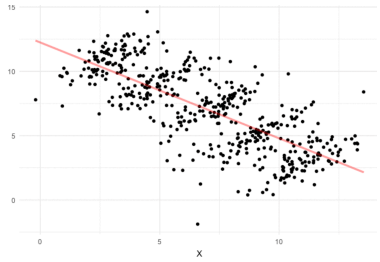
Multilevel Modeling

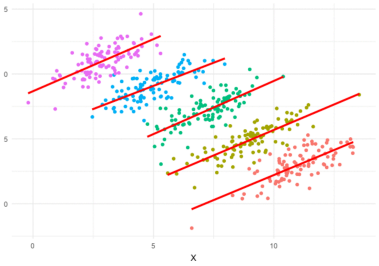
Introduction

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 in often embarrasingly incorrect conclusions - sometimes I think Simpson's paradox is named after Homer:

It's essential for an analyst to understand business drivers and outcomes at all levels. Multilevel models are also known as hierarchical linear models, nested data models, mixed models, random coefficient, random-effects models, random parameter models, or split-plot designs *(the reason for all these terms is that there's a conspiracy to create confusion by the over-engineering society)*. Simply put, Multilevel models are just models of parameters that vary in more than one data grouping.

Multilevel Modeling Exercises

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

$y = \alpha + \beta X + \epsilon$

and when you used varying intercepts models, the equation was altered to:

$y = \alpha + \gamma_m + \beta X + \epsilon$

where γ_m was the variance of the group level intercept from the base level intercept. These models varying intercept models minimize error within a fixed slope constraint. Of course, a fixed slope constraint does not always produce the model we want, and we can expand the model fit to include intercepts and slopes. The equation then expands to:

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

The γ is often called the "random" effect, but we'll just use the term "effect", which, in frequentist methodology, is generally determined by:

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

σ_m^2 is the variance within groups, and σ_b^2 is the variance between groups. This is called *Partial Pooling*, which weights intRA and intER group variances. In a way, it *regularizes* the coefficients within a groups balancing variances between groups.

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

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.

> [Note: Another dimension to modeling that we'll explore later is whether the data groups are nested *(or hierarchical - group membership and primary keys are distinct)*, or crossed *(each data group affects all transactions)*, but we'll save this for later.]

Load the following libraries

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

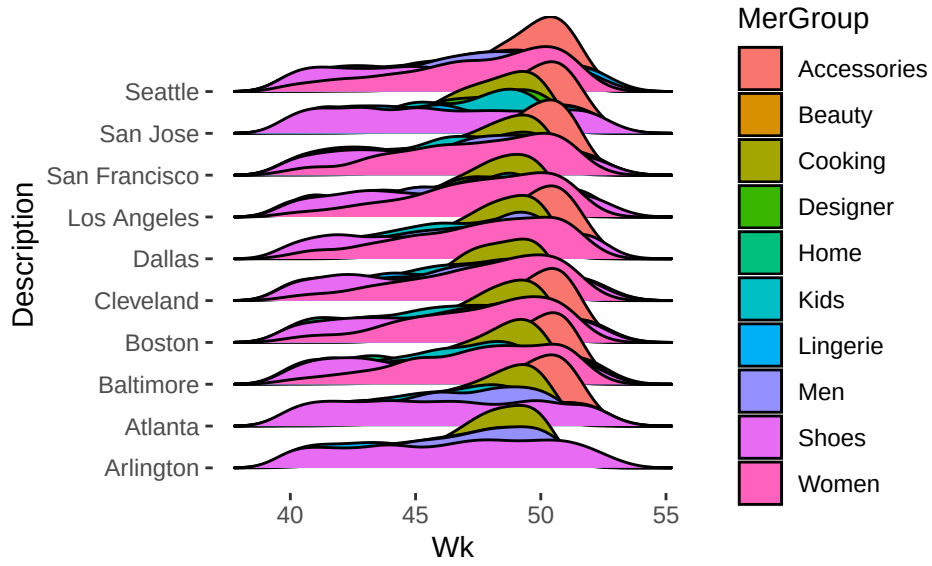
Load and transform data as follows *(this is sales data is for a chain of retail outlets with 10 merchandise groups across 10 locations)*:

```
SalesTrans = read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Sa
Location = read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Loca
MerGroup = read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/MerG
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)
```

Visualizing distributions across weeks:

```
ggplot(data.frame(SalesTrans),
       aes(y = Description, x = Wk, fill = MerGroup)) +
  geom_density_ridges()  +
  theme(panel.background = element_rect(fill = "white"))
```



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:

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

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

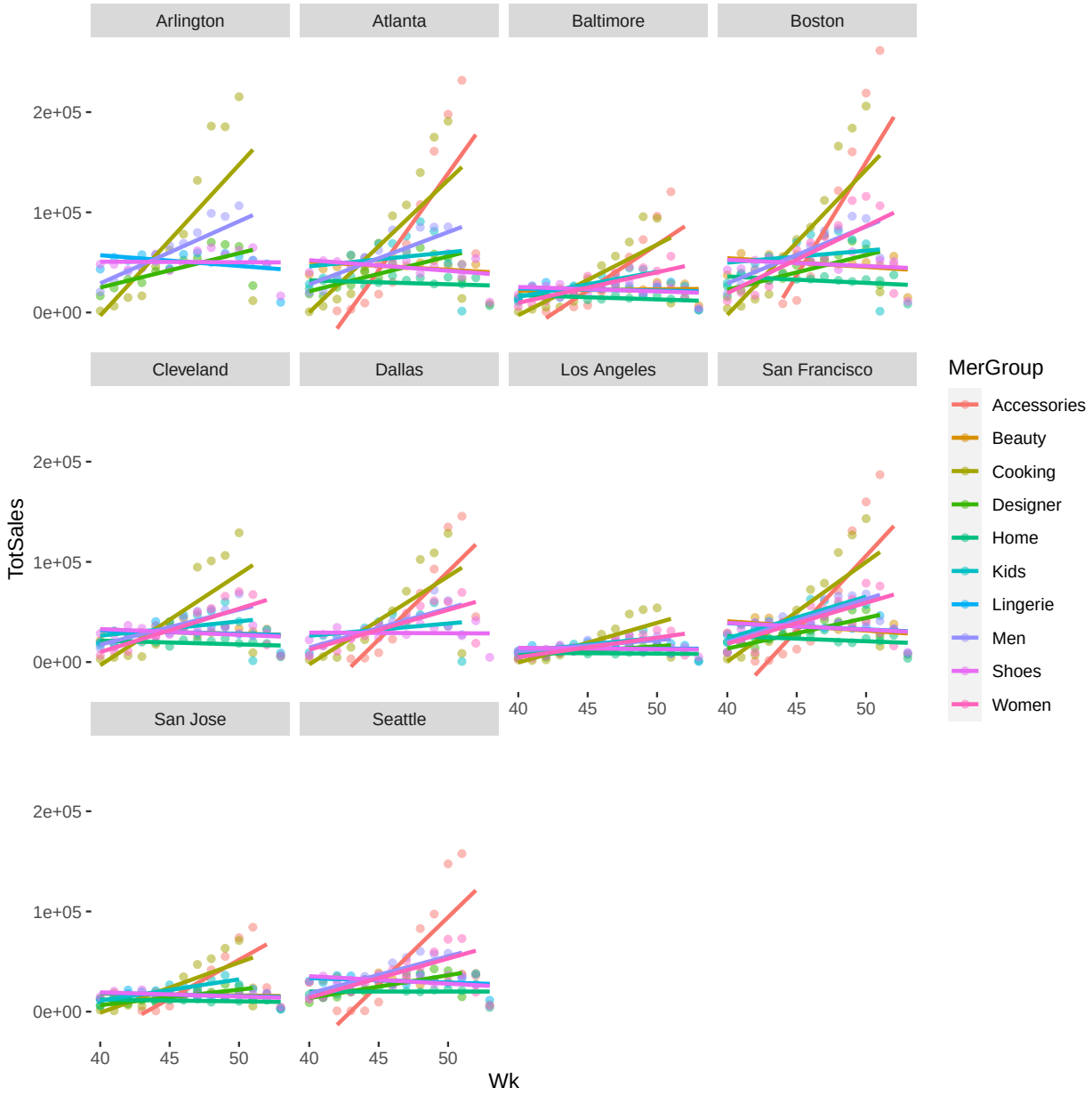
Description	n
Arlington	5
Atlanta	8
Baltimore	8
Boston	9
Cleveland	8
Dallas	6
Los Angeles	8
San Francisco	10
San Jose	8
Seattle	8

We'll use the lmer function *(lme4 package)* to build models here, but we'll migrate this approach to Bayesian modeling next, where we can fine tune how pooling is applied to groups.

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
```

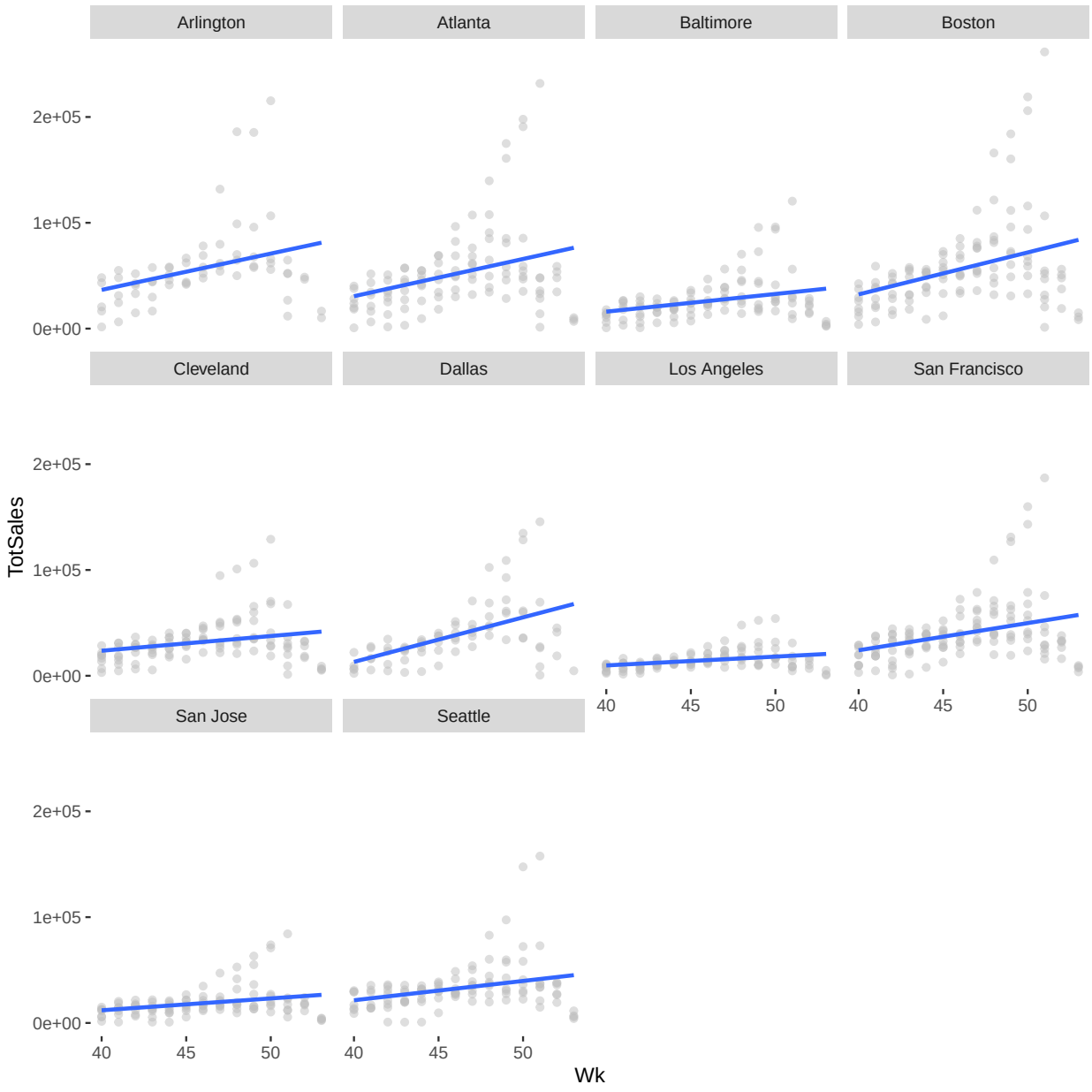


These models are truly independent *(not pooled)* - ggplot creates a separate model for each Description and MerGroup. So, in this case, 78 sepearte models! Keep in mind, business data is very dynamic - What did we learn from regularization? There's none here.

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:

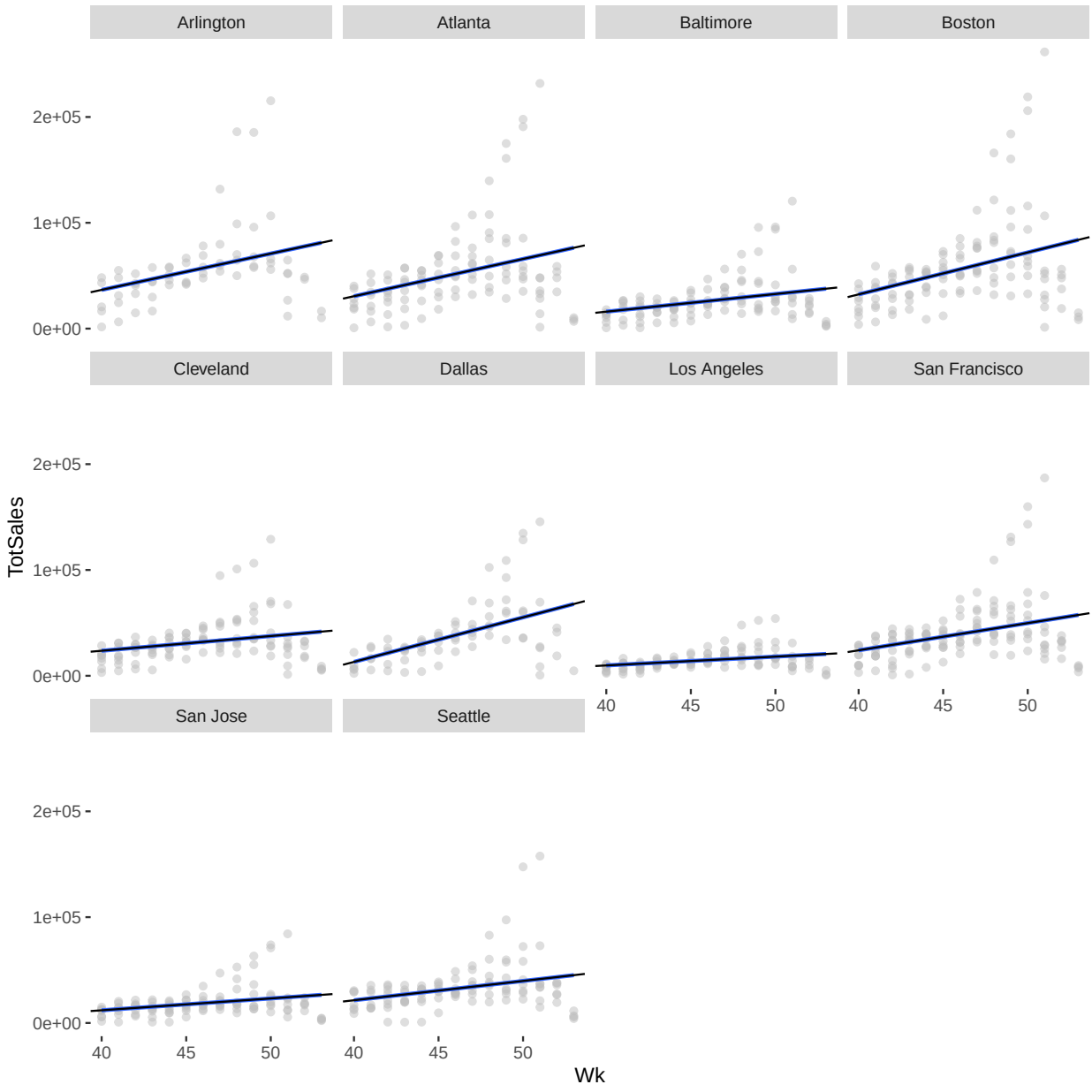
```
p =  ggplot(SalesTransSummary, aes(Wk, TotSales)) +
  geom_point(color = "gray", alpha = .5) +
  geom_smooth(method = "lm", se = F) +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```



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, the 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:

```
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
```

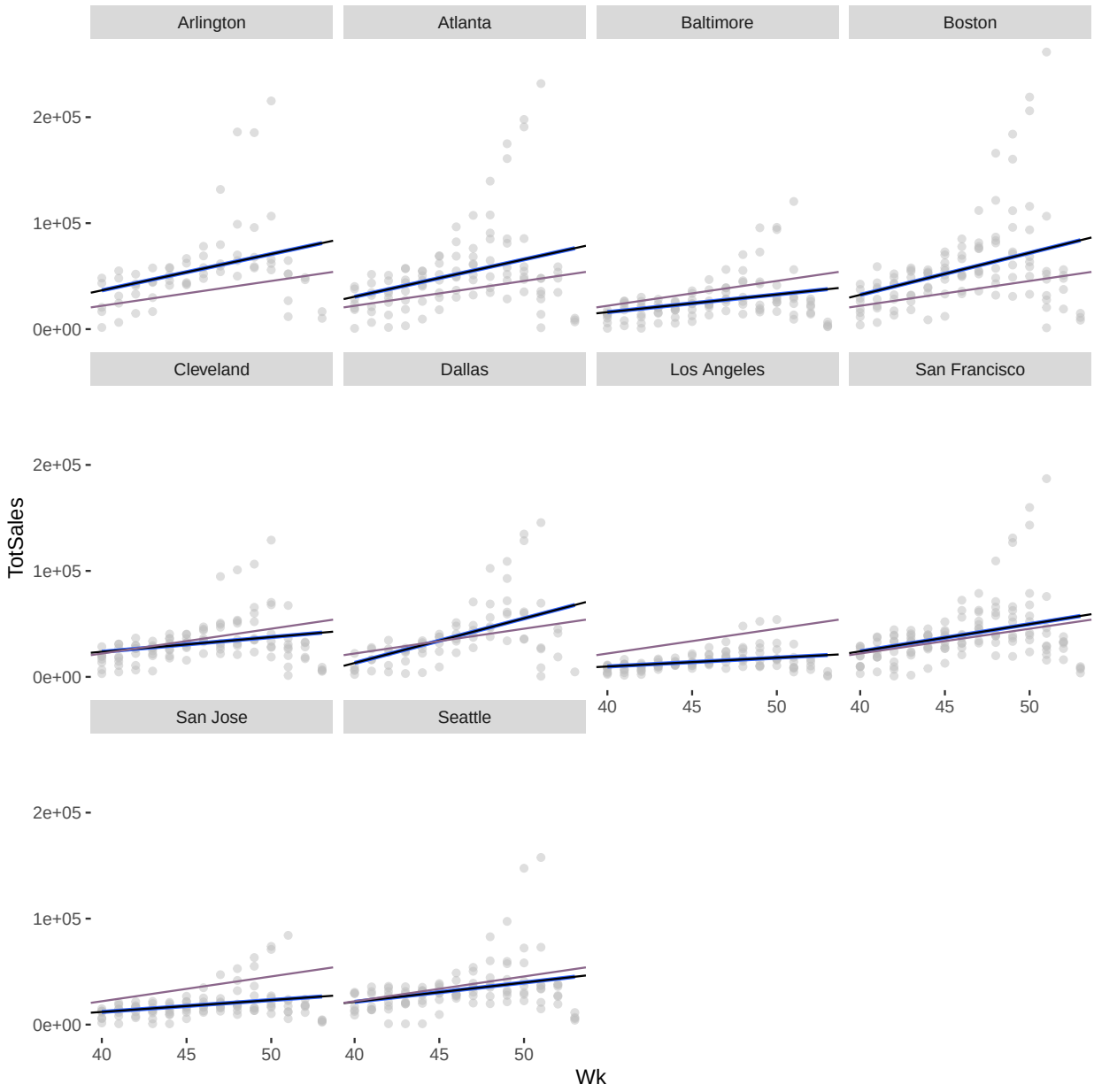


You don't see 2 sets of lines because lmer gets the same coefficients as lm here. *(lmer grouping syntax at the end).*

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
```



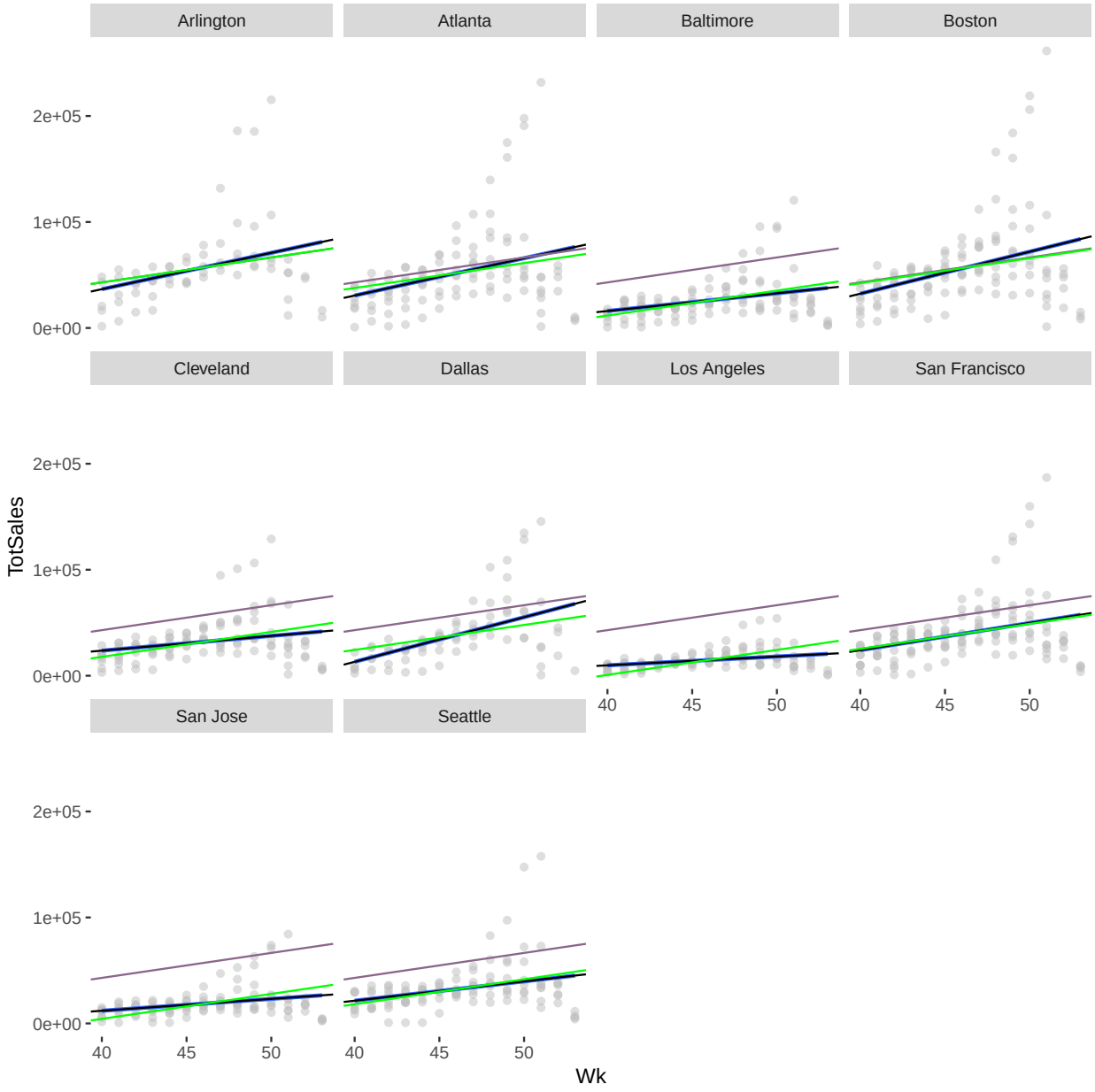
The plot above compares the fully pooled model with the independent, single group level model. Note how the fully pooled model doesn't change across groups.

Varying Intercept Models *(Partial Pooling)*

These are the models you're most familar with - as we spent a lot of time on categorical regression last semester *(recall the Auto price data, where each make of car was assigned a different intercept value while they all shared the same slope)*. So we won't spend a lot of time on this model other than to compare:

```
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
```



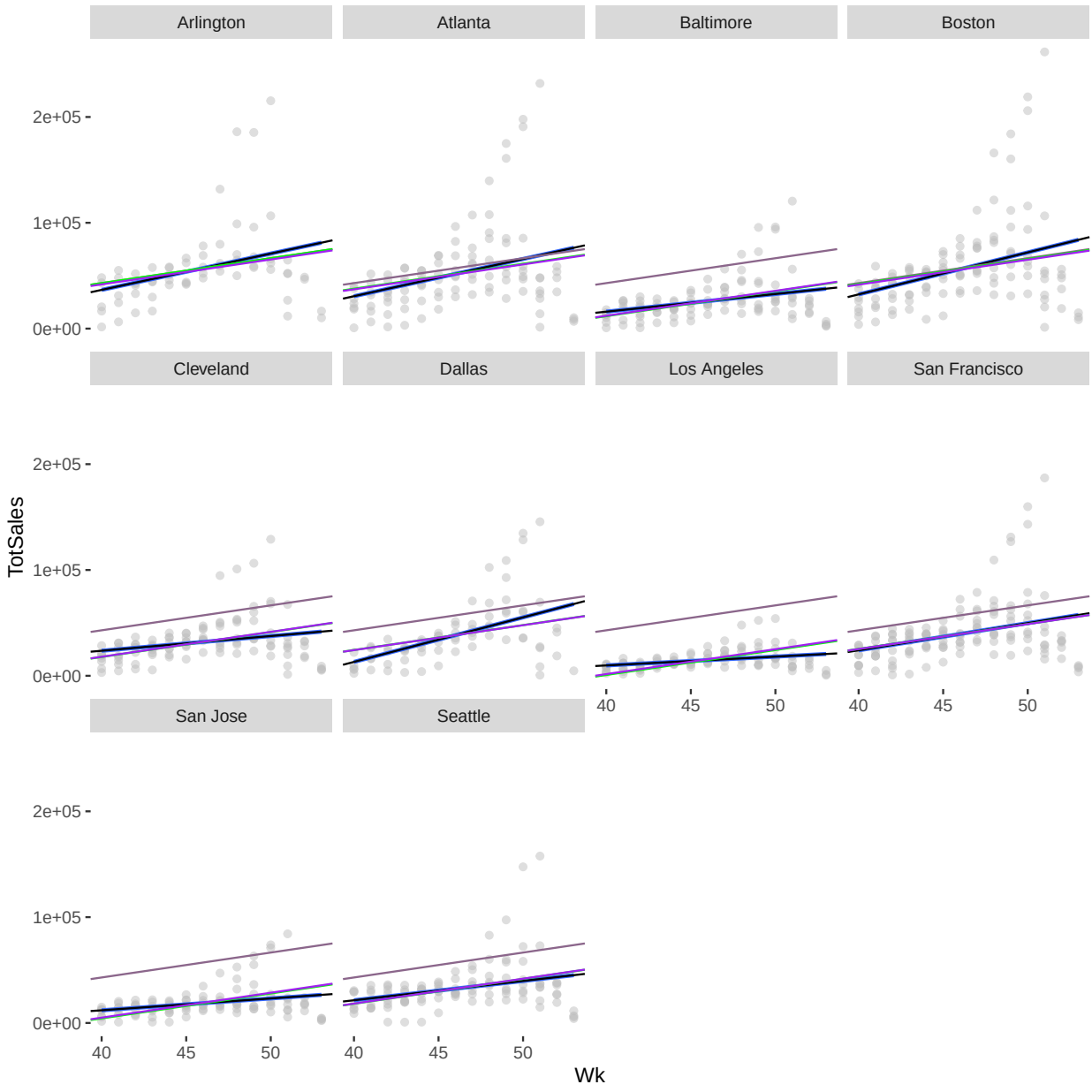
Also, we can use glm to fit linear models instead of lm - using maximum likelihood:

```
GLMVarIntMod = glm(TotSales ~ Wk + Description, SalesTransSummary, family = gaussian() )
GLMVarIntCoef = coef(GLMVarIntMod)
GLMVarIntCoef = data.frame(Description = unique(SalesTransSummary$Description),
```

```
                          fpLMI = c( coef(LMVarIntMod)[1], coef(LMVarIntMod)[1] + c(coef(LMVarIntMod)[3
                          fpLMS = c( rep(coef(LMVarIntMod)[2], 10)))
```

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 = "purple")
p
```



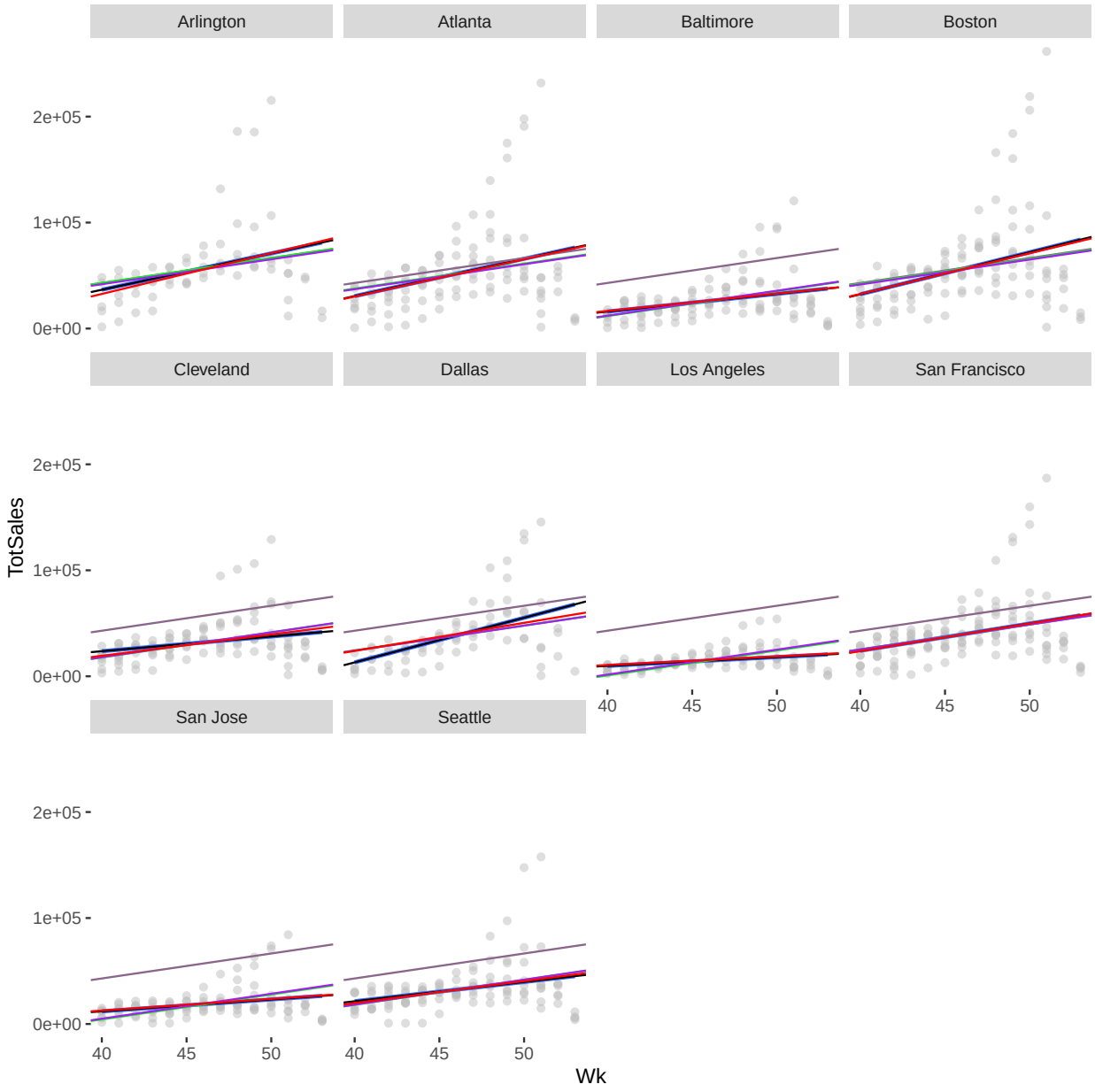
Note that the green lines from the lm varying intercept model have been overridden - the coefficients are the same. These models are *partially pooled*, they share correlation with other groups.

Varying Intercept and Slope Models *(Partial Pooling)*

We can also let both Intercept and slope vary *(we need lmer for this)*. The following lmer creates this type of model *(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, but first let's create an independent, no-pooling, model for comparison:

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

Now, lets take a look at a varying intercept and slope with 2 grouping levels. The syntax is shown below *(refer to manual)*:

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

Which creates the model:

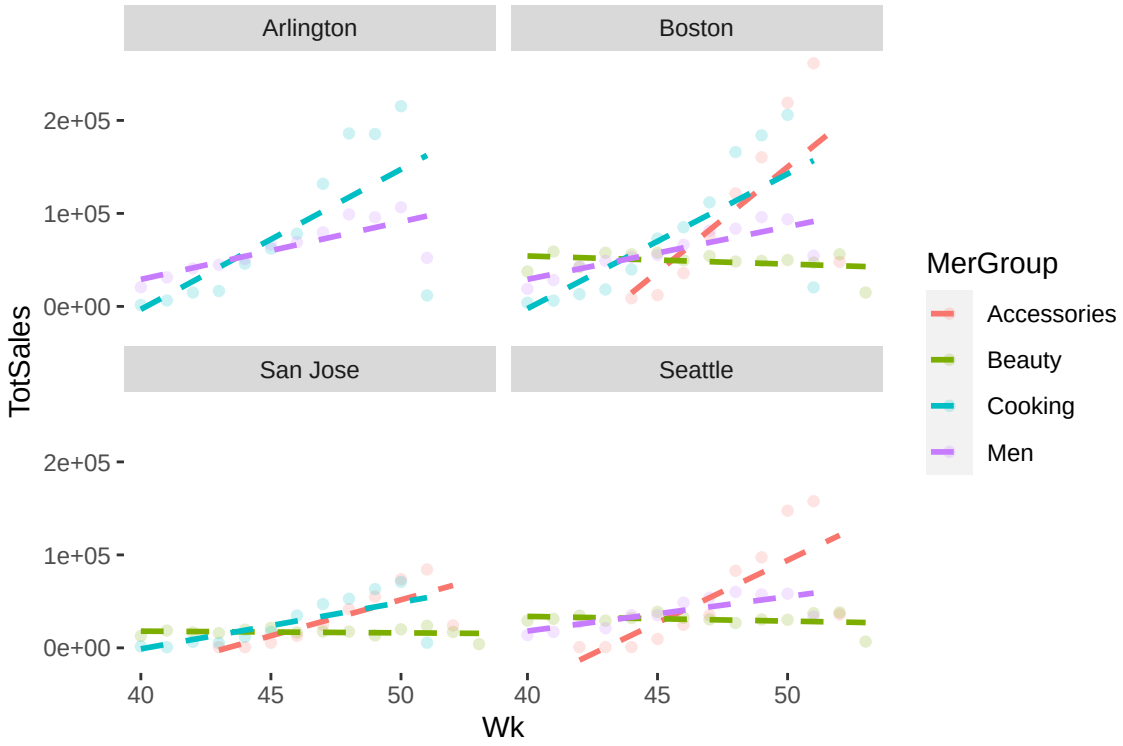
$y = \alpha_{m,n} + \gamma_{m,n} + (\beta_{m,n} + \gamma_{m,n})X + \epsilon$

and we can get the fixed $\alpha_{m,n}$ and $\beta_{m,n}$, and the random $\gamma_{m,n}$ effects from lmer models using the following functions:

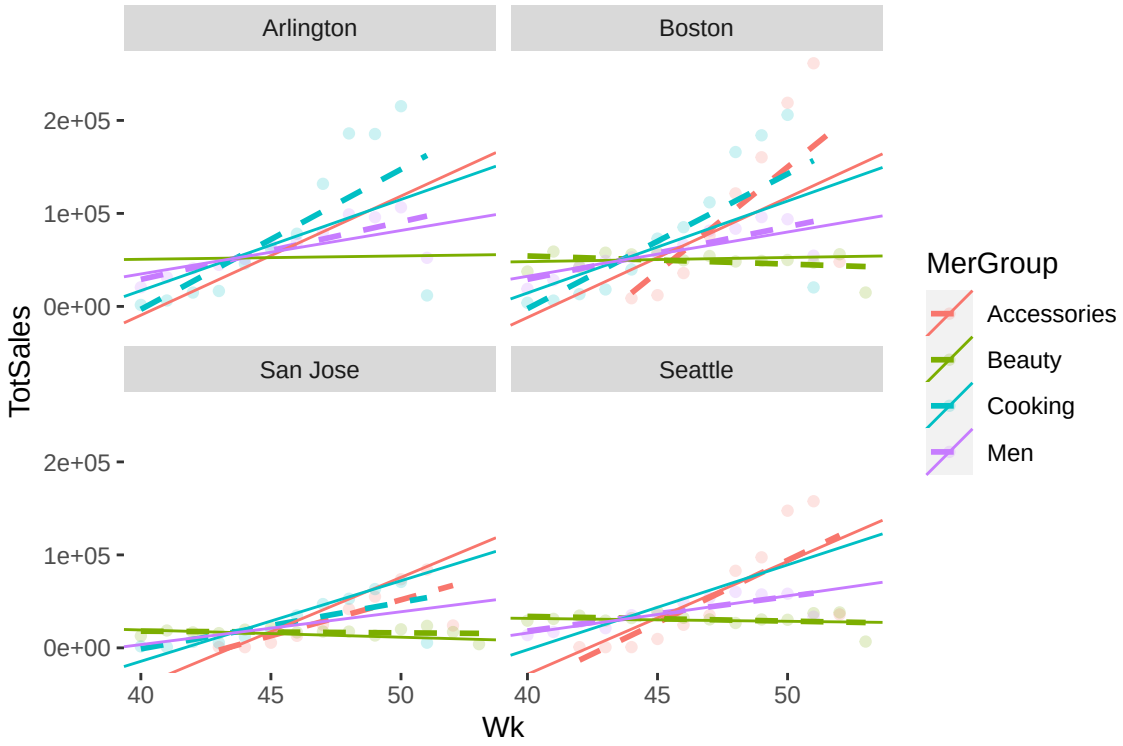
```
d1 = ranef(LMERVarIntSlpMod2L)$Description %>% rownames_to_column("Description") %>% select(Description
d2 = ranef(LMERVarIntSlpMod2L)$MerGroup %>% rownames_to_column("MerGroup") %>% select(MerGroup, "I2" =

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

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

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

Let's filter it down to 3 locations and 4 Mergroups so we can compare the models.

```
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") +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```




```
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 **pooled**, sharing correlation. One could say they're more generalized, but we're using pooling.
2. Notice how Accessories and Beauty are modeled for Arlington, even though they didn't sell any. That's because the coefficients are group effects which can be applied for any combination. So, if Arlington wanted to add Accesories or Beauty, we would already have a good start on predicting that effect.
3. These level effects become the quantification of the causal relationships between business drivers and business performance. **This is the central question of business planning and assurance**.