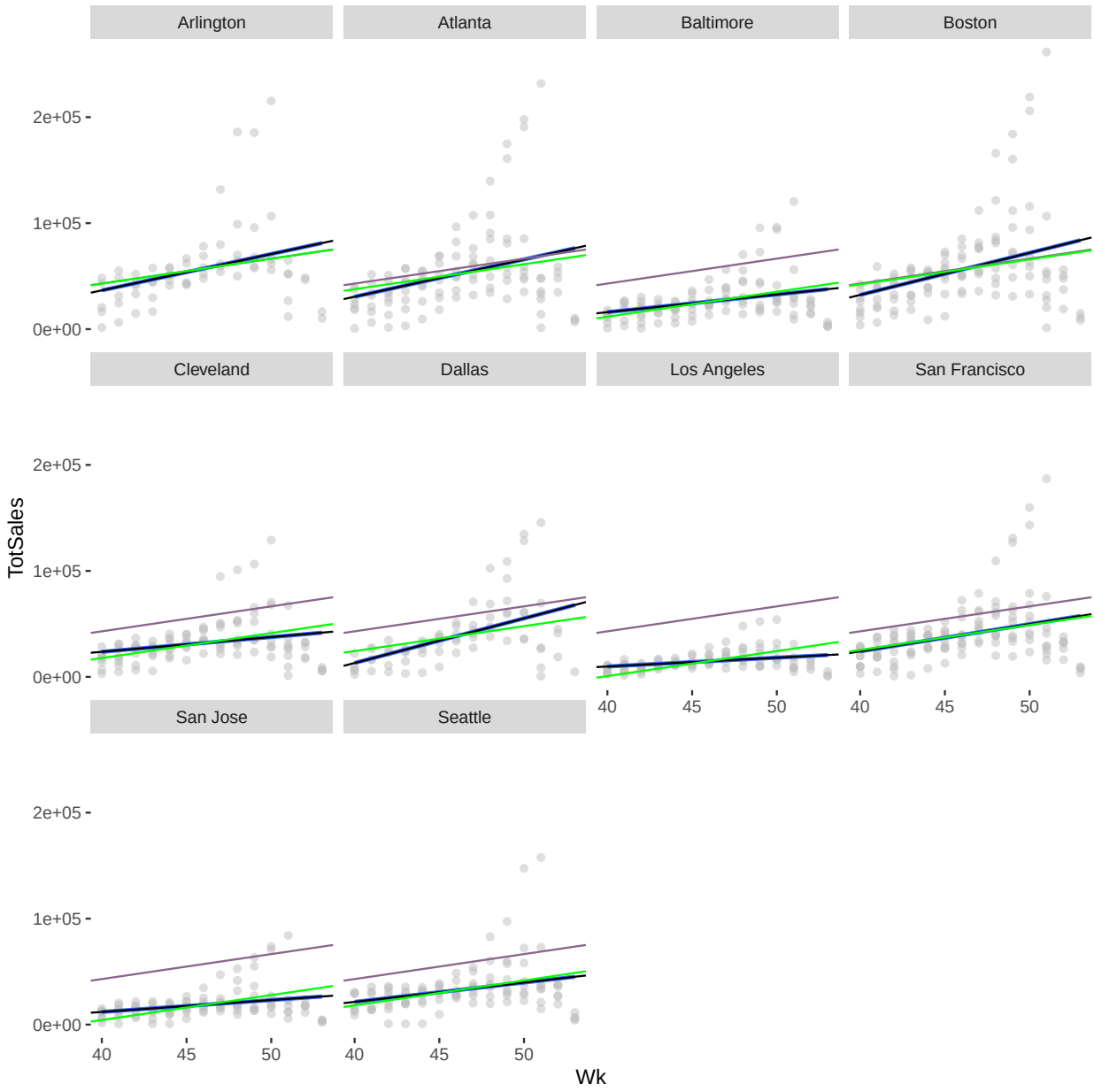
The plot above compares the fully pooled model with the independent, single group level model. Note how the fully pooled model doesn't change across groups. You might be thinking that a no-pooled model is more

accurate, but remember that these data are usually just snapshots of a broader, more dynamic environment. If you're planning, you will often find that the consistency *(lower variability)* of a fully pooled moded fits your purposes better. But these are not the only choices:

Varying Intercept Models *(Partial Pooling)*

Now let's apply a varying intercept model *(like the Auto Price data we modeled earlier)* for one level *(location)* using lm:

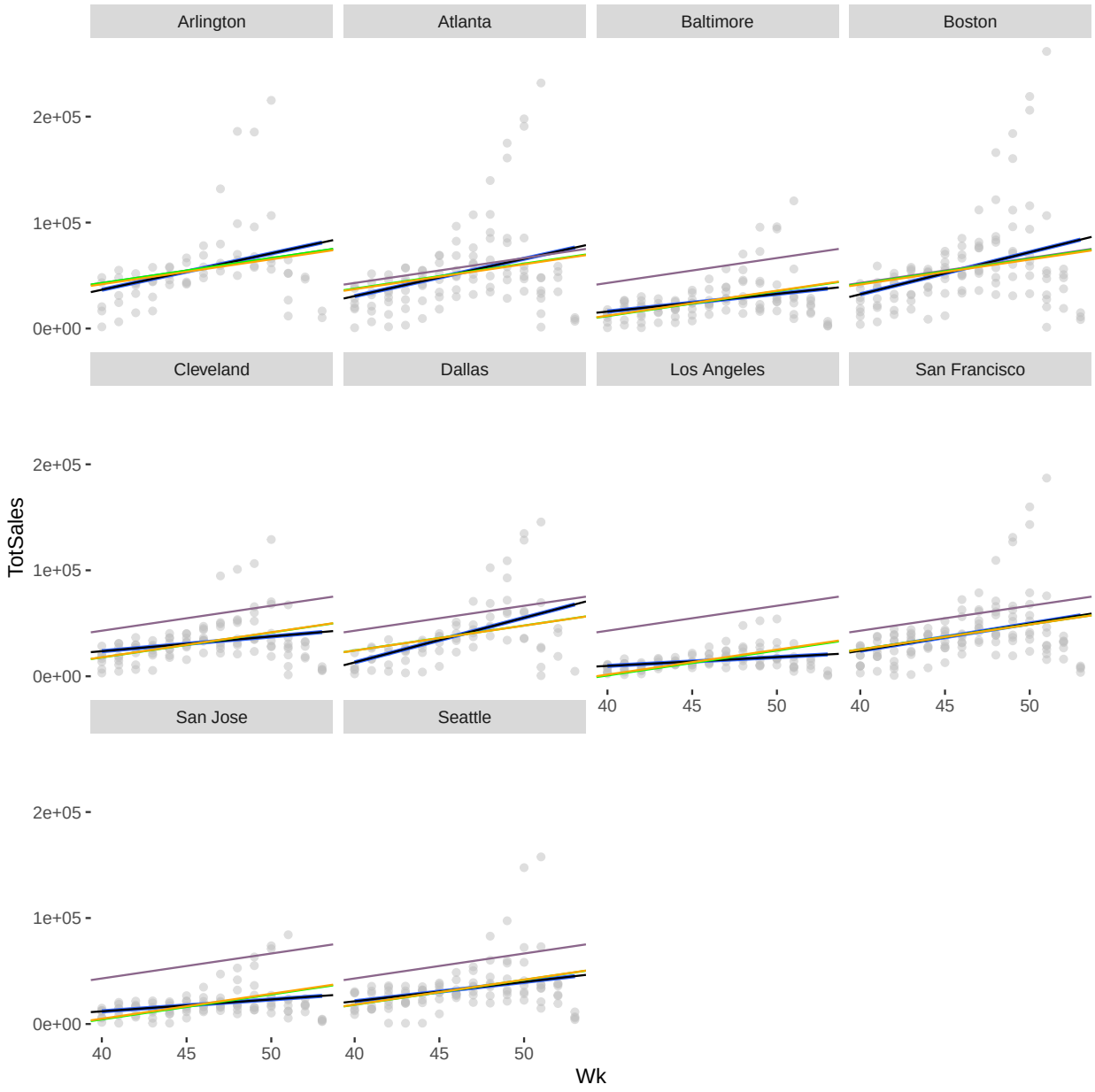
```
LMVarIntMod = lm(TotSales ~ Wk + Description, SalesTransSummary )
fpLMCoef = coef(LMVarIntMod)

LMVarIntCoef = data.frame(Description = unique(SalesTransSummary$Description),
                          fpLMI = c( coef(LMVarIntMod)[1],
                                     coef(LMVarIntMod)[1] +
                                       c(coef(LMVarIntMod)[3:11])),
                          fpLMS = c( rep(coef(LMVarIntMod)[2], 10)))




p = p + geom_abline(data = LMVarIntCoef,
                    aes(intercept = fpLMI, slope =  fpLMS),
                    color = "green")
p
```



And we can use lmer to do the same. This is how you create a varying intercept model in lmer *(note syntax and refer to manual)*:

```
LMERVarIntMod = lmer(TotSales ~ 1 + Wk + ( 1 | Description),
                     data = SalesTransSummary,
                     REML = FALSE)

LMERVarIntCoef = data.frame(coef(LMERVarIntMod)$Description) %>%
  rownames_to_column("Description") %>%
  rename(Intercept = `X.Intercept.`, Slope = Wk)

p = p + geom_abline(data = LMERVarIntCoef,
                    aes(intercept = LMERVarIntCoef$Intercept,
                        slope =  LMERVarIntCoef$Slope), color = "orange")
```

```
p
```



Observe that the green lines from the lm varying intercept model have been overridden - the coefficients are the same. These models are *partially pooled*, they share correlation with other groups.

> [Note: in the code above, I'm "collecting" coefficients and putting them in a dataframe that I can pass to ggplot and then use abline to plot the regression lines. A few things to consider:
>
> 1. ggplot will look at the column headers and establish names for intercept and slope in the first plot. You can all these whatever you like, but additional plots must use the same names
> 2. lm outputs coefficents differently than lmer. lm will give you the base intercept + the first member of the group *(alphabetically)*, and then the rest of the "effects" are adjusted to that base value. So I had to pick up the Arlington intercept *(which is listed first[1] in the coef() function)*, then skip the slope and add the rest of the intercept "effects" to that base *(yes, it's a pain - lm is "user friendly")*

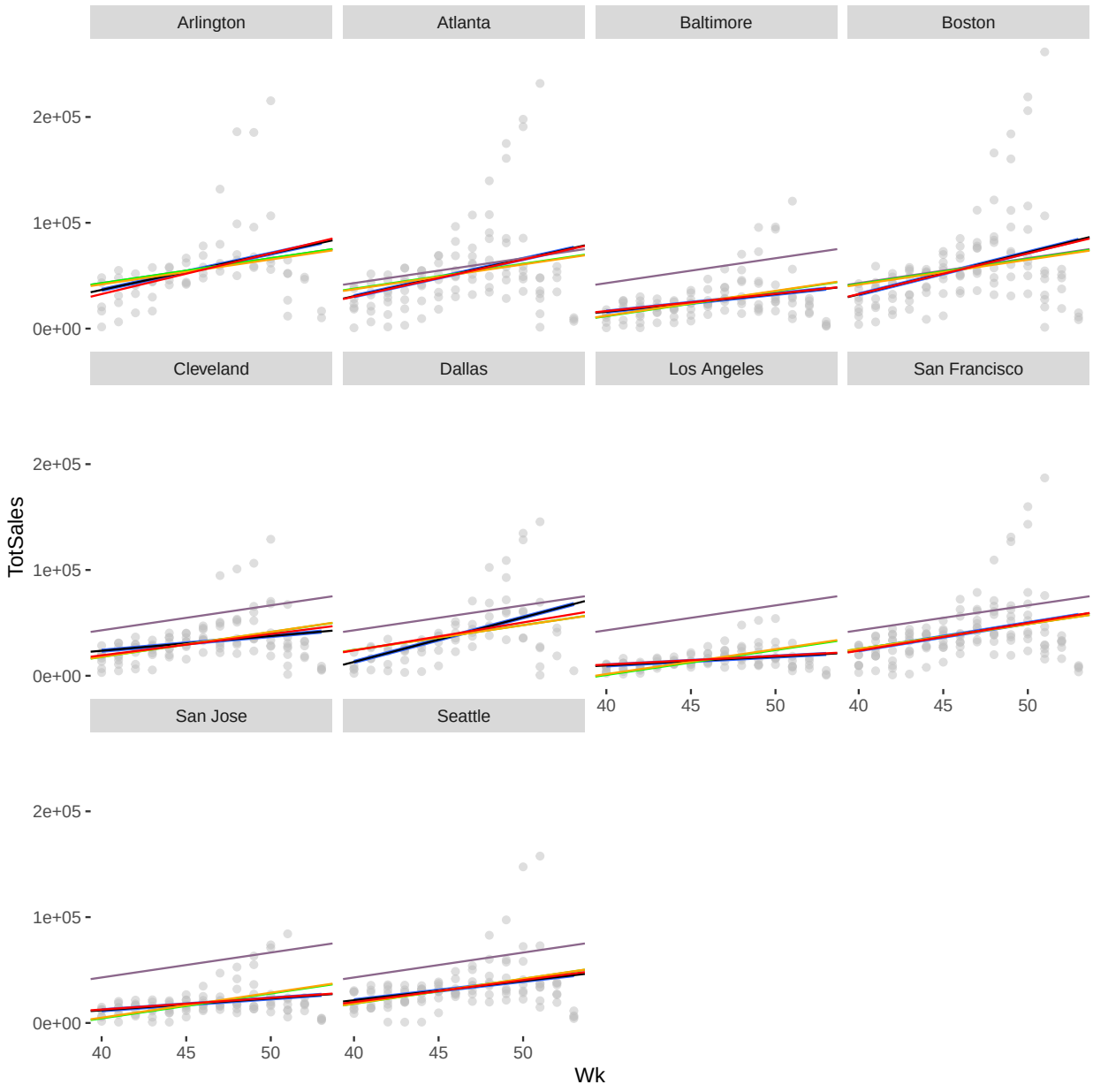
3. lmer is more advanced and workable. The coef() function will give you the intercept *(base + effect)*. Then, I can get the effects - divided into random and fixed effects - using ranef() an fixef() functions. Note: in Bayesian mulitlevel modeling, we see things a little different, so don't worry about this concept of fixed vs random effects.]

Varying Intercept and Slope Models *(Partial Pooling)*

We can also let both Intercept and slope vary *(we need lmer for this - lm can't do this)*. The following lmer creates this type of model:

Note: you might get convergence errors but it should get close enough for our purposes here - lmer struggles with complex data and we're not going to spend time debugging because our application of lmer is limited.

```
LMERVarIntSlpMod = lmer(TotSales ~ Wk + ( Wk | Description),
                     data = SalesTransSummary)

# if you get convergence errors, it can sometimes be resolved by setting the optimizer.

LMERVarIntSlpMod = update(LMERVarIntSlpMod,
                          control = lmerControl(optimizer="Nelder_Mead"))

LMERVarIntSlpModCoef = data.frame(coef(LMERVarIntSlpMod)$Description) %>%
  rownames_to_column("Description") %>%
  rename(Intercept = `X.Intercept.`, Slope = Wk)

p = p + geom_abline(data = LMERVarIntCoef,
                    aes(intercept = LMERVarIntSlpModCoef$Intercept,
                        slope =  LMERVarIntSlpModCoef$Slope),
                    color = "red")
p
```



Multilevel - Varying Inetercept and Slope - Partially Pooled

OK, we're going to start pooling between groups - varying intercept and slope with 2 grouping levels. The syntax is shown below *(refer to manual - but think of this as: TotSales is a function of Wk, which is grouped by Description and MerGroup)*:

```
LMERVarIntSlpMod2L = lmer(TotSales ~ Wk + ( Wk | Description) + (Wk | MerGroup),
                          data = SalesTransSummary)
```

Which creates the model:

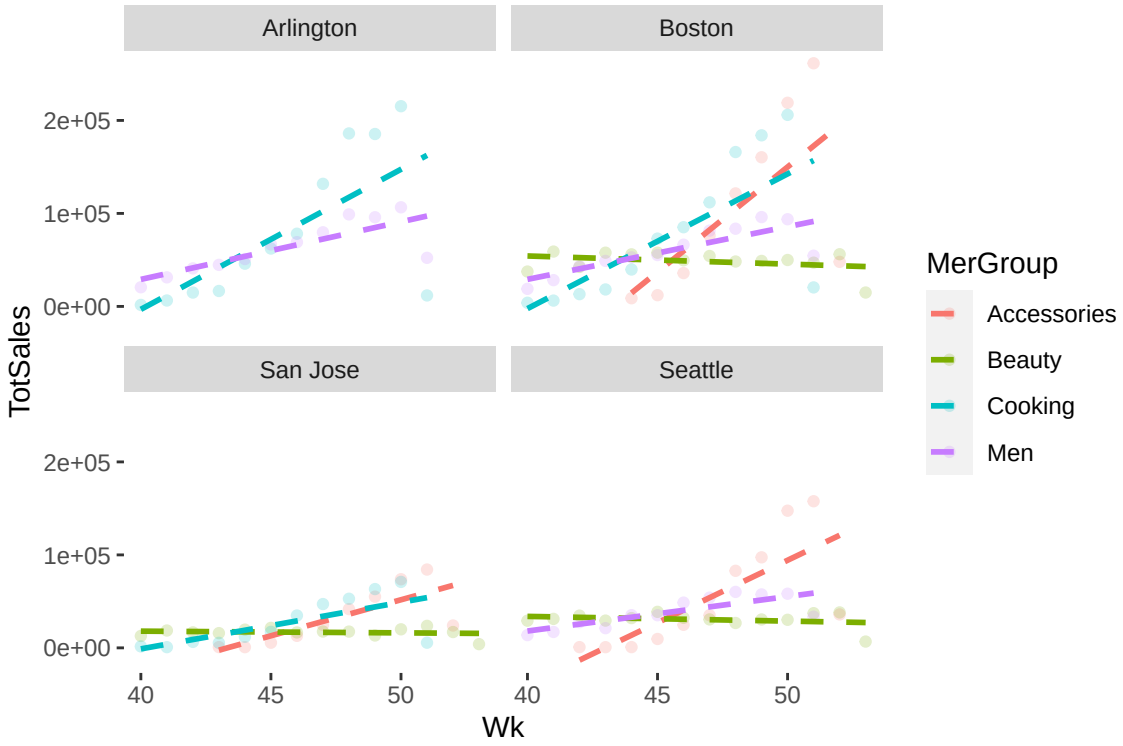
$y = \beta_0 + \alpha_{m,n} + (\beta_1 + \gamma_{m,n})X + \epsilon$

lmer breaks effects into fixed
$beta_0$ and β_x, and random components ($\alpha_{m,n}$ and $\gamma_{m,n}$) by group and we can get those values using ranef() and fixef().

```
# Here, we're getting the effects (intercept and slope) for the Description level

d1 = ranef(LMERVarIntSlpMod2L)$Description %>%
  rownames_to_column("Description") %>%
  select(Description, "I1" = "(Intercept)", "S1" = "Wk")

# Here, we're getting the effects (intercept and slope) for the MerGroup level

d2 = ranef(LMERVarIntSlpMod2L)$MerGroup %>%
  rownames_to_column("MerGroup") %>%
  select(MerGroup, "I2" = "(Intercept)", "S2" = "Wk")

# then, we get the fixed effects

I = fixef(LMERVarIntSlpMod2L)[1]
S = fixef(LMERVarIntSlpMod2L)[2]

# and we add them up to get the parameter values:

Coef = crossing(d1, d2) %>% mutate(Intercept = I + I1 + I2, Slope = S + S1 + S2)
```

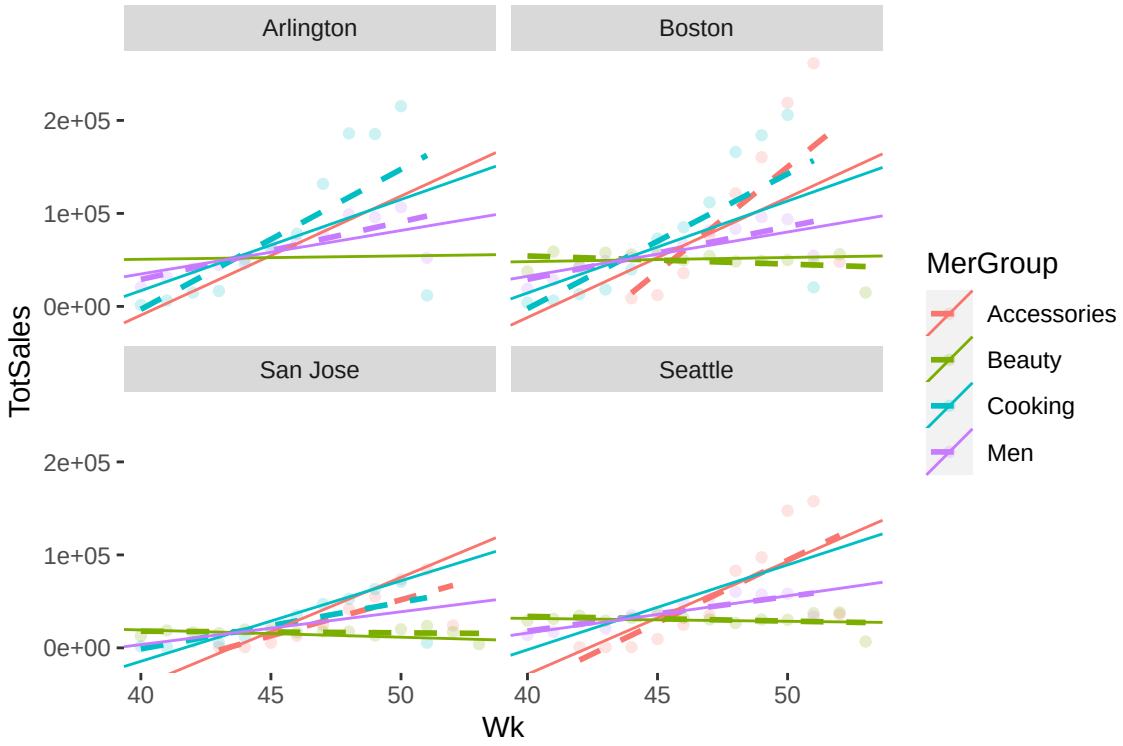
Let's filter it down to 3 locations and 4 Mergroups so it's easier to see, and generate lm models for baseline:

```
SalesSummarySub = filter(SalesTransSummary,
                         Description %in% c("Arlington", "Boston", "San Jose", "Seattle"),
                         MerGroup %in% c("Accessories", "Beauty", "Cooking", "Men" ))

p =  ggplot(SalesSummarySub, aes(Wk, TotSales, color = MerGroup)) +
  geom_point(alpha = .2) +
  geom_smooth(method = "lm", se = F, alpha = .05, linetype = "dashed", alpha = .5) +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```



and add our lmer multilevel models:

```
CoefSub = filter(Coef,
                         Description %in% c("Arlington", "Boston", "San Jose", "Seattle"),
                         MerGroup %in% c("Accessories", "Beauty", "Cooking", "Men" ))


p = p + geom_abline(data = CoefSub,
                    aes(intercept = CoefSub$Intercept, slope =  CoefSub$Slope, color = MerGroup))
p
```



Note:

1. The lmer models are more grouped - they **pool**, sharing correlation. Look at Boston and compare Accessories and Beauty - see how the hared group correlations pull Accessories down and Beauty up toward the "center"? How would this impact analysis in highly dynamic environments? What is a prediction, a forecast, how precise/variable do we really want forecast to be? How does this relate to the trade-offs between model bias and variability, the trade-offs between model interpretability and flexibility? We are introducing practices that allow us to tailor bias, variability, interpretability and flexibility.
2. Notice how lmer models Accessories and Beauty for Arlington, even though lm didn't. lm didn't becuause there's no data - Arlington didn't sell any. But in lmers case, once we have group effects, we can apply them to any combination. Suppose you're an analyst and your client wants a forecast for Accessories in Arlington - where would you begin?
3. These level effects will become an important basis for defining the causal relationships between business drivers and business outcomes **This is the central question of business planning and assurance**.