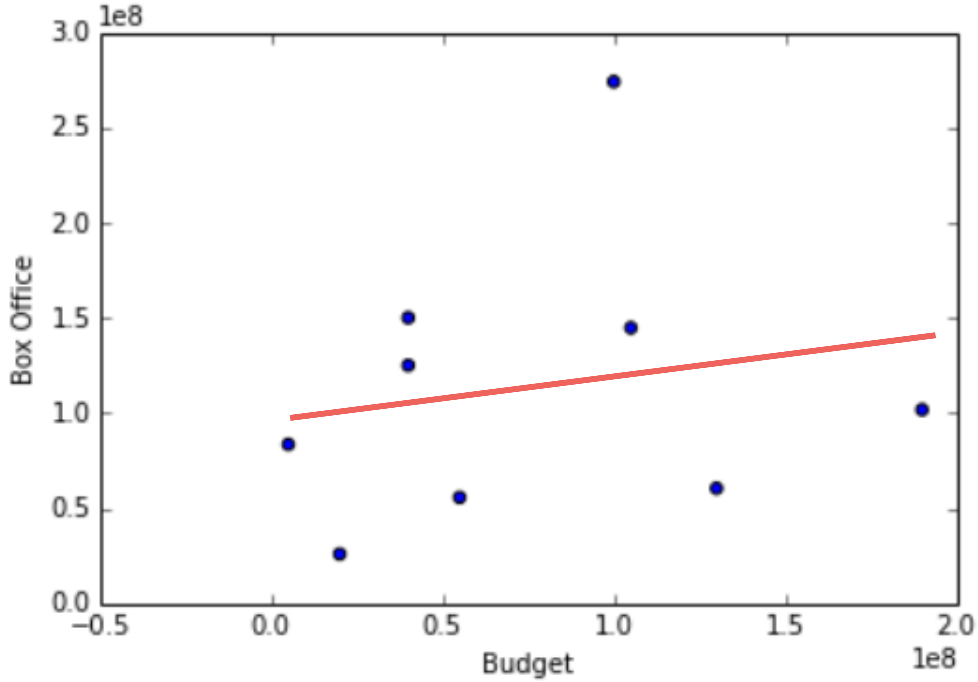
Time Series

DATA SCIENCE BOOTCAMP

Previously, on Linear Regression…



$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

$$\beta_0 = 94.68 million$$
$$\beta_1 = 0.1$$

We expect box office returns for a movie to be

$95 Million

+ 10% of its budget

+ luck factor

(what we can't predict)

(roll the Gaussian dice for about +- $100 Million)

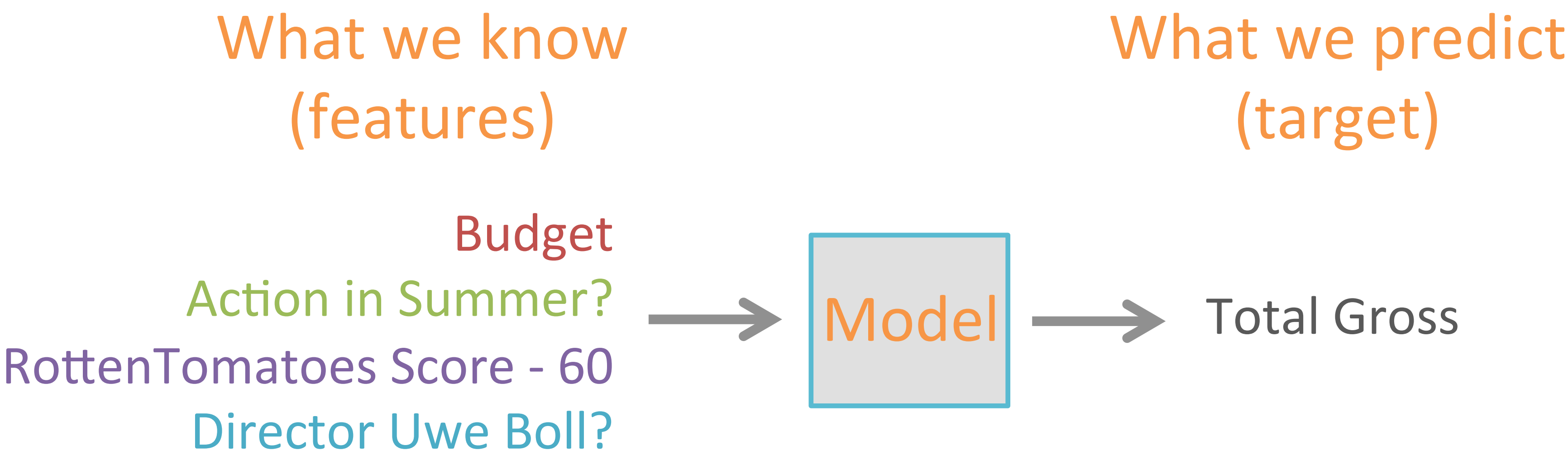
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

$$\beta_0 = 94.68 million$$
$$\beta_1 = 0.1$$

or something like...

We expect box office returns for a movie to be
$20 Million
+ 25% of its budget
+$60 Million if it is an action movie in summer
+$8Million per 10% rottentomatoes score over 60%
-$80 Million if directed by Uwe Boll
+ luck factor
(roll the dice for +- $70 Million)

$$y_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + + \beta_5 x_5 + \varepsilon$$


What we know
(features)
What we predict
(target)
Budget
Action in Summer?
RottenTomatoes Score - 60
Director Uwe Boll?
Model
Total Gross

What we know
(features)

Budget
Action in Summer?
RottenTomatoes Score - 60
Director Uwe Boll?

→ Model → Total Gross

What we predict
(target)

+ luck factor
means this is a
stochastic process

What if we wanted to predict weekly returns?

Mad Max: Fury Road

Domestic Total Gross: **$153,636,354**	
Domestic Lifetime Gross: $154,058,340	
Distributor: **Warner Bros.**	Release Date: **May 15, 2015**
Genre: **Sci-Fi Action**	Runtime: **2 hrs. 0 min.**
MPAA Rating: **R**	Production Budget: **$150 million**

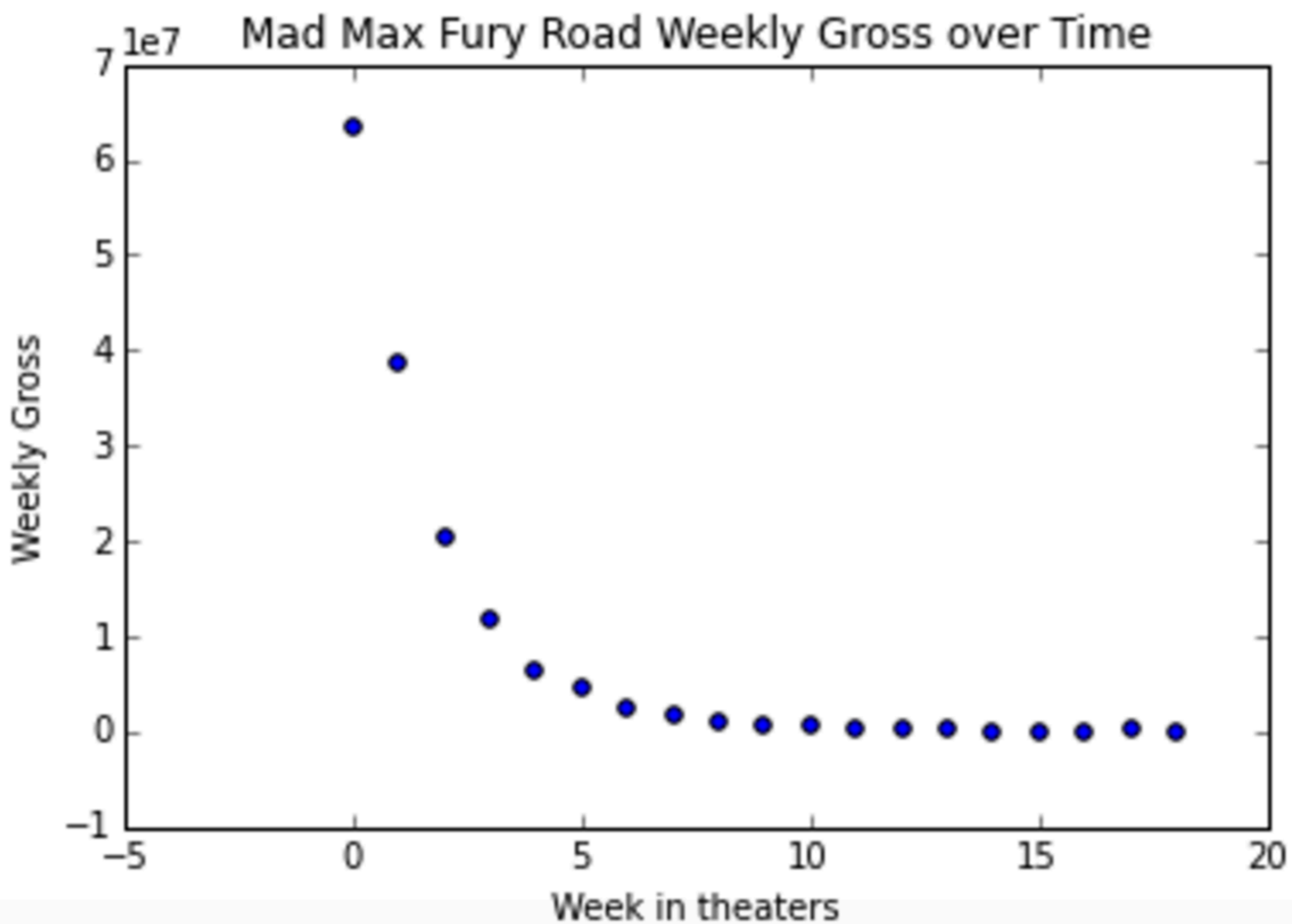
Get local showtimes at IMDb

Summary | Daily | Weekend | **Weekly** | Releases | Foreign | Similar Movies

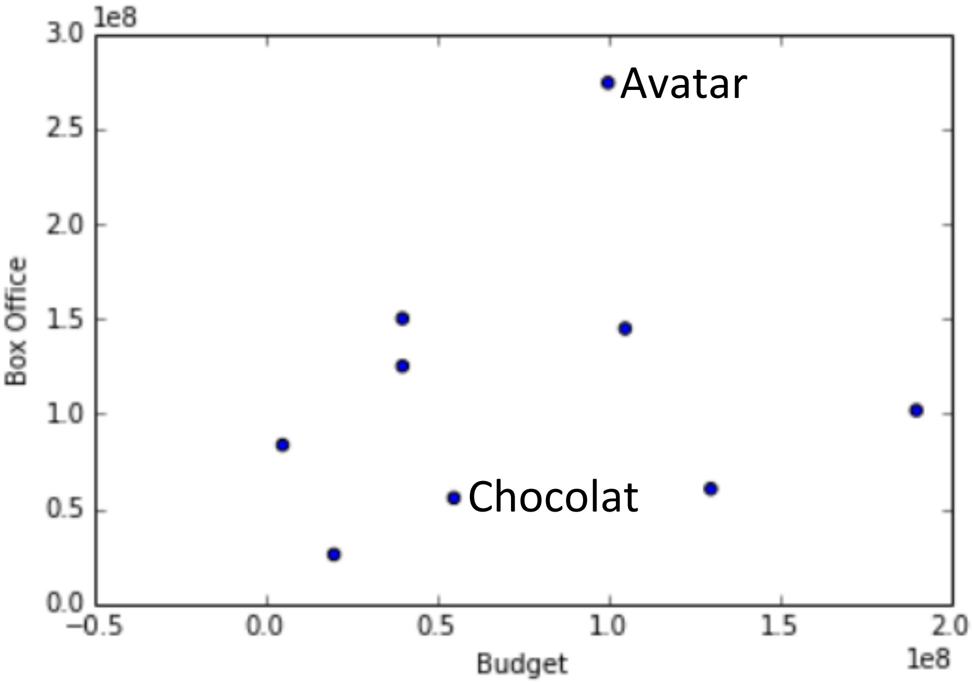
2015

Date (click to view chart)	Rank	Weekly Gross	% Change	Theaters	/ Change	Avg.	Gross-to-Date	Week #
May 15–21	2	$63,440,279	-	3,702	-	$17,137	$63,440,279	1
May 22–28	3	$38,849,255	-38.8%	3,722	+20	$10,438	$102,299,534	2
May 29–Jun 4	3	$20,544,731	-47.1%	3,255	-467	$6,312	$122,834,265	3
Jun 5–11	5	$11,643,562	-43.3%	2,720	-535	$4,281	$134,477,827	4
Jun 12–18	6	$6,309,002	-45.8%	2,234	-486	$2,824	$140,786,829	5
Jun 19–25	8	$4,555,993	-27.8%	1,424	-810	$3,199	$145,342,822	6
Jun 26–Jul 2	11	$2,648,047	-41.9%	561	-863	$4,720	$147,990,879	7

What if we wanted to predict weekly returns?



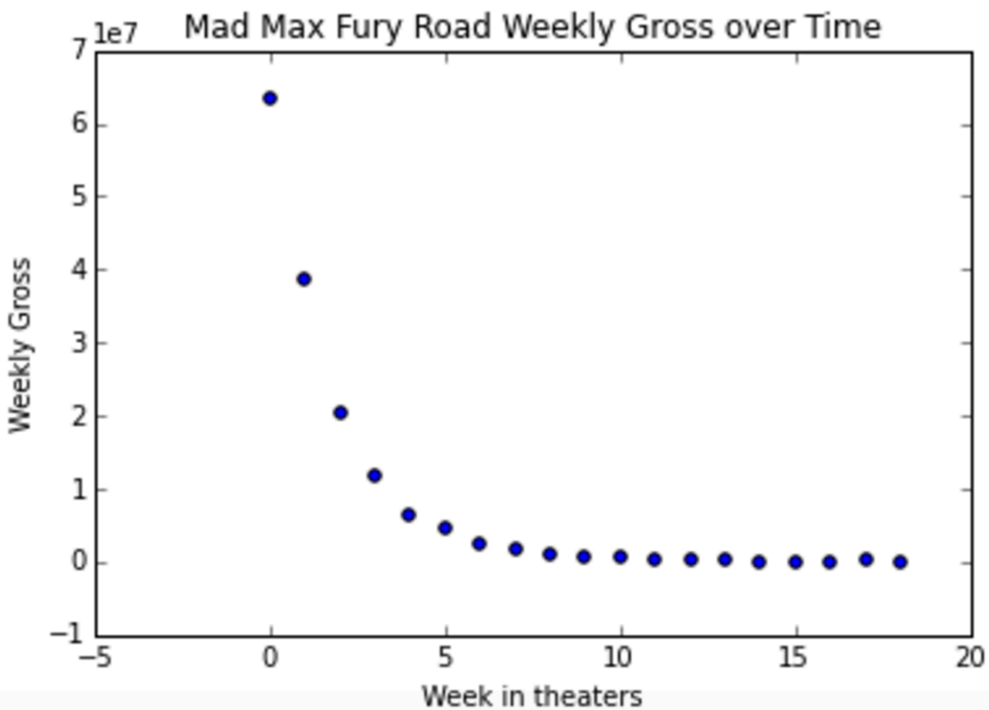
In the previous case, each point is independent.



Each point is an independent 'firing' of the stochastic process.

How much Avatar makes does not limit or influence by itself how much Chocolat makes

In a time series, points are **not** independent.



Points carry information about other points

Clearly, if the movie has made $600K on week 10, and $470K on week 11, you know that the movie will not make $2M on week 12.

But the approach does not need to be different from any other regression

What we know (features)

What we predict (target)

Values that carry information about the target → Model → Target Value

But the approach does not need to be different from any other regression

What we know (features)

What we predict (target)

Gross of Week 10 → Model → **Gross of Week 11**

Since past points carry information about the future points, use these past points as features

Target Feature

	weekly_gross	one_prev_weeks_gross
0	63440279	NaN
1	38849255	63440279
2	20544731	38849255
3	11643562	20544731
4	6309002	11643562
5	4555993	6309002
6	2648047	4555993
7	1645168	2648047
8	966275	1645168
9	601794	966275
10	663222	601794

Auto-Regressive Models

Regression, where one or more previously observed target values is used as (a) feature(s)

Target   Feature

	weekly_gross	one_prev_weeks_gross
0	63440279	NaN
1	38849255	63440279
2	20544731	38849255
3	11643562	20544731
4	6309002	11643562
5	4555993	6309002
6	2648047	4555993
7	1645168	2648047
8	966275	1645168
9	601794	966275
10	663222	601794

AR-1

A*uto****R****egressive model with lag* ***1***

Just one previous point is among the features

AR-1

What we know
(features)

What we predict
(target)

Gross of previous week → Model → **Gross of this week**

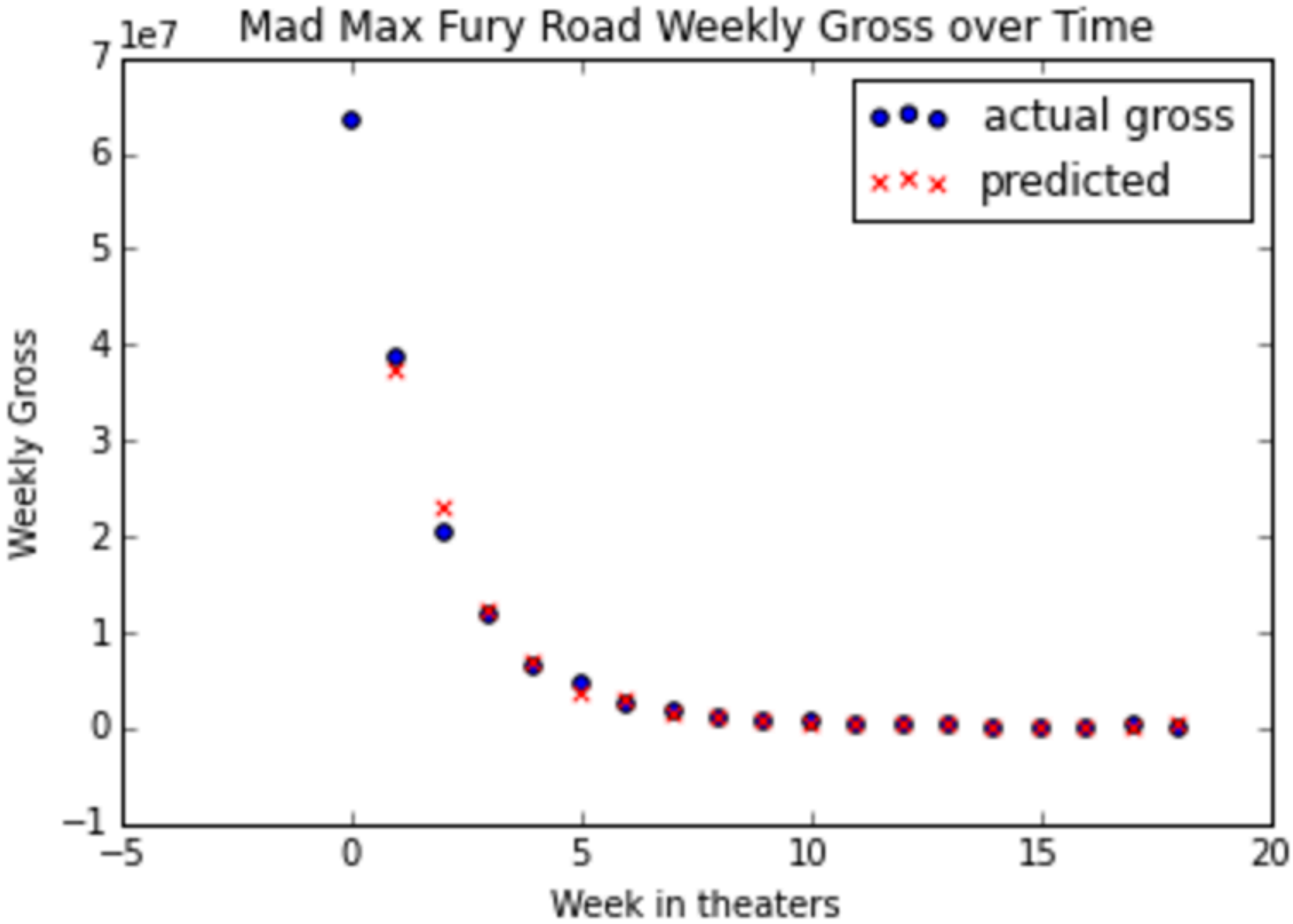
AR-1

We expect a week's gross for a movie to be
$10 Thousand
+ 60% of what it made last week
+ luck factor
(roll the dice for +- $110 Thousand)

$$x_t = \beta_0 + \beta_1 x_{t-1} + \varepsilon$$

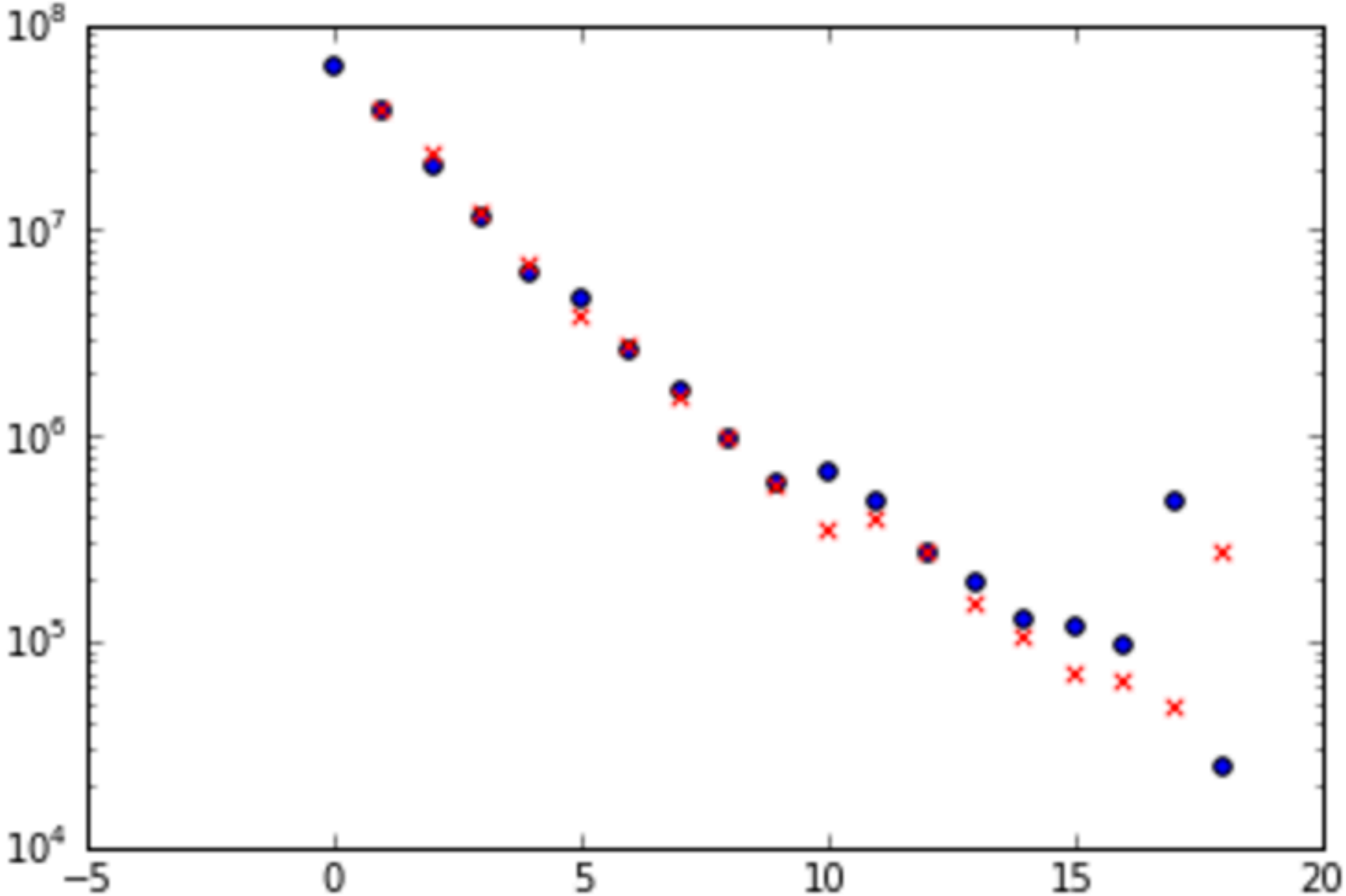
$$\beta_0 = 0.01 million$$
$$\beta_1 = 0.598$$

AR-1


Mad Max Fury Road Weekly Gross over Time
1e7
Weekly Gross
Week in theaters
actual gross
predicted
7
6
5
4
3
2
1
0
−1
−5
0
5
10
15
20

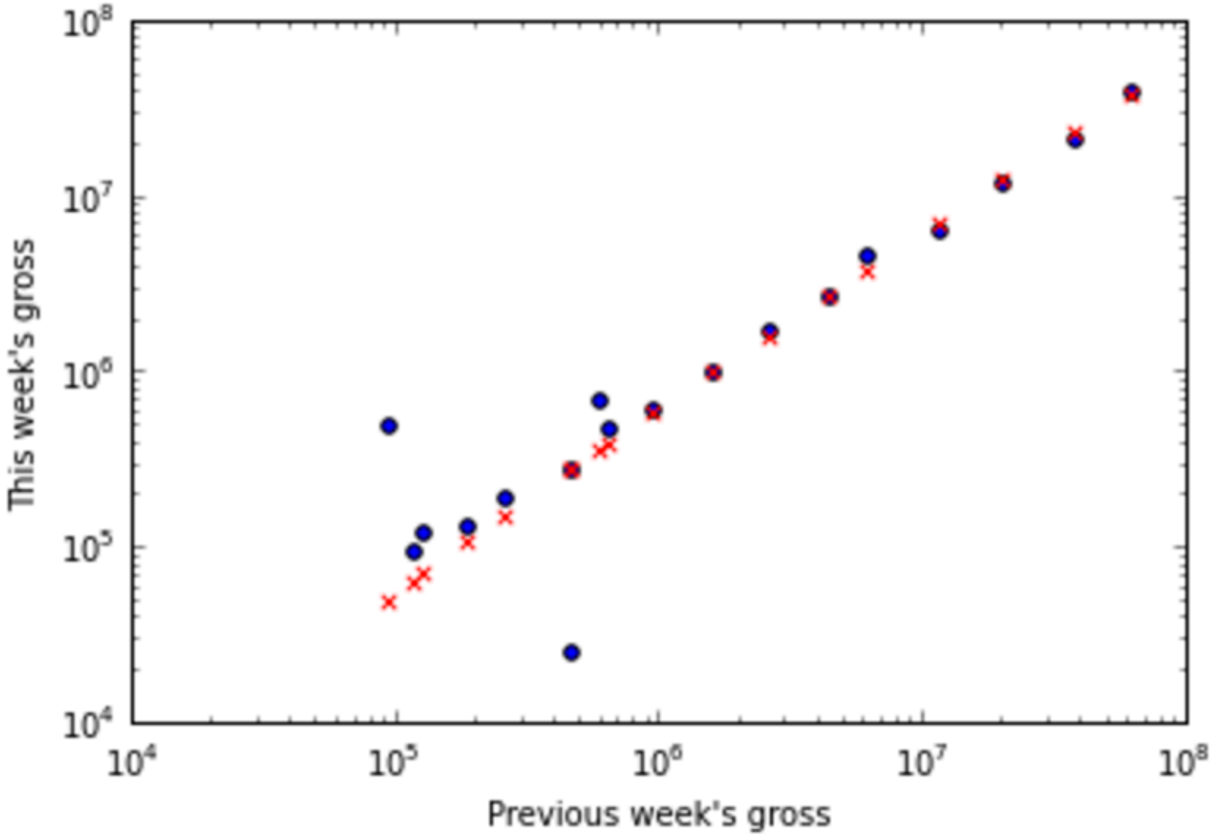
AR-1

Looking at the order of magnitude (log) of the gross instead to be able to see differences in later weeks



Previously, we were drawing target versus feature

Here is the AR1 model from the same angle: Drawing target vs an input feature



AR-2

What we know
(features)

What we predict
(target)

Gross of previous week
Gross of two weeks ago

Model

Gross of this week

Target Feature 1 Feature 2

	weekly_gross	one_prev_weeks_gross	two_prev_weeks_gross
0	63440279	NaN	NaN
1	38849255	63440279	NaN
2	20544731	38849255	63440279
3	11643562	20544731	38849255
4	6309002	11643562	20544731
5	4555993	6309002	11643562
6	2648047	4555993	6309002
7	1645168	2648047	4555993
8	966275	1645168	2648047
9	601794	966275	1645168
10	663222	601794	966275

AR-2

__A__uto__R__egressive model with lag __2__

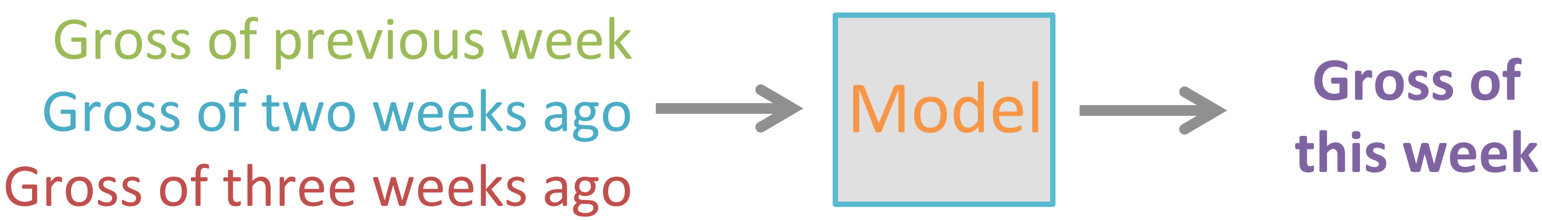
Two previous points as features

AR-3

and so on and so forth…

What we know (features)

What we predict (target)



You can do the lag processing yourself, or just use AR models in statsmodels to do it for you

```
import statsmodels.api as sm

AR1 = sm.tsa.AR(weekly_flu_numbers, freq="W").fit(maxlag=1)

AR2 = sm.tsa.AR(weekly_flu_numbers, freq="W").fit(maxlag=2)

AR2 = sm.tsa.AR(weekly_flu_numbers, freq="W").fit(maxlag=3)
```

Big splash in 2008: Google Flu Trends!

Google can predict flu volume by analyzing Google searches!

Big splash in 2008: Google Flu Trends!

Google can predict flu volume
by analyzing Google searches!

Majorly Accurate Flu
Model from Google!

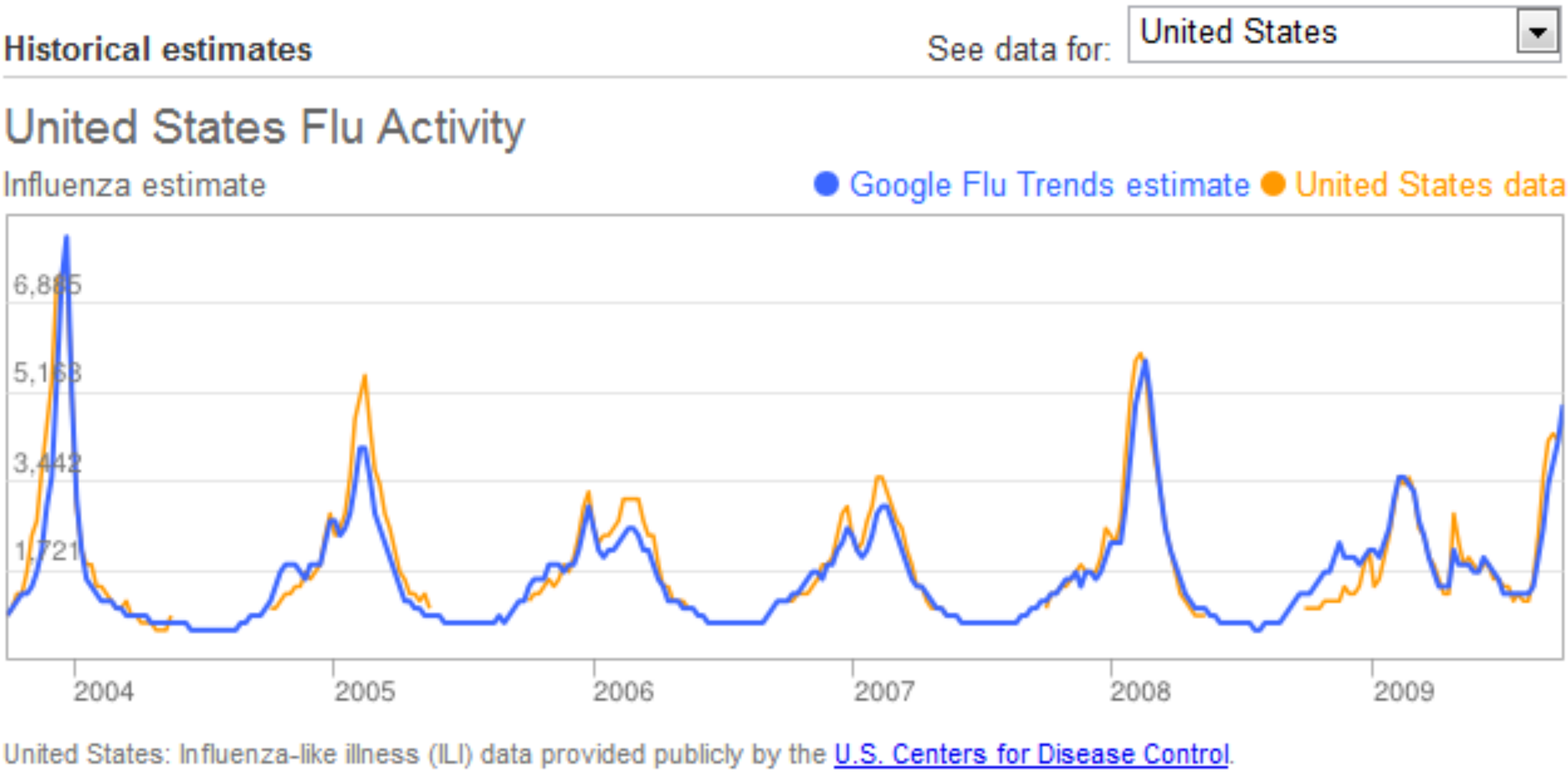
Big splash in 2008: Google Flu Trends!

By looking at how many people google for flu related terms, Google can predict flu epidemics!

Google can predict flu volume by analyzing Google searches!

Majorly Accurate Flu Model from Google!

Big splash in 2008: Google Flu Trends!



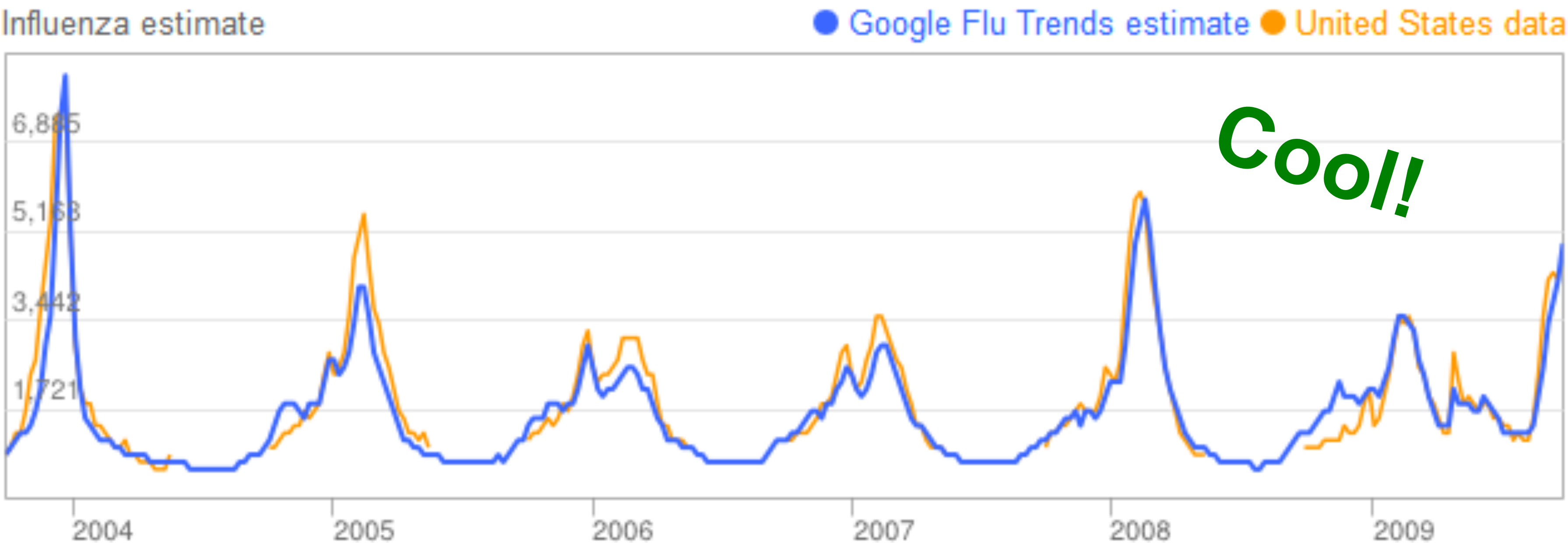
Neat!

OMG AMAZANNGG!
It's, like, the future!

Historical estimates

See data for: United States

United States Flu Activity



Cool!

United States: Influenza-like illness (ILI) data provided publicly by the U.S. Centers for Disease Control.

Google Flu Model

What we know (features) | | What we predict (target)

Search volume for flu related terms this week → Model → **Number of flu cases this week**

The bubble burst when we realized that an autoregressive model can produce the same accuracy

What we know
(features)

What we predict
(target)

Number of flu cases last week → Model → **Number of flu cases this week**

And actually adding google search data as a new feature gives only a tiny improvement in performance

What we know (features)

Search volume for flu related terms this week

Number of flu cases last week

→ Model →

What we predict (target)

Number of flu cases this week

AR can be powerful

What we know
(features)

What we predict
(target)

Number of flu cases last week → Model → **Number of flu cases this week**

This has been available even before the Internet

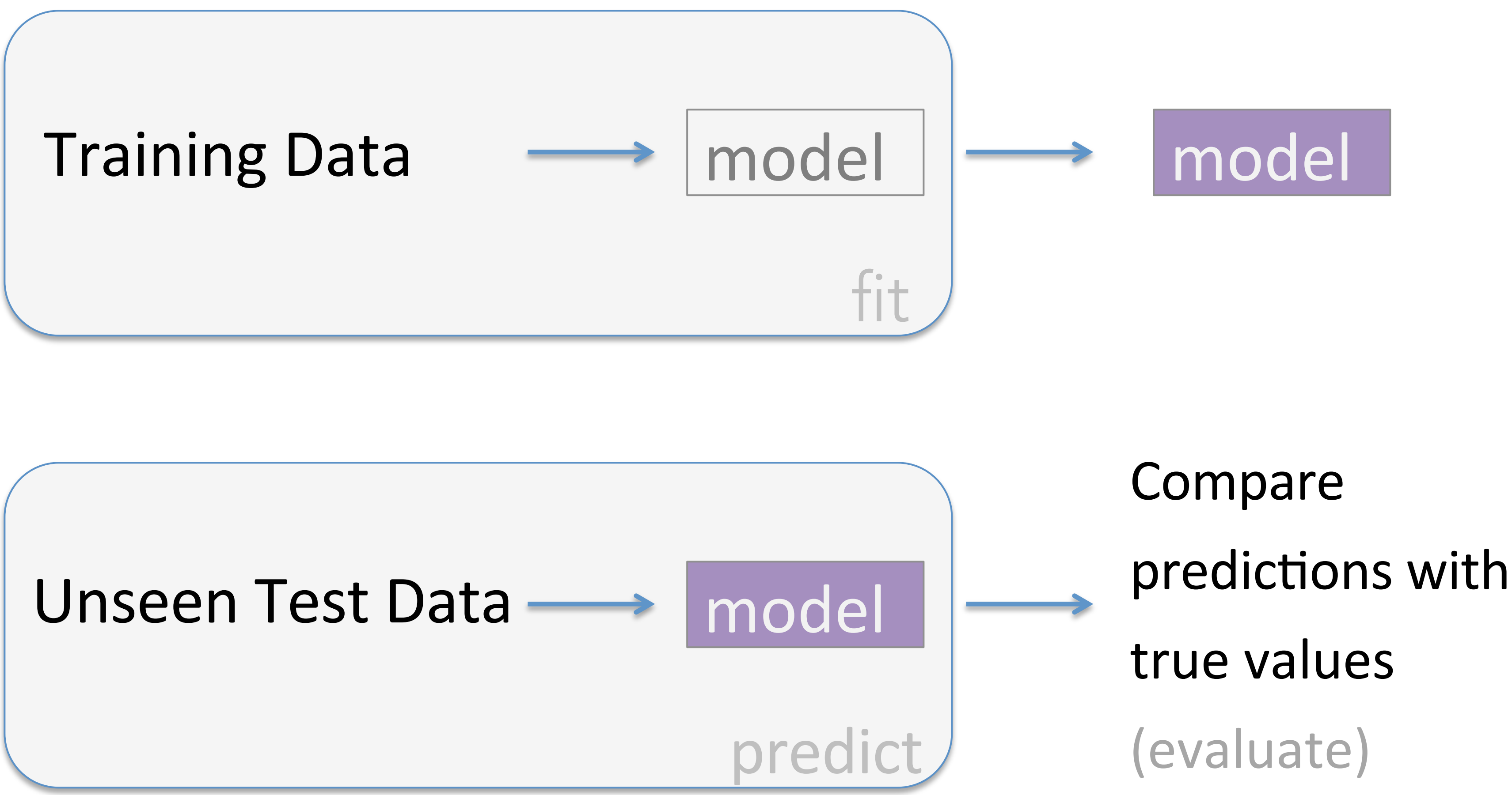
Think of lagged-features just as another potential source of usable information in a standard regression problem.

You can combine them with other features as well.

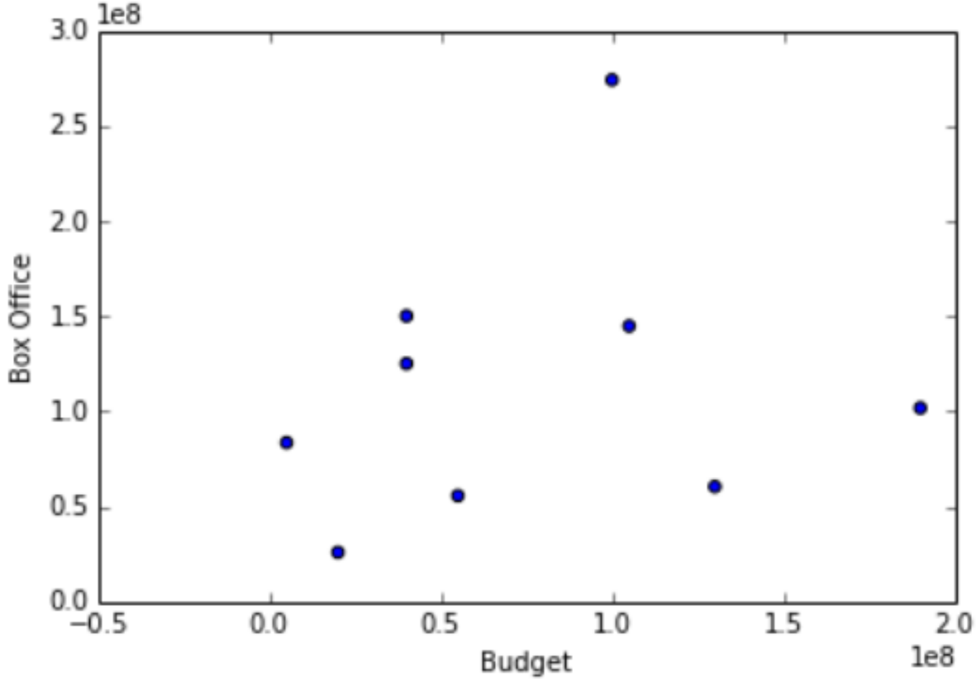
AR models can be quite powerful for **short term forecasting** (next time point), but their efficiency goes down when you try to forecast long term (as errors propagate from each next point you predict)

How do we do the train/test split?

As with any regression problem, it is important to measure performance on a separate test set.



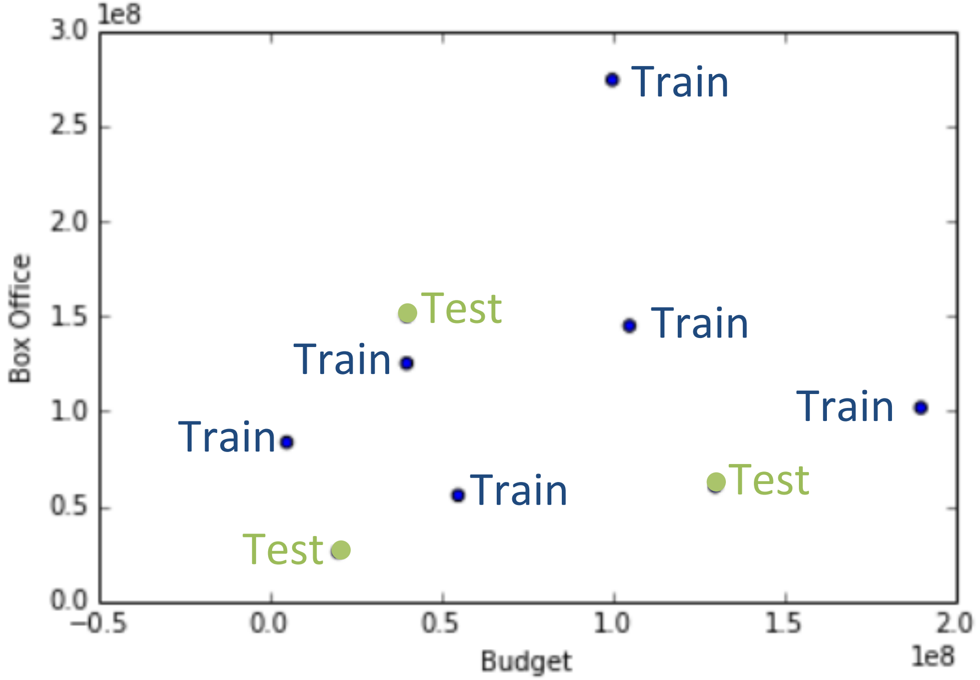
In previous cases, we could just randomly pick some points and set them aside as a test set



66% Training set

33% Test set

In previous cases, we could just randomly pick some points and set them aside as a test set

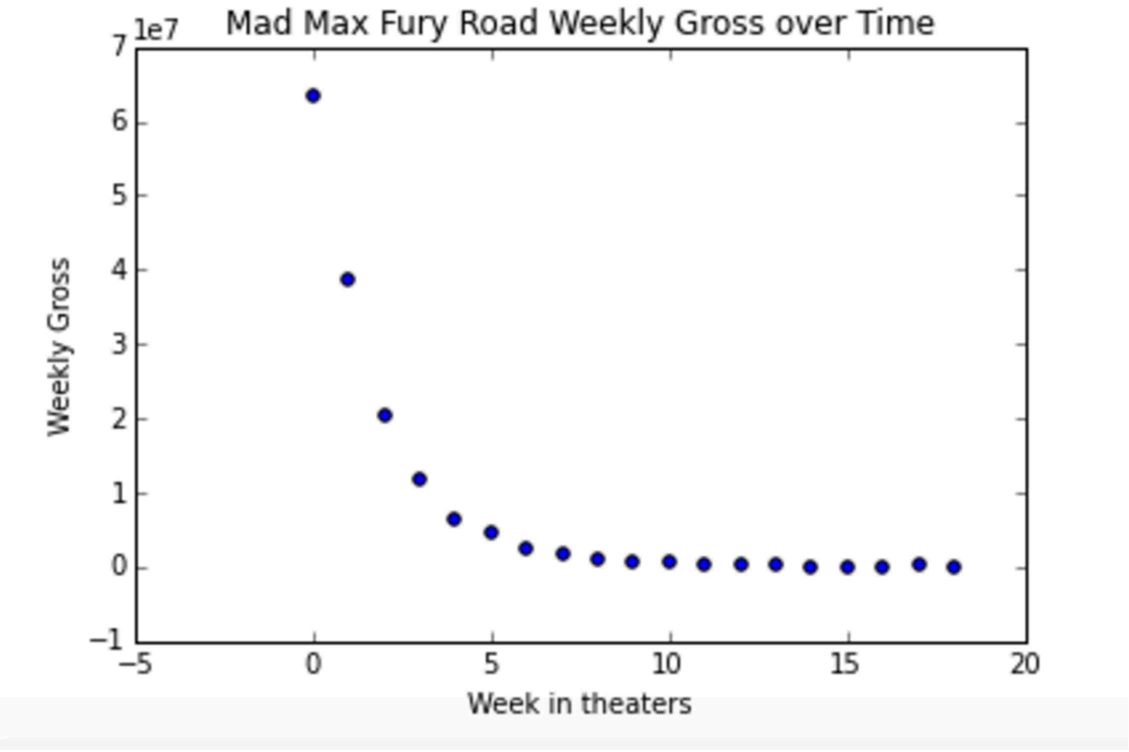


66% Training set

33% Test set

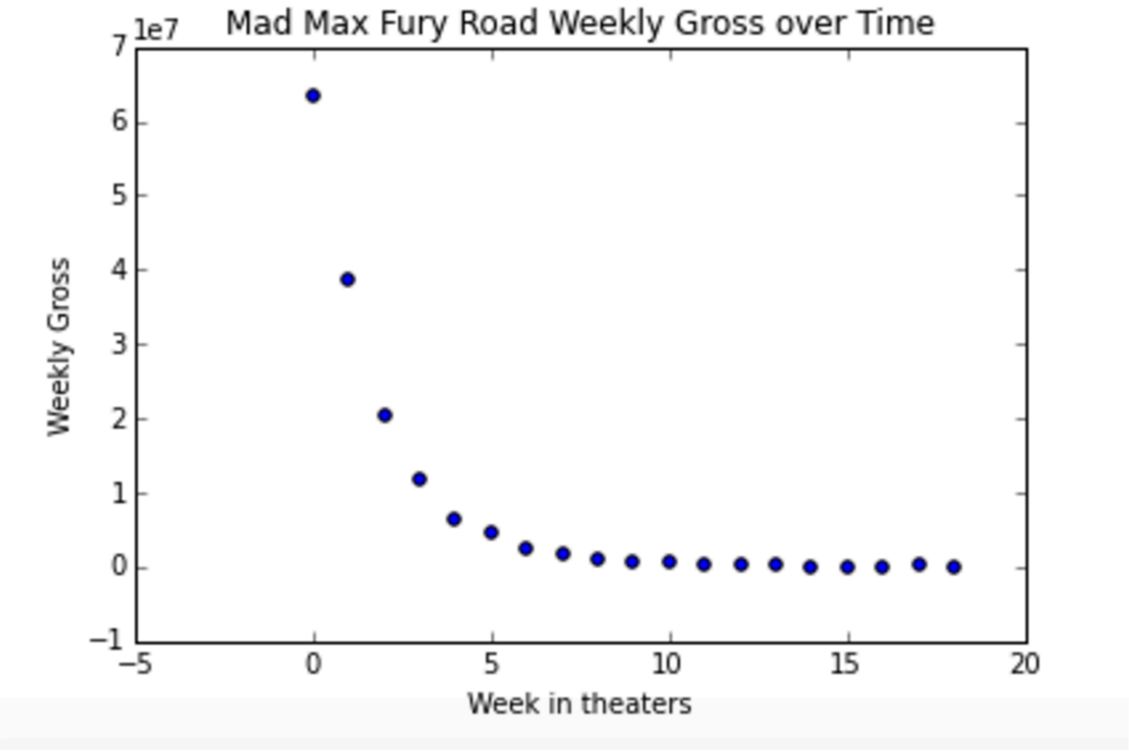
What about time series?

These points aren't independent



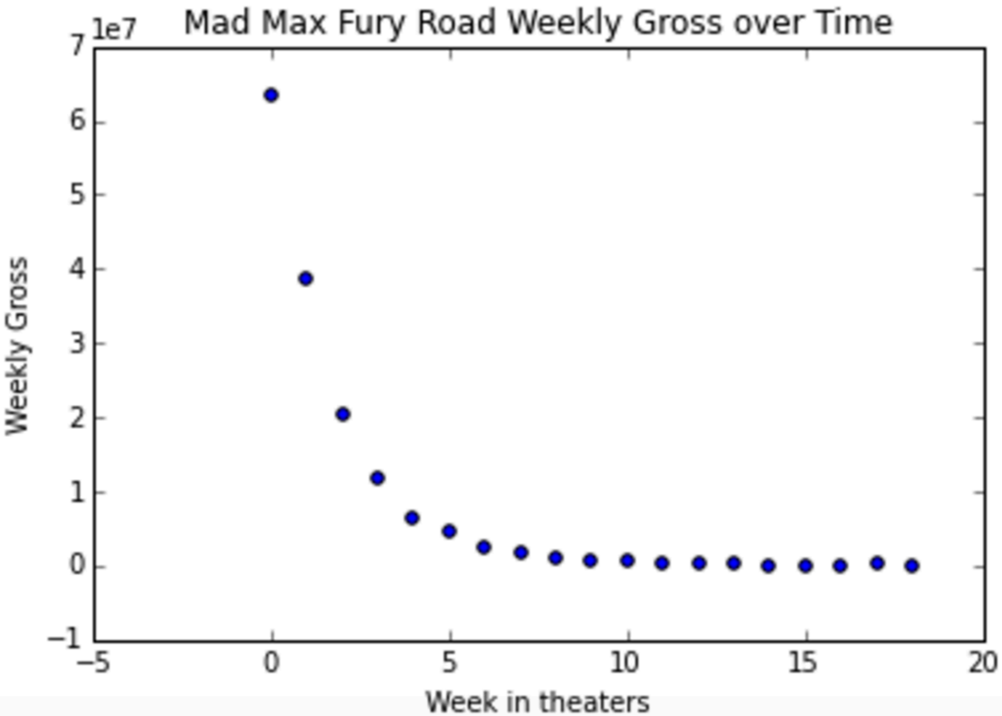
Depends on your use case!

Test set should simulate exactly what type of points you will make predictions on with the model



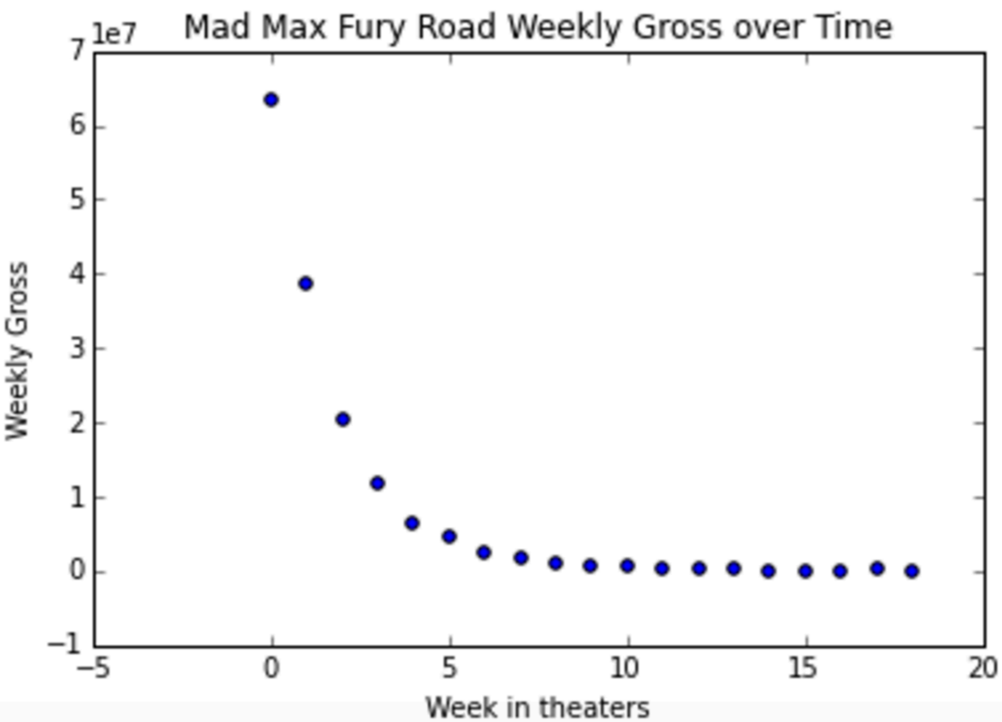
Train/test split for time series

Case 1: We model a with past observations to make predictions on how it will continue.



Train/test split for time series

Case 1: We model a with past observations to make predictions on how it will continue.

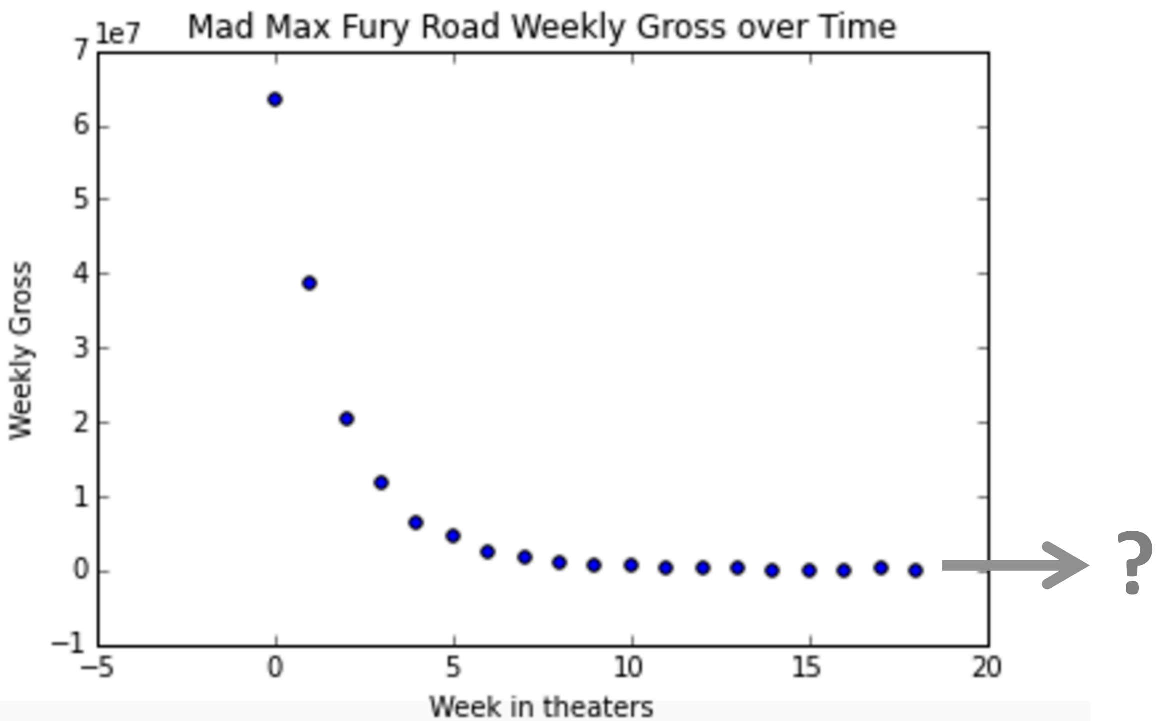


Mad Max has been running for 18 weeks.

Each week, I want to ask the model to forecast next week's gross, to decide if it's worth keeping it on one more week

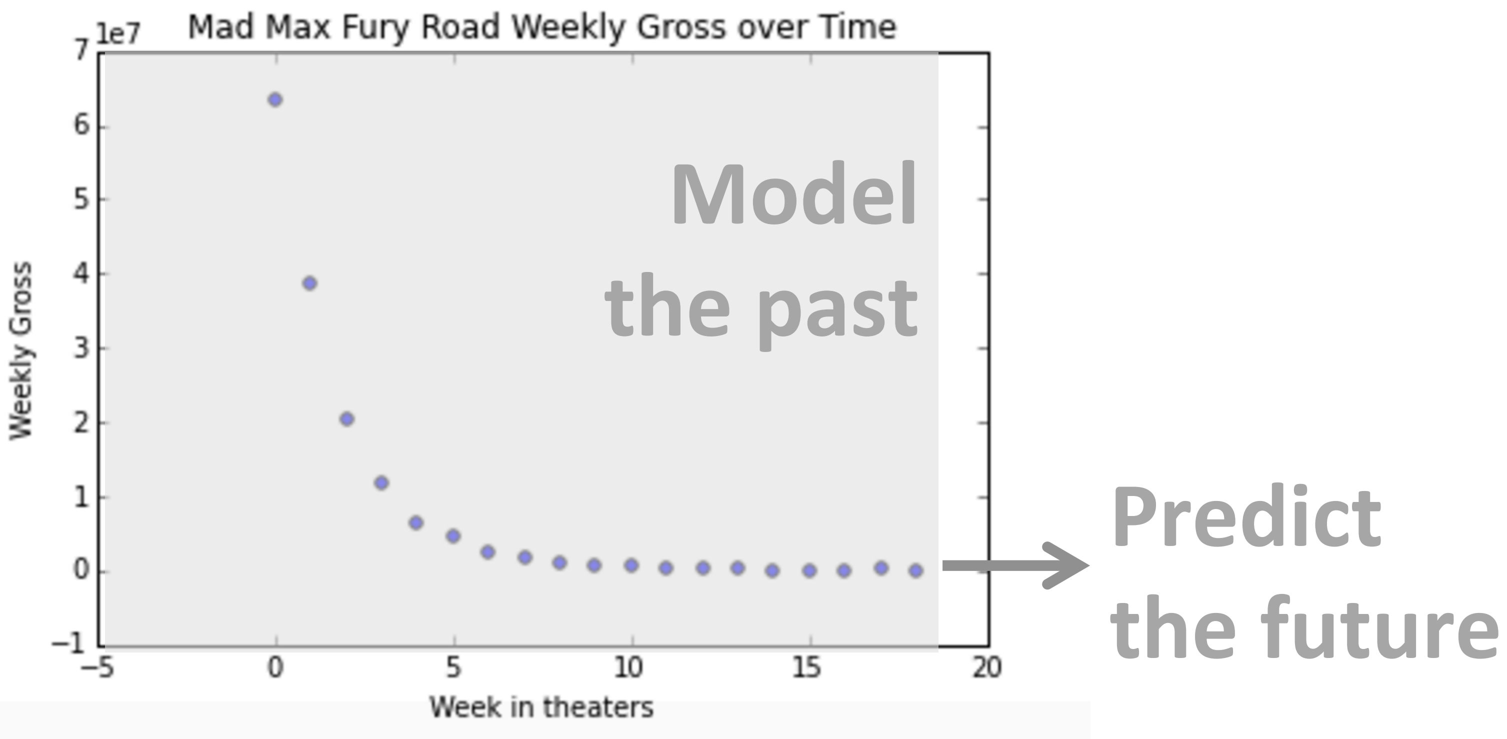
Train/test split for time series

Case 1: We model a with past observations to make predictions on how it will continue.



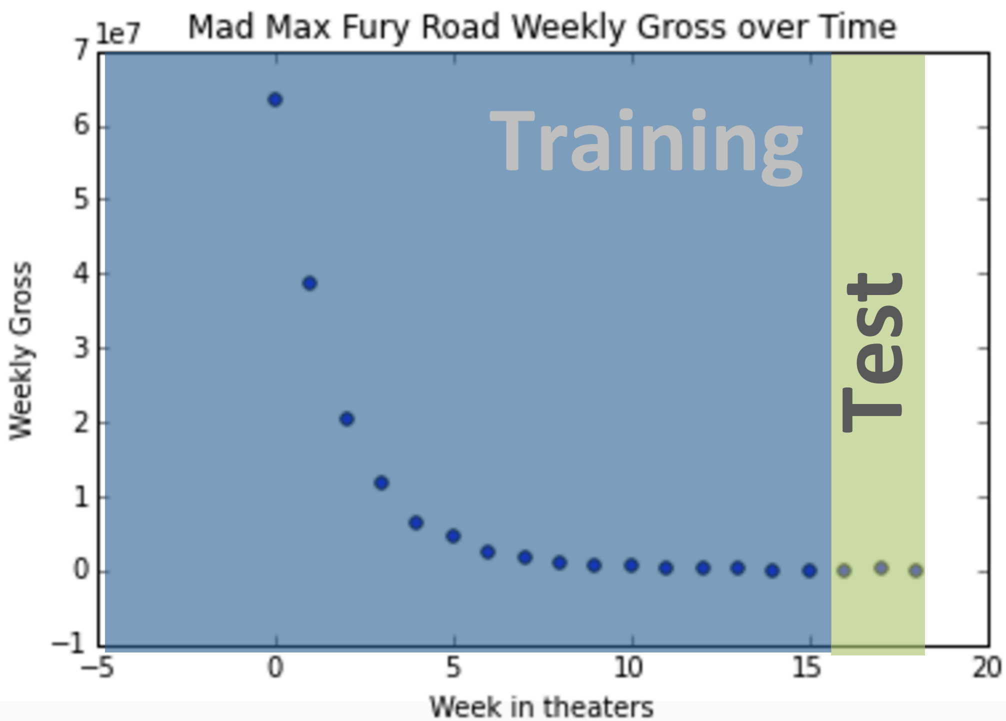
Train/test split for time series

Case 1: We model a with past observations to make predictions on how it will continue.



Train/test split for time series

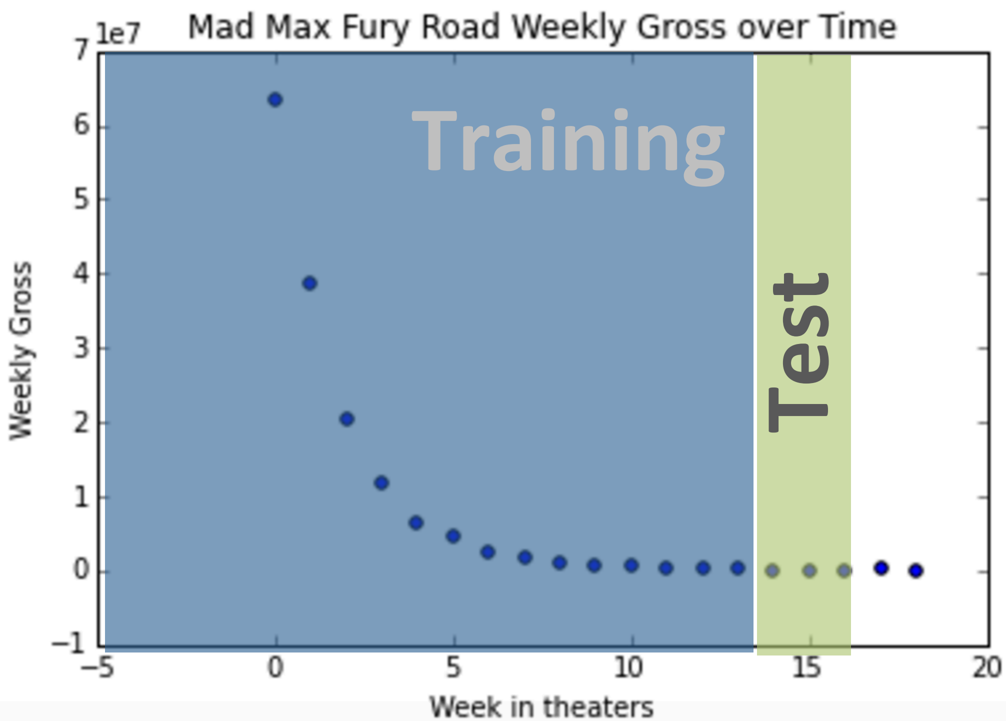
Case 1: We model a with past observations to make predictions on how it will continue.



Your training/test split should simulate building a model on past values and testing them on newer values!

Train/test split for time series

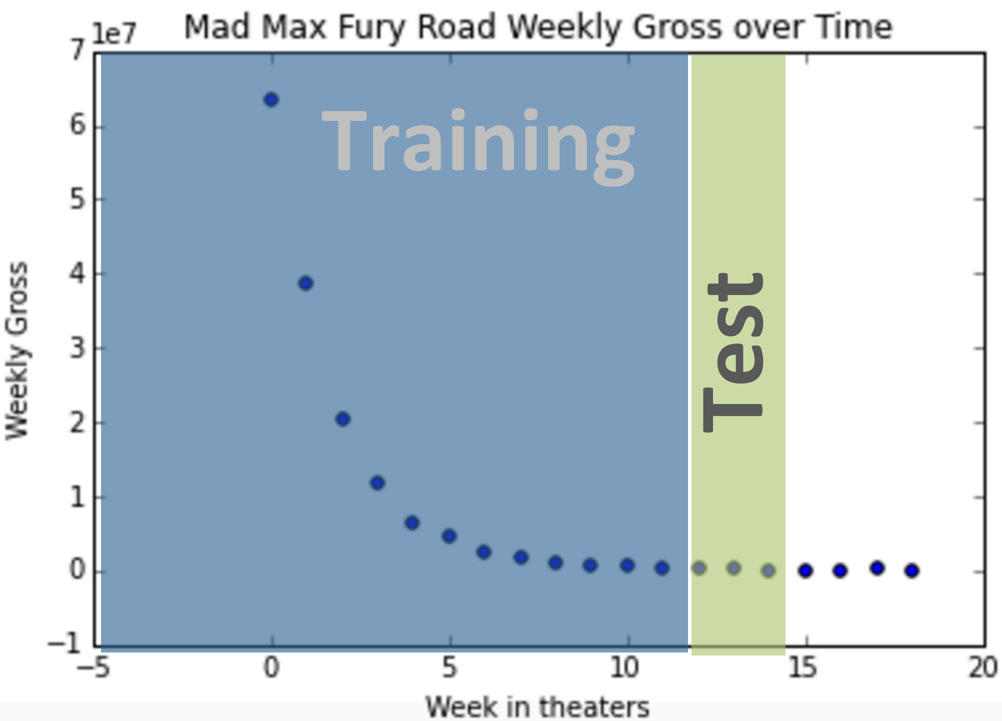
Case 1: We model a with past observations to make predictions on how it will continue.



You can even try it on different windows (kind of similar to cross-validation, but not quite)

Train/test split for time series

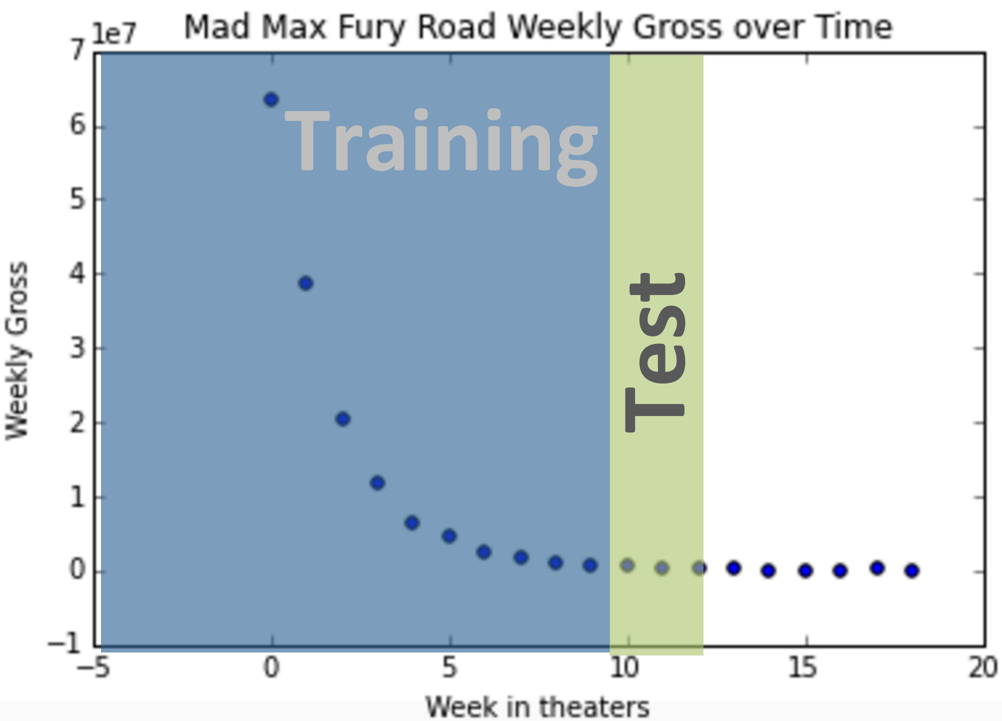
Case 1: We model a with past observations to make predictions on how it will continue.



You can even try it on different windows (kind of similar to cross-validation, but not quite)

Train/test split for time series

Case 1: We model a with past observations to make predictions on how it will continue.



You can even try it on different windows (kind of similar to cross-validation, but not quite)

Train/test split for time series

Case 2: We model gross decay over time in movies in the past to predict how it'll go in new movies.

?

We just opened The Force Awakens. We need a model that can study all past movies and predict what gross to expect each week.

Train/test split for time series

Case 2: We model gross decay over time in movies in the past to predict how it'll go in new movies.

Model observed movies

?

We just opened The Force Awakens. We need a model that can study all past movies and predict what gross to expect each week.

Predict new movie

Train/test split for time series

Case 2: We model gross decay over time in movies in the past to predict how it'll go in new movies.

Randomly assign movies into training and test sets, fit AR model to the points from the movies in the training set, evaluate performance on test set movies

Train/test split for time series

Depending on your use case, set aside a test set that simulates the way in which you will ask for predictions

How to choose the order of AR

AR1? AR2? AR3? Which do I use?

Previous point's value, two-previous point's value, etc. are different features.

Just like in any other regression, **you will try different feature sets and choose the model with the test performance.**

Using more lag-features can improve performance, too many can lead to overfitting. Depends on the specific problem.

How to choose the order of AR

AR1? AR2? AR3? Which do I use?

Note:

You can also look at the autocorrelation function to inform your process of choosing AR order, but in the end, a hypothesis-driven trial-and-error process with performance evaluation on separate test sets will be the ultimate best way of determining order.

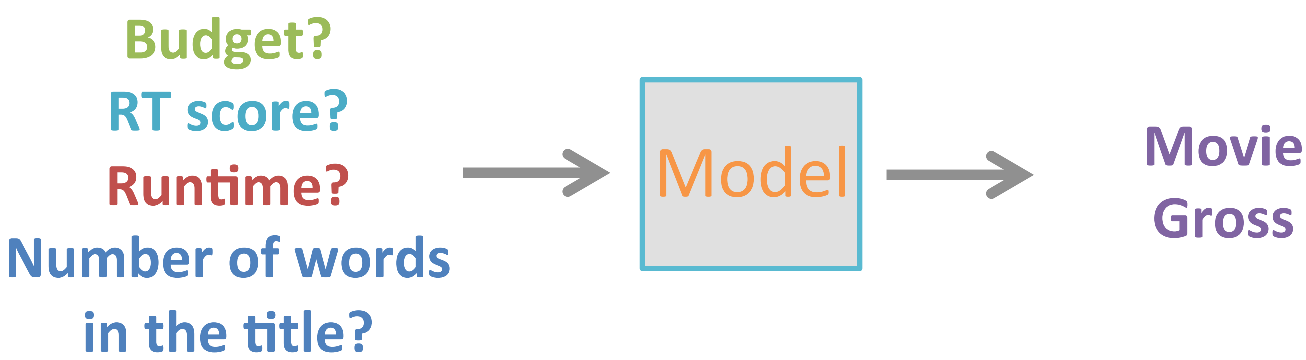
We will touch upon autocorrelation functions later in the bootcamp, when we revisit Time Series.

Autocorrelation

Another input to understand the time series better and help choose the order of the AR model

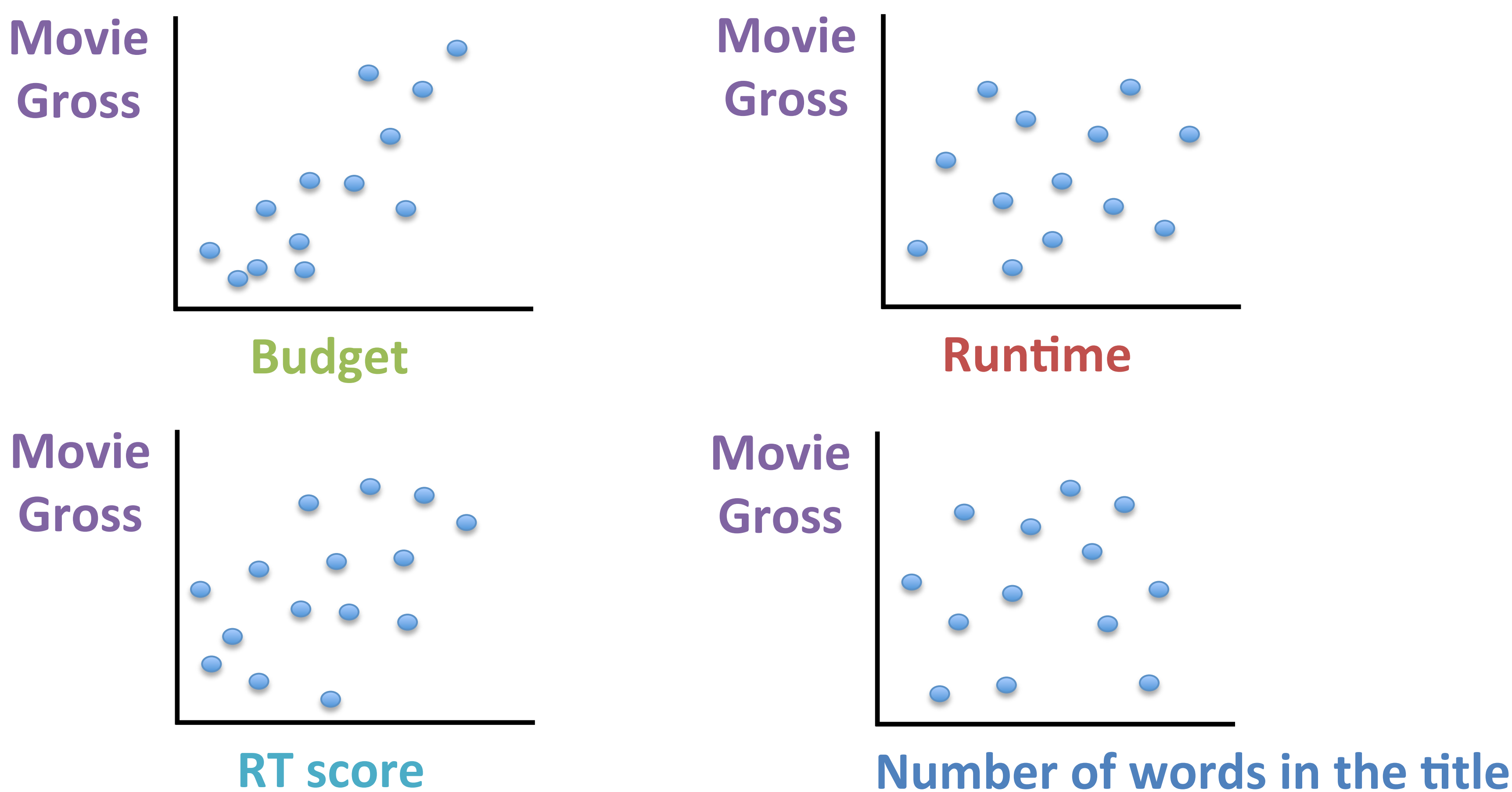
Autocorrelation

Imagine a simple regression model. No time series. We need to predict total movie gross, and we have several features we can use. We are trying to understand the system and choose features.



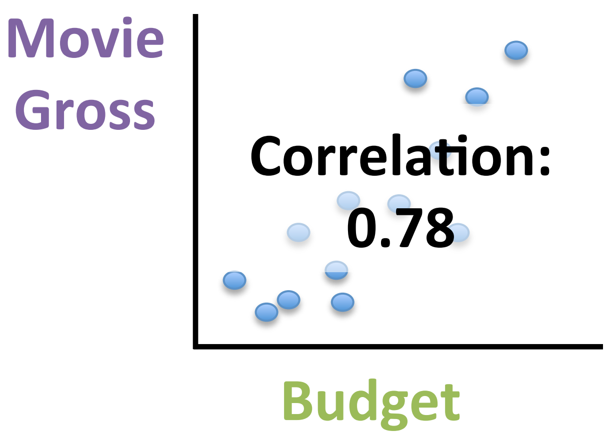
Autocorrelation

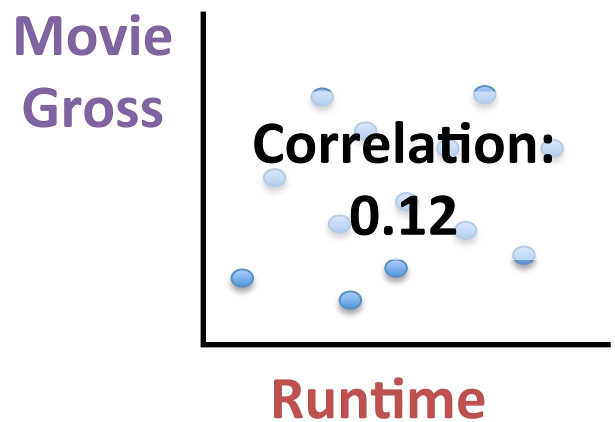
A good approach is to plot each feature by itself versus the target, and look at their correlation. This way we can see how much information each feature carries.

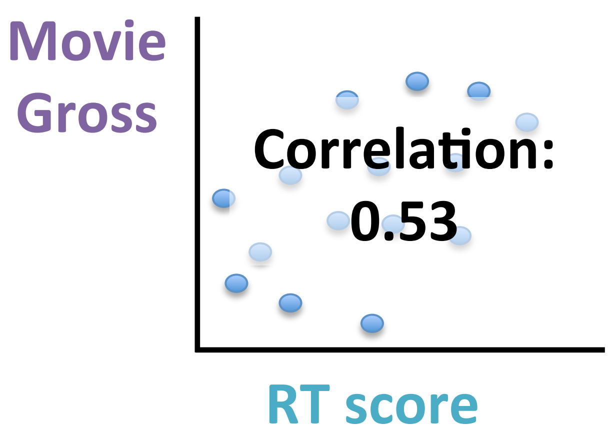


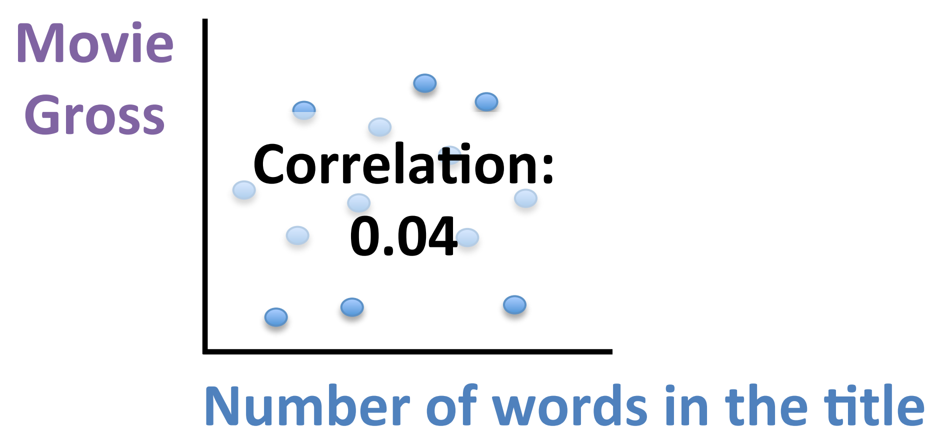
Autocorrelation

A good approach is to plot each feature by itself versus the target, and look at their correlation. This way we can see how much information each feature carries.



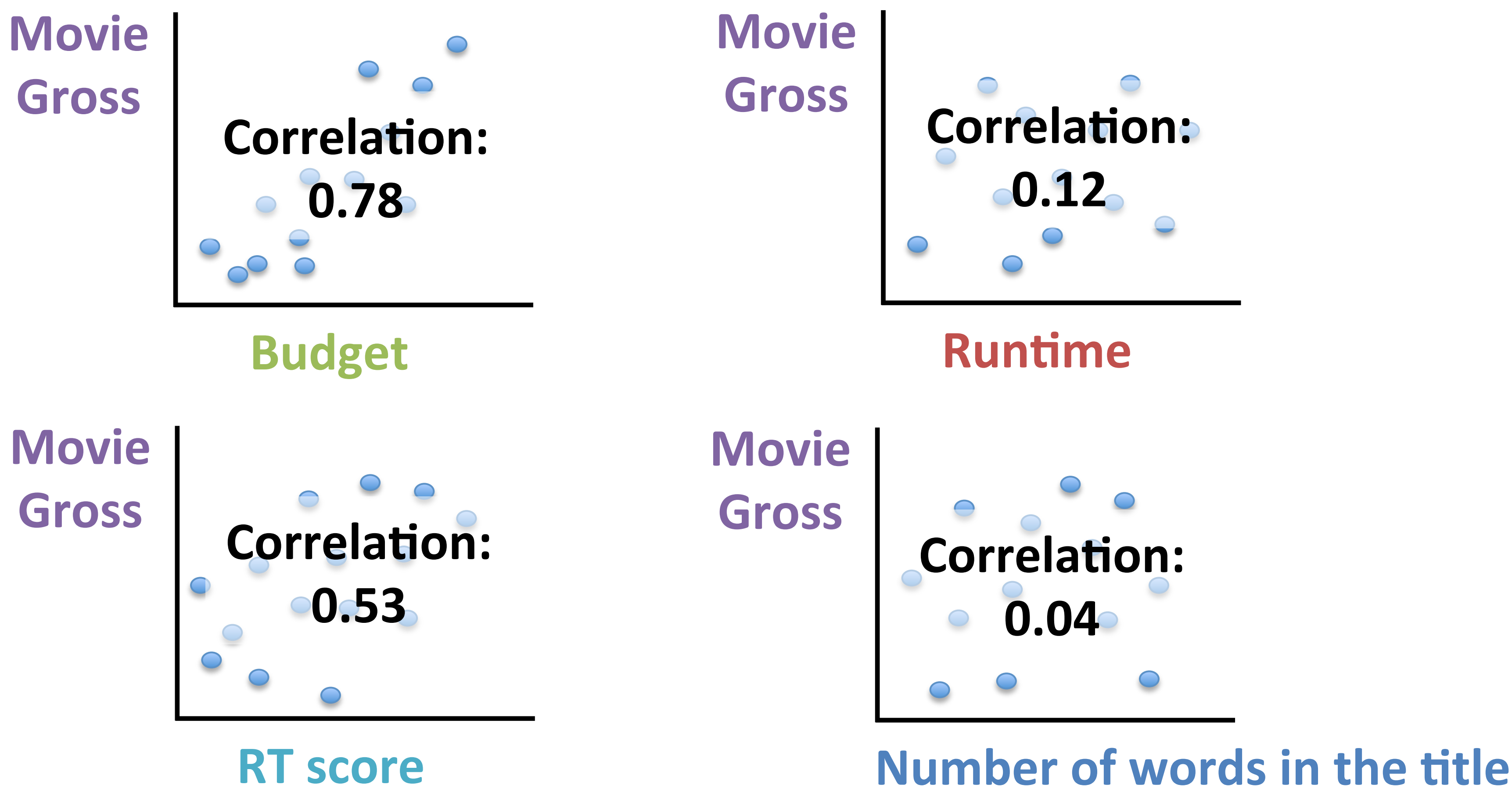






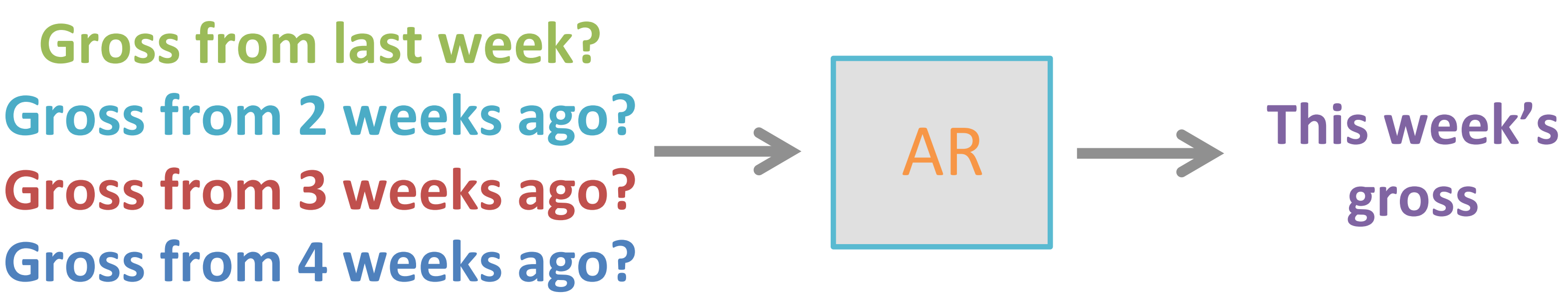
Autocorrelation

Of course, this by itself cannot be the sole determining information in selecting the feature set, but it helps us understand the contribution of information!



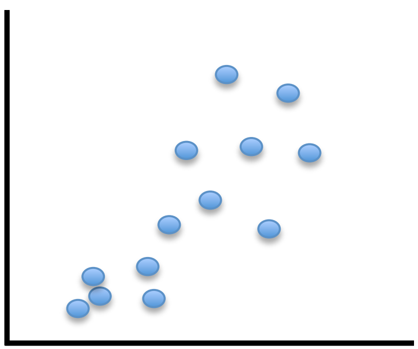
Autocorrelation

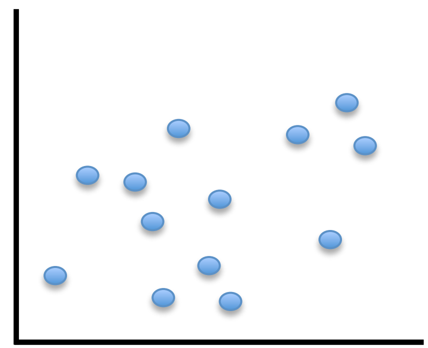
Now imagine our weekly gross problem again. Basically, trying to figure out what order AR to use is the same as trying to figure out which features to use.

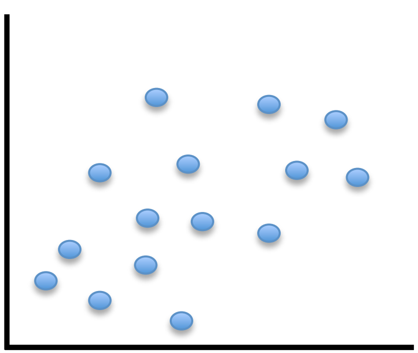


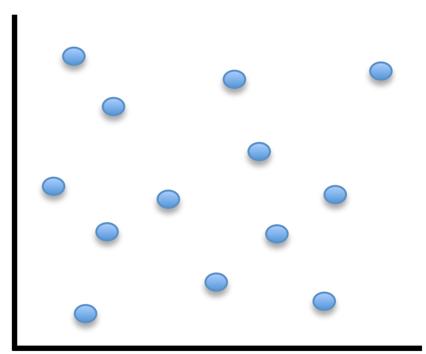
Autocorrelation

It is still a good idea to understand the correlations between each of these features and the target.



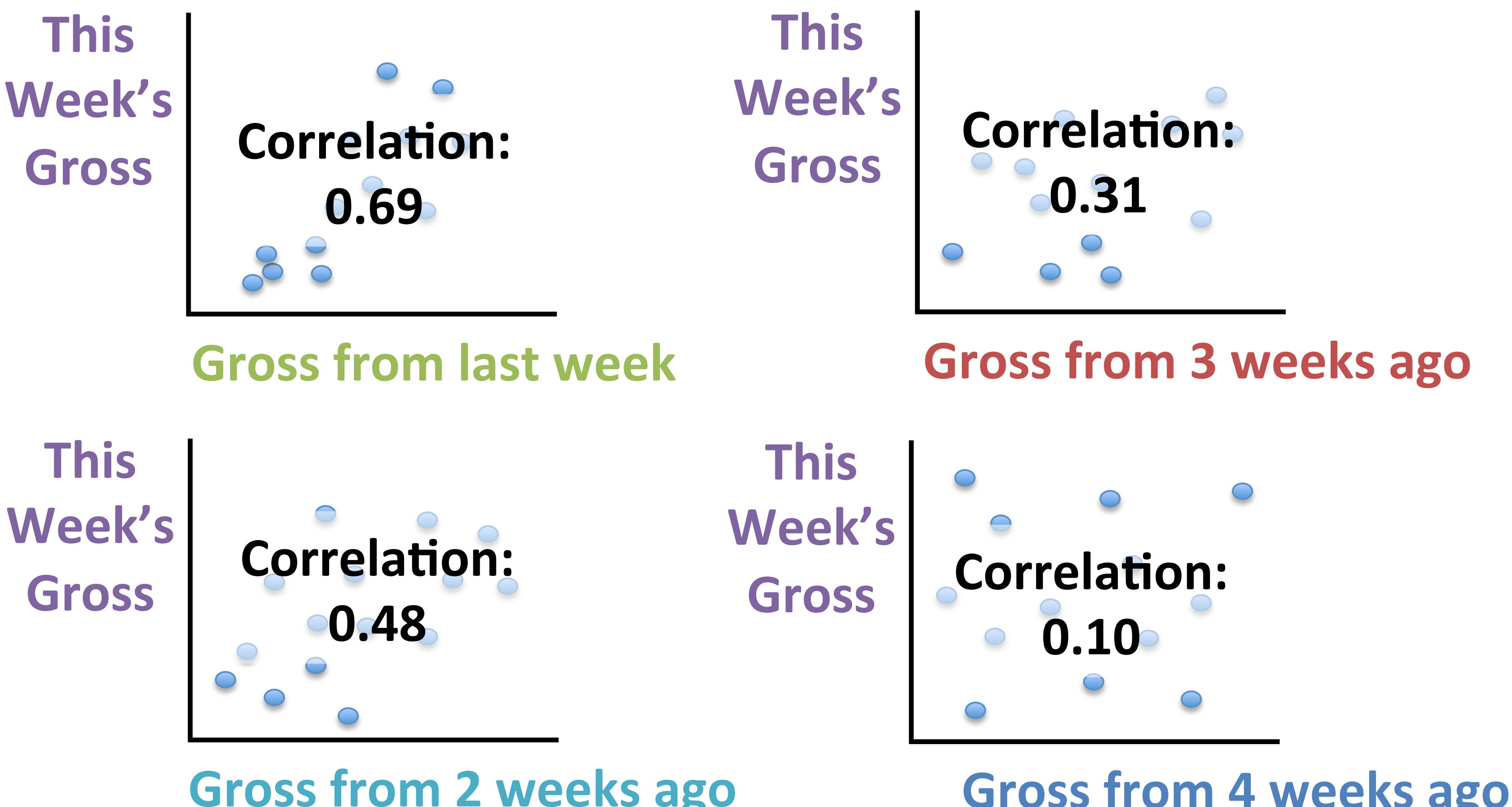






Autocorrelation

It is still a good idea to understand the correlations between each of these features and the target.



Autocorrelation

The autocorrelation function basically condenses this information into a single plot

Correlation: 0.69

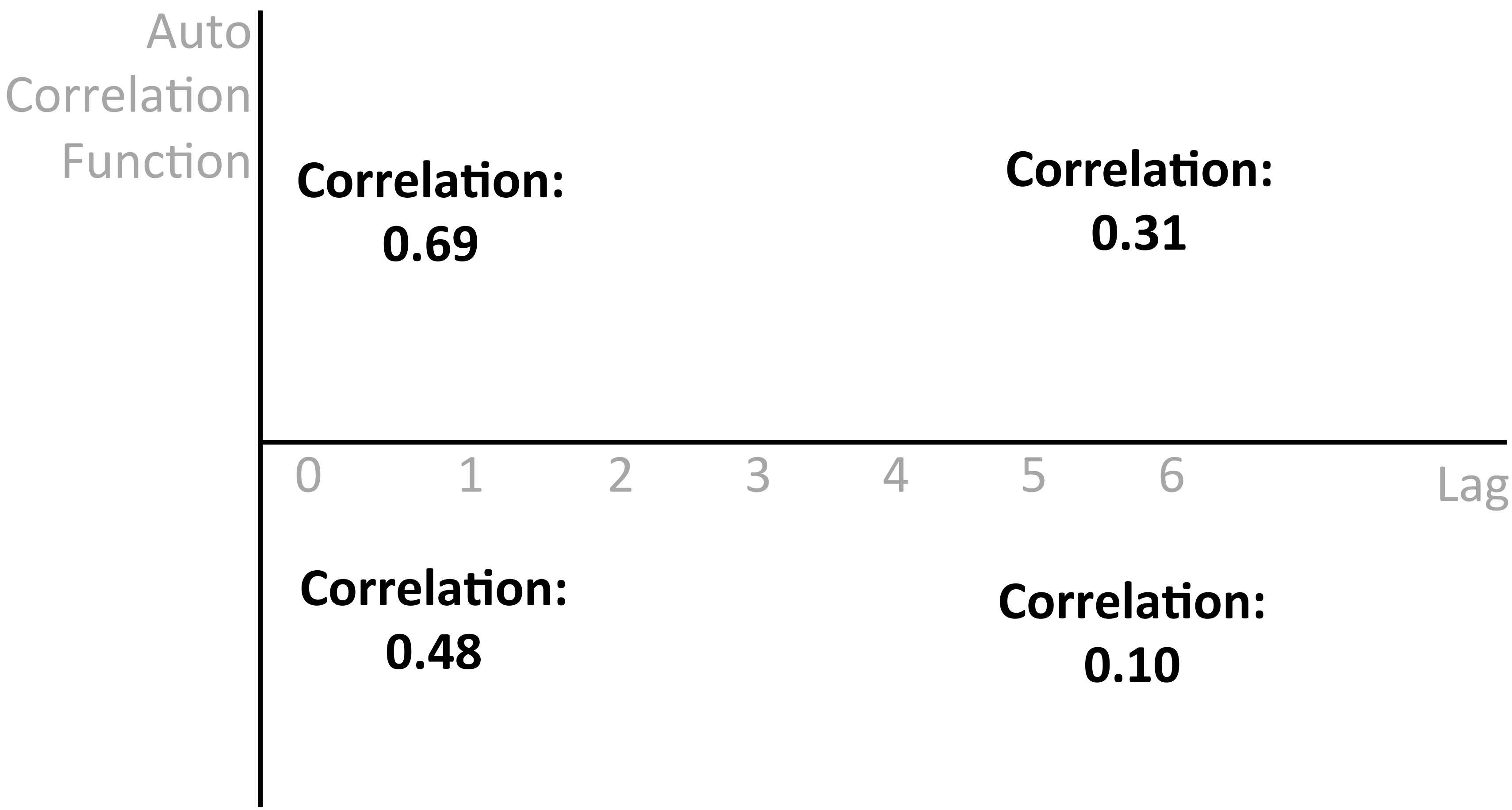
Correlation: 0.31

Correlation: 0.48

Correlation: 0.10

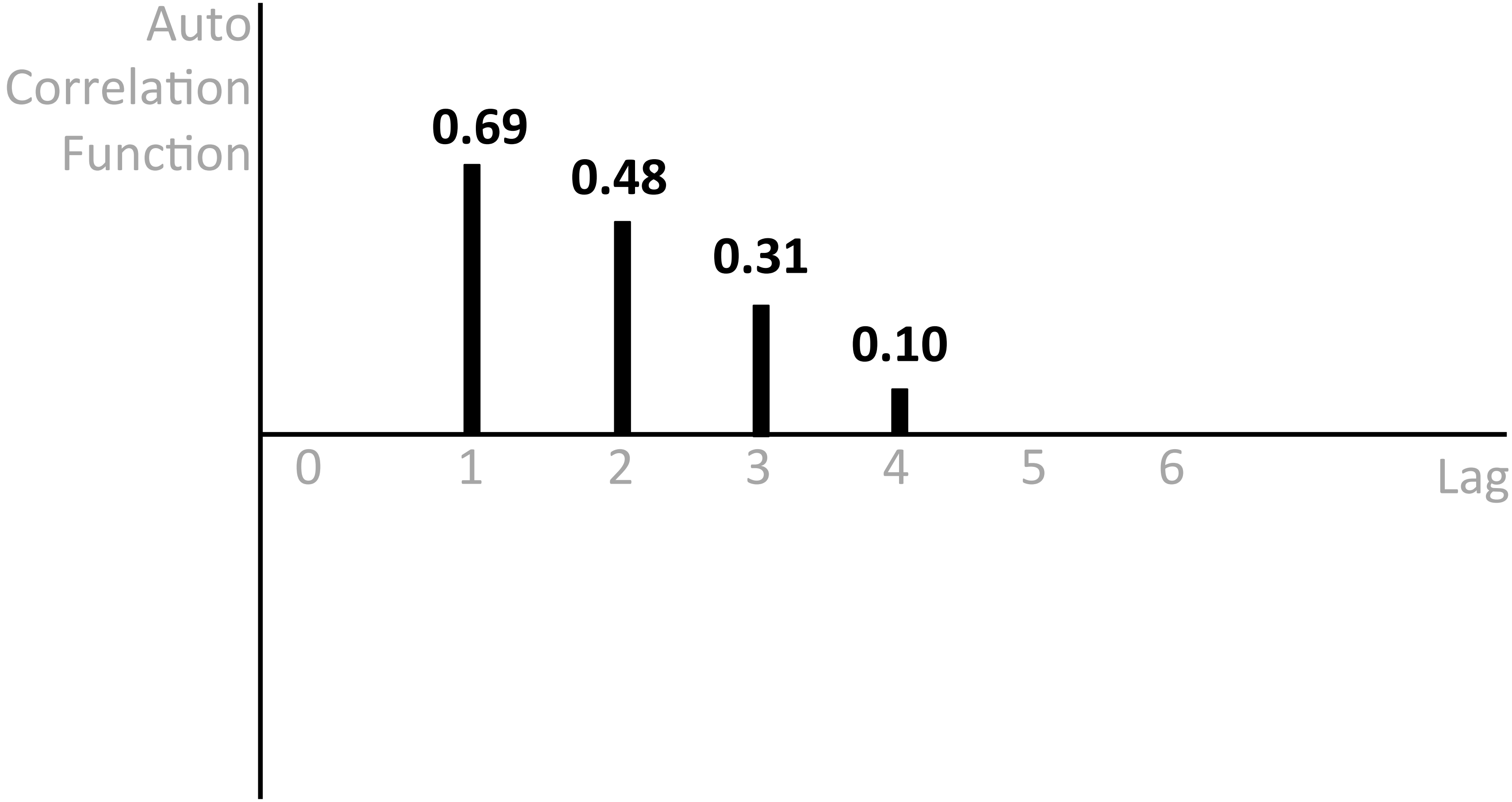
Autocorrelation

The autocorrelation function basically condenses this information into a single plot



Autocorrelation

The autocorrelation function basically condenses this information into a single plot



Autocorrelation

```
import statsmodels as sm
sm.tsa.stattools.acf(timeseries)
```

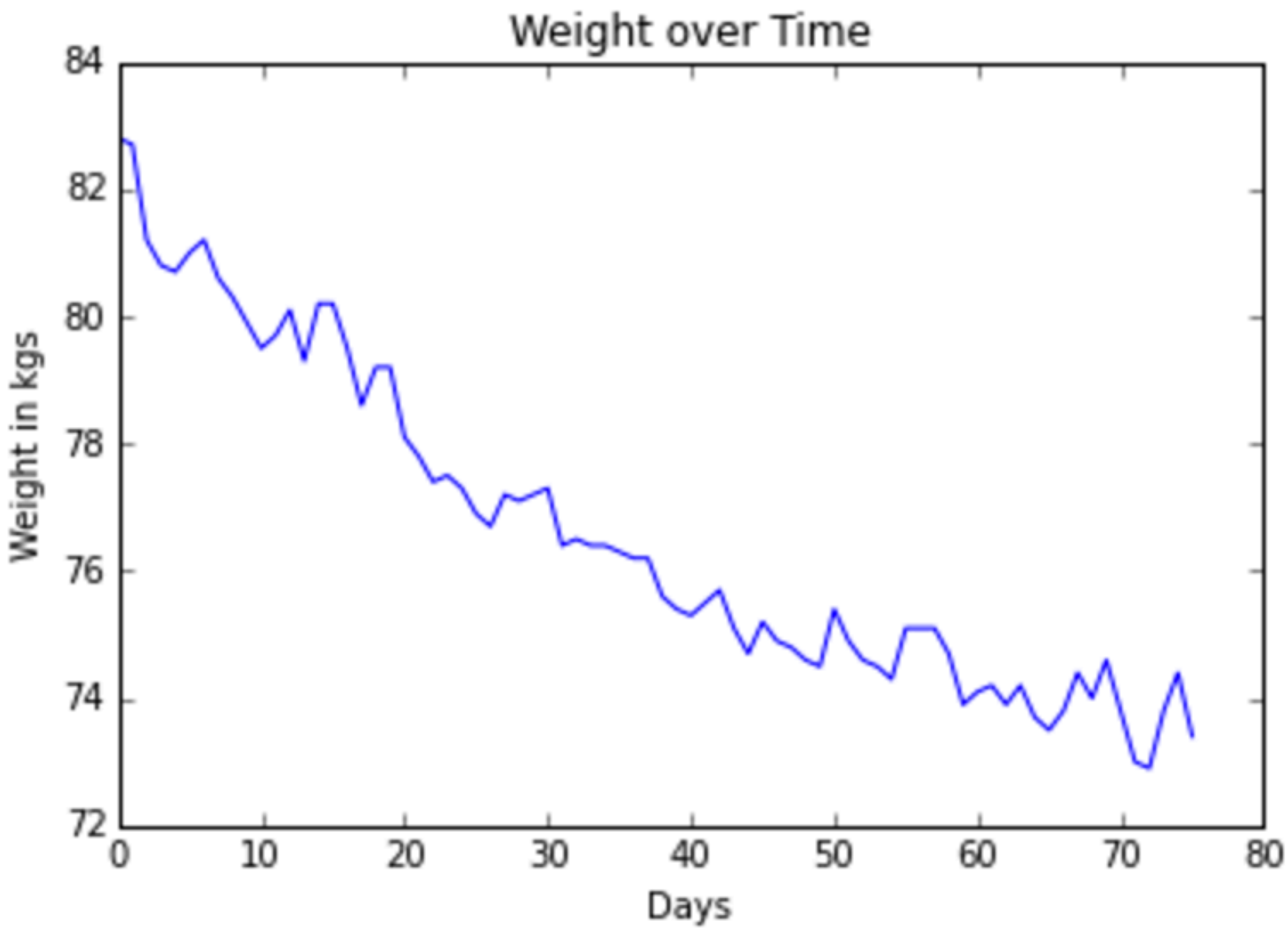
creates the autocorrelation function for you.

Preprocessing: Moving Average

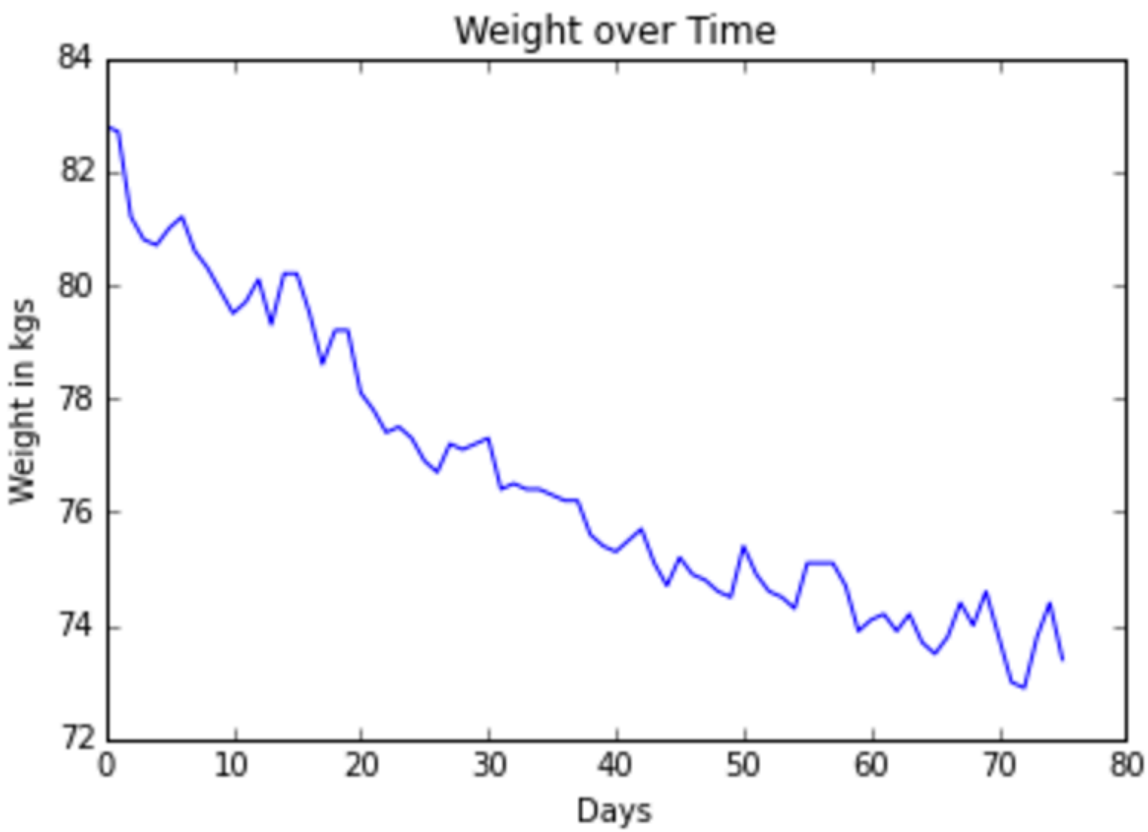
specific to time-series

Preprocessing: Moving Average

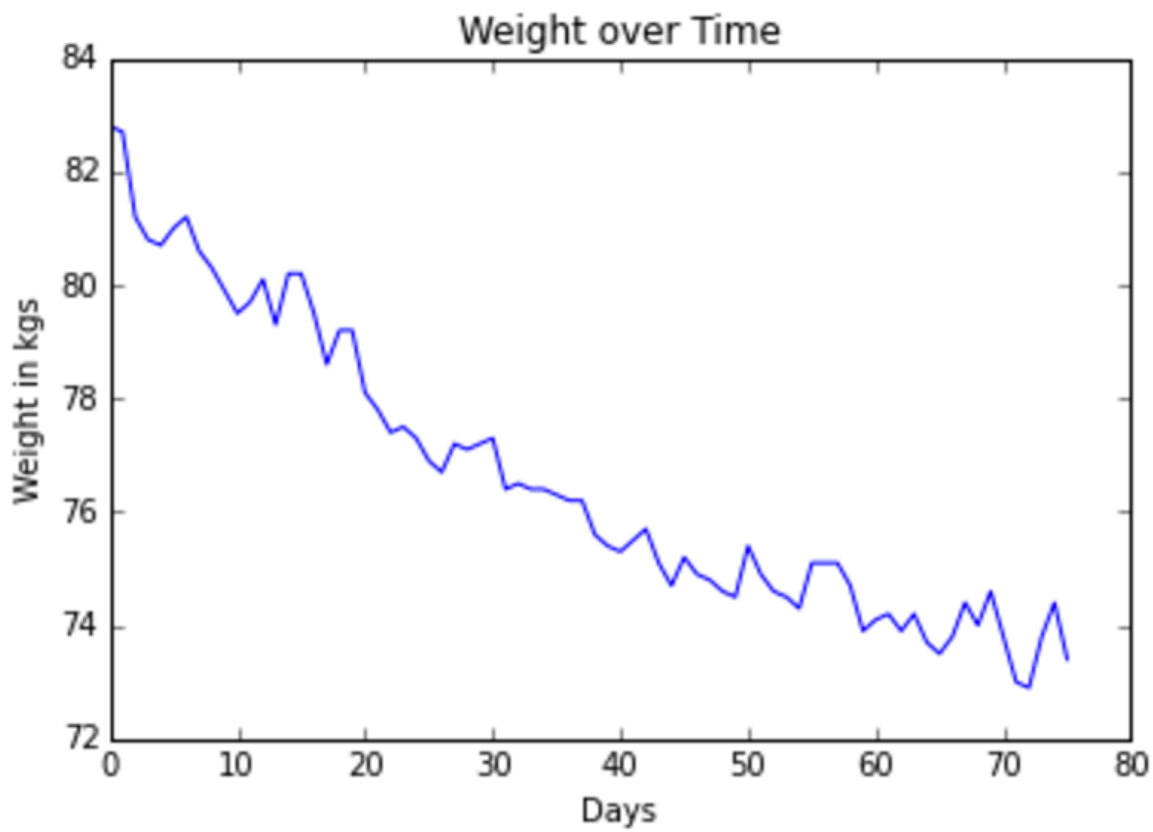
specific to time-series



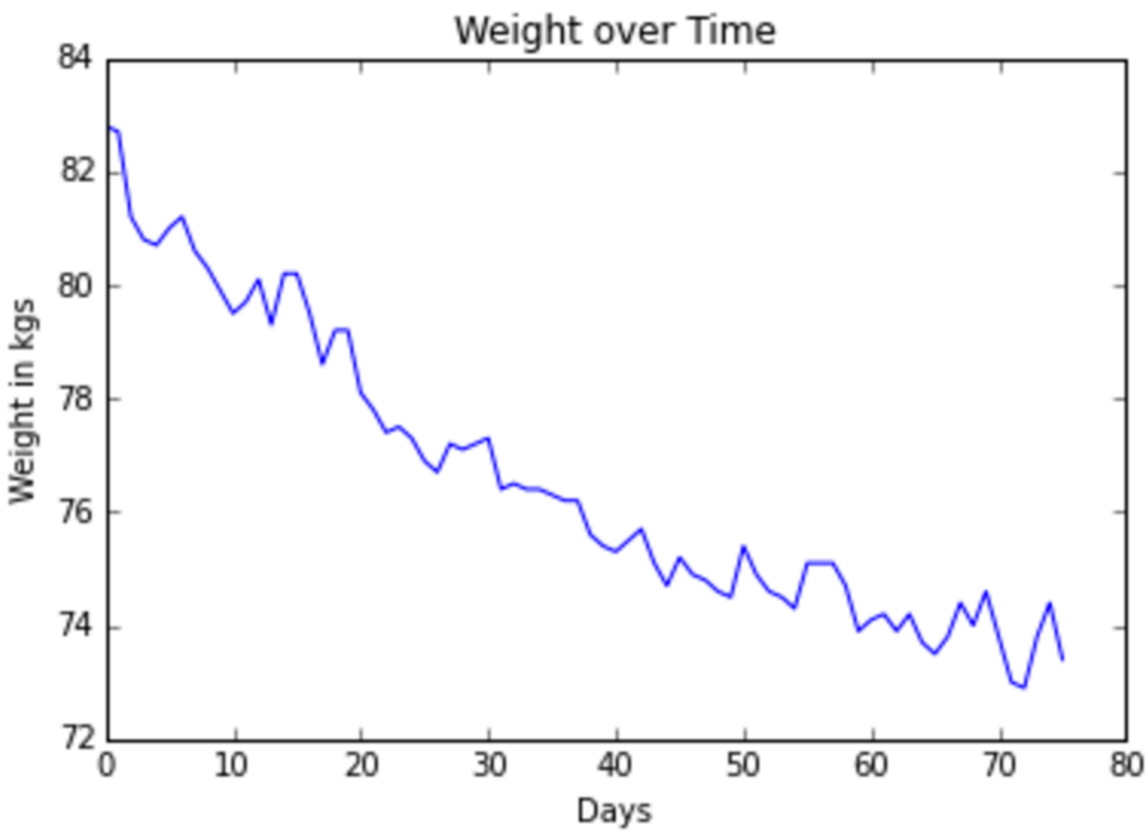
On a day to day basis, weight fluctuates up and down.
On a month to month basis, it is going down.



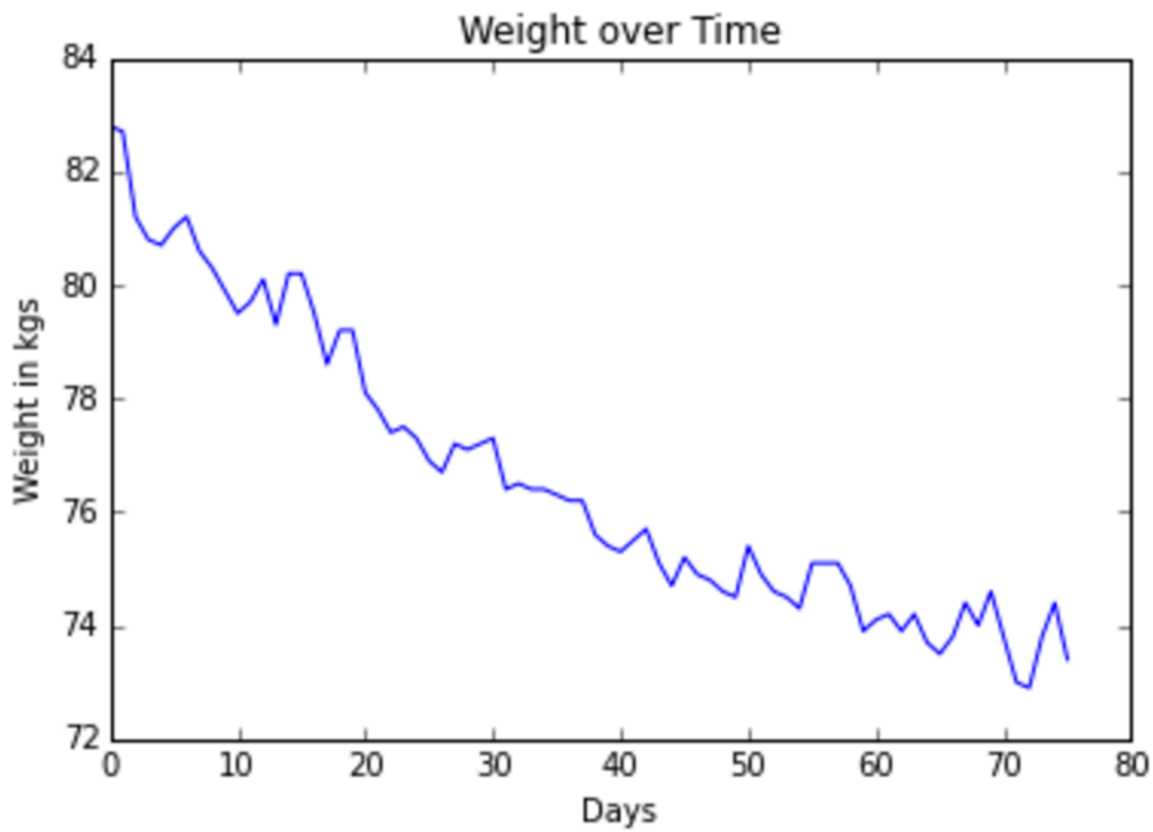
On a day to day basis, weight fluctuates up and down.
On a month to month basis, it is going down.
If we can make solid assumptions about what is noise and what is a trend, we can make the modeling problem easier



Assumption: Day to day fluctuations are "noise". What that really means that we don't aim to build a model that can predict such daily changes. We want to model longer-time-scale trends.

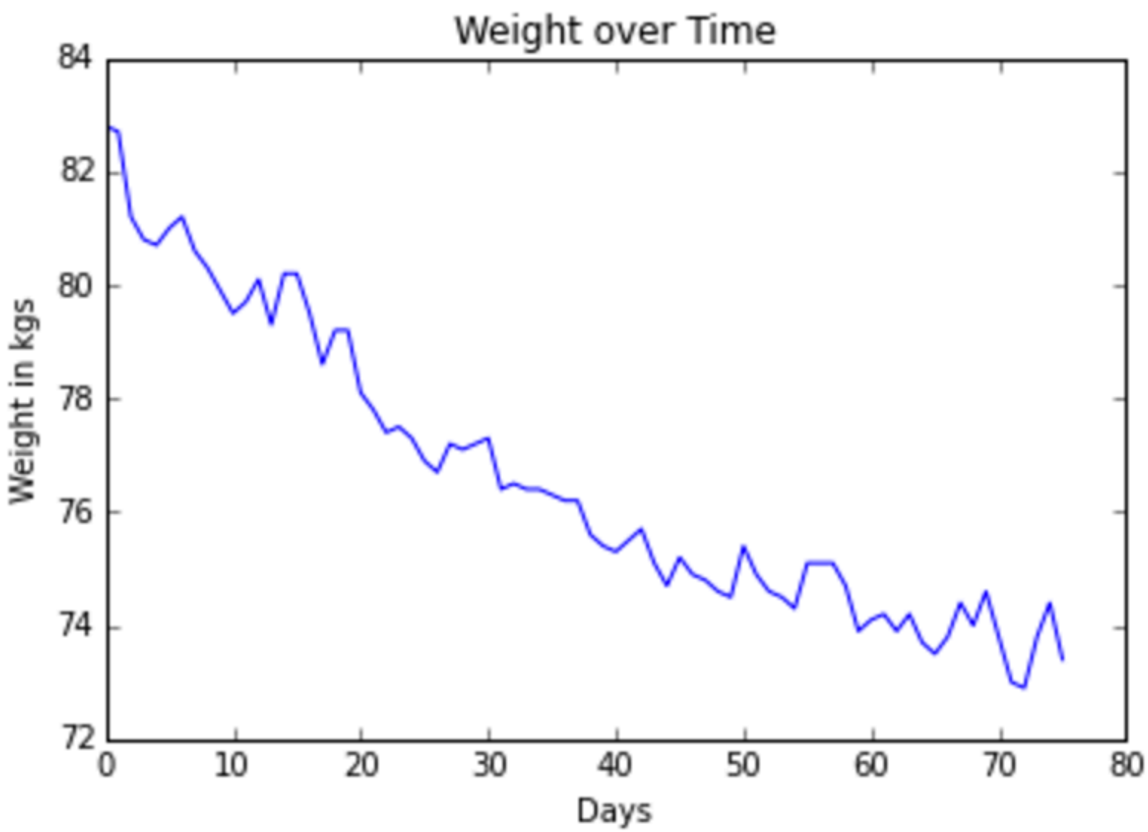


Important: We are not making a claim about what we think is "really" happening underneath.
We are declaring what we choose to model.
As long as you don't forget these decisions when you use the model for predictions, and stay tru to them, they are fine.



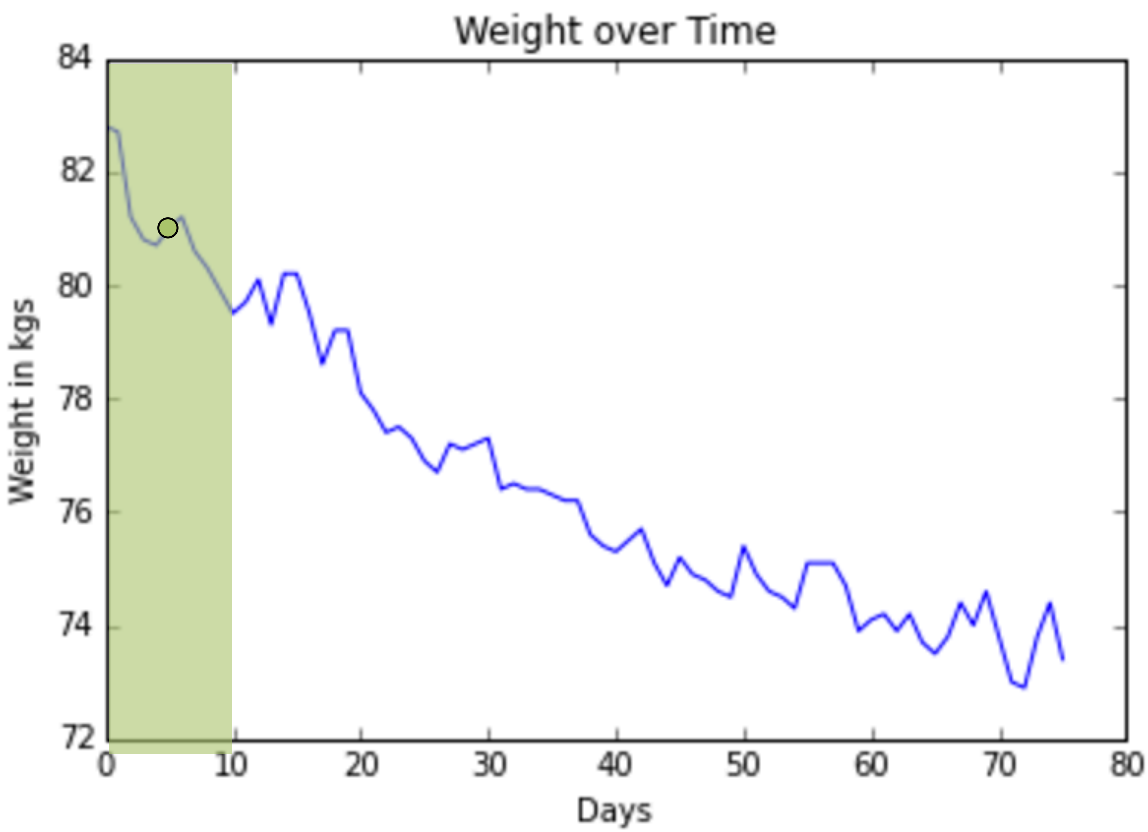
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



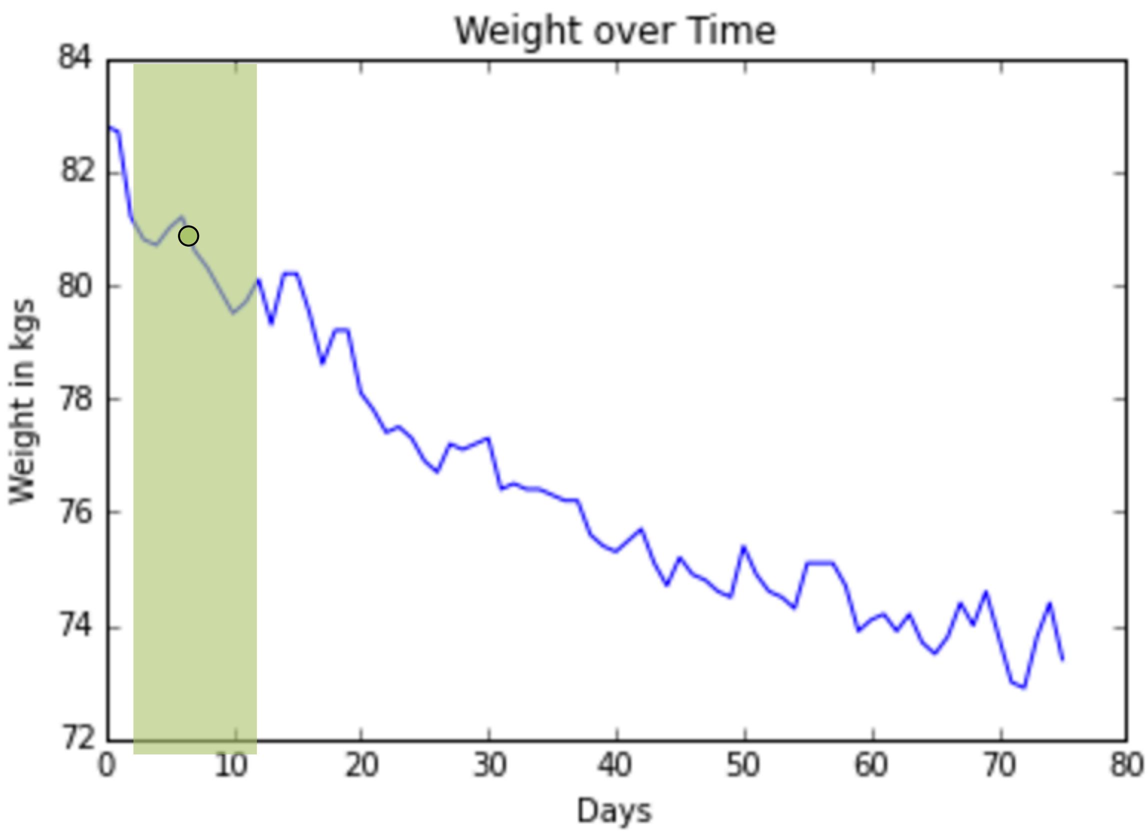
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



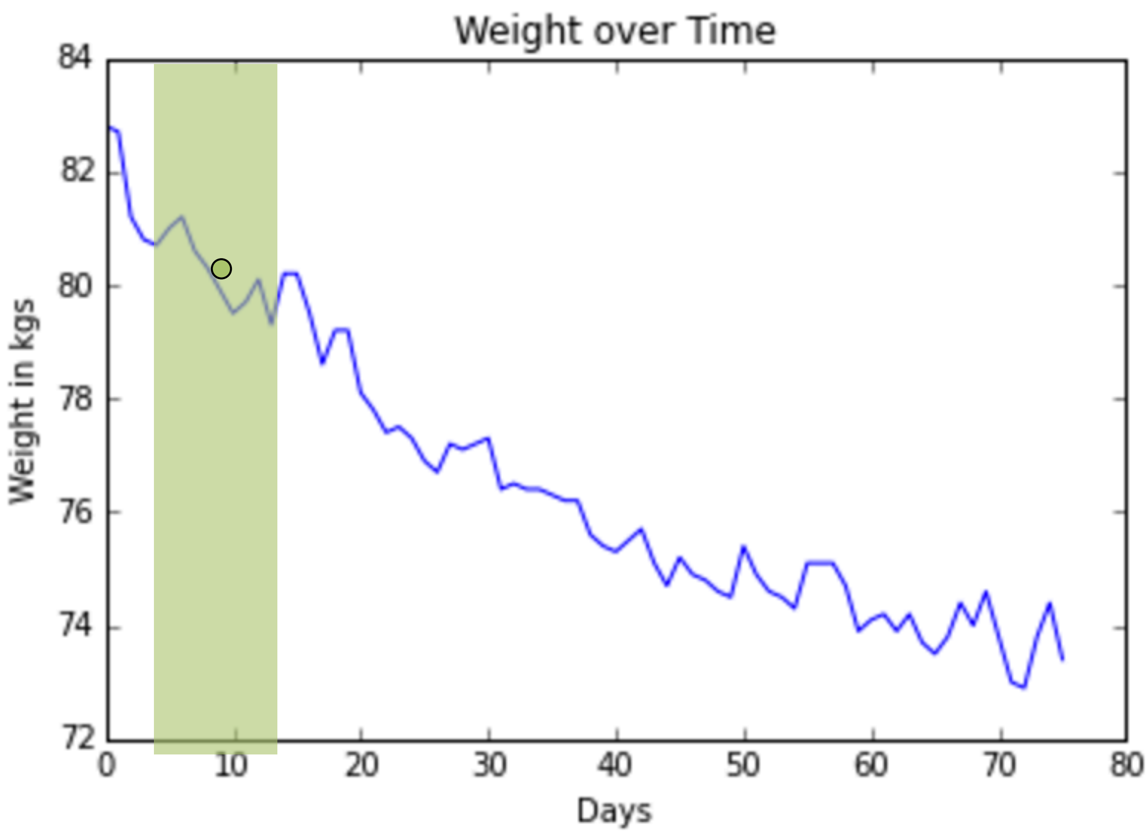
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



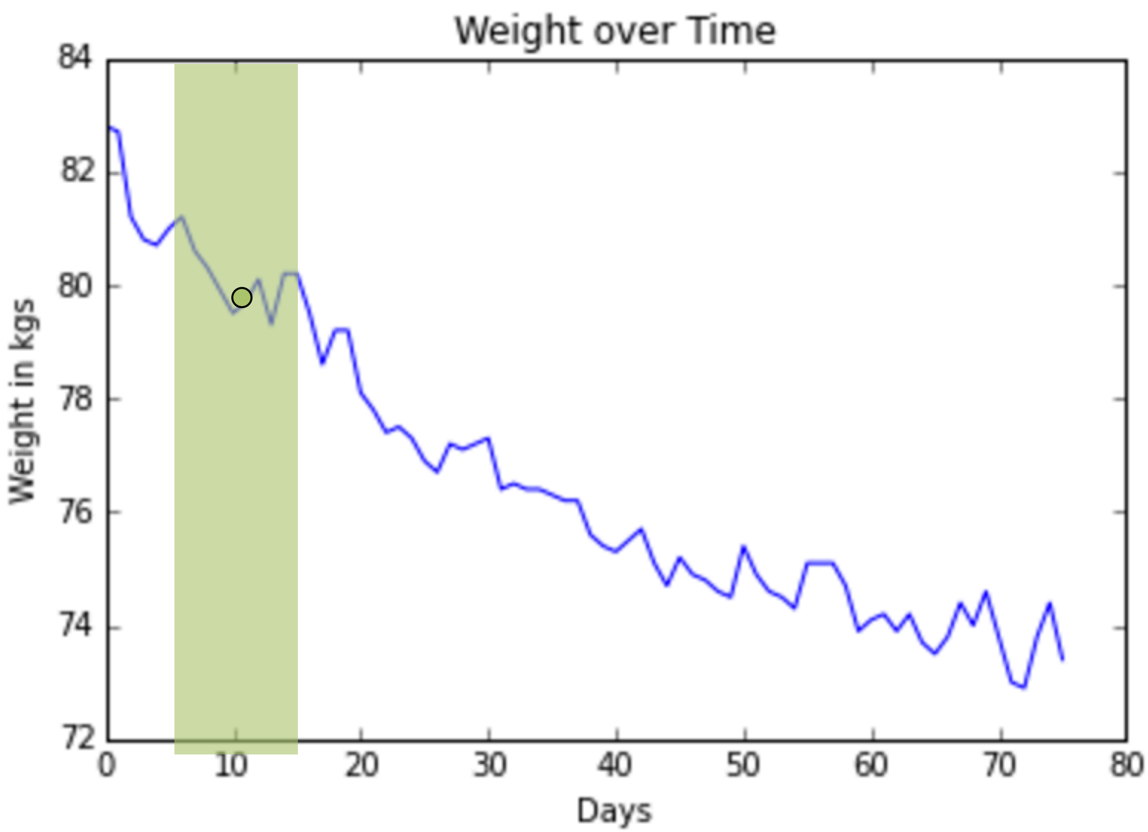
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



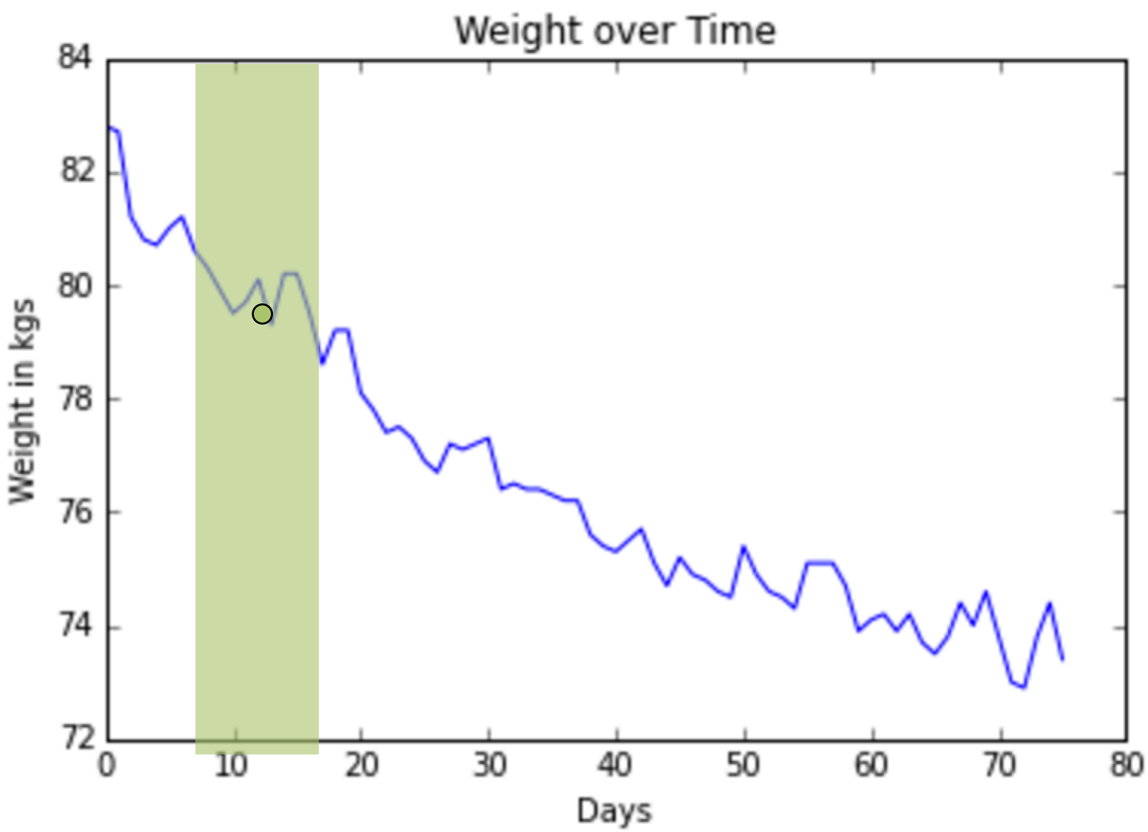
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



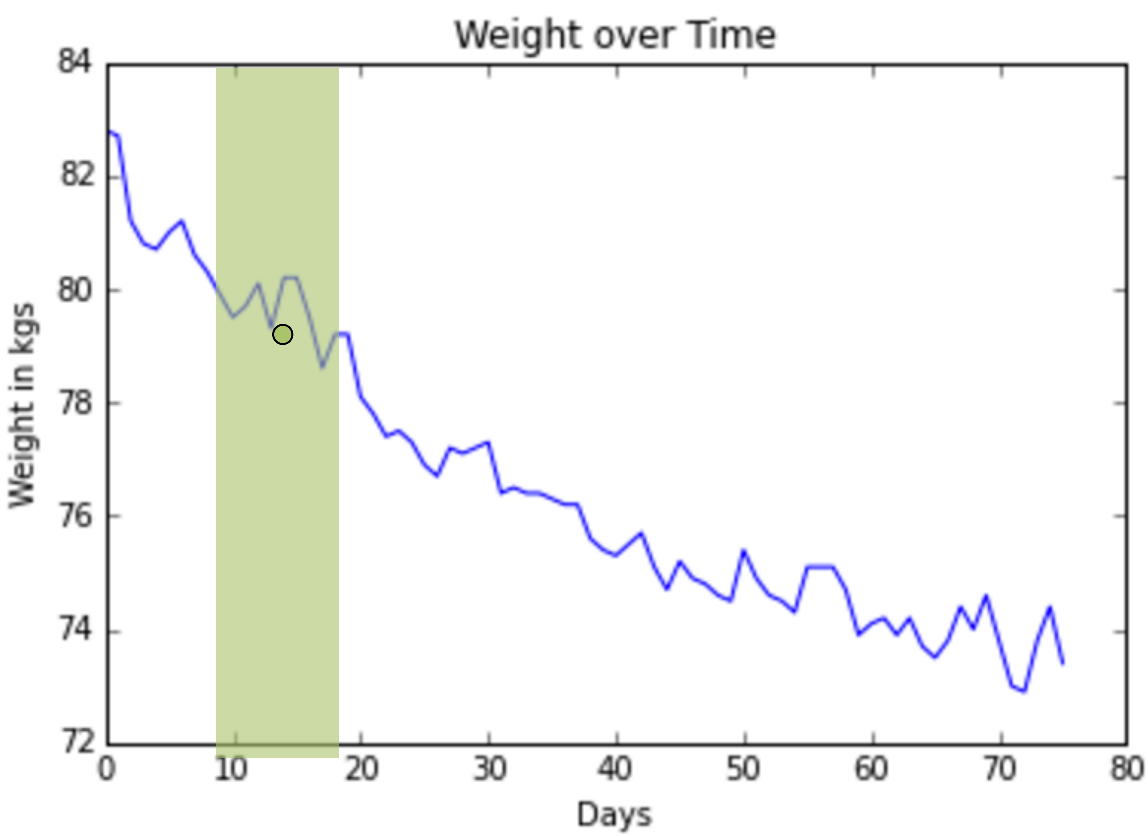
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



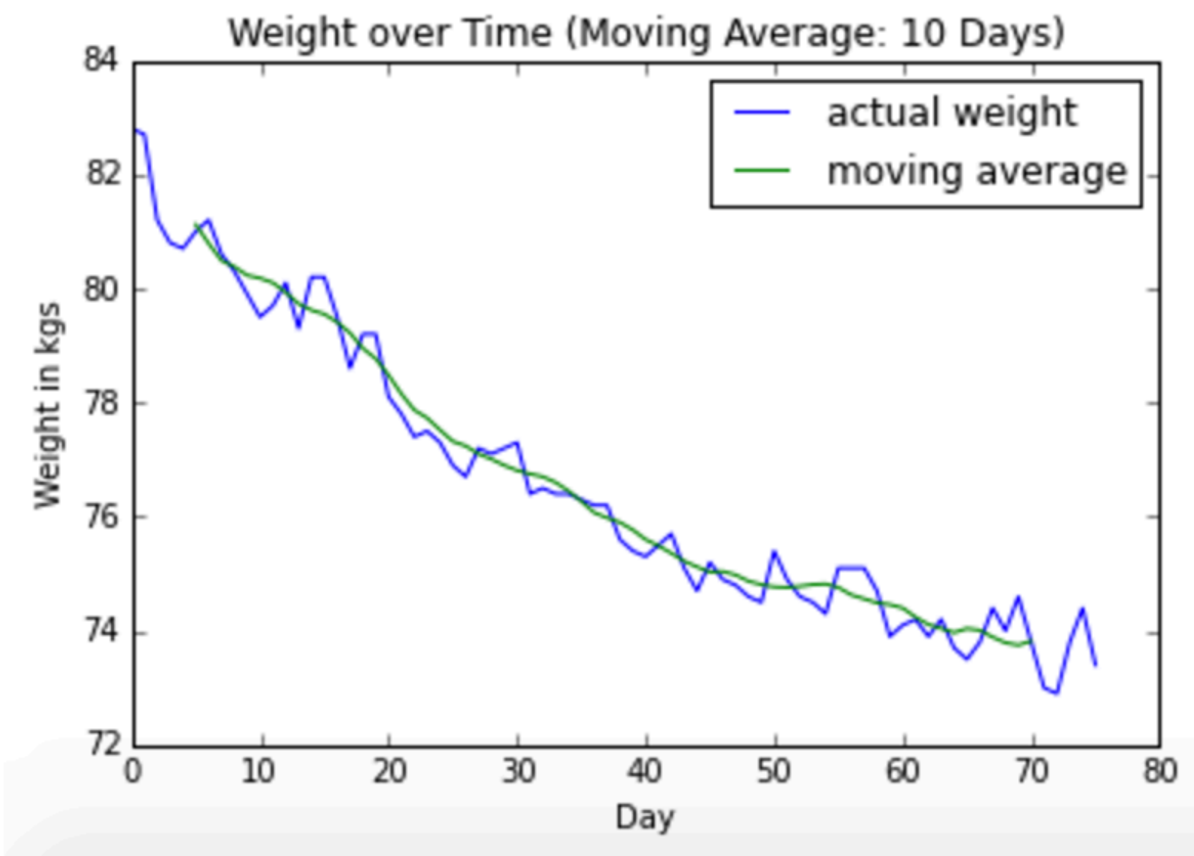
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



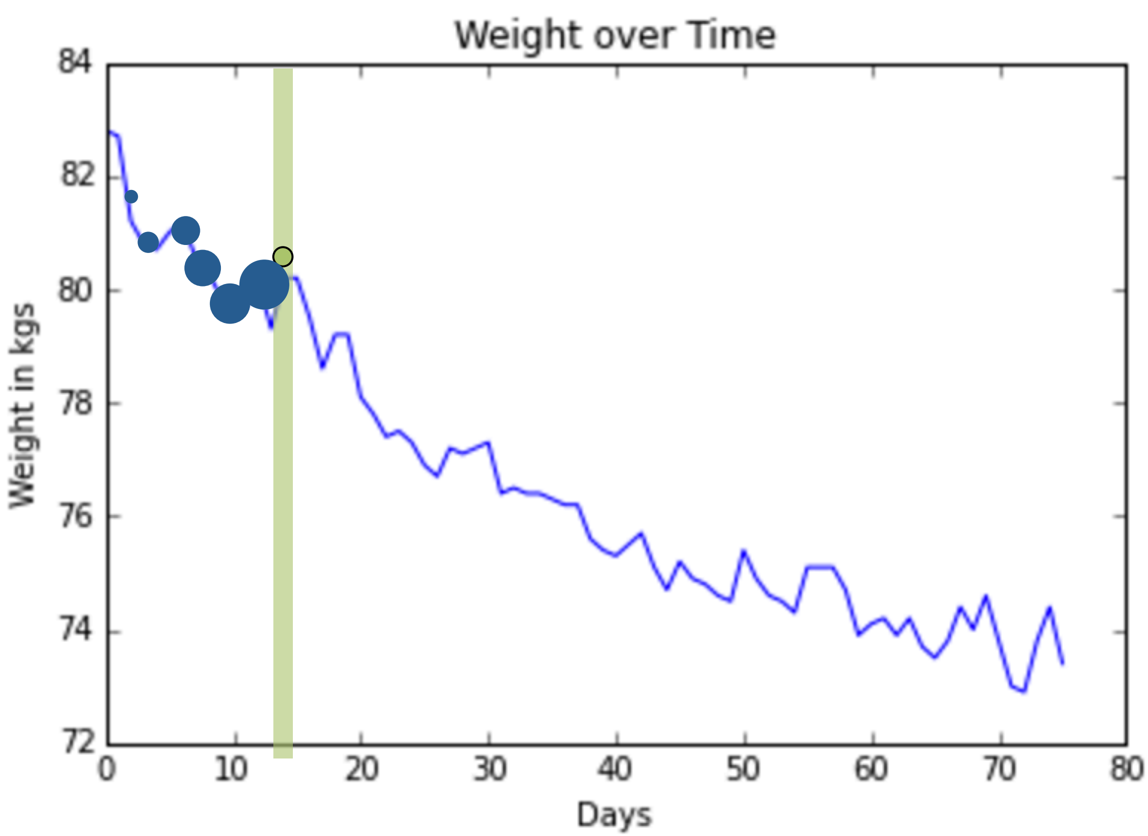
Assumption: Day to day fluctuations are "noise".

With this assumption I can remove some of that noise with a moving average window.



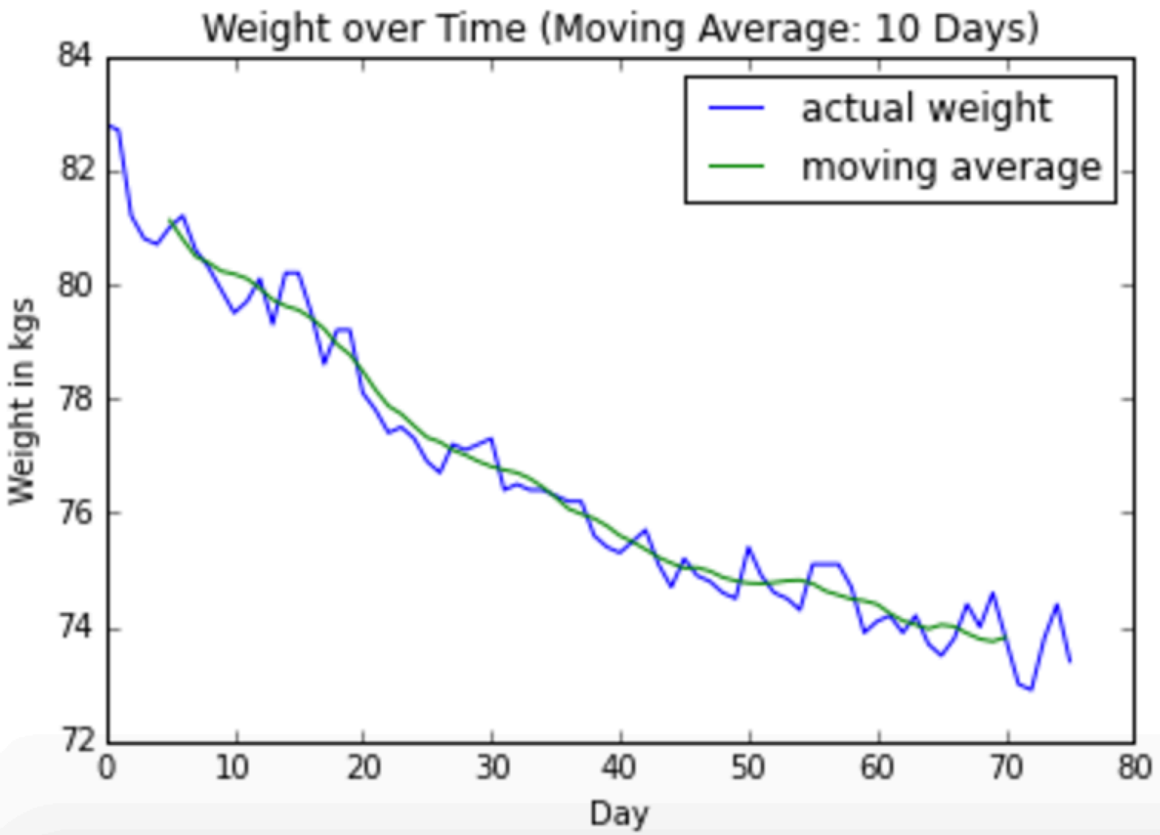
Exponential Moving Average

Instead of weighing each point in the window the same amount, average all past points, but decrease the weight exponentially as we go further back.



Preprocessing: Moving Average

Preprocessing can help avoid overfitting to noise.
Always be mindful of the assumptions you made.



Warning!

Moving Average Smoothing

is <u>**not**</u>

Moving Average Models (MA)

What we covered is a preprocessing technique in modeling time series.

Moving Average Models are (just like AR models) a framework to model time series.
They are inaccurately named (they don't involve averages).
We will cover them briefly in the future.