**1. Understanding and visualizing trends in customer data**

Awesome work, examining plots in this way can be a tremendous way to understand trends in user behavior!

**2. Further techniques for uncovering trends**

Often, plotting a graph is not enough and additional preprocessing is required to uncover the trend. Here we will consider two of these processing techniques and explore how they are useful.

**3. Subscribers Per Day**

To start, let’s look at the USA subscribers-per-day number for a new product without a trial period. Here we have a dataset of subscribers broken out by registration and subscription date. For a given day we will look at the number of subscribers who registered within the prior week. To do this we find the difference between registration and subscription dates. Then filter on this value, and finally group and aggregate by subscription date.

**4. Weekly seasonality and our pricing change**

There seems to be weekly seasonality. As evidenced by the peaks-and-valleys roughly every 7 days. Perhaps users are more likely to buy when they are using the app more heavily on the weekends than during the middle of the week Many metrics will have seasonality and it can easily obfuscate macro-level-trends in the data. Specifically, we have reason to be concerned that a recent pricing change is causing a dip in subscription volume. It might be hard to tell if the data has a high degree of seasonal movement as we see here.

**5. Correcting for seasonality with trailing averages**

We can correct for this movement, by calculating a trailing average over our data. A trailing average is a smoothing technique that sets the value for a given day as the average over the past n-days. To smooth weekly seasonality we want n-equals-seven. This has the effect of averaging over a week such that every day is pulled towards the weekly level limiting the day level effects.

**6. Calculating Trailing Averages**

First, we use the pandas `rolling()` method to find the rolling window. The primary parameters of `rolling()` are `window` which is our n value from above, and `center` which is a Boolean. If this is true then our averaged value will be placed at the middle of our window, if not it will be placed in the index that we are looking back from, which is what we want.

**7. Smoothing our USA subscription data**

Once we have this we call `mean` to find our average over the window. We can calculate this and store it in an additional variable As we can see, this smoothing does a lot to flatten out our line and reveal the dip, unobscured.

**8. Noisy data - Highest SKU purchases by date**

Beyond seasonality, data can simply be noisy. Let’s take a look at a graph of how many of our largest sku in-app-items are purchased per day. It is incredibly noisy, as the values vary widely from day-to-day. We can apply an exponential moving average so we can check if any macro trends are hidden among this noise.

**9. Smoothing with an exponential moving average**

This type of average weights the points such that the earlier ones are weighted less than the more recent ones within our window. This pulls our data back to any central trend, while maintaining any recent movements.

**10. Smoothed purchases by date**

We can use the `ewm()` method to find these weighted windows. To do this we specify the `span` argument to be our window size. Then we find the average of this weighted data. Here we apply this to our set of purchase data, using a 30-day window. Determining windows like this can require prior knowledge of the structure of the data or some trial-and-error. See that this does remove a lot of the noise and reveals a slight upward trend.

**11. Summary - Data Smoothing Techniques**

These techniques are very useful for uncovering trends. Next we will put them to use diving deeper into user data.

**12. Let's practice!**

Good luck, now let’s practice!

## 1. [1.] Understanding and visualizing trends in customer data [Müşteri verilerindeki eğilimleri anlama ve görselleştirme]

Awesome work, examining plots in this way can be a tremendous way to understand trends in user behavior! [Grafikleri bu şekilde incelemek, kullanıcı davranışındaki eğilimleri anlamak için harika bir yol olabilir!]

## 2. [2.] Further techniques for uncovering trends [Trendleri ortaya çıkarmak için diğer teknikler]

Often, plotting a graph is not enough and additional preprocessing is required to uncover the trend. [Çoğu zaman, bir grafiği çizmek yeterli değildir ve trendi ortaya çıkarmak için ek ön işleme gereklidir.] Here we will consider two of these processing techniques and explore how they are useful. [Burada bu işleme tekniklerinden ikisini ele alacağız ve bunların nasıl yararlı olduğunu keşfedeceğiz.]

## 3. [3.] Subscribers Per Day [Günlük Abone Sayısı]

To start, let’s look at the USA subscribers-per-day number for a new product without a trial period. [Başlamak için, deneme süresi olmayan yeni bir ürün için ABD'deki günlük abone sayısına bakalım.] Here we have a dataset of subscribers broken out by registration and subscription date. [Burada kayıt ve abonelik tarihine göre ayrılmış bir abone veri setimiz var.] For a given day we will look at the number of subscribers who registered within the prior week. [Belirli bir gün için önceki hafta içinde kaydolan abone sayısına bakacağız.] To do this we find the difference between registration and subscription dates. [Bunu yapmak için kayıt ve abonelik tarihleri ​​arasındaki farkı buluyoruz.] Then filter on this value, and finally group and aggregate by subscription date. [Ardından bu değeri filtreleyin ve son olarak abonelik tarihine göre gruplandırın ve toplayın.]

## 4. [4.] Weekly seasonality and our pricing change [Haftalık mevsimsellik ve fiyat değişikliğimiz]

There seems to be weekly seasonality. [Haftalık mevsimsellik var gibi görünüyor.] As evidenced by the peaks-and-valleys roughly every 7 days. [Tepeler ve vadiler tarafından kabaca her 7 günde bir kanıtlandığı gibi.] Perhaps users are more likely to buy when they are using the app more heavily on the weekends than during the middle of the week Many metrics will have seasonality and it can easily obfuscate macro-level-trends in the data. [Belki de kullanıcıların uygulamayı hafta ortasına kıyasla hafta sonları daha yoğun kullandıklarında satın alma olasılıkları daha yüksektir.] Specifically, we have reason to be concerned that a recent pricing change is causing a dip in subscription volume. [Spesifik olarak, yakın tarihli bir fiyat değişikliğinin abonelik hacminde bir düşüşe neden olduğundan endişe etmek için nedenlerimiz var.] It might be hard to tell if the data has a high degree of seasonal movement as we see here. [Burada gördüğümüz gibi verilerin yüksek derecede mevsimsel hareketlere sahip olup olmadığını söylemek zor olabilir.]

## 5. [5.] Correcting for seasonality with trailing averages [Takip eden ortalamalarla mevsimsellik için düzeltme]

We can correct for this movement, by calculating a trailing average over our data. [Verilerimiz üzerinden takip eden bir ortalama hesaplayarak bu hareketi düzeltebiliriz.] A trailing average is a smoothing technique that sets the value for a given day as the average over the past n-days. [İzleyen ortalama, belirli bir gün için değeri son n gün içindeki ortalama olarak ayarlayan bir yumuşatma tekniğidir.] To smooth weekly seasonality we want n-equals-seven. [Haftalık mevsimselliği yumuşatmak için n-eşittir-yedi istiyoruz.] This has the effect of averaging over a week such that every day is pulled towards the weekly level limiting the day level effects. [Bu, bir hafta boyunca ortalama alma etkisine sahiptir, öyle ki her gün, günlük seviye etkilerini sınırlayan haftalık seviyeye çekilir.]

## 6. [6.] Calculating Trailing Averages [Sondaki Ortalamaları Hesaplama]

First, we use the pandas `rolling()` method to find the rolling window. [İlk olarak, dönen pencereyi bulmak için pandas `rolling()` yöntemini kullanıyoruz.] The primary parameters of `rolling()` are `window` which is our n value from above, and `center` which is a Boolean. [Rolling()'in birincil parametreleri, yukarıdan n değerimiz olan 'window' ve bir Boole olan 'center'dır.] If this is true then our averaged value will be placed at the middle of our window, if not it will be placed in the index that we are looking back from, which is what we want. [Eğer bu doğruysa, ortalama değerimiz penceremizin ortasına yerleştirilecek, değilse geriye baktığımız dizine yerleştirilecek, ki bu bizim istediğimiz şey.]

## 7. [7.] Smoothing our USA subscription data [ABD abonelik verilerimizi yumuşatmak]

Once we have this we call `mean` to find our average over the window. [Bunu elde ettiğimizde, pencere üzerinden ortalamamızı bulmak için 'ortalama' diyoruz.] We can calculate this and store it in an additional variable As we can see, this smoothing does a lot to flatten out our line and reveal the dip, unobscured. [Bunu hesaplayabilir ve ek bir değişkende saklayabiliriz Gördüğümüz gibi, bu yumuşatma çizgimizi düzleştirmek ve eğimi belirsiz bir şekilde ortaya çıkarmak için çok şey yapıyor.]

## 8. [8.] Noisy data - Highest SKU purchases by date [Gürültülü veriler - Tarihe göre en yüksek SKU satın alma işlemleri]

Beyond seasonality, data can simply be noisy. [Mevsimselliğin ötesinde, veriler sadece gürültülü olabilir.] Let’s take a look at a graph of how many of our largest sku in-app-items are purchased per day. [Günde en büyük uygulama içi SKU öğelerimizden kaçının satın alındığına dair bir grafiğe bakalım.] It is incredibly noisy, as the values vary widely from day-to-day. [Değerler günden güne büyük ölçüde değiştiği için inanılmaz derecede gürültülüdür.] We can apply an exponential moving average so we can check if any macro trends are hidden among this noise. [Üstel bir hareketli ortalama uygulayabiliriz, böylece bu gürültü arasında herhangi bir makro eğilimin gizlenip gizlenmediğini kontrol edebiliriz.]

## 9. [9.] Smoothing with an exponential moving average [Üstel hareketli ortalama ile yumuşatma]

This type of average weights the points such that the earlier ones are weighted less than the more recent ones within our window. [Bu ortalama türü, puanları, penceremizdeki öncekiler daha yeni olanlardan daha az ağırlıklı olacak şekilde ağırlıklandırır.] This pulls our data back to any central trend, while maintaining any recent movements. [Bu, son hareketleri korurken verilerimizi herhangi bir merkezi eğilime geri çeker.]

## 10. [10.] Smoothed purchases by date [Tarihe göre yumuşatılmış satın almalar]

We can use the `ewm()` method to find these weighted windows. [Bu ağırlıklı pencereleri bulmak için 'ewm()' yöntemini kullanabiliriz.] To do this we specify the `span` argument to be our window size. [Bunu yapmak için, pencere boyutumuz olarak `span` argümanını belirledik.] Then we find the average of this weighted data. [Daha sonra bu ağırlıklı verilerin ortalamasını buluyoruz.] Here we apply this to our set of purchase data, using a 30-day window. [Burada bunu 30 günlük bir pencere kullanarak satın alma veri setimize uyguluyoruz.] Determining windows like this can require prior knowledge of the structure of the data or some trial-and-error. [Bunun gibi pencereleri belirlemek, verilerin yapısı hakkında önceden bilgi sahibi olmayı veya bazı deneme yanılmaları gerektirebilir.] See that this does remove a lot of the noise and reveals a slight upward trend. [Bunun çok fazla gürültüyü ortadan kaldırdığını ve hafif bir yükseliş eğilimi gösterdiğini görün.]

## 11. [11.] Summary - Data Smoothing Techniques [Özet - Veri Düzeltme Teknikleri]

These techniques are very useful for uncovering trends. [Bu teknikler trendleri ortaya çıkarmak için çok kullanışlıdır.] Next we will put them to use diving deeper into user data. [Daha sonra, onları kullanıcı verilerine daha derinlemesine dalmayı kullanmaya koyacağız.]

## 12. [12.] Let's practice! [Hadi pratik yapalım!]

Good luck, now let’s practice! [İyi şanslar, şimdi pratik yapalım!]