“Best data science job interview resource”

DataScienceBootcamps.com

A COLLECTION OF DATA SCIENCE TAKE-HOME CHALLENGES

GIULIO PALOMBO

Intro

Things to keep in mind while doing a take-home challenge:

- Unless otherwise specified, you want to use R or Python. Comment the code as much as possible. They are also evaluating how clear is your code. Also, anyone should be able to understand the conclusions of your take-home even if they are not familiar with the language you used.
- Check the data. Never assume data is right. Always check data reliability and, if you find that some data doesn't make sense, clean it. This is also a big part of a data scientist job. There are companies that send take-homes that are only about identifying everything that is wrong with the data!
- Take-home challenges are usually fairly open ended. Play to your strengths: this could mean spending more time on visualization, machine learning, product ideas, or business insights depending on your skills.
- Don't make the solution over complicated. Focus on a few things and make sure the overall message is clear and consistent.
- Along the same lines: when you have to build a machine learning model, don't spend days optimizing its accuracy (this is not Kaggle, it is real world). Pick a model, explain why you picked that model and use parameters that make sense. You can then say what you would do if you had 1 more week to optimize it.
- Focus on the business impact that your work could have. How would the company benefit from your analysis? What would you suggest as a next step?
- If you find anything interesting in the data, by any means show it even if it is not related to the questions. If you find some info in the data that not even the hiring manager knows is there, you will pass the take-home for sure. After all, that's exactly why they will be hiring you: to discover things they don't know yet.
- A take-home challenge is rarely the place where over emphasizing your theoretical knowledge (unless specifically required in a question).
- Before extracting insights from a model, make sure your model predicts well. If your model doesn't predict well, its coefficients, splits, variable importance, etc. are totally irrelevant.
- The solutions provided in this book are in R. Obviously, Python would be perfectly fine too. Solutions are by no means exhaustive, and they simply show one (of the many) possible approaches.

Conversion Rate

Goal

Optimizing conversion rate is likely the most common work of a data scientist, and rightfully so.

The data revolution has a lot to do with the fact that now we are able to collect all sorts of data about people who buy something on our site as well as people who don't. This gives us a tremendous opportunity to understand what's working well (and potentially scale it even further) and what's not working well (and fix it).

The goal of this challenge is to build a model that predicts conversion rate and, based on the model, come up with ideas to improve revenue.

This challenge is significantly easier than all others in this collection. There are no dates, no tables to join, no feature engineering required, and the problem is really straightforward. Therefore, it is a great starting point to get familiar with data science take-home challenges.

You should not move to the other challenges until you fully understand this one.

Challenge Description

We have data about users who hit our site: whether they converted or not as well as some of their characteristics such as their country, the marketing channel, their age, whether they are repeat users and the number of pages visited during that session (as a proxy for site activity/time spent on site).

Your project is to:

- Predict conversion rate
- Come up with recommendations for the product team and the marketing team to improve conversion rate

Data

We have 1 table downloadable by clicking **here**.

```
The table is "conversion_data". It has information about signed-in users
during one session. Each row is a user session.
```

Columns:

- **country** : user country based on the IP address
- **age** : user age. Self-reported at sign-in step
- **new_user** : whether the user created the account during this session or had already an account and simply came back to the site
- **source** : marketing channel source
    - Ads: came to the site by clicking on an advertisement
    - Seo: came to the site by clicking on search results
    - Direct: came to the site by directly typing the URL on the browser
- **total_pages_visited**: number of total pages visited during the session. This is a proxy for time spent on site and engagement during the session.
- **converted**: this is our label. 1 means they converted within the session, 0 means they left without buying anything. The company goal is to increase conversion rate: # conversions / total sessions.

Example

```
Let's now check the characteristics of the user in the first row.
```

head(conversion_data, 1)

Field	Value	Description
country	UK	the user is based in the UK
age	25	the user is 25 yr old
new_user	1	she created her account during this session
source	Ads	she came to the site by clicking on an ad
total_pages_visited	1	she visited just 1 page during that session
converted	0	this user did not buy during this session. These are the users whose behavior we want to change!

Spanish Translation A/B Test

Goal

A/B tests play a huge role in website optimization. Analyzing A/B tests data is a very important data scientist responsibility. Especially, data scientists have to make sure that results are reliable, trustworthy, and conclusions can be drawn.

Furthermore, companies often run tens, if not hundreds, of A/B tests at the same time. Manually analyzing all of them would require lot of time and people. Therefore, it is common practice to look at the typical A/B test analysis steps and try to automate as much as possible. This frees up time for the data scientists to work on more high level topics.

In this challenge, you will have to analyze results from an A/B test. Also, you will be asked to design an algorithm to automate some steps.

Challenge Description

Company XYZ is a worldwide e-commerce site with localized versions of the site.

A data scientist at XYZ noticed that Spain-based users have a much higher conversion rate than any other Spanish-speaking country. She therefore went and talked to the international team in charge of Spain And LatAm to see if they had any ideas about why that was happening.

Spain and LatAm country manager suggested that one reason could be translation. All Spanish-speaking countries had the same translation of the site which was written by a Spaniard. They agreed to try a test where each country would have its one translation written by a local. That is, Argentinian users would see a translation written by an Argentinian, Mexican users by a Mexican and so on. Obviously, nothing would change for users from Spain.

After they run the test however, they are really surprised cause the test is negative. I.e., it appears that the non-localized translation was doing better!

You are asked to:

- Confirm that the test is actually negative. That is, it appears that the old version of the site with just one translation across Spain and LatAm performs better
- Explain why that might be happening. Are the localized translations really worse?

- If you identified what was wrong, design an algorithm that would return FALSE if the same problem is happening in the future and TRUE if everything is good and the results can be trusted.

Data

We have 2 table downloadable by clicking **here**.

The 2 tables are:

```
"test_table" - general information about the test results
```

Columns:

- **user_id** : the id of the user. Unique by user. Can be joined to user id in the other table. For each user, we just check whether conversion happens the first time they land on the site since the test started.
- **date** : when they came to the site for the first time since the test started
- **source** : marketing channel: *Ads, SEO, Direct* . *Direct* means everything except for ads and SEO. Such as directly typing site URL on the browser, downloading the app w/o coming from SEO or Ads, referral friend, etc.
- **device** : device used by the user. It can be mobile or web
- **browser_language** : in browser or app settings, the language chosen by the user. It can be *EN, ES, Other* (Other means any language except for English and Spanish)
- **ads_channel** : if marketing channel is ads, this is the site where the ad was displayed. It can be: *Google, Facebook, Bing, Yahoo ,Other*. If the user didn't come via an ad, this field is NA
- **browser** : user browser. It can be: *IE, Chrome, Android_App, FireFox, Iphone_App, Safari, Opera*
- **conversion** : whether the user converted (1) or not (0). This is our label. A test is considered successful if it increases the proportion of users who convert.
- **test** : users are randomly split into test (1) and control (0). Test users see the new translation and control the old one. For Spain-based users, this is obviously always 0 since there is no change there.

```
"user_table" - some information about the user
```

Columns:

- **user_id** : the id of the user. It can be joined to user id in the other table
- **sex** : user sex: Male or Female
- **age** : user age (self-reported)
- **country** : user country based on ip address

Example

```
Let's check one user who went through the test
```

head(test_table, 1)

Column Name	**Value**	**Description**
user_id	315281	this is id of the user
date	2015-12-03	he came to the site on Dec, 3 for the first time since the test started
source	Direct	his marketing channel was direct. No SEO or Ads.
device	Web	he visited the site using "web" (i.e. laptop/desktop, but not mobile)
browser_language	ES	his browser language settings are Spanish
ads_channel	NA	didn't come via an ad, so this has to be NA
browser	IE	he used Internet Explorer!
conversion	1	he converted
test	0	he was in control. That is, he saw the old translation written by a Spaniard

```
Let's now check the characteristics of that user
```

subset(user_table,user_id == 315281)

Column Name	Value	Description
user_id	315281	same id as in the example above
sex	M	he is a Male
age	32	he is 32 y/o
country	Spain	he is based in Spain. So, in his case, he could have not been in the test no matter what.

Employee Retention

Goal

Employee turn-over is a very costly problem for companies. The cost of replacing an employee if often larger than 100K USD, taking into account the time spent to interview and find a replacement, placement fees, sign-on bonuses and the loss of productivity for several months.

It is only natural then that data science has started being applied to this area. Understanding why and when employees are most likely to leave can lead to actions to improve employee retention as well as planning new hiring in advance. This application of DS is sometimes called *people analytics* or *people data science* (if you see a job title: people data scientist, this is your job).

In this challenge, you have a data set with info about the employees and have to predict when employees are going to quit by understanding the main drivers of employee churn.

Challenge Description

We got employee data from a few companies. We have data about all employees who joined from 2011/01/24 to 2015/12/13. For each employee, we also know if they are still at the company as of 2015/12/13 or they have quit. Beside that, we have general info about the employee, such as avg salary during her tenure, dept, and yrs of experience.

As said above, the goal is to predict employee retention and understand its main drivers. Specifically, you should:

- Assume, for each company, that the headcount starts from zero on 2011/01/23. Estimate employee headcount, for each company, on each day, from 2011/01/24 to 2015/12/13. That is, if by 2012/03/02 2000 people have joined company 1 and 1000 of them have already quit, then company headcount on 2012/03/02 for company 1 would be 1000. **You should create a table with 3 columns: day, employee_headcount, company_id.**
- What are the main factors that drive employee churn? Do they make sense? Explain your findings.
- If you could add to this data set just one variable that could help explain employee churn, what would that be?

Data

We have 1 table downloadable by clicking **here**.

The table is:

```
"employee_retention" - comprehensive information about employees.
```

Columns:

- **employee_id** : id of the employee. Unique by employee per company
- **company_id** : company id.
- **dept** : employee dept
- **seniority** : number of yrs of work experience when hired
- **salary**: avg yearly salary of the employee during her tenure within the company
- **join_date**: when the employee joined the company, it can only be between 2011/01/24 and 2015/12/13
- **quit_date**: when the employee left her job (if she is still employed as of 2015/12/13, this field is NA)

Example

```
Let's now check the characteristics of the employee in first row.
```

head(employee_retention, 1)

Field	**Value**	**Description**
employee_id	13201	unique identifier of the employee
company_id	7	she works for company 7
dept	customer_service	she works in the customer service dept
seniority	28	she had 28 yrs of work experience when she was hired by company 7
salary	89000	her yearly salary is 89K USD
join_date	2014-03-24	she joined company 7 on March, 24 2014
quit_date	2015-10-30	she quit her job at company 7 on Oct, 30 2015. That is, she worked there for ~19 months

Identifying Fraudulent Activities

Goal

E-commerce websites often transact huge amounts of money. And whenever a huge amount of money is moved, there is a high risk of users performing fraudulent activities, e.g. using stolen credit cards, doing money laundry, etc.

Machine Learning really excels at identifying fraudulent activities. Any website where you put your credit card information has a risk team in charge of avoiding frauds via machine learning.

The goal of this challenge is to build a machine learning model that predicts the probability that the first transaction of a new user is fraudulent.

Challenge Description

Company XYZ is an e-commerce site that sells hand-made clothes.

You have to build a model that predicts whether a user has a high probability of using the site to perform some illegal activity or not. This is a super common task for data scientists.

You only have information about the user first transaction on the site and based on that you have to make your classification ("fraud/no fraud").

These are the tasks you are asked to do:

- For each user, determine her country based on the numeric IP address.
- Build a model to predict whether an activity is fraudulent or not. Explain how different assumptions about the cost of false positives vs false negatives would impact the model.
- Your boss is a bit worried about using a model she doesn't understand for something as important as fraud detection. How would you explain her how the model is making the predictions? Not from a mathematical perspective (she couldn't care less about that), but from a user perspective. What kinds of users are more likely to be classified as at risk? What are their characteristics?
- Let's say you now have this model which can be used live to predict in real time if an activity is fraudulent or not. From a product perspective, how would you use it? That is,

what kind of different user experiences would you build based on the model output?

Data

We have 2 table downloadable by clicking **here**.

The 2 tables are:

```
"Fraud_Data" - information about each user first transaction
```

Columns:

- **user_id** : Id of the user. Unique by user
- **signup_time** : the time when the user created her account (GMT time)
- **purchase_time** : the time when the user bought the item (GMT time)
- **purchase_value** : the cost of the item purchased (USD)
- **device_id** : the device id. You can assume that it is unique by device. I.e., 2 transactions with the same device ID means that the same physical device was used to buy
- **source** : user marketing channel: ads, SEO, Direct (i.e. came to the site by directly typing the site address on the browser).
- **browser** : the browser used by the user.
- **sex** : user sex: Male/Female
- **age** : user age
- **ip_address** : user numeric ip address
- **class** : this is what we are trying to predict: whether the activity was fraudulent (1) or not (0).

```
    "IpAddress_to_Country" - mapping each numeric ip address to its country.
For each country, it gives a range. If the numeric ip address falls within
the range, then the ip address belongs to the corresponding country.
```

Columns:

- **lower_bound_ip_address** : the lower bound of the numeric ip address for that country
- **upper_bound_ip_address** : the upper bound of the numeric ip address for that country
- **country** : the corresponding country. If a user has an ip address whose value is within the upper and lower bound, then she is based in this country.

Example

```
    Let's check the first user activity
```

head(Fraud_Data,1)

Column Name	Value	Description
user_id	22058	this is the user id
signup_time	2015-02-24 22:55:49	this user signed up on the site on Feb, 24, at 10:55 and 49sec PM GMT
purchase_time	2015-04-18 02:47:11	his first transaction happened on April, 18 at 2:47 AM GMT
purchase_value	34	the item he bought cost 34$
device_id	QVPSPJUOCKZAR	this is his device id
source	SEO	came to the site via SEO
browser	Chrome	was using Google Chrome
sex	M	he is a Male
age	39	he is 39 y/o
ip_address	732758369	this is his numeric ip address
class	0	his activity was not fraudulent

```
    Let's check where that user is based. Its IP address should fall within
some range in the IpAddress_to_Country table.
```

subset (IpAddress_to_Country, lower_bound_ip_address <= 732758369 & upper_bound_ip_address>= 732758369)

Column Name	Value	Description
lower_bound_ip_address	729808896	this the lower bound of the range that included that user numeric ip address
upper_bound_ip_address	734003199	this the upper bound of the range that included that user numeric ip address
country	Japan	the user is based in Japan since we found out that his ip address falls within the Japanese range

Funnel Analysis

Goal

The goal is to perform funnel analysis for an e-commerce website.

Typically, websites have a clear path to conversion: for instance, you land on the home page, then you search, select a product, and buy it. At each of these steps, some users will drop off and leave the site. The sequence of pages that lead to conversion is called 'funnel'.

Data Science can have a tremendous impact on funnel optimization. Funnel analysis allows to understand where/when our users abandon the website. It gives crucial insights on user behavior and on ways to improve the user experience. Also, it often allows to discover bugs.

Challenge Description

You are looking at data from an e-commerce website. The site is very simple and has just 4 pages:

- The first page is the **home page.** When you come to the site for the first time, you can only land on the home page as a first page.
- From the home page, the user can perform a search and land on the **search page**.
- From the search page, if the user clicks on a product, she will get to the **payment page**, where she is asked to provide payment information in order to buy that product.
- If she does decide to buy, she ends up on the **confirmation page**

The company CEO isn't very happy with the volume of sales and, especially, of sales coming from new users. Therefore, she asked you to investigate whether there is something wrong in the conversion funnel or, in general, if you could suggest how conversion rate can be improved.

Specifically, she is interested in :

- **A full picture of funnel conversion rate** for both desktop and mobile
- Some insights on **what the product team should focus on** in order to improve conversion rate as well as anything you might discover that could help improve

conversion rate.

Data

We have 5 tables downloadable by clicking **here**.
All the tables refer to *only* the user first experience on the site. The 5 tables are:

```
"user_table" - info about the user
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in all other tables
- **date** : the date when the user firstly landed on the site
- **device** : user device. Can be mobile or desktop
- **sex** : male/female

```
"home_page_table" - Users who landed on the home page
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in all other tables
- **page** : it is always home_page.

```
"search_page_table" - Users who landed on the search_page
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in all other tables
- **page** : it is always search_page

```
"payment_page_table" - Users who landed on the payment_page
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in all other tables
- **page** : it is always payment_page

```
    "payment_confirmation_table" - Users who landed on the
payment_confirmation_table. That is, these are the users who bought the product.
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in all other tables
- **page** : it is always payment_confirmation_page

Example

```
    Let's check one user through the funnel
```

subset(user_table,user_id == 1659)

Column Name	Value	Description
user_id	1659	The Id of the user
date	2015-01-01	This user firstly hit the site on Jan, 1st.
device	Mobile	User was using mobile
sex	Female	Was a female

```
    Let's check if she hit the home page (she should, since all users should hit
the home page on their first visit)
```

subset(home_page_table,user_id == 1659)

Column Name	Value	Description
user_id	1659	User id.
page	home_page	Yep, on Jan, 1st she visited the home page

```
Let's check if she hit the search page
```

subset(search_page_table,user_id == 1659)

Column Name	Value	Description
user_id	1659	User id.
page	search_page	Yep, on Jan, 1st she visited the search page too.

Let's check if she hit the payment page

subset(payment_page_table,user_id == 1659)

<0 rows> (or 0-length row.names) # No. She never visited the payment page. User 1659 landed on the site on Jan, 1st using mobile. From the home page, she then went to the search page and then left the site.

```
Let's check one user who actually bought the product
```

head (payment_confirmation_table, 1)

Column Name	Value	Description
user_id	123100	User id.
page	payment_confirmation_page	This user bought the product! Went all the way through the funnel! Success!

Pricing Test

Goal

Pricing optimization is, non surprisingly, another area where data science can provide huge value.

The goal here is to evaluate whether a pricing test running on the site has been successful. As always, you should focus on user segmentation and provide insights about segments who behave differently as well as any other insights you might find.

Challenge Description

Company XYZ sells a software for $39. Since revenue has been flat for some time, the VP of Product has decided to run a test increasing the price. She hopes that this would increase revenue. In the experiment, 66% of the users have seen the old price ($39), while a random sample of 33% users a higher price ($59).

The test has been running for some time and the VP of Product is interested in understanding how it went and whether it would make sense to increase the price for all the users.

Especially he asked you the following questions:

- **Should the company sell its software for $39 or $59?**
- The VP of Product is interested in having a holistic view into user behavior, especially focusing on actionable insights that might increase conversion rate. **What are your main findings looking at the data?**
- [Bonus] The VP of Product feels that the test has been running for too long and he should have been able to get statistically significant results in a shorter time. Do you agree with her intuition? **After how many days you would have stopped the test?** Please, explain why.

Data

We have two tables downloadable by clicking **here**.

The two tables are:

```
"test_results" - data about the test
```

Columns:

- **user_id** : the Id of the user. Can be joined to user_id in user_table
- **timestamp** : the date and time when the user hit for the first time company XYZ webpage. It is in user local time
- **source** : marketing channel that led to the user coming to the site. It can be:
  - ads-["google", "facebook", "bing", "yahoo", "other"]. That is, user coming from google ads, yahoo ads, etc.
  - seo - ["google", "facebook", "bing", "yahoo", "other"]. That is, user coming from google search, yahoo, facebook, etc.
  - friend_referral : user coming from a referral link of another user
  - direct_traffic: user coming by directly typing the address of the site on the browser
- **device** : user device. Can be mobile or web
- **operative_system** : user operative system. Can be: "windows", "linux", "mac" for web, and "android", "iOS" for mobile. "Other" if it is none of the above
- **test**: whether the user was in the test (i.e. 1 -> higher price) or in control (0 -> old/lower price)
- **price** : the price the user sees. It should match test
- **converted** : whether the user converted (i.e. 1 -> bought the software) or not (0 -> left the site without buying it).

```
"user_table" - Information about the user
```

Columns:

- **user_id** : the Id of the user. Can be joined to user_id in test_results table
- **city** : the city where the user is located. Comes from the user ip address
- **country** : in which country the city is located
- **lat** : city latitude - should match user city

- **long** : city longitude - should match user city

Example

```
Let's check the first user
```

head(test_results,1)

Column Name	Value	Description
user_id	604839	The Id of the user
timestamp	2015-05-08 03:38:34	The user landed on the site on May, 8 at 3 and 38AM (and 34 seconds).
source	ads_facebook	User came via Facebook ads
device	mobile	User was using mobile
operative_system	iOS	Was using iOS
test	0	Was in the control group, i.e. seeing the old price
price	39	Indeed, the price she saw was just $39
converted	0	Alas, left the site without purchasing the software

---

```
Let's check location info for that user
```

subset (user_table,user_id == 604839)

Column Name	Value	Description
user_id	604839	User id. Same user as in the previous table
city	Buffalo	She was based in Buffalo when she hit the site
country	USA	The country where Buffalo is
lat	42.89	Buffalo latitude
long	-78.86	Buffalo longitude

Marketing Email Campaign

Goal

Optimizing marketing campaigns is one of the most common data science tasks.
Among the many possible marketing tools, one of the most efficient is using emails.

Emails are great cause they are free and can be easily personalized. Email optimization involves personalizing the text and/or the subject, who should receive it, when should be sent, etc. Machine Learning excels at this.

Challenge Description

The marketing team of an e-commerce site has launched an email campaign. This site has email addresses from all the users who created an account in the past.

They have chosen a random sample of users and emailed them. The email let the user know about a new feature implemented on the site. From the marketing team perspective, a success is if the user clicks on the link inside of the email. This link takes the user to the company site.

You are in charge of figuring out how the email campaign performed and were asked the following questions:

- What percentage of users opened the email and what percentage clicked on the link within the email?
- The VP of marketing thinks that it is stupid to send emails to a random subset and in a random way. Based on all the information you have about the emails that were sent, can you build a model to optimize in future email campaigns to maximize the probability of users clicking on the link inside the email?
- By how much do you think your model would improve click through rate (defined as # of users who click on the link / total users who received the email). How would you test that?
- Did you find any interesting pattern on how the email campaign performed for different segments of users? Explain.

Data

We have 3 tables downloadable by clicking **here**.

The 3 tables are:

```
"email_table" - info about each email that was sent
```

Columns:

- **email_id** : the Id of the email that was sent. It is unique by email
- **email_text** : there are two versions of the email: one has "long text" (i.e. has 4 paragraphs) and one has "short text" (just 2 paragraphs)
- **email_version** : some emails were "personalized" (i.e. they had the name of the user receiving the email in the incipit, such as "Hi John,"), while some emails were "generic" (the incipit was just "Hi,").
- **hour** : the user local time when the email was sent.
- **weekday** : the day when the email was sent.
- **user_country** : the country where the user receiving the email was based. It comes from the user ip address when she created the account.
- **user_past_purchases** : how many items in the past were bought by the user receiving the email

---

```
"email_opened_table" -  the id of the emails that were opened at least once.
```

Columns:

- **email_id** : the id of the emails that were opened, i.e. the user clicked on the email and, supposedly, read it.

```
    "link_clicked_table" -  the id of the emails whose link inside was clicked
at least once. This user was then brought to the site.
```

Columns:

- **email_id** : if the user clicked on the link within the email, then the id of the email shows up on this table.

Example

```
    Let's check one email that was sent
```

head(email_table, 1)

Column Name	Value	Description
email_id	85120	The Id of the email
email_text	short_email	That was a short email
email_version	personalized	It was personalized with the user name in the text
hour	2	It was sent at 2AM user local time
weekday	Sunday	It was sent on a Sunday
user_country	US	The user is based in the US
user_past_purchases	5	The user in the past has bought 5 items from the site

```
    Let's check if that email was opened
```

subset(email_opened_table, email_id == 85120) >

<0 rows> (or 0-length row.names) # Nop. The user never opened it.

```
    We would obviously expect that the user never clicked on the link, since you
need to open the email in the first place to be able to click on the link
inside. Let's check:
```

subset(link_clicked_table, email_id == 85120)

<0 rows> (or 0-length row.names) # The user obviously never clicked on the link.

Song Challenge

Goal

Company XYZ is a very early stage startup. They allow people to stream music from their mobile for free. Right now, they still only have songs from the Beatles in their music collection, but they are planning to expand soon.

They still have all their data in json files and they are interested in getting some basic info about their users as well as building a very preliminary song recommendation model in order to increase user engagement.

Working with json files is important. If you join a very early stage start-up, they might not have a nice database and all data will be in jsons. Third party data are often stored in json files as well.

Challenge Description

You are the fifth employee at company XYZ. The good news is that if the company becomes big, you will become very rich with the stocks. The bad news is that, at such an early stage, the data is usually very messy. All their data is stored in json files.

The company CEO asked you very specific questions:

- What are the top 3 and the bottom 3 states in terms of number of users?
- What are the top 3 and the bottom 3 states in terms of user engagement? You can choose how to mathematically define user engagement. What the CEO cares about here is in which states users are using the product a lot/very little.
- The CEO wants to send a gift to the first user who signed-up for each state. That is, the first user who signed-up from California, from Oregon, etc. Can you give him a list of those users?
- Build a function that takes as an input any of the songs in the data and returns the most likely song to be listened next. That is, if, for instance, a user is currently listening to "Eight Days A Week", which song has the highest probability of being played right after it by the same user? This is going to be v1 of a song recommendation model.

- How would you set up a test to check whether your model works well and is improving engagement?

Data

We have 1 json file downloadable by clicking **here**.

The json is:

```
    "data" - all the data is here. Each row represents a song that was listened
by a user.
```

Fields:

- **id** : unique identifier of the row
- **user_id** : user id who listened to a given song
- **user_state** : where the user is based
- **user_sign_up_date** : when the user signed-up
- **song_played** : the song that was listened
- **time_played** : at which time the user started listening to the song (local time)

Example

```
    Let's check the first row in the json file:
```

Field Name	**Value**	**Description**
id	GOQMMKSQQH	The unique Id of this event
user_id	122	The person who listened to that song.
user_state	Louisiana	She is based in Louisiana
user_sign_up_date	2015-05-16	She created her account on May, 16.
song_played	Hey Jude	She listened to Hey Jude
time_played	2015-06-11 21:51:35	She started listening to the song on June, 11 at 9:51 and 35 secs PM

Clustering Grocery Items

Goal

Online shops often sell tons of different items and this can become very messy very quickly!

Data science can be extremely useful to automatically organize the products in categories so that they can be easily found by the customers.

The goal of this challenge is to look at user purchase history and create categories of items that are likely to be bought together and, therefore, should belong to the same section.

Challenge Description

Company XYZ is an online grocery store. In the current version of the website, they have manually grouped the items into a few categories based on their experience.

However, they now have a lot of data about user purchase history. Therefore, they would like to put the data into use!

This is what they asked you to do:

- The company founder wants to meet with some of the best customers to go through a focus group with them. You are asked to send the ID of the following customers to the founder:
    - the customer who bought the most items overall in her lifetime
    - for each item, the customer who bought that product the most
- Cluster items based on user co-purchase history. That is, create clusters of products that have the highest probability of being bought together. The goal of this is to replace the old/manually created categories with these new ones. Each item can belong to just one cluster.

Data

We have 2 table downloadable by clicking **here**.

The 2 tables are:

```
"item_to_id" - for each item, it gives the corresponding id
```

Columns:

- **Item_name** : the name of the item
- **Item_id** : the id of the item. Can be joined to the id in the other table. It is unique by item

---

```
"purchase_history" - for each user purchase, the items bought
```

Columns:

- **user_id** : the id of the user.
- **id** : comma-separated list of items bought together in that transaction.

Example

```
    Let's check what one user bought:
```

head(purchase_history,1)

Column Name	Value	Description
user_id	222087	this is simply the user id
id	27,26	this means the user bought together item 27 and 26 in that purchase event. All co-purchased items are listed under the same column and separated by a comma .

```
Let's check what is item 26 for instance:
```

subset(item_to_id,Item_id==26)

Column Name	Value	Description
Item_name	spaghetti sauce	she bought spaghetti sauce
Item_id	26	spaghetti sauce id is 26.

Credit Card Transactions

Goal

One of the greatest challenges in fraud, and in general in that area of data science related to catching illegal activities, is that you often find yourself one step behind.

Your model is trained on past data. If users come up with a totally new way to commit a fraud, it often takes you some time to be able to react. By the time you get data about that new fraud strategy and retrain the model, many frauds have been already committed.

A way to overcome this is to use unsupervised machine learning, instead of supervised. With this approach, you don't need to have examples of certain fraud patterns in order to make a prediction. Often, this works by looking at the data and identify sudden clusters of unusual activities.

This is the goal of this challenge. You have a dataset of credit card transactions and you have to identify unusual/weird events that have a high chance of being a fraud.

Challenge Description

Company XYZ is a major credit card company. It has information about all the transactions that users make with their credit card.

Your boss asks you to do the following:

- Your boss wants to identify those users that in your dataset never went above the monthly credit card limit (calendar month). The goal of this is to automatically increase their limit. Can you send him the list of Ids?
- On the other hand, she wants you to implement an algorithm that as soon as a user goes above her monthly limit, it triggers an alert so that the user can be notified about that.We assume here that at the beginning of the new month, user total money spent gets reset to zero (i.e. she pays the card fully at the end of each month). Build a function that for each day, returns a list of users who went above their credit card monthly limit on that day.
- Finally, your boss is very concerned about frauds cause they are a huge cost for credit card companies. She wants you to implement an unsupervised algorithm that returns all transactions that seem unusual and are worth being investigated further.

Data

We have 2 table downloadable by clicking **here**.

The 2 tables are:

```
"cc_info" - general information about the credit card and its holder
```

Columns:

- **credit_card** : credit card number. Can be joined to credit_card in the table below
- **city** : where the credit card holder lives
- **state** : in which state the credit card holder lives
- **zipcode** : credit card holder zip code
- **credit_card_limit** : this is the credit card monthly limit. Credit card holders should be careful in not going above this limit in total money spent per month. The limit is by calendar month

---

```
    transactions - information about each transaction that happens between Aug,
1 and Oct, 30 for the credit cards in cc_info.
```

Columns:

- **credit_card** : credit card number. Can be joined to credit_card in the other table
- **date** : when the transaction happened (GMT time)
- **transaction_dollar_amount** : transaction amount in dollars
- **Long** : longitude of where the transaction happened
- **Lat** : latitude of where the transaction happened

Example

```
    Let's check one transaction
```

head(transactions,1)

Column Name	Value	Description
credit_card	1003715054175576	this is the credit card number
date	2015-09-11 00:32:40	the transaction happened on Sept, 11 around midnight GMT time
transaction_dollar_amount	43.78	the credit card was charged 43.78$
Long	-80.17413	the transaction happened at this longitude
Lat	40.26737	and this latitude. It means it happened here, in Pennsylvania.

```
    Let's check info about that credit card
```

subset (cc_info, credit_card==1003715054175576)

Column Name	Value	Description
credit_card	1003715054175576	same as in the example above
city	Houston	city where the credit card holder lives
state	PA	it makes sense. Before we saw the cc holder bought something in Pennsylvania and now we found out that she actually also lives in Pennsylvania
zipcode	15342	her zip code
credit_card_limit	20000	her monthly credit card limit is 20K USD

User Referral Program

Goal

Almost all sites have a user referral program: you can invite new users to try a given product. Typically, after the new user completes a transaction, you get rewarded with a certain amount of money or credit to be used on the site.

The goal of this challenge is to analyze the data from a referral program and draw conclusions about its effectiveness.

Challenge Description

Company XYZ has started a new referral program on Oct, 31. Each user who refers a new user will get 10$ in credit when the new user buys something.

The program has been running for almost a month and the Growth Product Manager wants to know if it's been successful. She is very excited cause, since the referral program started, the company saw a spike in number of users and wants you to be able to give her some data she can show to her boss.

- Can you estimate the impact the program had on the site?
- Based on the data, what would you suggest to do as a next step?
- The referral program wasn't really tested in a rigorous way. It simply started on a given day for all users and you are drawing conclusions by looking at the data before and after the test started. What kinds of risks this approach presents? Can you think of a better way to test the referral program and measure its impact?

Data

We have just 1 table downloadable by clicking **here**.

The table is:

```
    "referral" - provides information about each transaction that happens on
the site and whether the user came from the referral program or not.
```

Columns:

- **user_id** : the id of the user
- **date** : date of the purchase
- **country** : user country based on ip address
- **money_spent** : how much the item bought cost (USD)
- **is_referral** : whether the user came from the referral program(1) or not (0)
- **device_id** : It is an identifier of the device used to make the purchase. You can assume here that for a given physical device, its id never changes

Example

```
    Let's check one purchase event
```

head (referral, 1)

Column Name	Value	Description
user_id	2	this is the user id
date	2015-10-03	she bought something on Oct, 3
country	FR	she is in France
money_spent	65	her purchase cost 65$
is_referral	0	she didn't come via the referral program
device_id	EVDCJTZMVMJDG	to make the purchase, she used a device identified by this id

```
    Let's now also check one purchase event from a user who came via the
referral program
```

head(subset(referral,is_referral==1), 1)

Column Name	**Value**	**Description**
user_id	5016	this is the user
date	2015-10-31	she bought on Oct, 31
country	DE	she is in Germany
money_spent	45	her purchase cost 45$
is_referral	1	she joined the site via another user referral
device_id	OGAZARJAGCPUQ	this is the device she used to complete the purchase

Loan granting

Goal

Another area where data science and machine learning play a huge role is in choosing if granting a loan. This is a particularly hot field as many start-ups feel that bank loan models can be improved. Therefore, there is space to come up with better loan strategies that can benefit both the lender and the borrower.

In this challenge, you will have access to loan data from a bank and will have to improve their model.

Challenge Description

We have access to a specific bank loan data. We have data about all loans asked to the bank, whether the bank decided to grant it and, finally, whether the borrower managed to repay it. We also have info about the borrower at the moment she is asking for the loan.

You have to come up with a better strategy to grant loans. Specifically you should:

- Build a model which is better than the bank model. For simplicity, assume that:
    - If you grant the loan and the it doesn't get repaid, you lose 1
    - If you grant the loan and the it does get repaid, you gain 1
    - If you don't grant the loan, you gain 0
- Using the rules above, compare bank profitability vs your model profitability
- Describe the impact of the most important variables on the prediction. Also, focus on the variable "is_employed", which describes whether the borrower is employed when she asks for the loan. How does this variable impact the model? Explain why
- Are there any other variables, not in the data provided, that you'd have liked to include in the model?

Data

We have 2 table downloadable by clicking here.

The 2 tables are:

```
"loan_table" - general information about the loan
```

Columns:

- **loan_id** : the id of the loan. Unique by loan. Can be joined to loan id in the other table
- **loan_purpose** : the reason for asking the loan: *investment, business, emergency_funds, home, other,*
- **date** : when the loan was asked
- **loan_granted** : whether the loan was granted
- **loan_repaid** : whether the loan was repaid. NA means that the loan was not granted

---

```
"borrower_table" - information about the borrower
```

Columns:

- **loan_id** : the id of the the loan. Unique by loan. Can be joined to loan id in the other table
- **is_first_loan** : did she ask for any other loans in her lifetime?
- **fully_repaid_previous_loans** : did she pay on time all of her previous loans? If this is the first loan, it is NA
- **currently_repaying_other_loans** : is she currently repaying any other loans? If this is the first loan, it is NA
- **total_credit_card_limit** : total credit card monthly limit
- **avg_percentage_credit_card_limit_used_last_year** : on an average, how much did she use of her credit card limit in the previous 12 months. This number can be >1 since it is possible to go above the credit card limit
- **saving_amount** : total saving account balance when she asked for the loan
- **checking_amount** : total checking account balance when she asked for the loan
- **is_employed** : whether she is employed (1) or not (0)
- **yearly_salary** : how much she earned in the previous year
- **age** : her age
- **dependent_number** : number of people she claims as dependent

Example

```
Let's check one loan:
```

head (loan_table,1)

Column Name	Value	Description
loan_id	19454	this is id of the loan
loan_purpose	investment	she needed the loan in order to make an investment
date	2012-03-15	the loan was asked on March, 15.
loan_granted	0	unfortunately, the bank didn't grant this loan
loan_repaid	NA	therefore, loan_repaid is NA since there was no loan granted in the first place.

```
Let's now check the characteristics of that user who was denied the loan
```

subset (borrower_table, loan_id == 19454)

Column Name	Value	Description
loan_id	19454	same id as in the example above
is_first_loan	1	this is the first time this person asks for a loan
fully_repaid_previous_loans	NA	has to be NA since she never asked for a loan in the past
currently_repaying_other_loans	NA	has to be NA since she never asked for a loan in the past
total_credit_card_limit	8600	her monthly credit card limit is 8600 USD
avg_percentage_credit_card_limit_used_last_year	0.79	on an average, per month, she used 79% of her credit card limit in the previous yr.
saving_amount	1491	she has 1491 USD in her saving bank account
checking_amount	6285	she has 6285 USD in her checking account
is_employed	1	she has a job when she asked for the loan
yearly_salary	45200	and she was making 45200 USD a yr
age	42	she is 42 y/o
dependent_number	7	she claims 7 dependents

Json City Similarities

Goal

This is another challenge where your data is stored in a JSON file.

Analyzing user behavior within the same session is often crucial. Clustering users based on their browsing behavior is probably the most important step if you want to personalize your site.

The goal of this challenge is to build the foundation of personalization by indentifying searches likely to happen together and cluster users based on their session searches.

Challenge Description

Company XYZ is an Online Travel Agent, such as Expedia, Booking.com, etc.

They store their data in JSON files. Each row in the json shows all different cities which have been searched for by a user within the same session (as well as some other info about the user). That is, if I go to company XYZ site and look for hotels in NY and SF within the same session, the corresponding JSON row will show my user id, some basic info about me and the two cities.

You are given the following tasks:

- There was a bug in the code and one country didn't get logged. It just shows up as an empty field (""). Can you guess which country was that? How?
- For each city, find the most likely city to be also searched for within the same session.
- Travel sites are browsed by two kinds of users. Users who are actually planning a trip and users who just dream about a vacation. The first ones have obviously a much higher purchasing intent. Users planning a trip often search for cities close to each other, while users who search for cities far away from each other are often just dreaming about a vacation. That is, a user searching for LA, SF and Las Vegas in the same session is much more likely to book a hotel than a user searching for NY, Paris, Kuala Lumpur (makes sense, right?). Based on this idea, come up with an algorithm that clusters sessions into two groups: high intent and low intent. Explain all assumptions you make along the way.

Data

We have 1 file downloadable by clicking **here**.

```
"city_search" - a list of searches happening within the same session.
```

Fields:

- **session_id** : session id. Unique by row
- **unix_timestamp** : unix timestamp of when the session started
- **cities** : the unique cities which were searched for within the same session by a user
- **user** : it is has the following nested fields:

  user_id: the id of the user
  joining_date: when the user created the account
  country: where the user is based

Example

```
Let's now check the characteristics of the first row in the JSON. The
interesting fields of the URL are after *hotelsearch?* and are separated by *&*
```

head (city_search, 1)

Field	Value	Description
session_id	X061RFWB06K9V	unique identifier of the search session
unix_timestamp	1442503708	unix timestamp of when the session started. That means: Thu, 17 Sep 2015 15:28:28 GMT
cities	New York NY, Newark NJ	the user searched for hotels in two cities: NY and Newark
user_id	2024	id of the user
joining_date	2015-03-22	she joined the site on March, 22
country	UK	she is a based in UK

Optimization of Employee Shuttle Stops

Goal

It is a common practice for tech companies to use shuttle buses to ferry their employees from home to the office. The goal of this exercise is to figure out the optimal stops for a bus shuttle. The company is based in Mountain View and the shuttle provides transportation for employees based in San Francisco.

With the explosion of user location data, data science can be used to optimize many of the services cities offer to their citizens. Transportation optimization is an example of that, but there are so many other possible applications. All this often goes under the (buzz name) "smart city" and it is one of the most interesting future applications of data science.

Challenge Description

Company XYZ has decided to offer a shuttle bus to help its employees commute from San Francisco to Mountain View. The city of San Francisco has given the company a list of potential bus stop locations to choose from and asked to not have more than 10 stops within the city.

You have been given the home address of all employees interested in taking the shuttle and asked to come up with the ten most efficient stops. While you have been given a certain freedom in defining what is "efficient", the general consensus within the company is that the most efficient way to select the bus stops is to minimize the overall walking distance between employee homes and the closest bus stop.

Estimating all possible 10 stop combinations would require a lot of time (how many combinations would that be?). Therefore, your boss is fine with simplifying the problem and returning 10 stops that have a high probability of being the best stops.

You should write an algorithm that returns the best 10 stops in your opinion. Also, please explain the rationale behind the algorithm.

Data

We have 2 tables downloadable by clicking **here**.

The 2 tables are:

```
    "Potentail_Bust_Stops" - this is the list of potential bus stops given by
San Francisco to the company. It is a list of intersections in the city.
```

Columns:

- **Street_One** : one of the two streets intersecting
- **Street_Two** : the other street intersecting

```
    "Employee_Addresses" - the home address of each employee interested in
taking the shuttle.
```

Columns:

- **address** : employee address
- **employee_id** : employee id, unique by employee

Example

```
    Let's check the first bus stop candidate:
```

head (Potentail_Bust_Stops, 1)

Column Name	Value	Description
Street_One	MISSION ST	One of the two streets intersecting is Mission St
Street_Two	ITALY AVE	The second street is Italy Ave

That is , the bus stop candidate is here.

```
Let's now check where one employee lives
```

head (Employee_Addresses,1)

Column Name	Value	Description
address	98 Edinburgh St, San Francisco, CA 94112, USA	She lives in Edinburgh St
employee_id	206	Her employee id is 206

That is, that employee lives here.

Diversity in the Workplace

Goal

Diversity, unconscious bias in the workplace and, in general, the way companies treat their employees are a very important topic. Data science can help discover potential discriminations by looking at the data and see if there are segments of employees that are treated worse.

Challenge Description

There has been lots of talking about diversity in the workplace, especially in technology. The Head of HR at your company is very concerned about that and has asked you to analyze internal data about employees and see whether results suggest that the company is treating its employees fairly.

Specifically, she gave you the following tasks:

- In the company there are 6 levels (described below). Identify, for each employee, her corresponding level.
  - Individual Contributors (IC) - they don't manage anyone
  - Middle Managers (MM) - they are the direct bosses of IC
  - Directors (D) - they are the direct bosses of MM
  - VP - D direct bosses
  - Executives (E) - VP direct bosses
  - CEO - The direct boss of E.
- How many people each employee manages? You should have a table with employee ids and the number of people managed. Consider that if John directly manages 2 people and these two people manage 5 people each, then we conclude that John manages 12 people.
- Build a model to predict the salary of each employee.
- Describe the main factors impacting employee salaries. Do you think the company has been treating its employees fairly? Do you see any bias? What are the next steps you would suggest to the Head of HR?

Data

We have 2 tables downloadable by clicking **here**.

The 2 tables are:

```
"company_hierarchy" - info about each employee direct boss and her dept
```

Columns:

- **employee_id** : the Id of the employee. It is unique by employee and can be joined to employee id in the other table
- **boss id** : the id of the boss of employee id. It is unique by employee and can also be joined to employee id in the other table.
- **dept** : employee id dept. There are the following departments:
    - Engineering (data science is under engineering)
    - Marketing
    - Sales
    - HR
    - + the "CEO" dept just for the CEO, since he belongs to all the above dept

---

```
    employee - info about each employee
```

Columns:

- **employee_id** : the Id of the employee. It is unique by employee and can be joined to employee id and/or boss id in the other table
- **signing_bonus** : whether the employee got a signing bonus when she joined the company (1 -> yes, 0 -> no)
- **salary** : the current salary of that employee in USD
- **degree level**: the highest degree received by the employee.
- **sex**: Male/Female
- **yrs_experience**: employee years of work experience

Example

```
Let's check one random employee. Let's say, employee 51535:
```

subset (company_hierarchy, employee_id == 51535)

Column Name	**Value**	**Description**
employee_id	51535	The Id of the employee
boss_id	61554	The employee id of his boss. That is, 51535 directly reports to 61554
dept	engineering	Employee 51535 works in the engineering dept

---

```
Let's check employee 51535 information:
```

subset (employee, employee_id == 51535)

Column Name	**Value**	**Description**
employee_id	51535	The Id of the employee
signing_bonus	1	He got a signing bonus when he joined the company
salary	650000	He makes 650K$ a year. Not bad...
degree_level	PhD	He has a PhD
sex	M	He is a male
yrs_experience	33	He worked for 33 yrs

```
Let's now check his boss, employee 61554:
```

subset(employee,employee_id == 61554)

Column Name	Value	Description
employee_id	61554	The Id of the employee
signing_bonus	1	He also got a signing bonus when he joined the company
salary	750000	He makes 750K$ a year! Must have a high position...
degree_level	PhD	He also has a PhD
sex	M	He is a male
yrs_experience	7	He worked for only 7 yrs though

URL Parsing Challenge

Goal

Being able to efficiently parse URLs and extract info from them is a very important skill for a data scientist.

Firstly, if you join a very early stage startup, it might be that a lot of data are just stored in the URLs visited by the users. And, therefore, you will have to build a system that parses the URLs, extracts fields out of it, and saves them into a table that can be easily queried (not the most fun of the jobs, but very useful!).

Secondly, often using external data can help a lot your models. A way to get external data is by scraping websites. And this is often done by being able to play with a given site URL structure (assuming it is allowed by the external site ToS or it is not allowed and you don't get caught).

The goal of this project is to parse a sequence of URLs about user searches and extract some basic info out of it.

Challenge Description

Company XYZ is an Online Travel Agent site, such as Expedia, Booking.com, etc.

They haven't invested in data science yet and all the data they have about user searches are simply stored in the URLs generated when users search for a hotel. If you are not familiar with URLs, you can run a search on any OTA site and see how all search parameters are present in the URL.

You are asked to:

- Create a clean data set where each column is a field in the URL, each row is a given search and the cells are the corresponding URL values.
- For each search query, how many amenities were selected?
- Often, to measure the quality of a search algorithm, data scientists use some metric based on how often users click on the second page, third page, and so on. The idea here is that a great search algorithm should return all interesting results on the first page and never force users to visit the other pages (how often do you click on the second page

results when you search on Google? Almost never, right?).

  - Create a metric based on the above idea and find the city with the worst search algorithm.

Data

We have 1 file downloadable by clicking here.

The file is

```
    "url_list" - a list of search URLs generated by the users when searching
for hotels
```

Fields:

- **hotel.checkin** : checkin date in the search query. It is mandatory
- **hotel.customMinimumPriceFilter** : This filter allows to only return hotels whose nightly price is above a certain threshold (in USD). Useful to filter out the really bad hotels
- **hotel.customMaximumPriceFilter** : This filter allows to only return hotels whose nightly price is below a certain threshold (in USD). Useful to filter out the hotels you can't afford
- **hotel.freeCancellation** : It is a check box. If the user selects it, only hotels with free cancellation are returned.
- **hotel.stars_5** : It is a check box. If the user selects it, 5-star hotels are returned. Multiple choices are allowed. For instance, a user can select 5 and 4 star hotels by checking this box and the one below. If no check box is selected, all hotels are returned.
- **hotel.stars_4** : It is a check box. If the user selects it, 4-star hotels are returned
- **hotel.stars_3** : It is a check box. If the user selects it, 3-star hotels are returned
- **hotel.stars_2** : It is a check box. If the user selects it, 2-star hotels are returned
- **hotel.stars_1** : It is a check box. If the user selects it, 1-star hotels are returned
- **hotel.max_score** : This filter allows to only return hotels whose score is below a certain threshold. Score is 1-5 with high being good (you can think of TravelAdvisor score to get an idea of what it is)
- **hotel.min_score** : This filter allows to only return hotels whose score is above a certain threshold
- **hotel.couponCode** : If the user uses a coupon in her search, this fields gets populated with "Yes"
- **hotel.adults** : Number of adults in the search query. This is the number of adults who would stay in the same room. It is mandatory
- **hotel.city** : Which city is the user searching for. It is mandatory

- **hotel.children** : Is the user traveling with children? This field returns the number of children in the search query
- **hotel.amenities** : There are a few amenities that the user can select in her search via different check boxes. The possible amenities are:
    - *shuttle*: free shuttle transportation from the airport
    - *internet*: free internet
    - *breakfast* : free breakfast
    - *lounge* : does the hotel have a lounge
    - *yes_smoking* : are there rooms where smoking is allowed
    - *yes_pet* : is it allowed to bring pets
- **hotel.checkout** : Check out date. It is mandatory
- **hotel.search_page** : Search page visited. 1 means the user is on the first page results, 2 -> clicked on the second page etc. This will be used to estimate the goodness of ranking for different cities

Example

```
Let's now check the characteristics of the first URL. The interesting fields of
the URL are after hotelsearch? and are separated by &
```

head (user_list, 1)

Field	**Value**	**Description**
hotel.checkin	2015-09-19	check-in date is Sept, 19
hotel.stars_4	yes	the user only selected 4 star hotels.
hotel.min_score	4	hotel score must be at least 4
hotel.adults	3	she is looking for a room for 3 adults
hotel.city	New+York,+NY,+United+States	Looking for a hotel in NY
hotel.checkout	2015-09-20	check-out on Sept, 20. That is, it is just for 1 night
hotel.search_page	1	user is on the first page result. Seeing many high numbers here is often taken an indicator that the search algorithm isn't that good.

If a field is not in the URL, it means the user didn't filter by it. So, for instance, in this search the user didn't filter by a certain max/min price, amenities, etc.

Engagement Test

Goal

Many sites make money by selling ads. For these sites, the number of pages visited by users on each session is one of the most important metric, if not **the most important** metric.

Data science plays a huge role here, especially by building models to suggest personalized content. In order to check if the model is actually improving engagement, companies then run A/B tests.

It is often data scientist responsibility to analyze test data and understand whether the model has been successful. The goal of this project is to look at A/B test results and draw conclusions.

Challenge Description

The company of this exercise is a social network. They decided to add a feature called: Recommended Friends, i.e. they suggest people you may know.

A data scientist has built a model to suggest 5 people to each user. These potential friends will be shown on the user newsfeed. At first, the model is tested just on a random subset of users to see how it performs compared to the newsfeed without the new feature.

The test has been running for some time and your boss asks you to check the results. You are asked to check, for each user, the number of pages visited during their first session since the test started. If this number increased, the test is a success.

Specifically, your boss wants to know:

- Is the test winning? That is, should 100% of the users see the Recommended Friends feature?
- Is the test performing similarly for all user segments or are there differences among different segments?
- If you identified segments that responded differently to the test, can you guess the reason? Would this change your point 1 conclusions?

Data

We have 2 tables downloadable by clicking here.

The 2 tables are:

```
"user_table" - info about each user sign-up date
```

Columns:

- **user_id** : the Id of the user. It is unique by user and can be joined to user id in the other table
- **signup_date** : when the user joined the social network

```
    "test_table" - data about the test results. For each user, we only consider
her first session since the date when the test started. That is, if the test
started on Jan 1, and user 1 visited the site on Jan, 2 and Jan, 3, we only care
about how many pages she visited on Jan, 2.
```

Columns:

- **user_id** : the Id of the user
- **date** : the date of the first session since the test started
- **browser** : user browser during that session
- **test**: 1 if the user saw the new feature, 0 otherwise
- **pages_visited**: the metric we care about. # of pages visited in that session

Example

```
    Let's check the first row:
```

head(test_table,1)

Column Name	Value	Description
user_id	600597	this is the user
date	2015-08-13	her first session since the test started was on Aug, 13.
browser	IE	she used Internet Explorer to visit the site
test	0	she was in the control group, i.e. didn't see the new feature
pages_visited	2	she visited 2 pages during that session and then left the site

---

```
    Let's check when the user Id from the previous table joined the site:
```

subset (user_table, user_id== 600597)

Column Name	Value	Description
user_id	600597	this is the user we care about
signup_date	2015-01-19	she created her account on Jan, 19.

On-Line Video Challenge

Goal

The company of this challenge allows users to upload videos online, just like YouTube.

This company is interested in knowing whether a video is "hot" (i.e. trending up in terms of popularity), stable or going down. Understanding this would allow to optimize the videos promoted on the home-page and, therefore, maximize ads revenue.

Challenge Description

Company XYZ is an online video streaming company, just like YouTube or Dailymotion.

The Head of Product has identified as a major problem for the site a very high home page drop-off rate. That is, users come to the home-page and then leave the site without taking any action or watching any video. Since customer acquisition costs are very high, this is a huge problem: the company is spending a lot of money to acquire users who don't generate any revenue.

Currently, the videos shown on the home page to new users are manually chosen. The Head of Product had this idea of creating a new recommended video section on the home page.

He asked you the following:

- Classify each video into one the 3 categories below and explain your approach:
    - "Hot" - means trending up. These videos are candidate to be shown.
    - "Stable and Popular" - video view counts are flat, but very high. These videos are candidates to be shown too.
    - "Everything else" - these videos won't be shown.
- What are the main characteristics of the "hot videos"?
- After having identified the characteristics of the hot videos, how would you use this information from a product standpoint?

Data

We have 2 tables downloadable by clicking **here**.

The 2 tables are:

```
    "video_count" - provides information about how many times each video was
seen by day
```

Columns:

- **video_id** : video id, unique by video and joinable to the video id in the other table
- **count** : total count of views for each video by day
- **date** : on which day that video was watched that many times

---

```
"video_features" - characteristics of the video.
```

Columns:

- **video_id** : video id, unqiue by video and joinable to the video id in the other table
- **video_length** : length of the video in seconds
- **video_language** : language of the video, as selected by the user when she uploaded the video
- **video_upload_date** : when the video was uploaded
- **video_quality** : quality of the video. It can be [240p, 360p, 480p, 720p, 1080p]

Example

```
Let's check one video: how many times was it seen on a given day?
```

head (video_count, 1)

Column Name	Value	Description
video_id	2303	id of the video
count	22	it was seen just 22 times.
date	2015-01-07	on Jan, 7th.

```
Let's now check the characteristics of that video 2303:
```

subset(video_features, video_id == 2303)

Column Name	Value	Description
video_id	2303	It is the video we care about. Same as above
video_length	1071	the video lasts almost 18 min (1071 seconds)
video_language	Cn	the video is in Chinese
video_upload_date	2014-12-10	was uploaded on Dec, 10
video_quality	1080p	video quality is 1080p, i.e. very high

Subscription Retention Rate

Goal

Subscriptions are a great business model. There are so many advantages for businesses in having subscribers compared to single purchase users: revenue by customer is much higher, it is possible to cross-sell to the subscribers, future revenue is easily predictable, there is a significant cost (time/effort/etc.) for the customer in canceling the subscription, etc.

It is no surprise then that so many companies have subscription business models (or try very hard to come up with one!).

The goal of this challenge is to model subscription retention rate.

Challenge Description

Company XYZ started a subscription model in January, 2015. You get hired as a first data scientist at the end of August and, as a first task, you are asked to help executives understand how the subscription model is doing.

Therefore, you decide to pull data from all the users who subscribed in January and see, for each month, how many of them unsubscribed. In particular, your boss is interested in:

- A model that predicts monthly retention rate for the different subscription price points
- Based on your model, for each price point, what percentage of users is still subscribed after at least 12 months?
- How do user country and source affect subscription retention rate? How would you use these findings to improve the company revenue?

Data

We have just 1 table downloadable by clicking **here**.

The table is:

```
"subscription" - gives information about the user and her subscription status
```

Columns:

- **user_id** : the id of the user. Unique by user.
- **subscription_signup_date** : when the user signed up for the subscription. It is always Jan, 2015 in this table.
- **subscription_monthly_cost** : how much the user pays each month for the subscription (USD)
- **source** : marketing acquisition channel (SEO/Ads/Friend Referral)
- **billing_cycles** : total billing cycles as of the end of August.
- **is_active** : whether the subscription is still active (1) or not (0). If billing cycles is 8, it means the user has still an active subscription.

Example

```
    Let's check one subscription event:
```

head (subscription, 1)

Column Name	Value	Description
user_id	1459	This is the user
subscription_signup_date	January, 2015	The user signed up for the subscription in Jan. This is a constant within the table
subscription_monthly_cost	29	The subscription costs 29$ per month
country	Spain	User is based in Spain
source	ads	When the user subscribed, she came to the site via an ad
billing_cycles	4	The user has been subscribed for 4 months before canceling the subscription. That is, in total she paid 4*29$
is_active	0	Consequently, her subscription is not active

```
Let's now check one active subscription:
```

head(subset(subscription, is_active==1) ,1)

Column Name	**Value**	**Description**
user_id	7341	this is the user
subscription_signup_date	January, 2015	the user signed up for the subscription in Jan. This is a constant within the table
subscription_monthly_cost	29	This subscription also costs 29$ per month
country	US	User is based in the US
source	ads	also this user came to the site via an ad
billing_cycles	8	Makes sense, since it is still active, there have been 8 billing cycles since Jan.
is_active	1	Consequently, the subscription is active

Ads Analysis

Goal

Maybe the first industry to heavily rely on data science was the online ads industry. Data Science is used to choose which ads to show, how much to pay, optimize the ad text and the position as well as in countless of other related applications.

Optimizing ads is one of the most intellectually challenging jobs a data scientist can do. It is a really complex problem given the huge (really really huge) size of the datasets as well as number of features that can be used. Moreover, companies often spend huge amounts of money in ads and a small ad optimization improvement can be worth millions of dollars for the company.

The goal of this project is to look at a few ad campaigns and analyze their current performance as well as predict their future performance.

Challenge Description

Company XYZ is a food delivery company. Like pretty much any other site, in order to get customers, they have been relying significantly on online ads, such as those you see on Google or Facebook.

At the moment, they are running 40 different ad campaigns and want you to help them understand their performance.

Specifically, you are asked to:

- If you had to identify the 5 best ad groups, which ones would they be? Which metric did you choose to identify the best ad groups? Why? Explain the pros of your metric as well as the possible cons.
- For each group, predict how many ads will be shown on Dec, 15 (assume each ad group keeps following its trend).
- Cluster ads into 3 groups: the ones whose avg_cost_per_click is going up, the ones whose avg_cost_per_click is flat and the ones whose avg_cost_per_click is going down.

Data

We have 1 table downloadable by clicking **here**.

The table is:

```
"ad_table" - aggregate information about ads
```

Columns:

- **date** : all data are aggregated by date
- **shown** : the number of ads shown on a given day all over the web. Impressions are free. That is, companies pay only if a user clicks on the ad, not to show it
- **clicked** : the number of clicks on the ads. This is what companies pay for. By clicking on the ad, the user is brought to the site
- **converted** : the number of conversions on the site coming from ads. To be counted, a conversion has to happen on the same day as the ad click.
- **avg_cost_per_click** : on an average, how much it cost each of those clicks
- **total_revenue** : how much revenue came from the conversions
- **ad** : we have several different ad groups. This shows which ad group we are considering

Example

```
    Let's check one ad group for one day:
```

head(ad_table, 1)

Column Name	**Value**	**Description**
date	2015-10-01	aggregate stats are about Oct, 1.
shown	65877	all over the web, this ad group was shown 65877 times
clicked	2339	it received 2339 clicks. Clicks/Shown, often called click-through-rate, was therefore about 3.5%
converted	43	of those 2339 users coming to the site on Oct,1 via the ad, 43 converted on the same day. That is, they bought something on Oct, 1.
avg_cost_per_click	0.9	on an average, each click cost 0.9 USD
total_revenue	641.62	those 43 conversions, in total, generated 641.62 USD
ad	ad_group_1	this is the ad group 1. It is one of the many different ad variations we have.

Solution: Conversion Rate

```
#libraries needed
require(dplyr)
require(rpart)
require(ggplot2)
require(randomForest)
```

Let’s read the dataset into R.

```
data = read.csv('conversion_data.csv')
```

We should get something like this:

```
head(data)
```

```
##   country age new_user source total_pages_visited converted
## 1      UK  25        1    Ads                   1         0
## 2      US  23        1    Seo                   5         0
## 3      US  28        1    Seo                   4         0
## 4   China  39        1    Seo                   5         0
## 5      US  30        1    Seo                   6         0
## 6      US  31        0    Seo                   1         0
```

Let’s check the structure of the data:

```
str(data)
```

```
## 'data.frame':    316200 obs. of  6 variables:
##  $ country            : Factor w/ 4 levels "China","Germany",..: 3 4 4 1 4 4 1
4 3 4 ...
##  $ age                : int  25 23 28 39 30 31 27 23 29 25 ...
##  $ new_user           : int  1 1 1 1 1 0 1 0 0 0 ...
##  $ source             : Factor w/ 3 levels "Ads","Direct",..: 1 3 3 3 3 3 3 1 2
1 ...
##  $ total_pages_visited: int  1 5 4 5 6 1 4 4 4 2 ...
##  $ converted          : int  0 0 0 0 0 0 0 0 0 0 ...
```

Now, let’s inspect the data to look for weird behavior/wrong data. Data is never perfect in real life and requires to be cleaned. Often takehome challenges have wrong data which has been put there on purpose. **Identifying the wrong data and dealing with it is part of the challenge.**

R summary function is usually the best place to start:

```
summary(data)
```

```
##     country            age            new_user         source     
##  China  : 76602   Min.   : 17.00   Min.   :0.0000   Ads   : 88740  
##  Germany: 13056   1st Qu.: 24.00   1st Qu.:0.0000   Direct: 72420  
##  UK     : 48450   Median : 30.00   Median :1.0000   Seo   :155040  
##  US     :178092   Mean   : 30.57   Mean   :0.6855                  
##                   3rd Qu.: 36.00   3rd Qu.:1.0000                  
##                   Max.   :123.00   Max.   :1.0000                  
##  total_pages_visited   converted      
##  Min.   : 1.000      Min.   :0.00000  
##  1st Qu.: 2.000      1st Qu.:0.00000  
##  Median : 4.000      Median :0.00000  
##  Mean   : 4.873      Mean   :0.03226  
##  3rd Qu.: 7.000      3rd Qu.:0.00000  
##  Max.   :29.000      Max.   :1.00000  
```

A few quick observations:

- the site is probably a US site, although it does have a large Chinese user base as well
- user base is pretty young
- conversion rate at around 3% is industry standard. It makes sense.
- everything seems to make sense here except for max age 123 yrs! Let’s investigate it:

```
sort(unique(data$age), decreasing=TRUE)
```

```
##  [1] 123 111  79  77  73  72  70  69  68  67  66  65  64  63  62  61  60
## [18]  59  58  57  56  55  54  53  52  51  50  49  48  47  46  45  44  43
## [35]  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26
## [52]  25  24  23  22  21  20  19  18  17
```

Those 123 and 111 values seem unrealistic. How many users are we talking about:

```
subset(data, age>79)
```

```
##        country age new_user source total_pages_visited converted
## 90929  Germany 123        0    Seo                  15         1
## 295582      UK 111        0    Ads                  10         1
```

It is just 2 users! In this case, we can remove them, nothing will change. In general, depending on the problem, you can:

- remove the entire row saying you don’t trust the data
- treat those values as NAs
- if there is a pattern, try to figure out what went wrong.

In doubt, always go with removing the row. It is the safest choice.

You probably also want to emphasize in the text that wrong data is worrisome and can be an indicator of some bug in the logging code. Therefore, you'd like to talk to the software engineer who implemented the code to see if, perhaps, there are some bugs which affect the data significantly.

Anyway, here is probably just users who put wrong data. So let's remove them:

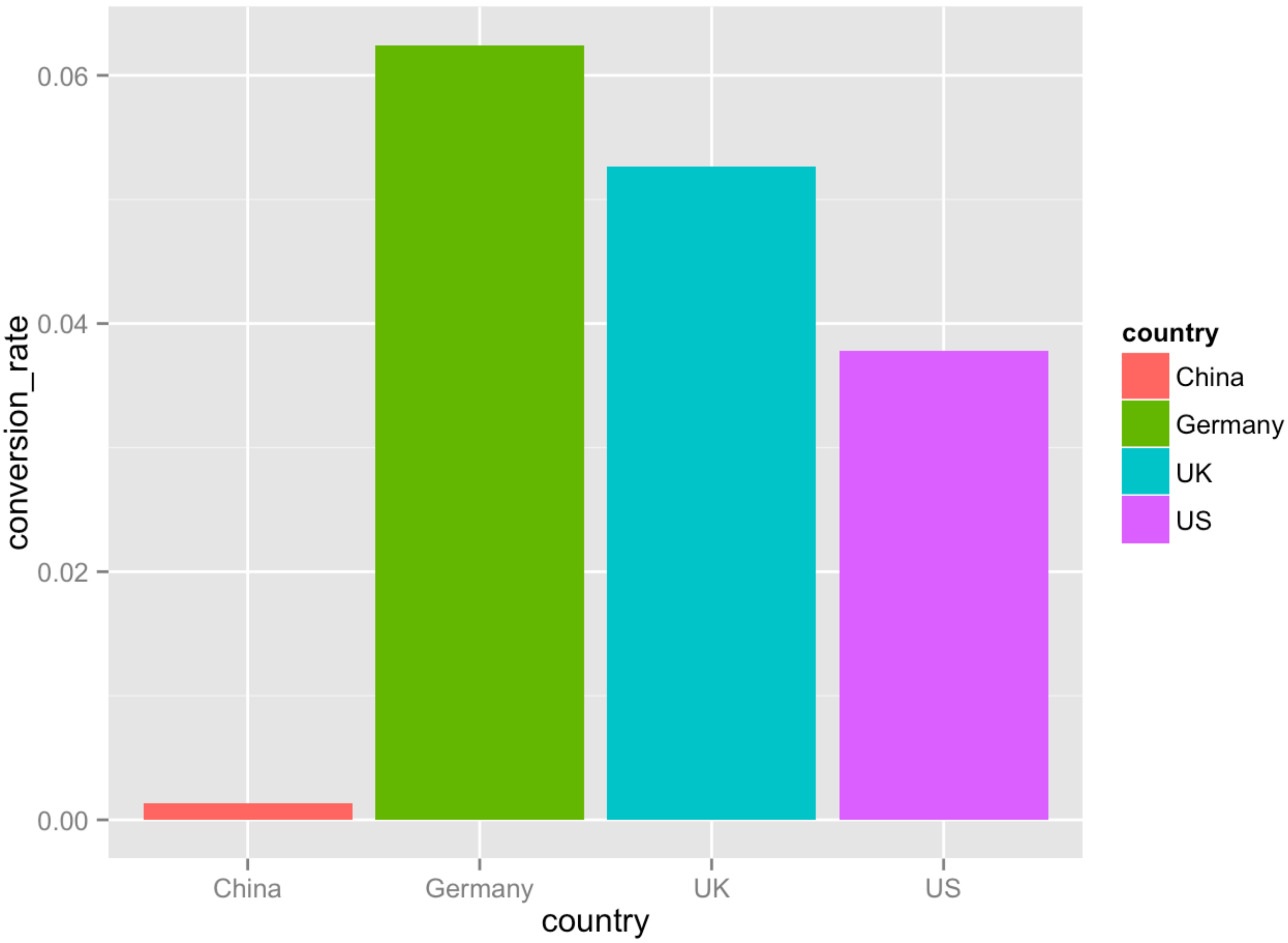
```
data = subset(data, age<80)
```

Now, let's quickly investigate the variables and how their distribution differs for the two classes. This will help us understand whether there is any information in our data in the first place and get a sense of the data.

Never start by blindly building a machine learning model. Always first get a sense of the data.

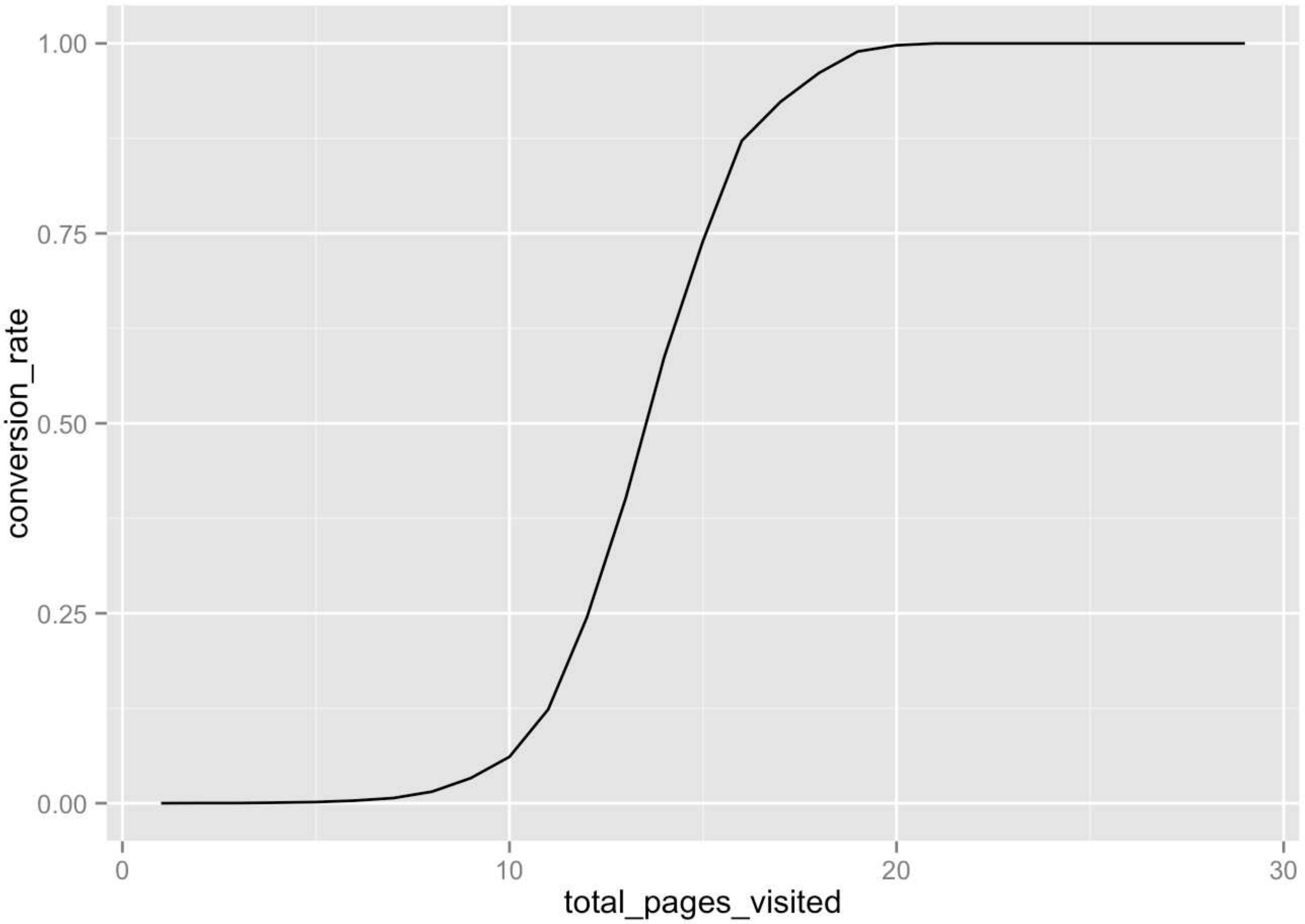
Let's just pick a couple of vars as an example, but you should do it with all:

```
data_country = data %>%
                      group_by(country) %>%
                       summarise(conversion_rate = mean(converted))

ggplot(data=data_country, aes(x=country, y=conversion_rate))+
           geom_bar(stat = "identity", aes(fill = country))
```



Here it clearly looks like Chinese convert at a much lower rate than other countries!

```
 data_pages = data %>%
               group_by(total_pages_visited) %>%
               summarise(conversion_rate = mean(converted))
qplot(total_pages_visited, conversion_rate, data=data_pages, geom="line")
```



Definitely spending more time on the site implies higher probability of conversion!

Focus on your strengths in the challenge. If visualization is your main strength, spend as much time as you wish on that and come up with something great. You might be hired as a great data scientist - visualization. If you have other strengths, spend more time on those. Take-home challenges are pretty open ended by design. By seeing where you spend more time, hiring managers can also understand your strengths, what you like doing the most, and where you would fit best.

Machine Learning

Let's now build a model to predict conversion rate. Outcome is binary and you care about insights to give product and marketing team some ideas. You should probably choose among the following models:

- Logistic regression
- Decision Trees
- RuleFit (this is often your best choice)

- Random Forest in combination with partial dependence plots

Pick the one you know the best. Don’t spend too much time optimizing it. Just explain why you picked it and say that with more time you would spend ~1 day trying other models/different params and you would pick the best.

Ex: I am going to pick a random forest to predict conversion rate. I pick a random forest cause: it usually requires very little time to optimize it (its default params are often close to the best ones) and it is strong with outliers, irrelevant variables, continuous and discrete variables. I will use the random forest to predict conversion, then I will use its partial dependence plots and variable importance to get insights about how it got information from the variables. Also, I will build a simple tree to find the most obvious user segments and see if they agree with RF partial dependence plots.

Firstly, “Converted” should really be a factor here as well as new_user. So let’s change them:

```
data$converted = as.factor(data$converted) # let's make the class a factor
data$new_user = as.factor(data$new_user) #also this a factor
levels(data$country)[levels(data$country)=="Germany"]="DE" # Shorter name, easier
to plot.
```

Create test/training set with a standard 66% split (if the data were too small, I would cross-validate) and then build the forest with standard values for the 3 most important parameters (100 trees, trees as large as possible, 3 random variables selected at each split).

```
train_sample = sample(nrow(data), size = nrow(data)*0.66)
train_data = data[train_sample,]
test_data = data[-train_sample,]

rf = randomForest(y=train_data$converted, x = train_data[, -ncol(train_data)],
                  ytest = test_data$converted, xtest = test_data[, -ncol(test_data)],

                  ntree = 100, mtry = 3, keep.forest = TRUE)

rf
```

```
## 
## Call:
##  randomForest(x = train_data[, -ncol(train_data)], y = train_data$converted, 
xtest = test_data[, -ncol(test_data)], ytest = test_data$converted,      ntree = 1
00, mtry = 3, keep.forest = TRUE) 
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
## 
##         OOB estimate of  error rate: 1.47%
## Confusion matrix:
##        0    1 class.error
## 0 201008  935 0.004630019
## 1   2133 4614 0.316140507
##                 Test set error rate: 1.41%
## Confusion matrix:
##        0    1 class.error
## 0 103589  468 0.004497535
## 1   1045 2406 0.302810779
```

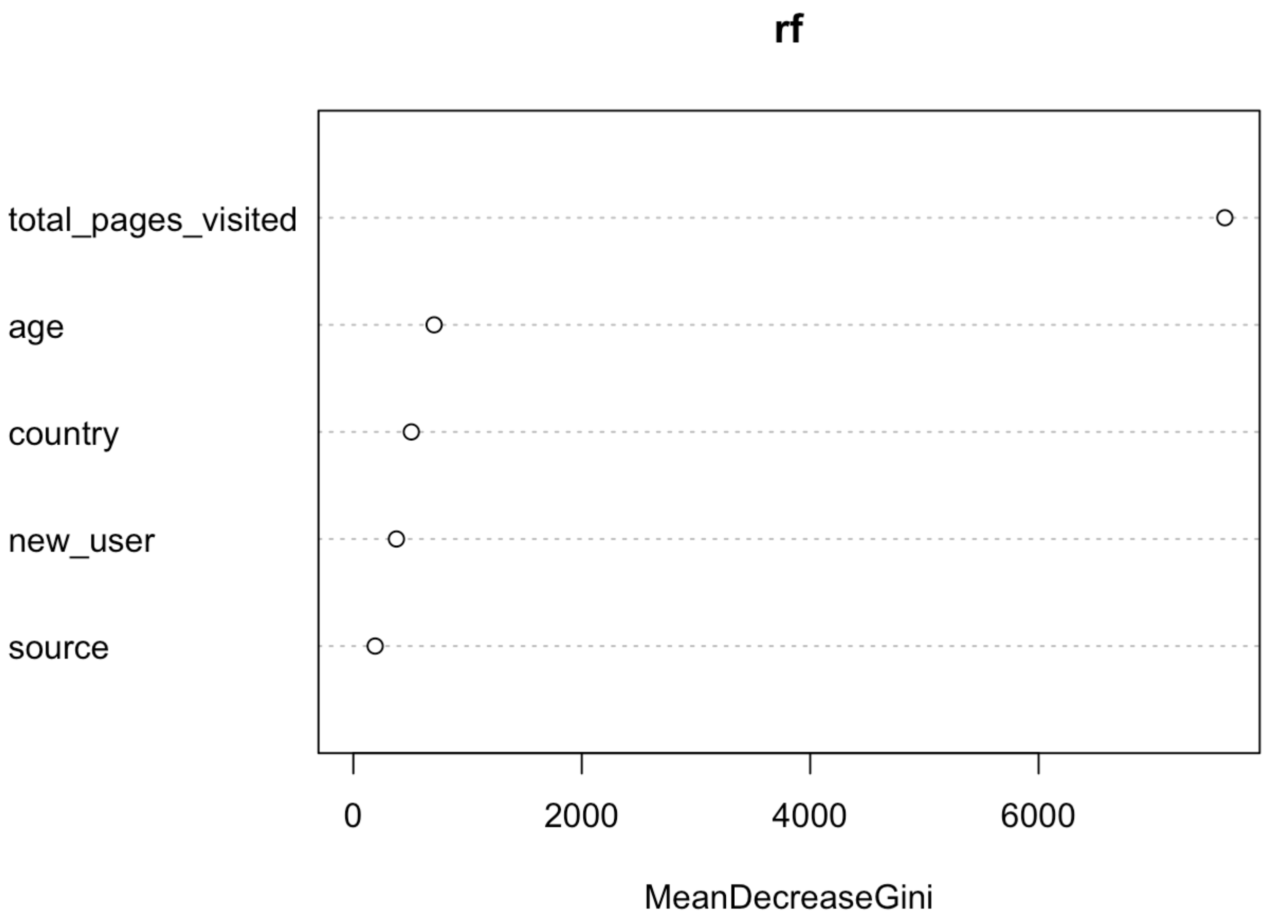
So, OOB error and test error are pretty similar: 1.5% and 1.4%. We are confident we are not overfitting. Error is pretty low. However, we started from a 97% accuracy (that's the case if we classified everything as "non converted"). So, 98.5% is good, but nothing shocking. Indeed, 30% of conversions are predicted as "non conversion".

If we cared about the very best possible accuracy or specifically minimizing false positive/false negative, we would also use ROCR and find the best cut-off point. Since in this case that doesn't appear to be particularly relevant, we are fine with the default 0.5 cutoff value used internally by the random forest to make the prediction. Again, if ROC and cut-off analysis is something you know very well, you should do it no matter what.

If you care about insights, building a model is just the first step. You need to check that the model predicts well and, if it does, you can now extract insights out of it.

Let's start checking variable importance:

```
varImpPlot(rf,type=2)
```



Total pages visited is the most important one, by far. Unfortunately, it is probably the least “actionable”. People visit many pages cause they already want to buy. Also, in order to buy you have to click on multiple pages.

Let’s rebuild the RF without that variable. Since classes are heavily unbalanced and we don’t have that very powerful variable anymore, let’s change the weight a bit, just to make sure we will get something classified as 1.

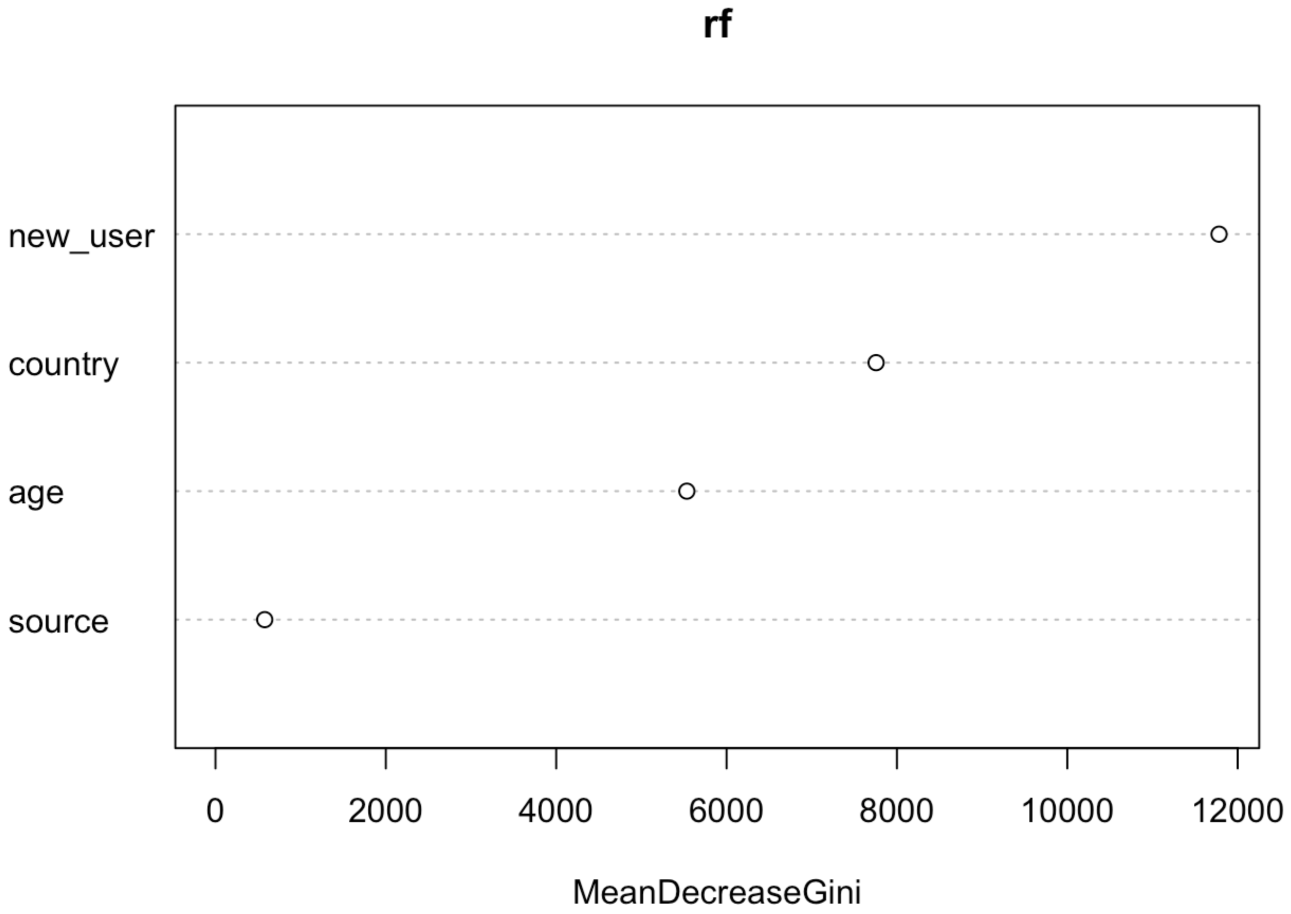
```
rf = randomForest(y=train_data$converted, x = train_data[, -c(5, ncol(train_dat
a))],
          ytest = test_data$converted, xtest = test_data[, -c(5, ncol(train_dat
a))],
                  ntree = 100, mtry = 3, keep.forest = TRUE, classwt = c(0.7,0.3))

rf
```

```
##
## Call:
##  randomForest(x = train_data[, -c(5, ncol(train_data))], y = train_data$convert
ed,      xtest = test_data[, -c(5, ncol(train_data))], ytest = test_data$converte
d,      ntree = 100, mtry = 3, classwt = c(0.7, 0.3), keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 14.09%
## Confusion matrix:
##        0     1 class.error
## 0 175596 26347   0.1304675
## 1   3051  3696   0.4522010
##                 Test set error rate: 13.91%
## Confusion matrix:
##       0     1 class.error
## 0 90703 13354   0.1283335
## 1  1601  1850   0.4639235
```

Accuracy went down, but that’s fine. The model is still good enough to give us insights.

Let’s recheck variable importance:

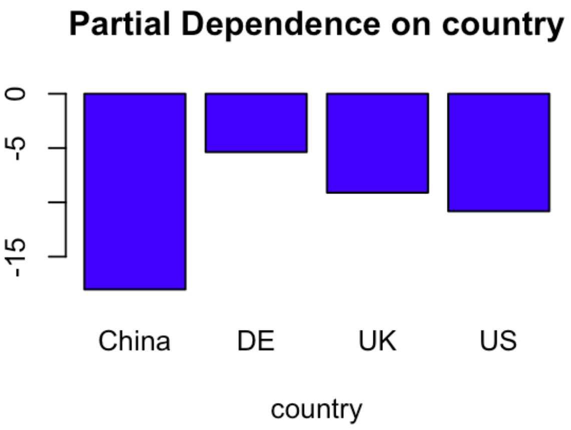
```
varImpPlot(rf,type=2)
```

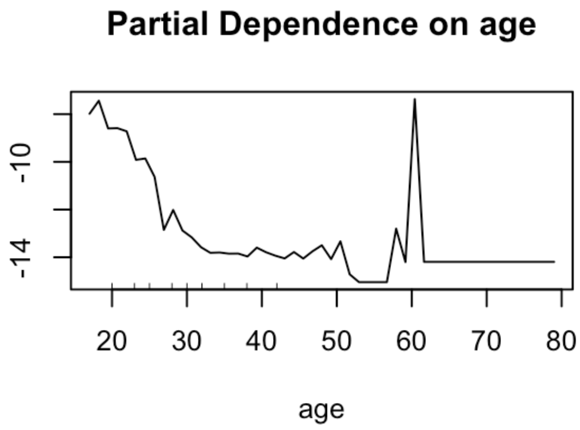


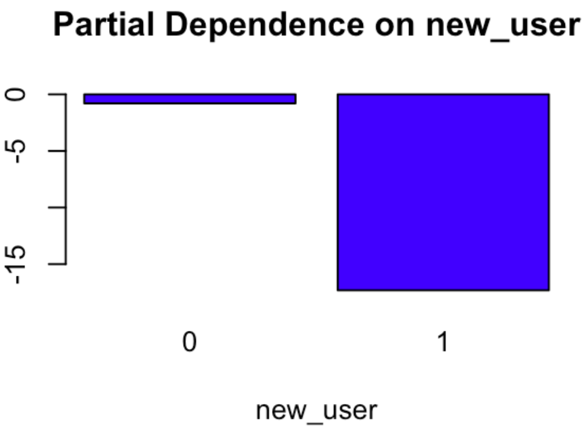
Interesting! New user is the most important one. Source doesn’t seem to matter at all.

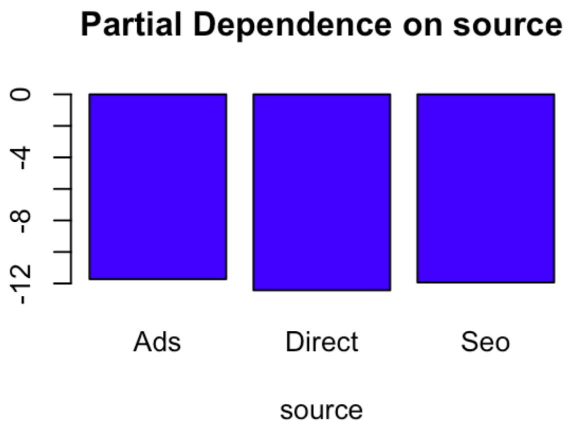
Let’s check partial dependence plots for the 4 vars:

```
op <- par(mfrow=c(2, 2))
partialPlot(rf, train_data, country, 1)
partialPlot(rf, train_data, age, 1)
partialPlot(rf, train_data, new_user, 1)
partialPlot(rf, train_data, source, 1)
```









In partial dependence plots, we just care about the trend, not the actual y value. So this shows that:

- Users with an old account are much better than new users
- China is really bad, all other countries are similar with Germany being the best
- The site works very well for young people and bad for less young people (>30 yrs old)
- Source is irrelevant

Let’s now build a simple decision tree and check the 2 or 3 most important segments:

```
tree = rpart(data$converted ~ ., data[, -c(5,ncol(data))],
             control = rpart.control(maxdepth = 3),
             parms = list(prior = c(0.7, 0.3))
             )
tree
```

```
## n= 316198
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 316198 94859.4000 0 (0.70000000 0.30000000)
##    2) new_user=1 216744 28268.0600 0 (0.84540048 0.15459952) *
##    3) new_user=0 99454 66591.3400 0 (0.50063101 0.49936899)
##      6) country=China 23094   613.9165 0 (0.96445336 0.03554664) *
##      7) country=DE,UK,US 76360 50102.8100 1 (0.43162227 0.56837773)
##       14) age>=29.5 38341 19589.5200 0 (0.57227507 0.42772493) *
##       15) age< 29.5 38019 23893.0000 1 (0.33996429 0.66003571) *
```

A simple small tree confirms exactly the random forest findings.

Some conclusions and suggestions:

1. The site is working very well for young users. Definitely let’s tell marketing to advertise and use marketing channel which are more likely to reach young people.

2. The site is working very well for Germany in terms of conversion. But the summary showed that there are few Germans coming to the site: way less than UK, despite a larger population. Again, marketing should get more Germans. Big opportunity.

3. Users with old accounts do much better. Targeted emails with offers to bring them back to the site could be a good idea to try.

4. Something is wrong with the Chinese version of the site. It is either poorly translated, doesn’t fit the local culture, some payment issue or maybe it is just in English! Given how many users are based in China, fixing this should be a top priority. Huge opportunity.

5. Maybe go through the UI and figure out why older users perform so poorly? From 30 y/o conversion clearly starts dropping.

6. If I know someone has visited many pages, but hasn’t converted, she almost surely has high purchase intent. I could email her targeted offers or sending her reminders. Overall, these are probably the easiest users to make convert.

As you can see, conclusions usually end up being about:

1. tell marketing to get more of the good performing user segments
2. tell product to fix the experience for the bad performing ones

Solution: Spanish Translation A/B Test

```
#libraries needed
require(dplyr)
require(rpart)
require(ggplot2)
```

```
#read data
user = read.csv("Translation_Test/user_table.csv")
test = read.csv("Translation_Test/test_table.csv")

#let's create one data set
length(unique(test$user_id)) == length(test$user_id) # are there dupes?
```

```
## [1] TRUE
```

```
length(unique(user$user_id)) == length(user$user_id) # are there dupes?
```

```
## [1] TRUE
```

```
length(user$user_id) - length(test$user_id) # everyone in one table also in the
other one?
```

```
## [1] -454
```

Looks like the user table is busted and we have some user ids missing. When joining, we have to be careful to do not lose the user ids in the test table, but not in the user table.

```
data = merge(test,user, by = "user_id", all.x = TRUE) # this way we don't lose data
data$date = as.Date(data$date)
summary(data)
```

```
##     user_id             date               source          device      
##  Min.   :      1   Min.   :2015-11-30   Ads   :181877   Mobile:201756  
##  1st Qu.: 249816   1st Qu.:2015-12-01   Direct: 90834   Web   :251565  
##  Median : 500019   Median :2015-12-03   SEO   :180610                  
##  Mean   : 499938   Mean   :2015-12-02                                  
##  3rd Qu.: 749522   3rd Qu.:2015-12-04                                  
##  Max.   :1000000   Max.   :2015-12-04                                  
##                                                                        
##  browser_language   ads_channel            browser         conversion     
##  EN   : 63137     Bing    : 13689   Android_App:155135   Min.   :0.00000  
##  ES   :377547     Facebook: 68425   Chrome     :101929   1st Qu.:0.00000  
##  Other: 12637     Google  : 68180   FireFox    : 40766   Median :0.00000  
##                   Other   :  4148   IE         : 61715   Mean   :0.04958  
##                   Yahoo   : 27435   Iphone_App : 46621   3rd Qu.:0.00000  
##                   NA's    :271444   Opera      :  6090   Max.   :1.00000  
##                                     Safari     : 41065                    
##       test          sex              age             country      
##  Min.   :0.0000   F   :188382   Min.   :18.00   Mexico   :128484  
##  1st Qu.:0.0000   M   :264485   1st Qu.:22.00   Colombia : 54060  
##  Median :0.0000   NA's:   454   Median :26.00   Spain    : 51782  
##  Mean   :0.4764                 Mean   :27.13   Argentina: 46733  
##  3rd Qu.:1.0000                 3rd Qu.:31.00   Peru     : 33666  
##  Max.   :1.0000                 Max.   :70.00   (Other)  :138142  
##                                 NA's   :454     NA's     :   454
```

First question is: check test results. But even before that, let’s make sure it is true Spain converts much better than the rest of LatAm countries.

```
data_conversion_country =  data %>%
                           group_by(country) %>%
                           summarize( conversion = mean(conversion[test == 0])) 
%>%# we check the old version
                           arrange (desc(conversion))

head(data_conversion_country)
```

```
## Source: local data frame [6 x 2]
## 
##       country conversion
##        (fctr)      (dbl)
## 1       Spain 0.07971882
## 2          NA 0.07755102
## 3 El Salvador 0.05355404
## 4   Nicaragua 0.05264697
## 5  Costa Rica 0.05225564
## 6    Colombia 0.05208949
```

Yes. Definitely true.

```
#a simple t-test here should work. We have collected ~0.5MM data and test/control split is ~50/50.
data_test = subset(data, country != "Spain") #nothing changed in Spain, so no point in keeping those users

t.test(data_test$conversion[data_test$test == 1], data_test$conversion[data_test$test == 0])
```

```
## 
##  Welch Two Sample t-test
## 
## data:  data_test$conversion[data_test$test == 1] and data_test$conversion[data_test$test == 0]
## t = -7.3539, df = 385260, p-value = 1.929e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.006181421 -0.003579837
## sample estimates:
##  mean of x  mean of y 
## 0.04341116 0.04829179
```

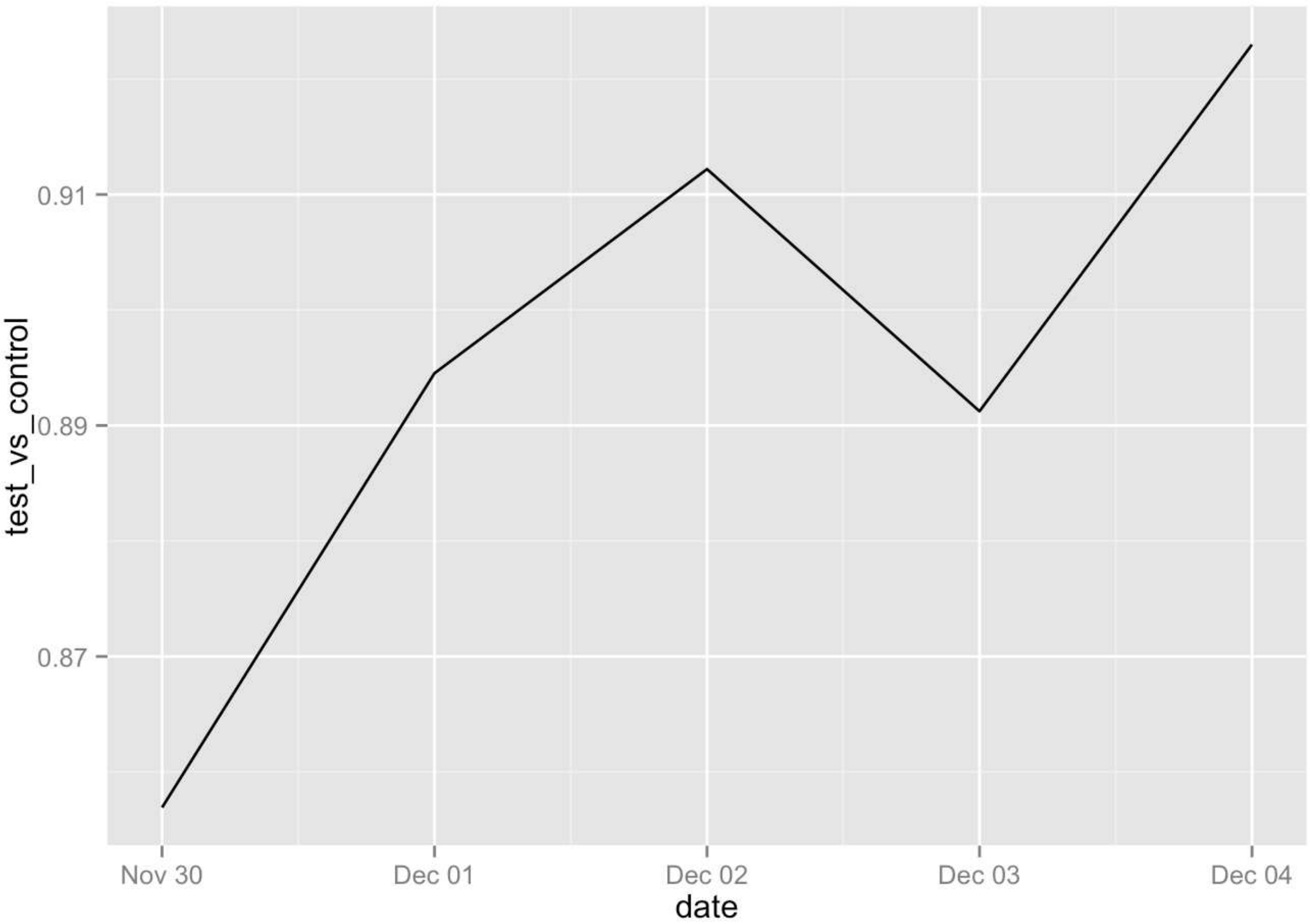
Mmh…not in the test are converting at 4.8% while users in the test just at 4.3%. That's a 10% drop, which would be dramatic if it were true. The most likely reason for weird A/B test results are:

1. We didn't collect enough data.
2. Some bias has been introduced in the experiment so that test/control people are not really random.

In data science, whenever results appear too bad or too good to be true, they are not true.

Firstly, let's plot day by day, to see if these weird results have been constantly happening or they just started happening all of a sudden.

```
data_test_by_day = data_test %>%
                   group_by(date) %>%
                   summarize(test_vs_control = mean(conversion[test==1])/
                                               mean(conversion[test==0])
                            )
qplot(date, test_vs_control, data= data_test_by_day, geom = "line")
```



From the plot, we notice a couple of things:

1. Test has constantly been worse than control and there is relatively little variance across days. That probably means that we do have enough data, but there was some bias in the experiment set up.
2. On a side note, we just ran it for 5 days. We should always run the test for at least 1 full week to capture weekly patterns, 2 weeks would be much better.

Time to find out the bias! Likely, there is for some reason some segment of users more likely to end up in test or in control, this segment had a significantly above/below conversion rate and this affected the overall results.

In an ideal world, the distribution of people in test and control for each segment should be the same. There are many ways to check this. One way is to build a decision tree where the variables are the user dimensions and the outcome variable is whether the user is in test or control. If the tree splits, it means that for given values of that variable you are more likely to end up in test or control. But this should be impossible! Therefore, if the randomization worked, the tree should not split at all (or at least not be able to separate the two classes well).

Let’s check this:

```
tree = rpart(test ~ .,data_test[,-8], #we remove conversion. Doesn't matter now.
             control = rpart.control(minbucket = nrow(data_test)/100, maxdepth =
2) # we only look for segments representing at least 1% of the populations.
             )
tree # here we are not too interested in predictive power, we are mainly using the
tree as a descriptive stat tool.
```

```
## n= 401085
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 401085 99692.820 0.5379757
##   2) country=Bolivia,Chile,Colombia,Costa Rica,Ecuador,El Salvador,Guatemala,Ho
nduras,Mexico,Nicaragua,Panama,Paraguay,Peru,Venezuela 350218 87553.970 0.4987693
*
##   3) country=Argentina,Uruguay 50867  7894.097 0.8079108 *
```

Looks very interesting. The randomization is perfect for the countries on one side of the split (country=Bolivia, Chile, Colombia, Costa Rica, Ecuador, EL Salvador, Guatemala, Honduras, Mexico, Nicaragua, Panama, Paraguay, Peru, Venezuela). Indeed, in that leaf the test/control ratio is 0.498! However, Argentina and Uruguay together have 80% test and 20% control! So let's check the test results after controlling for country. That is, we check for each country how the test is doing:

```
data_test_country =  data_test %>%
                     group_by(country) %>%
                     summarize( p_value = t.test( conversion[test==1],conversion[t
est==0])$p.value,
                                conversion_test = t.test( conversion[test==1],conv
ersion[test==0])$estimate[1],
                                conversion_control = t.test( conversion[test==1],c
onversion[test==0])$estimate[2]
                             ) %>%
                     arrange (p_value)

data_test_country
```

```
## Source: local data frame [16 x 4]
##
##         country   p_value conversion_test conversion_control
##          (fctr)     (dbl)           (dbl)              (dbl)
## 1        Mexico 0.1655437      0.05118631         0.04949462
## 2  El Salvador 0.2481267      0.04794689         0.05355404
## 3         Chile 0.3028476      0.05129502         0.04810718
## 4     Argentina 0.3351465      0.01372502         0.01507054
## 5      Colombia 0.4237191      0.05057096         0.05208949
## 6      Honduras 0.4714629      0.04753981         0.05090576
## 7     Guatemala 0.5721072      0.04864721         0.05064288
## 8     Venezuela 0.5737015      0.04897831         0.05034367
## 9    Costa Rica 0.6878764      0.05473764         0.05225564
## 10       Panama 0.7053268      0.04937028         0.04679552
## 11      Bolivia 0.7188852      0.04790097         0.04936937
## 12         Peru 0.7719530      0.05060427         0.04991404
## 13    Nicaragua 0.7804004      0.05417676         0.05264697
## 14      Uruguay 0.8797640      0.01290670         0.01204819
## 15     Paraguay 0.8836965      0.04922910         0.04849315
## 16      Ecuador 0.9615117      0.04898842         0.04915381
```

After we control for country, the test clearly appears non significant. Not a great success given that the goal was to improve conversion rate, but at least we know that a localized translation didn’t make things worse!

Solution: Employee Retention

```
#libraries needed
require(dplyr)
require(rpart)
require(ggplot2)
require(scales)
```

```
data = read.csv("employee_retention_data.csv") #read data set
str(data) #check the structure
```

```
## 'data.frame':    24702 obs. of  7 variables:
##  $ employee_id: int  13021 825355 927315 662910 256971
...
##  $ company_id : int  7 7 4 7 2 4 4 2 9 1 ...
##  $ dept       : Factor w/ 6 levels "customer_service",..: 1 5 5 1 2 2 1 1 4 6
...
##  $ seniority  : int  28 20 14 20 23 14 21 4 7 7 ...
##  $ salary     : num  89000 183000 101000 115000 276000 165000 107000 30000 1600
00 104000 ...
##  $ join_date  : Factor w/ 995 levels "2011-01-24","2011-01-25",..: 643 459 758
264 148 205 558 633 380 280 ...
##  $ quit_date  : Factor w/ 664 levels "2011-10-13","2011-10-14",..: 643 364 NA 2
29 428 267 NA NA 640 NA ...
```

```
data$company_id = as.factor(data$company_id) # this is a categorical var
data$join_date = as.Date(data$join_date) #make it a date
data$quit_date = as.Date(data$quit_date) #make it a date
summary(data) # everything seems to make sense, some simple plots would help doubl
e check that
```

```
##   employee_id      company_id               dept        seniority
##  Min.   :    36   1      :8486   customer_service:9180   Min.   : 1.00
##  1st Qu.:250134   2      :4222   data_science    :3190   1st Qu.: 7.00
##  Median :500793   3      :2749   design          :1380   Median :14.00
##  Mean   :501604   4      :2062   engineer        :4613   Mean   :14.13
##  3rd Qu.:753137   5      :1755   marketing       :3167   3rd Qu.:21.00
##  Max.   :999969   6      :1291   sales           :3172   Max.   :99.00
##                   (Other):4137
##      salary         join_date            quit_date
##  Min.   : 17000   Min.   :2011-01-24   Min.   :2011-10-13
##  1st Qu.: 79000   1st Qu.:2012-04-09   1st Qu.:2013-06-28
##  Median :123000   Median :2013-06-24   Median :2014-06-20
##  Mean   :138183   Mean   :2013-06-29   Mean   :2014-05-02
##  3rd Qu.:187000   3rd Qu.:2014-09-17   3rd Qu.:2015-03-27
##  Max.   :408000   Max.   :2015-12-10   Max.   :2015-12-09
##                                        NA's   :11192
```

Let’s answer this question: You should create a table with 3 columns: day, employee_headcount, company_id.

```
unique_dates = seq(as.Date("2011/01/24"), as.Date("2015/12/13"), by = "day") # cre
ate list of unique dates for the table
unique_companies = unique(data$company_id) #ceate list of unique companies
data_headcount = merge(unique_dates, unique_companies, by = NULL) #cross join so I
get all combinations of dates and companies. Will need it later.
colnames(data_headcount) = c("date", "company_id")

#now I get for each day/company, how many people quit/got hired on that day
data_join = data %>%
            group_by(join_date, company_id) %>%
            summarise(join_count = length(join_date))

data_quit = data %>%
            group_by(quit_date, company_id) %>%
            summarise(quit_count = length(quit_date))

#Now I left outer join with data_headcount.
#NA means no people were hired/quit on that day cause there is no match.

data_headcount = merge (data_headcount, data_join,
                        by.x = c("date", "company_id"),
                        by.y = c("join_date", "company_id"),
                        all.x = TRUE)

data_headcount = merge (data_headcount, data_quit,
                        by.x = c("date", "company_id"),
                        by.y = c("quit_date", "company_id"),
                        all.x = TRUE)
#replace the NAs with 0
data_headcount$join_count[is.na(data_headcount$join_count)] = 0
data_headcount$quit_count[is.na(data_headcount$quit_count)] = 0

#Now I need the sum by company_id. Data set is already ordered by date,
# so I can simply use dplyr to group by company_id and do cumsum

data_headcount = data_headcount %>%
            group_by(company_id) %>%
            mutate ( join_cumsum = cumsum(join_count),
                     quit_cumsum = cumsum(quit_count)
                    )
# finally, for each date I just take join_count - quit_count and I am done
data_headcount$count = data_headcount$join_cumsum - data_headcount$quit_cumsum
data_headcount_table = data.frame(data_headcount[, c("date", "company_id","coun
t")])

#Another way to do it would be with a for loop.
#While you often hear that you should avoid for loops in R as much as possible,
#in some cases you don't care that much about processing time, and you are
#willing to have something slower but more understandable.
#Other data scientists reading your code in future (or even yourself) will appreci
```

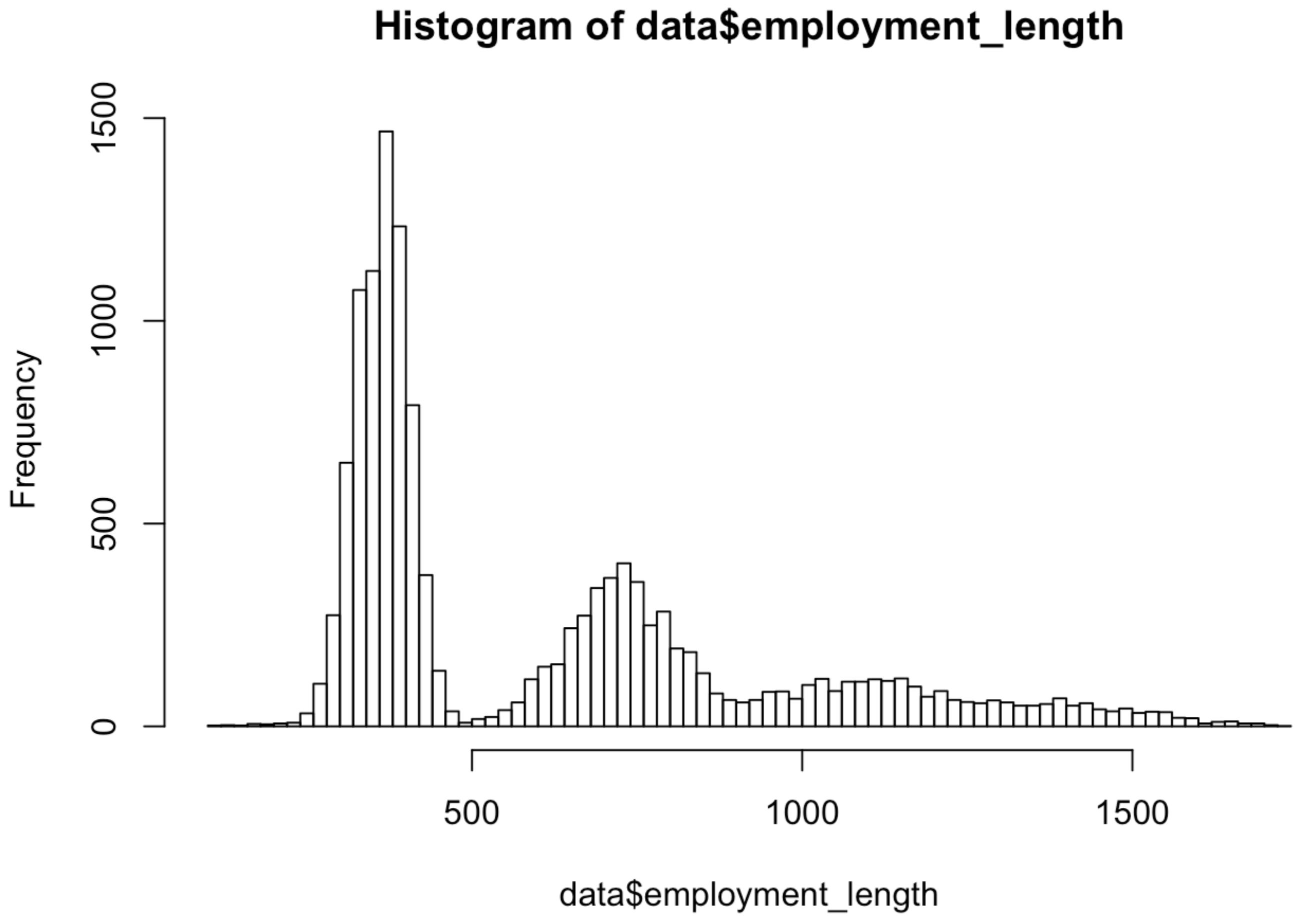
```
ate.

#Let's try with the for loop. Again here we optimize for future readibility!
# This is as slow as it can possibly be, but much clearer.

loop_cumsum = c() #intialize empty vector
loop_date = c()
loop_company = c()
for (i in seq(as.Date("2011/01/24"), as.Date("2015/12/13"), by = "day")) { #loop t
hrough all days
   for (j in unique(data$company_id)){ # loop through all companies
        tmp_join = nrow(subset(data, join_date <= i & company_id == j)) # count jo
ins until that day
        tmp_quit = nrow(subset(data, quit_date <= i & company_id == j)) # count qu
its
        loop_cumsum = c(loop_cumsum, tmp_join - tmp_quit )
        loop_date = c(loop_date, i)
        loop_company = c(loop_company, j)
   }
data_headcount_table_loop = data.frame(date = as.Date(loop_date, origin = '1970-0
1-01'), #fix R date
                                        company_id = loop_company,
                                        count = loop_cumsum)
}



#Let's finally check the two data sets are exactly the same:
identical (data_headcount_table[order(data_headcount_table[,1],
                                      as.numeric(as.character(data_headcount_tabl
e[,2] )))
                               ,],
            data_headcount_table[order(data_headcount_table[,1],
                                  as.numeric(as.character(data_headcount_table[,2]
)))
                                ,]
            )
```

```
## [1] TRUE
```

Now let’s try to understand employee retention. Here the main challenge is about feature engineering. That is, extract variables from the quitting_date column.

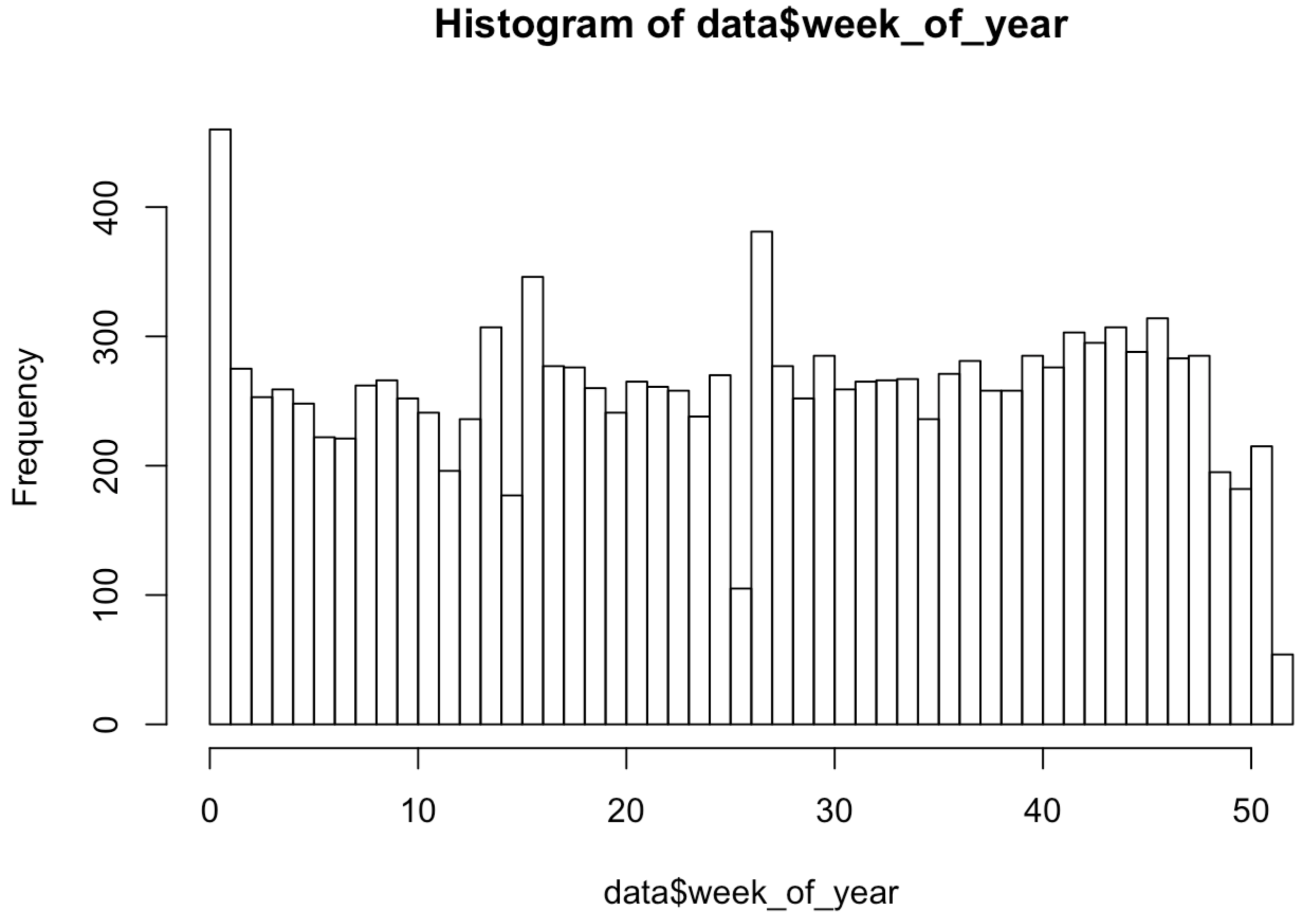
```
# how many days was she employed? This should matter.
#People might get bored in the same place for too long
data$employment_length = as.numeric(data$quit_date - data$join_date)

#In general, whenever you have a date, extract week of the year, and day of the we
ek. They tend to give an idea of seasonlity and weekly trends.
#In this case, weekly trends probably don't matter. So let's just get week of the
year
data$week_of_year = as.numeric(format(data$quit_date, "%U"))
```

```
#Let's plot employee length in days
hist(data$employment_length, breaks = 100)
```



Very interesting, there are peaks around each employee year anniversary!

```
#Let's plot week of the year
hist(data$week_of_year, breaks = length(unique(data$week_of_year)))
```



And it also peaks around the new year. Makes sense, companies have much more money to hire at the beginning of the year.

Now, let's see if we find the characteristics of the people who quit early. Looking at the histogram of employment_length, it looks like we could define early quitters as those people who quit within 1 yr or so. So, let's create two classes of users : quit within 13 months or not (if they haven't been in the current company for at least 13 months, we remove them).

```
#Create binary class
data = subset(data, data$join_date < as.Date("2015/12/13") - (365 + 31)) # only ke
ep people who had enough time to age
data$early_quitter = as.factor(ifelse( is.na(data$quit_date) | as.numeric(data$qui
t_date - data$join_date) > 396, 0, 1))
```

Let's now build a model. Here we can just care about: seniority, salary, dept and company. A simple decision tree is probably more than enough.

```
tree = rpart(early_quitter ~ .,data[, c("company_id", "dept", "seniority", "earl
y_quitter", "salary")], #put salary
             control = rpart.control(minbucket = 30, maxdepth = 3 , cp = 0.00000
1),
             parms = list(prior = c(0.5,0.5))
             )
tree #we are not too interested in predictive power, we are mainly using the tree
as a descriptive stat tool.
```

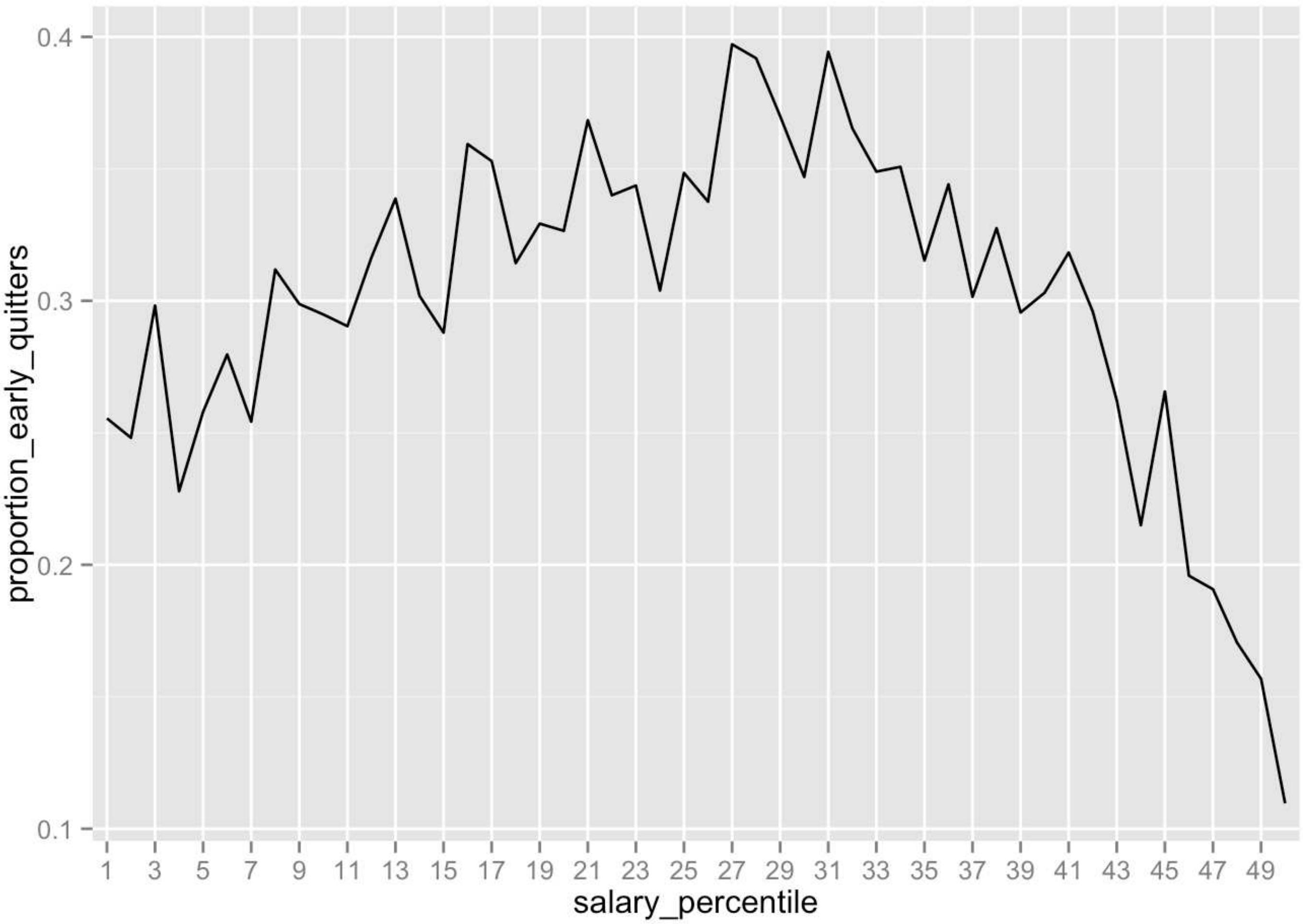
```
## n= 19270
## 
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
## 
## 1) root 19270 9635.0000 0 (0.5000000 0.5000000)
##   2) salary>=224500 2764  855.3351 0 (0.6528040 0.3471960) *
##   3) salary< 224500 16506 8026.7840 1 (0.4776014 0.5223986)
##     6) salary< 62500 2887 1249.7210 0 (0.5498859 0.4501141) *
##     7) salary>=62500 13619 6500.0510 1 (0.4632968 0.5367032) *
```

Not very surprising! Salary is what matters the most. After all, it probably has within it information about the other variables too. That is, seniority, dept and company impact salary. So salary carries pretty much all the information available.

It is interesting though that, looking at the terminal nodes, the way the tree split is: If salary between 62500 and 224500, the employee has higher probability of being an early quitter, otherwise she doesn’t. That means that **people who make a lot of money and very little are not likely to quit** (“little money” by Silicon Valley standards).

By plotting the proportion of early quitter by salary percentile, this becomes quite clear:

```
data$salary_percentile = cut(data$salary, breaks = quantile(data$salary, probs = s
eq(0, 1, 0.02)), 
                             include.lowest = TRUE, labels = 1:50)
data_proportion_by_percentile = data %>%
                                group_by(salary_percentile) %>%
                                summarize(proportion_early_quitters = length(earl
y_quitter[early_quitter==1])/length(early_quitter)
                                          )
qplot(salary_percentile, proportion_early_quitters, data=data_proportion_by_percen
tile, geom="line", group =1) + scale_x_discrete(breaks = seq(1,50, by=2))
```



Conclusions

1. Given how important is salary, I would definitely love to have as a variable the salary the employee who quit was offered in the next job. Otherwise, things like: promotions or raises received during the employee tenure would be interesting.
2. The major findings are that employees quit at year anniversaries or at the beginning of the year. Both cases make sense. Even if you don't like your current job, you often stay for 1 yr before quitting + you often get stocks after 1 yr so it makes sense to wait. Also, the beginning of the year is well known to be the best time to change job: companies are hiring more and you often want to stay until end of Dec to get the calendar year bonus.
3. Employees with low and high salaries are less likely to quit. Probably because employees with high salaries are happy there and employees with low salaries are not that marketable, so they have a hard time finding a new job.

Solution: Identifying Fraudulent Activities

```
#libraries needed
require(dplyr)
require(randomForest)
require(ROCR)
```

```
#read data
data = read.csv("Fraud/Fraud_Data.csv")
ip_addresses = read.csv("Fraud/IpAddress_to_Country.csv")

#are there duplicates?
nrow(data) == length(unique(data$user_id))
```

```
## [1] TRUE
```

```
#Let's add the country to the original data set by using the ip address
data_country = rep(NA, nrow(data))
for (i in 1: nrow(data)){
  tmp = as.character(ip_addresses [data$ip_address[i] >= ip_addresses$lower_boun
d_ip_address & data$ip_address[i] <= ip_addresses$upper_bound_ip_address,
                      "country"])
  if (length(tmp) == 1)  {data_country[i] = tmp}

}

data$country = data_country

data[, "signup_time"] = as.POSIXct(data[, "signup_time"], tz="GMT")
data[, "purchase_time"] = as.POSIXct(data[, "purchase_time"], tz="GMT")


summary(as.factor(data$country))
```

```
##                   United States                           China
##                           58049                           12038
##                           Japan                  United Kingdom
##                            7306                            4490
##               Korea Republic of                         Germany
##                            4162                            3646
##                          France                          Canada
##                            3161                            2975
##                          Brazil                           Italy
##                            2961                            1944
##                       Australia                     Netherlands
##                            1844                            1680
##              Russian Federation                           India
##                            1616                            1310
## Taiwan; Republic of China (ROC)                          Mexico
##                            1237                            1121
##                          Sweden                           Spain
##                            1090                            1027
##                    South Africa                     Switzerland
##                             838                             785
##                          Poland                       Argentina
##                             729                             661
##                       Indonesia                          Norway
##                             649                             609
##                        Colombia                          Turkey
##                             602                             568
##                        Viet Nam                         Romania
##                             550                             525
##                         Denmark                       Hong Kong
##                             490                             471
##                         Finland                         Austria
##                             460                             435
##                         Ukraine                           Chile
##                             429                             417
##                         Belgium      Iran (ISLAMIC Republic Of)
##                             409                             389
##                           Egypt                  Czech Republic
##                             359                             349
##                        Thailand                     New Zealand
##                             291                             278
##                          Israel                    Saudi Arabia
##                             272                             264
##                       Venezuela                         Ireland
##                             251                             240
##                  European Union                          Greece
##                             238                             231
##                        Portugal                         Hungary
##                             229                             211
##                        Malaysia                       Singapore
##                             210                             208
##                       Pakistan                     Philippines
##                            186                             177
##                       Bulgaria                         Morocco
##                            166                             158
##                        Algeria                            Peru
##                            122                             119
##                        Tunisia            United Arab Emirates
##                            118                             114
##                        Ecuador                       Lithuania
##                            106                              95
##                     Seychelles                           Kenya
##                             95                              93
##                     Kazakhstan                      Costa Rica
##                             92                              90
##                         Kuwait                        Slovenia
##                             90                              87
##     Slovakia (SLOVAK Republic)                         Uruguay
##                             86                              80
## Croatia (LOCAL Name: Hrvatska)                         Belarus
##                             79                              72
##                     Luxembourg                          Serbia
##                             72                              69
##                        Nigeria                          Latvia
##                             67                              64
##                         Panama                         Bolivia
##                             62                              53
##             Dominican Republic                          Cyprus
##                             51                              43
##                        Estonia                            Oman
##                             42                              41
##                     Bangladesh             Moldova Republic of
##                             37                              37
##                       Paraguay                         Georgia
##                             35                              32
##                      Sri Lanka          Bosnia and Herzegowina
##                             31                              30
##                    Puerto Rico                          Jordan
##                             30                              28
##                        Lebanon                     El Salvador
##                             28                              25
##                          Qatar                           Sudan
##                             25                              25
##                         Angola                       Macedonia
##                             24                              24
##           Syrian Arab Republic                      Azerbaijan
##                             24                              23
##                        Namibia                           Malta
##                             23                              22
##                        (Other)                            NA's
##                            550                           21966
```

Before jumping into building a model, think about whether you can create new powerful variables. This is called feature engineering and it is the most important step in machine learning. However, feature engineering is quite time consuming. In a take-home you should just give an idea of how you would do it and emphasize that with more time you would go deeper into it.

A few obvious variables that can be created here could be:

- Time difference between sign-up time and purchase time
- If the device id is unique or certain users are sharing the same device (many different user ids using the same device could be an indicator of fake accounts)
- Same for the ip address. Many different users having the same ip address could be an indicator of fake accounts
- Usual week of the year and day of the week from time variables

```r
#time difference between purchase and signup
data$purchase_signup_diff =  as.numeric(difftime(as.POSIXct(data$purchase_time, tz="GMT"), as.POSIXct(data$signup_time, tz="GMT"), unit="secs"))

#check for each device id how many different users had it
data = data %>%
       group_by(device_id) %>%
       mutate (device_id_count = n())

#check for each ip address how many different users had it
data = data.frame(data %>%
                  group_by(ip_address) %>%
                  mutate (ip_address_count = n())
                  )
#day of the week
data$signup_time_wd  = format(data$signup_time, "%A")
data$purchase_time_wd = format(data$purchase_time, "%A" )

#week of the yr
data$signup_time_wy   = as.numeric(format(data$signup_time, "%U"))
data$purchase_time_wy = as.numeric(format(data$purchase_time, "%U" ))

#data set for the model. Drop first 3 vars and device id.
data_rf = data[, -c(1:3, 5)]


#replace the NA in the country var
data_rf$country[is.na(data_rf$country)]="Not_found"
#just keep the top 50 country, everything else is "other"
data_rf$country = ifelse(data_rf$country %in% names(sort(table(data_rf$country), decreasing = TRUE ) )[51:length(unique(data_rf$country))], # after top 50 countries
                         "Other", as.character(data_rf$country)
                         )

#make class a factor
data_rf$class = as.factor(data_rf$class)

#all characters become factors
data_rf[sapply(data_rf, is.character)] <- lapply(data_rf[sapply(data_rf, is.character)], as.factor)

# train/test split
train_sample = sample(nrow(data_rf), size = nrow(data)*0.66)
train_data = data_rf[train_sample,]
test_data = data_rf[-train_sample,]

# a takehome is rarely the place where to optimize the model/the params. Choose something that makes sense, explain why it makes sense and then say that with more time you would try 2/3 different models and different params (specify which models you would try and why)
```

```
rf = randomForest(y=train_data$class, x = train_data[, -7],
                  ytest = test_data$class, xtest = test_data[, -7],
                  ntree = 50, mtry = 3, keep.forest = TRUE)
rf
```

```
##
## Call:
##  randomForest(x = train_data[, -7], y = train_data$class, xtest = test_data[,
-7], ytest = test_data$class, ntree = 50, mtry = 3, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.4%
## Confusion matrix:
##       0    1  class.error
## 0 90399   17 0.0001880198
## 1  4371 4946 0.4691424278
##                 Test set error rate: 4.24%
## Confusion matrix:
##       0    1  class.error
## 0 46542    3 6.445375e-05
## 1  2175 2659 4.499379e-01
```

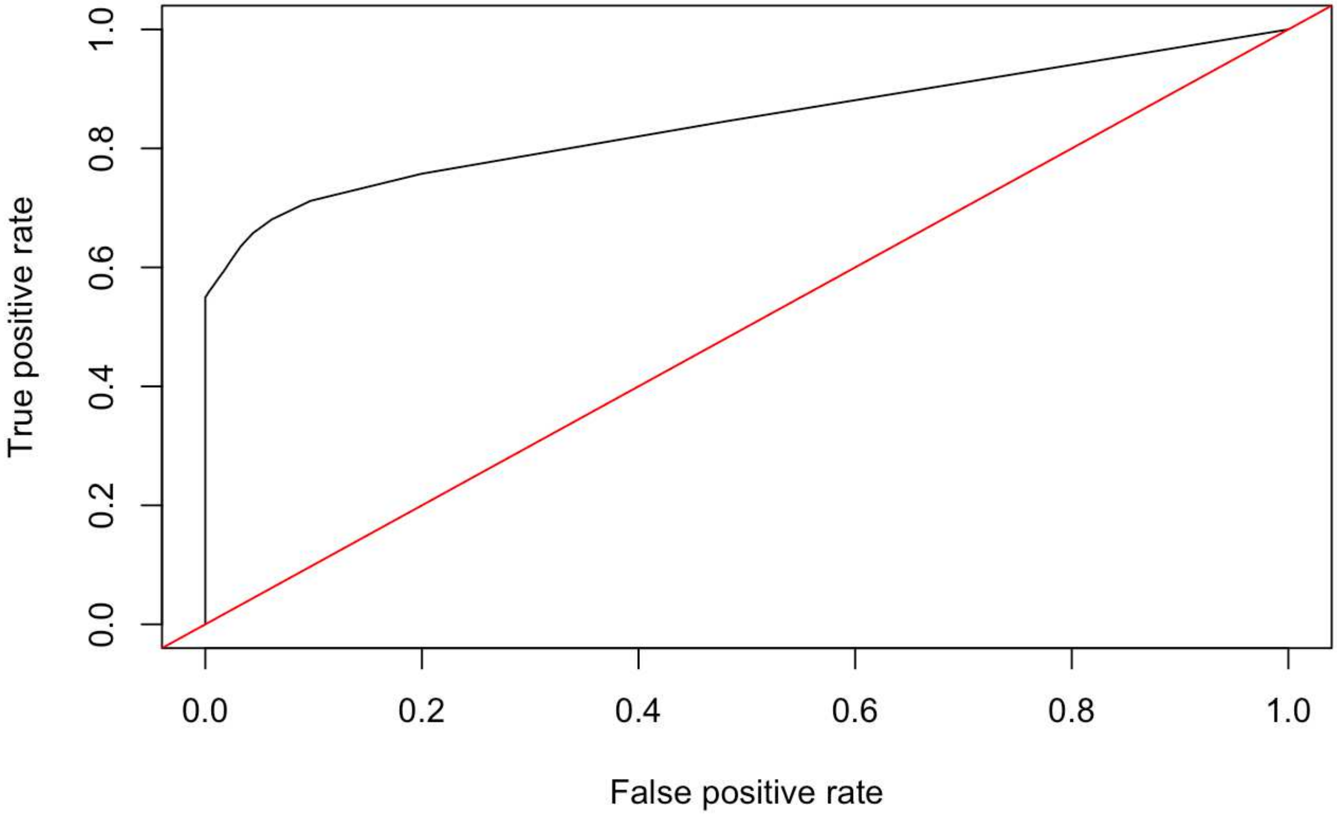
Since the challenge asks about false positive and false negative, this usually implies building the ROC and look for possible cut-off points. Luckily, in R this is very simple using the excellent ROCR package.

```
#let's combine in one data set model predictions and actual values.
#The first column are the actual classes in our test set and the second one the rf
predicted scores
rf_results = data.frame (true_values = test_data$class,
                         predictions = rf$test$votes[,2]
                         )
# we can double check that with a 0.5 cut-off we get the same results shown by rf
output(since randomforest internally uses 0.5)
identical( as.numeric(as.character(rf$test$predicted)), ifelse (rf_results$predict
ions>0.5, 1, 0))
```

```
## [1] TRUE
```

Cool. So they are the same! However, is really 0.5 the best possible cut-off? It really depends on what we are optimizing for (accuracy? true positive? true negative? etc.)

```
#this creates an object with all the information you can possibly need about how
d ifferent cutoff values impact all possible metrics: true positive, true negative,
false positive, false negative...
pred = prediction (rf_results$predictions, rf_results$true_values)

#now let's just plot the ROC and look at true positive vs false positive
perf = performance (pred, measure = 'tpr', x.measure = "fpr")
plot(perf) + abline(a=0, b=1, col = 'red') # the red line is randomness
```



```
## numeric(0)
```

Based on the ROC, if we care about minimizing false positive, we would choose a cut-off that would give us true positive rate of ~0.5 and false positive rate almost zero (this was essentially the random forest output). However, if we care about maximizing true positive, we will have to decrease the cut-off. This way we will classify more events as “1”: some will be true ones (so true positive goes up) and many, unfortunately, will be false ones (so false positive will also go up).

Understanding how ROC works is really important. So please review it if you are not familiar with it. In many practical ML applications, there is no reason to use the default 0.5 cut-off value

In terms of getting insights, this is similar to the conversion rate challenge, so we won’t redo it. The most effective strategy is again:

Look at random forest variable importance, plot partial dependence plots of the most important variables and finally build a simple and small tree and look at the main splits

Regarding “how to use this from a product perspective”: you now have a model that assigns to each user a probability of committing a fraud. You want to think about creating different experiences based on that. For instance:

1. If predicted fraud probability < *X*, the user has the normal experience (the high majority should fall here)
2. If *X* <= predicted fraud probability < *Z* (so the user is at risk, but not too much), you can create an additional verification step, like verify your phone number via a code sent by SMS or log in via Facebook.
3. If predicted fraud probability >= *Z* (so here is really likely the user is trying to commit a fraud), you can tell the user his session has been put on hold, send this user info to someone who reviews it manually and either blocks the user or decides it is not a fraud so the session is resumed.

This is just an example and there are many different ways to build products around some fraud score. However, it is important because it highlights that a **ML model is often really useful when it is combined with a product which is able to take advantage of its strengths and minimize its possible drawbacks (like false positives).**

V4