Pivotal Beamer Template

Example Presentation

Ronert Obst, Data Scientist

January 2, 2014

Outline

1. First Section
2. Second Section
3. Some Code

First Section

Buzzwords

- Big data
  1. Next generation data science
  2. Elastic deep learning
- Industry 8.0
- IoT 5.0
- More cloud $\rightarrow$ more winning

Two columns

- Considering the aspect ratio is 16:9
- It is probably a good idea to use two columns
- To avoid really long lines
- Typesetting just looks nicer than in PowerPoint
- And it can be versioned!

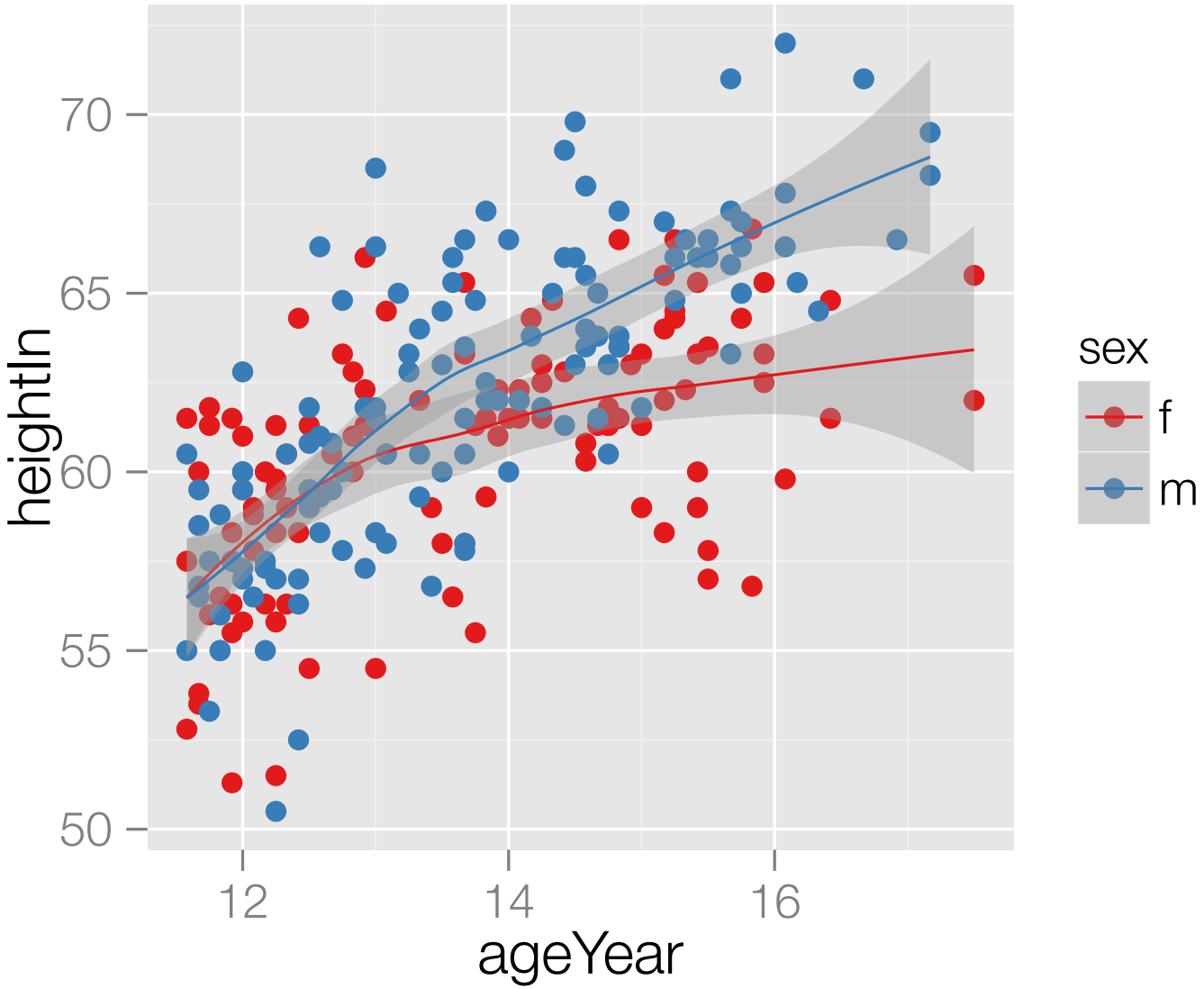
- Some other content on this side
- Big data
- More big data
- Hadoop
- GPDB
- MADlib
- $x + y = z$

Let us create a plot with `ggplot2`

- `ggplot2` is a really nice plotting library
- It can export graphics using TikZ
- Which look really nice in LaTeX

```
1  library(ggplot2)
2  library(gcookbook)
3  sps <- ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=sex))
4  + geom_point()
5  + scale_colour_brewer(palette="Set1")
6  sps + geom_smooth()
```

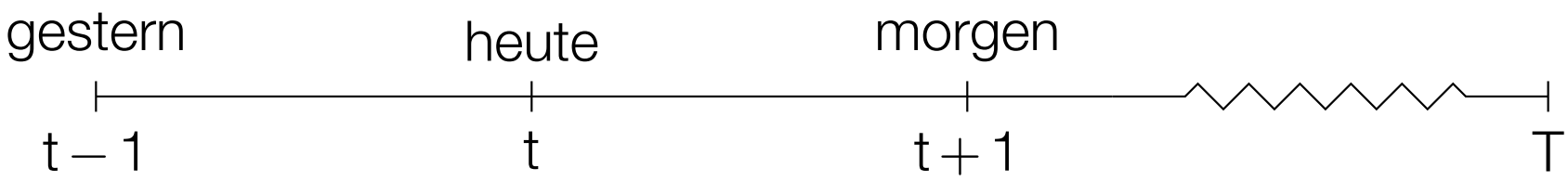
The plot using TikZ



Pivotal

Second Section

A timeline using TikZ



Explaining equations (TEXample.net)

- Coriolis acceleration

$$\vec{a}_p = \vec{a}_o + \frac{{}^b d^2}{dt^2}\vec{r} + 2\vec{w}_{ib} \times \frac{{}^b d}{dt}\vec{r} + \vec{\alpha}_{ib} \times \vec{r} + \vec{w}_{ib} \times (\vec{w}_{ib} \times \vec{r})$$

Explaining equations (TEXample.net)

- Coriolis acceleration

$$\vec{a}_p = \vec{a}_o + \frac{{}^b d^2}{dt^2}\vec{r} + 2\vec{w}_{ib} \times \frac{{}^b d}{dt}\vec{r} + \vec{\alpha}_{ib} \times \vec{r} + \vec{w}_{ib} \times (\vec{w}_{ib} \times \vec{r})$$

- Transversal acceleration

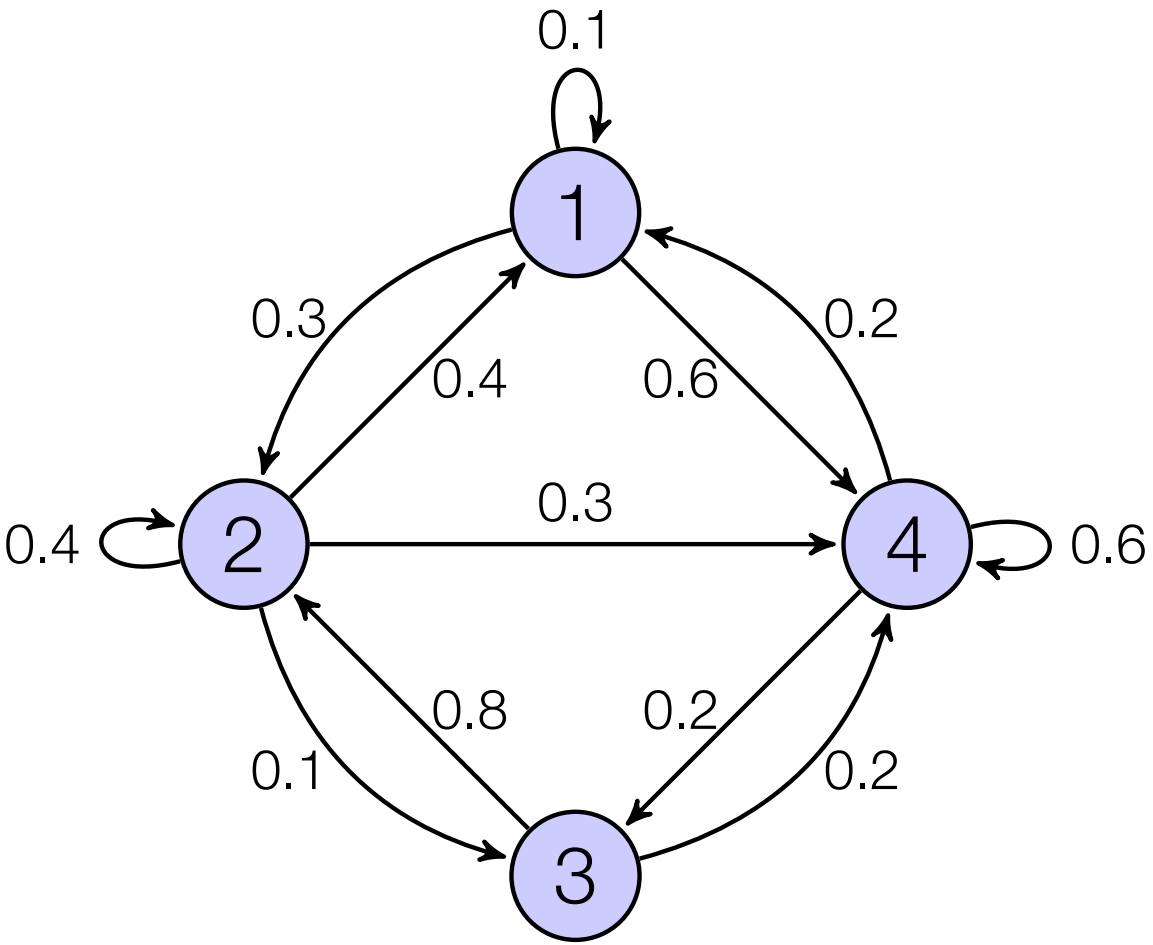
Explaining equations (TEXample.net)

- Coriolis acceleration

$$\vec{a}_p = \vec{a}_o + \frac{{}^b d^2}{dt^2}\vec{r} + 2\vec{w}_{ib} \times \frac{{}^b d}{dt}\vec{r} + \vec{\alpha}_{ib} \times \vec{r} + \vec{w}_{ib} \times (\vec{w}_{ib} \times \vec{r})$$

- Transversal acceleration
- Centripetal acceleration

A graph (TEXample.net)



Pivotal

Some Code

Python

- Pygments via `minted` has syntax highlighting for all major languages

```
1   # List comprehensions
2   num = [1, 4, -5, 10, -7, 2, 3, -1]
3   filtered_and_squared = [ x**2 for x in num if x > 0]
4   print filtered_and_squared
5
6   # Generators
7   num = [1, 4, -5, 10, -7, 2, 3, -1]
8   filtered_and_squared = ( x**2 for x in num if x > 0 )
9   print filtered_and_squared
10
11  for item in filtered_and_squared:
12      print item
```

New dplyr package

```
1   library(dplyr)
2   # Built in data frame
3   head(hflights)
4
5   # Coerce to data table
6   hflights_dt <- tbl_dt(hflights)
7
8   # Caches data in local SQLite db
9   hflights_db1 <- tbl(hflights_sqlite(), "hflights")
10
11  # Caches data in local postgres db
12  hflights_db2 <- tbl(hflights_postgres(), "hflights")
13
14  carriers_df  <- group_by(hflights, UniqueCarrier)
15  carriers_dt  <- group_by(hflights_dt, UniqueCarrier)
16  carriers_db1 <- group_by(hflights_db1, UniqueCarrier)
17  carriers_db2 <- group_by(hflights_db2, UniqueCarrier)
```

SQL

```
1   select public.diabetes.diabetes,
2       madlib.logregr_predict( array[1, times_pregnant,
3               plasma,
4               diastolic_blood_pressure,
5               triceps_skin_fold_thickness,
6               serum_insulin,
7               bmi,
8               diabetes_pedigree,
9               age],
10          public.diabetes_log_reg_ronert.coef
11          )::integer as predict
12  from public.diabetes, public.diabetes_log_reg_ronert limit 100;
```