

Tips and Tricks for Producing Time-Series Cohort Data

Nate Derby

Stakana Analytics
Seattle, WA

Golden Horseshoe SAS Users Group
10/26/18

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date.**
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer *relative to that specific date.*
- Lots of programming, but *these tips and tricks let the programs do the work for us.*

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date**.
- Lots of programming, but **these tips and tricks let the programs do the work for us.**

Introduction

- *Time-Series Cohort* = group of people defined by some time-related event.
 - *All customers who opened their first accounts in 1/18.*
- Goal: track people over time, compare to other cohorts.
 - *All customers who opened their first accounts in 2/18, 3/18,*
- We track activity relative to the event in question.
 - *Activity in the 1st/2nd/3rd month after getting a first account.*
- We find the date each customer started, then collect data for that customer **relative to that specific date**.
- Lots of programming, but **these tips and tricks let the programs do the work for us**.

Use Macros

- We need to apply *the exact same process* to each cohort.
- This is what macros were made for!

%collectCohortData Macro

```
%MACRO collectCohortData( cohortDate );  
  
    [CODE FOR COLLECTING DATA FOR THIS COHORT]  
  
%MEND collectCohortData;
```

Use Macros

- We need to apply *the exact same process* to each cohort.
- This is what macros were made for!

%collectCohortData Macro

```
%MACRO collectCohortData( cohortDate );  
  
    [CODE FOR COLLECTING DATA FOR THIS COHORT]  
  
%MEND collectCohortData;
```

Use Macros

- We need to apply *the exact same process* to each cohort.
- This is what macros were made for!

%collectCohortData Macro

```
%MACRO collectCohortData( cohortDate );  
  
    [CODE FOR COLLECTING DATA FOR THIS COHORT]  
  
%MEND collectCohortData;
```

Use Macros

- How we would collect data for January and February 2018:

```
%collectCohortData( 18m1 )  
%collectCohortData( 18m2 )
```

- Using `YYmM` for month `M` and year `YY` makes it easy to calculate the start date of our cohort.

Months

```
%LET month = %SCAN( &cohortDate, 2, m );  
%LET year = %EVAL( 2000 + %SCAN( &cohortDate, 1, m ) );  
%LET cohortStart = %SYSFUNC( MDY( &month, 1, &year ) );
```

- Above should all be *local* macro variables.

Use Macros

- How we would collect data for January and February 2018:

```
%collectCohortData( 18m1 )  
%collectCohortData( 18m2 )
```

- Using `YYmM` for month `M` and year `YY` makes it easy to calculate the start date of our cohort.

Months

```
%LET month = %SCAN( &cohortDate, 2, m );  
%LET year = %EVAL( 2000 + %SCAN( &cohortDate, 1, m ) );  
%LET cohortStart = %SYSFUNC( MDY( &month, 1, &year ) );
```

- Above should all be *local* macro variables.

Use Macros

- How we would collect data for January and February 2018:

```
%collectCohortData( 18m1 )  
%collectCohortData( 18m2 )
```

- Using `YYmM` for month `M` and year `YY` makes it easy to calculate the start date of our cohort.

Months

```
%LET month = %SCAN( &cohortDate, 2, m );  
%LET year = %EVAL( 2000 + %SCAN( &cohortDate, 1, m ) );  
%LET cohortStart = %SYSFUNC( MDY( &month, 1, &year ) );
```

- Above should all be *local* macro variables.

Divide the Code into Macro Units

- Divide our code into macro units that describe a specific function:

```
%collectData;  
%graphData;  
%makeForecasts;  
%graphForecasts;
```

- For example:

%collectData Macro

```
%MACRO collectData;  
  
%collectCohortData( 18m1 );  
%collectCohortData( 18m2 );  
%collectCohortData( 18m3 );  
%collectCohortData( 18m4 );  
  
%MEND collectData;
```

Divide the Code into Macro Units

- Divide our code into macro units that describe a specific function:

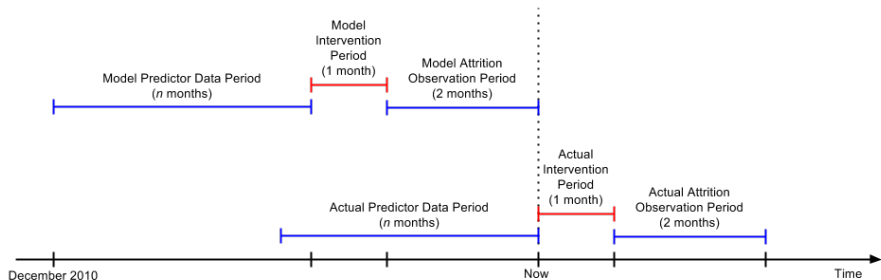
```
%collectData;  
%graphData;  
%makeForecasts;  
%graphForecasts;
```

- For example:

%collectData Macro

```
%MACRO collectData;  
  
%collectCohortData( 18m1 );  
%collectCohortData( 18m2 );  
%collectCohortData( 18m3 );  
%collectCohortData( 18m4 );  
  
%MEND collectData;
```

Define Dates in Terms of Specific Dates



Define Dates in Terms of Specific Dates

- We *never* want to hard code our dates!

%prepareData Macro

```
%MACRO prepareData( dataSet );  
  
%IF &dataSet = modeling %THEN %DO;  
    %LET predictorStartDate = &startDate;  
    %LET predictorEndDate = %EVAL( &now - 84 );  
%END;  
  
%ELSE %IF &dataSet = scoring %THEN %DO;  
    %LET predictorStartDate = %EVAL( &startDate + 84 );  
    %LET predictorEndDate = &now;  
%END;  
  
...  
  
%MEND prepareData;
```

Define Dates in Terms of Specific Dates

- We *never* want to hard code our dates!

%prepareData Macro

```
%MACRO prepareData( dataSet );  
  
  %IF &dataSet = modeling %THEN %DO;  
    %LET predictorStartDate = &startDate;  
    %LET predictorEndDate = %EVAL( &now - 84 );  
    %END;  
  %ELSE %IF &dataSet = scoring %THEN %DO;  
    %LET predictorStartDate = %EVAL( &startDate + 84 );  
    %LET predictorEndDate = &now;  
    %END;  
  
  ...  
  
%MEND prepareData;
```

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Define One Month as Four Weeks

Months are messy.

- Inconsistent number of days, almost never multiple of 7.
- Some months have more Saturdays (or other days) than others.
 - Months with 5 Saturdays typically have higher retail sales.
 - Months with 5 Sundays typically have higher church attendance.
 - ...
- This is a *weekly seasonality effect*.
- Avoid all this by **defining a month as four weeks**.
 - Holidays are still a problem, but lesser so.

Caveats

- This only applies for *generic* months.
- Inconsistency issues for 10+ months, since 10 months \neq 40 weeks.
- Explicitly mention that we're doing this.

Caveats

- This only applies for *generic* months.
- Inconsistency issues for 10+ months, since 10 months \neq 40 weeks.
- Explicitly mention that we're doing this.

Caveats

- This only applies for *generic* months.
- Inconsistency issues for 10+ months, since 10 months \neq 40 weeks.
- Explicitly mention that we're doing this.

Caveats

- This only applies for *generic* months.
- Inconsistency issues for 10+ months, since 10 months \neq 40 weeks.
- Explicitly mention that we're doing this.

Use Date/Time Functions and Formats

- INTCK: Calculates the number of intervals between two date/datetime values.
- INTNX: Calculates the date/datetime of the start/middle/end of the interval a specified number of intervals from the interval that contains a given date/datetime.
- Calculate the last day of the month:

```
%LET cohortEnd =  
    %SYSFUNC( INTNX( month, &cohortStart, 0, end ) );
```

Use Date/Time Functions and Formats

- **INTCK:** Calculates the number of intervals between two date/datetime values.
- **INTNX:** Calculates the date/datetime of the start/middle/end of the interval a specified number of intervals from the interval that contains a given date/datetime.
- Calculate the last day of the month:

```
%LET cohortEnd =  
    %SYSFUNC( INTNX( month, &cohortStart, 0, end ) );
```

Use Date/Time Functions and Formats

- **INTCK:** Calculates the number of intervals between two date/datetime values.
- **INTNX:** Calculates the date/datetime of the start/middle/end of the interval a specified number of intervals from the interval that contains a given date/datetime.
- Calculate the last day of the month:

```
%LET cohortEnd =  
    %SYSFUNC( INTNX( month, &cohortStart, 0, end ) );
```

Use Date/Time Functions and Formats

- **INTCK:** Calculates the number of intervals between two date/datetime values.
- **INTNX:** Calculates the date/datetime of the start/middle/end of the interval a specified number of intervals from the interval that contains a given date/datetime.
- Calculate the last day of the month:

```
%LET cohortEnd =  
    %SYSFUNC( INTNX( month, &cohortStart, 0, end ) );
```

Empirically Calculate End Dates

- Don't assume what the end date is.
- The end date has the most important data.

%prepareData Macro

```
PROC SQL NOPRINT;  
  SELECT MAX( lastTreatmentDate )  
    INTO :end1  
  FROM patientData;  
  SELECT MAX( treatmentDate )  
    INTO :end2  
  FROM treatments;  
QUIT;  
  
%LET end = %SYSFUNC( MIN( &end1, &end2 ) );
```

Empirically Calculate End Dates

- Don't assume what the end date is.
- **The end date has the most important data.**

%prepareData Macro

```
PROC SQL NOPRINT;  
  SELECT MAX( lastTreatmentDate )  
    INTO :end1  
  FROM patientData;  
  SELECT MAX( treatmentDate )  
    INTO :end2  
  FROM treatments;  
QUIT;  
  
%LET end = %SYSFUNC( MIN( &end1, &end2 ) );
```

Empirically Calculate End Dates

- Don't assume what the end date is.
- **The end date has the most important data.**

%prepareData Macro

```
PROC SQL NOPRINT;  
  SELECT MAX( lastTreatmentDate )  
    INTO :end1  
  FROM patientData;  
  SELECT MAX( treatmentDate )  
    INTO :end2  
  FROM treatments;  
QUIT;  
  
%LET end = %SYSFUNC( MIN( &end1, &end2 ) );
```

Further Resources

Too many to list—see the paper at nderby.org/publications

Nate Derby: nate@stakana.com