PROJETO STONE

Autor: Pedro Luiz Fernandes Assumpção

1) Objetivos

Objetivo primário

Desenvolver uma metodologia para avaliar a efetividade na comunicação com clientes que apresentam atraso na quitação do saldo devedor, avaliando os diferentes perfis para cada cliente.

Objetivos secundários:

1. Prever um fluxo de pagamento sustentável, com base no fluxo de caixa do cliente e levando em conta a sazonalidade.
2. Análise de correlações que possam trazer algum valor para o processo de comunicação atual.
3. Explorar correlações de dados internos da Stone com dados de fontes públicas.

2) Validações Iniciais

a) Fazendo joins de todas as tabelas (ajustanto as relações one-to-many) entre as chaves, teríamos as seguintes etapas:

1. *portfolio_geral* ⇒ 14265 clientes distintos
2. portfolio_geral inner join *portfolio_comunicados* ⇒ 11871 clientes distintos
3. merge anterior inner join *portfolio_tpv* ⇒ 11865 clientes distintos
4. merge anterior inner join com *portfolio_clientes* ⇒ 11865 clientes distintos

b) Ainda não aplicamos nenhum filtro de exclusão, mas será preciso. Seguem alguns exemplos:

Valores com DSP > DSPP na *portfolio_geral*

Valores de vendas negativas na tabela *portfolio_tpv*

3) Fluxo de Etapas Realizadas

Para poder mediar a efetividade das ações de comunicação, primeiramente iremos focar nossa análise nas tabelas de *portfolio_comunicados* e *portfolio_geral* , pois será a partir dessa tabela que iremos obter nossa variável target que será utilizada na nossa modelagem preditiva.

Para o carregamento e manipulação dos dados, utilizamos o pacote *data.table* da linguagem R. Esse pacote apresenta um processamento de dados otimizado e robusto, permitindo o processamento de grandes volumes de dados. Após carregar a 4 tabelas, fizemos uma ordenação pelas respectivas variáveis chaves de cada uma, a fim de garantir que os registros dos nossos conjuntos de dados estão organizados da forma correta.

Para assegurar a integridade dos dados, criamos uma função para percorrer o nosso dataset e identificar a existência/ou não de valores missing. Aplicamos essa função aos 4 datasets compartilhados e não encontramos nenhum valor missing.

Uma boa prática para verificar se as variáveis estão codificadas da forma correta, é fazer uma contagem dos valores distintos para cada variável. Em geral, quando temos muitos valores distintos, a variável se trata de uma variável numérica. Portanto, fizemos essa análise para todas as tabelas. Segue abaixo, como exemplo, as primeiras 5 variáveis com mais valores distintos para a tabela *portfolio_comunicados* :

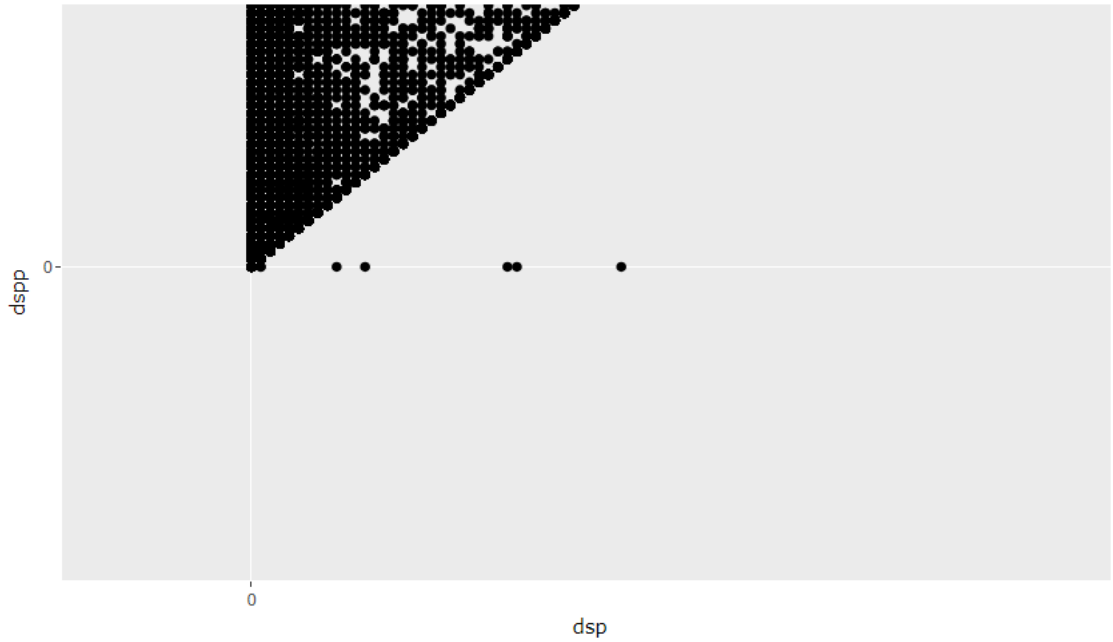
VARIÁVEL	TIPO	VALORES ÚNICOS
vlr_saldo_devedor_esperado	numeric	3050274
vlr_saldo_devedor	numeric	2879232
vlr_pgto_realizado	numeric	111430
contrato_id	character	14756
nr_documento	character	14265

Notamos que temos alguns clientes que possuem mais de um contrato em aberto. Por esse motivo, iremos criar uma variável que contabiliza o número de contratos disponíveis por clientes, conforme o script abaixo:

```
portfolio_geral[,var_created_contratos_por_cliente:=uniqueN(contrato_id), by = nr_documento]
```

Para cada variável, fizemos uma análise na consistência dos dados, com o intuito de identificar possíveis anomalias nos dados. Para facilitar as visualizações dos dados, obtivemos uma pequena amostra aleatória de 100k de linhas.

Encontramos no total de 706 registros de DSP que estão com valores inconsistentes (DSPP>DSP), como podemos notar pelo gráfico abaixo. Esse valores precisam ser tratados antes de executarmos qualquer modelagem ou análise. Ao todo, são 30 contratos que apresentaram essa inconsistência. A fim de identificar esses casos, criamos uma nova variável para com base nos valores de DSP e DSPP, como pode ser visto no script abaixo.



Criamos as categorias *inadimplente ativo*, *inadimplente inativo* e *ok* , para facilitar a identificação da situação do cliente conforme essa duas métricas de inadimplência (DSPP e DSP). Dessa forma, podemos identificar, por exemplo, os clientes que, mesmo não pagando o principalmente estão transacionando. O script abaixo mostrar como determinamos essa variável.

```
portfolio_geral[,var_created_situacao_inadimplencia:=fcase(

  dspp == 0 & dsp == 0, "ok",
  dspp > 0 & dsp  > 0, "inadimplente_inativo",
  dspp > 0 & dsp  == 0, "inadimplente_ativo",
  default = "erro"

)]
```

Notamos que para a tabela *portfolio_comunicados*, as chaves mostradas no diagrama de MER (*contrato_id* e *dt_ref_portfolio*) apresentam valores repetidos. Por conta disso, teremos de tratar essa tabela com intuito de ter apenas valores únicos nas chaves antes de interligar essa tabela com a *portfolio_geral*. Os passos para essa finalidade estão descritos em detalhes na tabela abaixo:

Notamos, com base na imagem abaixo, que a coluna *tipo_acao* contém informações que podem ser separadas em duas variáveis distintas, chamadas HSM e EMAIL. Para isso, vamos transformar a tabela para o formato wide, usando a função *dcast*.

contrato_id	dt_ref_portfolio	data_acao	tipo_acao	acao	status	conta
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	HSM	campanhaboletoquitado	ENTREGUE	4
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	HSM	campanhaobservacao	NAO ENTREGUE	4
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	EMAIL	campanhaboletoquitado	NAO ENTREGUE	4
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	EMAIL	campanhaobservacao	ENTREGUE	4

```
comunicados_wide = dcast(portfolio_comunicados,formula = contrato_id + dt_ref_portfolio + data_acao + acao ~ tipo_acao, value.var = 'status
```

Após a transformação da tabela, ainda temos alguns registros duplicados de chaves por conta do tipo de ação das duas réguas, que podem apresentar valores conflitantes. Iremos

contrato_id	dt_ref_portfolio	data_acao	acao	EMAIL	HSM	conta
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	campanhaboletoquitado	NAO ENTREGUE	ENTREGUE	2
000c35a61297edadc2842f6d5b4028e1	2021-04-24	2021-04-26	campanhaobservacao	ENTREGUE	NAO ENTREGUE	2
000c35a61297edadc2842f6d5b4028e1	2021-05-09	2021-05-11	campanhaparcelamento	ENTREGUE	ENTREGUE	2
000c35a61297edadc2842f6d5b4028e1	2021-05-09	2021-05-11	campanhaprenegativacao	NAO ENTREGUE	LIDO	2

Como nosso objetivo é tratar da efetividade da comunicação, iremos transformar a informação representada nas duas novas variáveis com base no critério exposto na tabela abaixo:

VALOR	NOVO VALOR
LIDO	OK
RESPONDIDO	OK
ENTREGUE	NAO_CONCLUSIVO
NAO ENTREGUE	FALHA

Com base nesse critério, podemos aplicar um filtro na tabela de comunicados, para remover todos os registros (linhas) que não tiveram êxito tanto na comunicação por nenhum dos canais (EMAIL e HSM). Essas linhas não terão poder preditivo nenhum pois não houve sucesso no contato com o cliente. O script para a remoção está descrito abaixo:

```
comunicados_wide = comunicados_wide[!(EMAIL=='NAO ENTREGUE' & HSM == 'NAO ENTREGUE')]
```

O próximo passo foi fazer um filtro na tabela de *portfolio_geral* para considerar somente os clientes que foram comunicados pelo menos uma vez com base na tabela de *comunicados_wide ,* atribuindo essa tabela final com o nome *portifolio_analisado*. Iremos performar um left join (*portifolio_analisado x comunicados_wide*) Como há chaves duplicadas na tabela *comunicados_wide,* haverá um aumento no total de registros (relação one-to-many), o que teremos tratar posteriormente (descobrimos que essa duplicidade se trata quando as réguas apresentam resultados conflitantes). Esse join foi armazenado na tabela *target_intermed*, de onde iremos extrair a nossa variável alvo (alguma relação entre comunicação e pagamento). Os códigos para performar o processo descrito estão abaixo. Inicialmente, vamos performar o join fazendo um filtro de colunas na tabela *comunicados_wide* apenas para as colunas *dt_ref_portfolio*,*contrato_id*, *acao.*

```
#Filtro de contatos
contratos_questionados = unique(comunicados_wide[,contrato_id])

#Aplicacao do filtro na tabela portifolio geral
portifolio_analisado = copy(portfolio_geral[contrato_id %in% contratos_questionados,.(contrato_id,dt_ref_portfolio,nr_documento,dsp,dspp,re
```

```
#Left Join
target_intermed = merge(portifolio_analisado[,.(contrato_id,dt_ref_portfolio,dsp,dspp)],comunicados_wide[,.(contrato_id,dt_ref_portfolio,ac
```

Como agora temos os valores de DSP e DSSP atrelados aos valores de ação da comunicação, iremos fazer a codificação conforme os dados expressos nas réguas. Criamos uma escala de campanha (de 1 a 6), está descrito abaixo:

```
target_intermed[,codificao_acao_comunicado:=fcase(

  dsp==5  & acao == 'campanhaobservacao',1,
  dsp==10 & acao ==  'campanhaparcelamento',2,
  dsp==15 & acao == 'campanhaboletoquitado',3,
  dsp==30 & acao == 'campanhaprenegativacao',4,
  dsp==60 & acao ==  'campanhanegativacao',5,
  dsp==90 & acao == 'campanhaboletoquitado',6,
  dspp==15 & acao == 'campanhaobservacao',1,
  dspp==30 & acao ==  'campanhaparcelamento',2,
  dspp==45 & acao == 'campanhaboletoquitado',3,
  default = NA

)]
```

Há dias em que os clientes foram aparentemente contactados 2 vezes, devido a conflitos entre as duas réguas. Visto isso, vamos fazer a suposição de que o contato mais drástico (conforme a escala acima) é o mais representativo (por exemplo, se o cliente recebeu no mesmo dia um aviso de negativação e outro de observação, será considerada apenas a negativação). Dessa forma, iremos fazer um filtro na nossa tabela, resolvendo o problema da chave duplicada. Seguem abaixo os códigos para executar essa operação:

```
#Selecao da acao mais drastica para o mesmo dia (caso haja 2)

target_intermed[,acao_mais_drastica:=max(codificao_acao_comunicado), by=.(contrato_id,dt_ref_portfolio)]

#Aplicacao da selecao mais drastica

target_intermed = target_intermed[contatos_por_dia==1 | (codificao_acao_comunicado==acao_mais_drastica)]

#Limpeza das linhas nulas e das colunas nao necessarias

target_intermed = target_intermed[!is.na(acao_mais_drastica)]
target_intermed = unique(target_intermed[,.(contrato_id,data_acao,contatos_por_dia,acao_mais_drastica)])
```

Note que agora temos exatamente qual foi a ação mais drástica informada ao cliente em relação ao dia exato da comunicação:

contrato_id	data_acao	contatos_por_dia	acao_mais_drastica
000180509391a5ac66ff83cae603ffb8	2020-12-29	1	1
000180509391a5ac66ff83cae603ffb8	2021-01-05	1	2
000c35a61297edadc2842f6d5b4028e1	2020-10-28	1	1
000c35a61297edadc2842f6d5b4028e1	2021-01-25	1	1
000c35a61297edadc2842f6d5b4028e1	2021-02-15	1	1
000c35a61297edadc2842f6d5b4028e1	2021-04-14	1	1
000c35a61297edadc2842f6d5b4028e1	2021-04-19	1	2
000c35a61297edadc2842f6d5b4028e1	2021-04-26	2	3

Agora iremos performar um novo left join , entre a tabela target_intermed () e a tabela previamente filtrada *portifolio_analisado* (relativa a portfolio_geral). Com base nisso, temos a junção da informação de comunicação com os dados financeiros do cliente, o que representa o nosso objetivo.

```
#Left Join entre os derivados das tabelas comunicado e geral

target = merge(portifolio_analisado,target_intermed, by.x = c('contrato_id','dt_ref_portfolio'),by.y = c('contrato_id','data_acao') , all.x
```

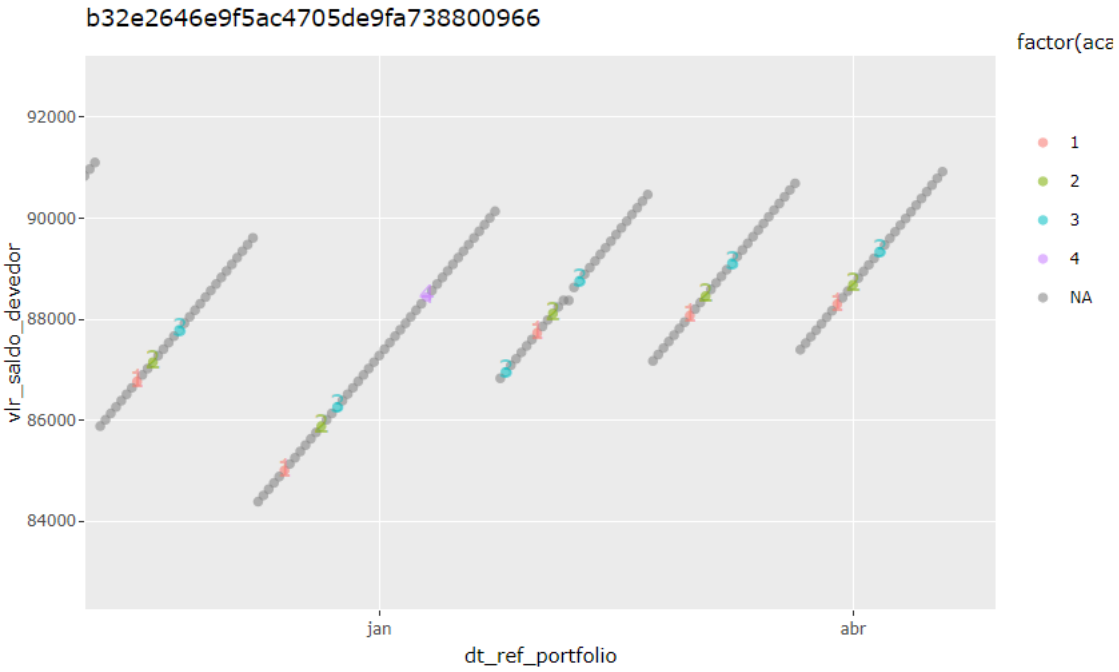
Finalmente, fizemos o left joint com as tabelas de *portfolio_tpv ,* obtendo a tabela *target* com 6478878 registros. A tabela de *portfolio_clientes* apresenta problema de chaves duplicadas. Por esse motivo iremos deixar ela para uma segunda etapa e focar apenas na tabela *target* , que é a junção das outras 3 tabelas e nos fornecerá insumos suficientes para uma compreensão dos dados iniciais.

```
#left join com a target com a portfolio_tpv
target = merge(target, portfolio_tpv, by.x = c('nr_documento','dt_ref_portfolio'), by.y = c('nr_documento','dt_transacao'),all.x = T, suffi

#ordenando a target
target = target[order(contrato_id,dt_ref_portfolio)]
```

Como veremos na seção de análise exploratória, iremos focar nos regimes/ciclos de pagamento dos clientes com base no **saldo devedor**. Acreditamos que esse seja o padrão que nos guiará para o objetivo de calibrar a régua de contato. O script abaixo vai a separação nos regimes para cada cada contrato, delimitando o ponto de separação como DSP=0.

```
#Determinacao dos regimes/ciclos de divida
target[,var_created_pagamento_efetuado:=ifelse(dsp==0,1,0)]
target[,regime_ativo:=rleid(var_created_pagamento_efetuado), by = contrato_id]
target[,regime_ativo:=ifelse(var_created_pagamento_efetuado==0 & shift(var_created_pagamento_efetuado, type = 'lead')!=0,regime_ativo, shif
```

A imagem abaixo (obtida com um filtro da tabela *target*), nos ilustra são esses ciclos para um cliente específico. No caso, podemos ver 5 regimes completos (5 retas). Os números representam o momento a campanha e, conforme vimos, 1 = observação, 2 = observação e assim por diante. Os regimes de pagamento podem apresentar padrões de repetição, e é nisso que iremos focar para determinar a nossa variável target.



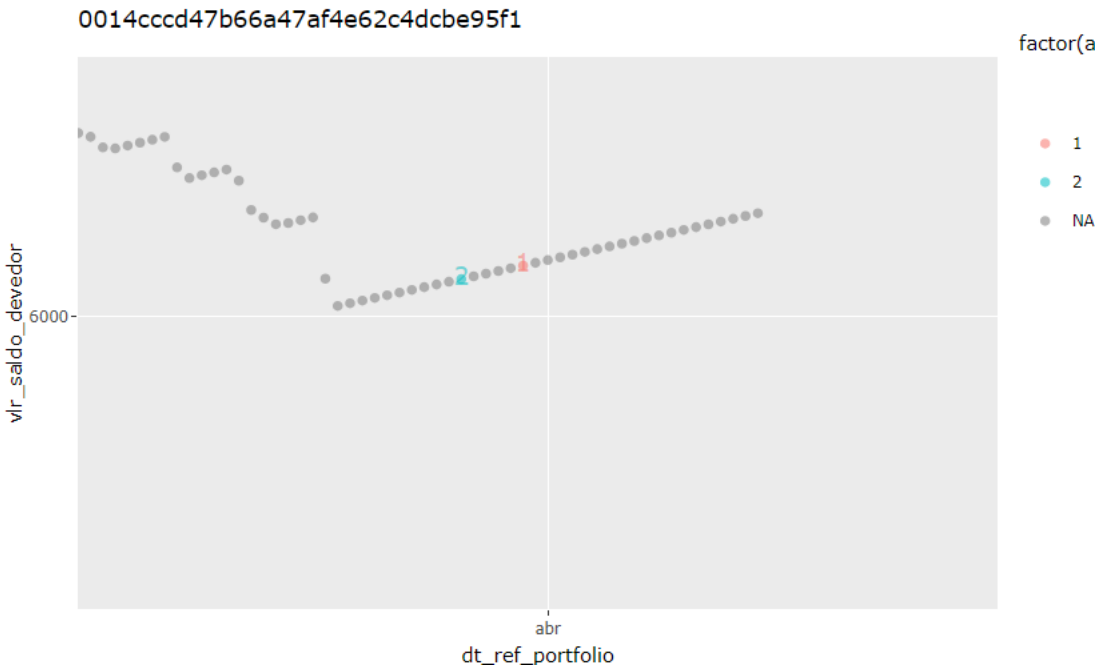
Agora, iremos agregar a tabela de target a nível de regime. O script abaixo exemplifica esse processo. Basicamente estamos substituindo as variáveis numéricas em somas, mínimos, máximos ou médias, com base nos regimes identificados para cada contrato. Queremos entender como essas variáveis se relacionam com as dos demais regimes e se ela tem algum valor preditivo com o próximo regime. Após o processo, a nova tabela *preditivas_agregadas* apresentou 3473279 registros.

```
#Fazendo uma copia da tabela target
preditivas_agregadas = copy(target)

#variaveis consideradas chave
var_unicas = c('contrato_id','regime_ativo','var_created_situacao_inadimplencia')

#tranformacao das variaveis
preditivas_agregadas[,var_agg_dt_ref_portfolio_min:=min(dt_ref_portfolio), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_dt_ref_portfolio_max:=max(dt_ref_portfolio), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_vlr_pgto_realizado_sum:=sum(vlr_pgto_realizado, na.rm = T),by = .(regime_ativo,contrato_id)]
preditivas_agregadas[,var_agg_vlr_saldo_devedor_min:=min(vlr_saldo_devedor), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_vlr_saldo_devedor_max:=max(vlr_saldo_devedor), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_vlr_saldo_devedor_esperado_min:=min(vlr_saldo_devedor_esperado), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_vlr_saldo_devedor_esperado_max:=max(vlr_saldo_devedor_esperado), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_dsp_max:=max(dsp), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_flag_transacao_med:=mean(flag_transacao), by = .(regime_ativo,contrato_id) ]
preditivas_agregadas[,var_agg_qtd_transacoes_sum:=sum(qtd_transacoes, na.rm = T),by = .(regime_ativo,contrato_id)]
preditivas_agregadas[,var_agg_vlr_tpv_sum:=sum(vlr_tpv, na.rm = T),by = .(regime_ativo,contrato_id)]

#selecao da variavel de situação de inadimplencia para o inicio do regime
preditivas_agregadas[,var_created_situacao_inadimplencia:=ifelse(dsp==0,var_created_situacao_inadimplencia,NA)]
```

```
preditivas_agregadas = preditivas_agregadas[!is.na(var_created_situacao_inadimplencia)]

#obtencao dos nomes das variaveis agregadas
variveis_agregadas = stringr::str_subset(colnames(preditivas_agregadas),'var_agg_')

#Aplicando o filtro de colunas, executando o distinct/unique e ordenando as colunas
filtro_colunas = c(var_unicas,variveis_agregadas)
preditivas_agregadas = unique(preditivas_agregadas[ ,colnames(preditivas_agregadas) %in% filtro_colunas, with=FALSE])
setcolorder(preditivas_agregadas,filtro_colunas)
```

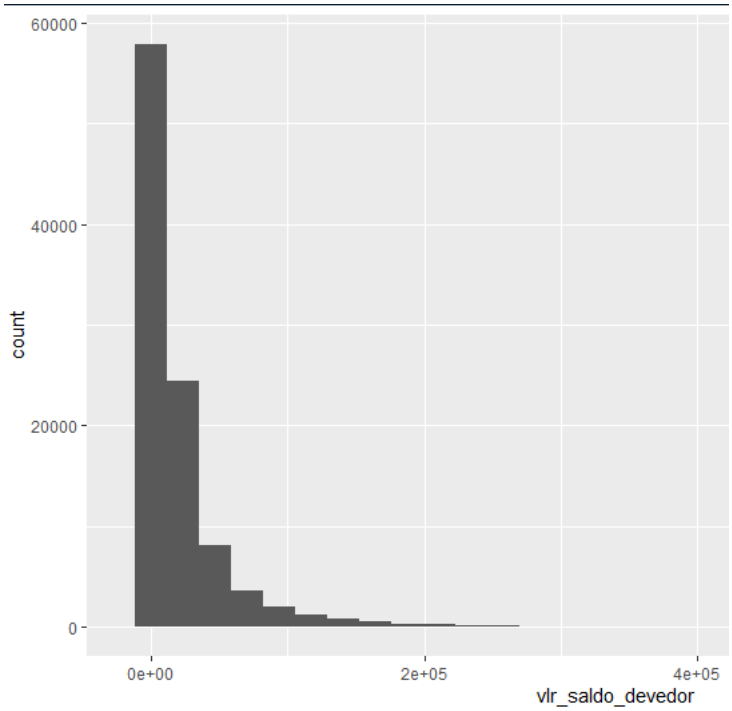
Agora que já temos a nossa tabela principal com as variáveis preditivas (*preditivas_agregadas*), iremos focar no cálculo da variável target, que será uma relação entre o tipo de campanha executada em cada regime. Nosso objetivo será focar na última comunicação de campanha de cada regime, pois essa será a mais provável de influenciar no pagamento (mudança de regime/reta). Porém, devido ao conflito de réguas, notamos alguns casos em que há comunicação menos drástica depois de uma mais drástica, conforme podemos observar no gráfico abaixo, onde a campanha 2 (parcelamento) foi informada antes da campanha 1 (observação). Por esse motivo, iremos criar duas variáveis que podem ter influenciado na decisão do cliente, a última campanha comunicada e a campanha mais crítica informada ao cliente. Na maioria das vezes elas são idênticas. AS informações serão armazenadas na tabela *campanha_regime*, conforme o script abaixo:

```
#extração de todos os registros da tabela target que apresentaram alguma comunicação
campanha_regime = unique(copy(target[!is.na(acao_mais_drastica),.(contrato_id,regime_ativo,acao_mais_drastica,dsp)]))
campanha_regime[,numero_dias_contactados_pagamento:=.N, by=.(contrato_id,regime_ativo)]
campanha_regime[,max_dsp:=max(dsp), by = .(contrato_id,regime_ativo)]
campanha_regime[,max_campanha:=max(acao_mais_drastica), by = .(contrato_id,regime_ativo)]


#vamos filtrar os casos o ultimo caso informado ao cliente e a campanha mais grave do ciclo
campanha_regime = campanha_regime[dsp==max_dsp | acao_mais_drastica==max_campanha]
#criacao das variaveis ultima campanha e campnha mais drastica
campanha_regime[dsp==max_dsp ,ultima_campanha_informada:=acao_mais_drastica]
campanha_regime[acao_mais_drastica==max_campanha ,principal_campanha_informada:=acao_mais_drastica]

campanha_regime[dsp==max_dsp ,dsp_ultima_campanha_informada:=dsp]
campanha_regime[acao_mais_drastica==max_campanha ,dsp_principal_campanha_informada:=dsp]

#ajuste dos valores missing
campanha_regime[,ultima_campanha_informada:=max(ultima_campanha_informada, na.rm = T), by = .(contrato_id,regime_ativo)]
campanha_regime[,principal_campanha_informada:=max(principal_campanha_informada, na.rm = T), by = .(contrato_id,regime_ativo)]
campanha_regime[,dsp_ultima_campanha_informada:=max(dsp_ultima_campanha_informada, na.rm = T), by = .(contrato_id,regime_ativo)]
campanha_regime[,dsp_principal_campanha_informada:=max(dsp_principal_campanha_informada, na.rm = T), by = .(contrato_id,regime_ativo)]


#Distinct
campanha_regime = unique(campanha_regime[,.(contrato_id,regime_ativo,dsp_ultima_campanha_informada,ultima_campanha_informada,dsp_principal_
```



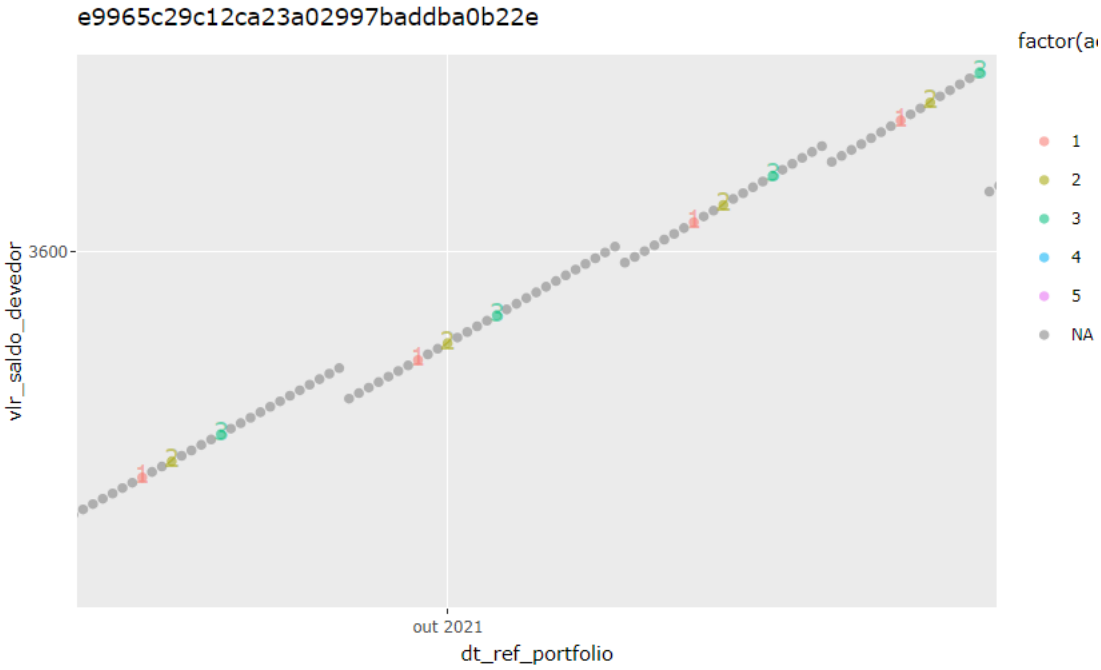
4) Insights (Análise exploratória)

Para as variáveis com valores financeiros, notamos uma distribuição com assimetria à direita, motivo pelo qual teremos de aplicar posteriormente uma transformação logarítmica nos dados. Segue abaixo o exemplo para a variável *vlr_saldo_devedor*.



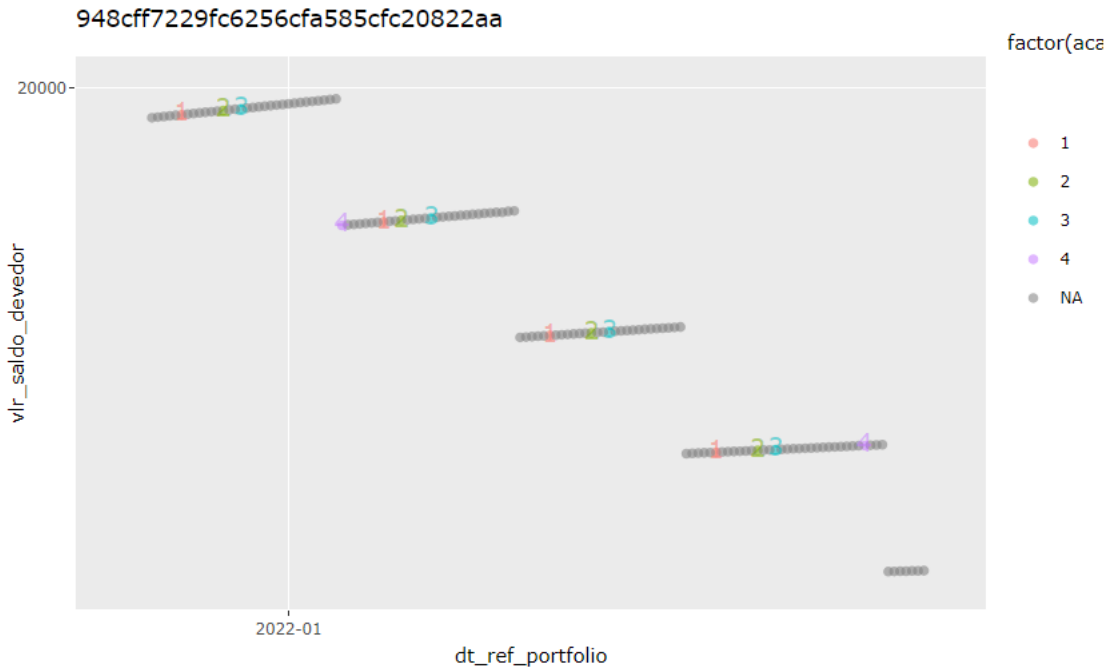
Para entender o comportamento das dívidas dos clientes, analisamos diversos gráficos de clientes distintos. Notamos que há um padrão nessa variável e ela opera por regimes. É fundamental entender como operam esses regimes (duração, dia esperado de reversão) e como eles transitam entre si (dado os regimes predecessores, qual será o regime esperado. Ou seja, qual é a probabilidade de que o cliente tenha um regime de diminuição ou de aumento nas dívidas).Abaixo temos alguns exemplos ilustrados por clientes:

4.1) Cliente aleatório 1 - padrão vlr_saldo_devedor



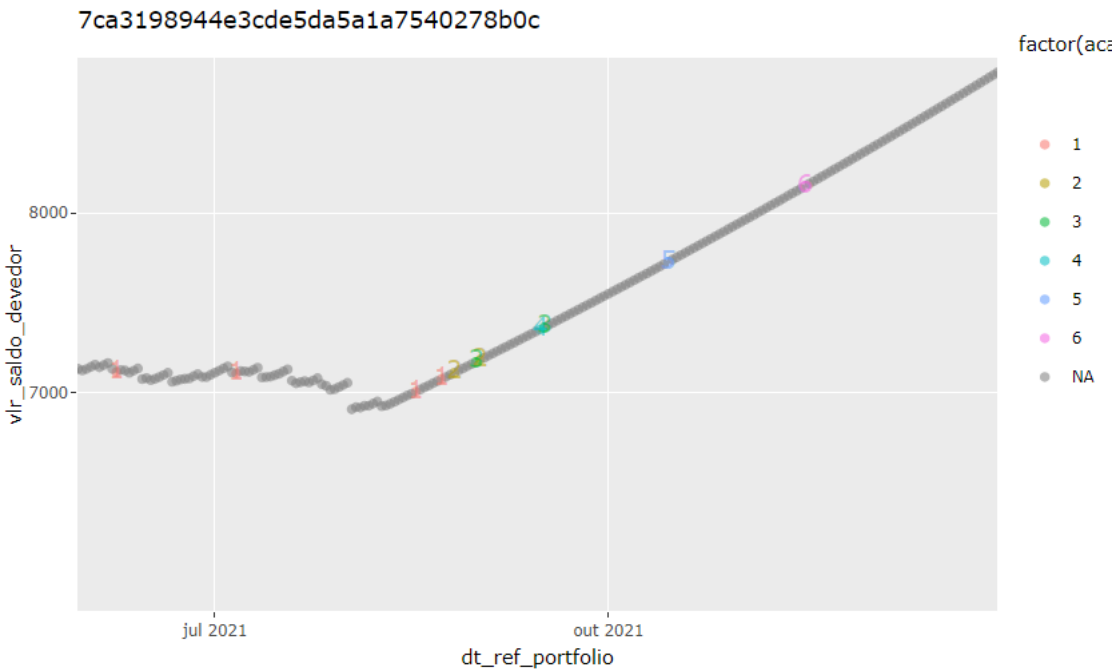
Pelo gráfico acima, notamos que esse cliente apresenta um aumento no fluxo de dívidas. Provavelmente os pagamentos não sejam suficientes para cobrir a dívida e tenhamos que reduzir um pouco os juros aplicado ao cliente. A reversão ocorre em espaços regularmente previsíveis e que podem ser determinados. No timeframe analisado, a reversão ocorre sempre após a campanha de quitação (número 3 em verde).

4.2) Cliente aleatório 2 - vlr_saldo_devedor



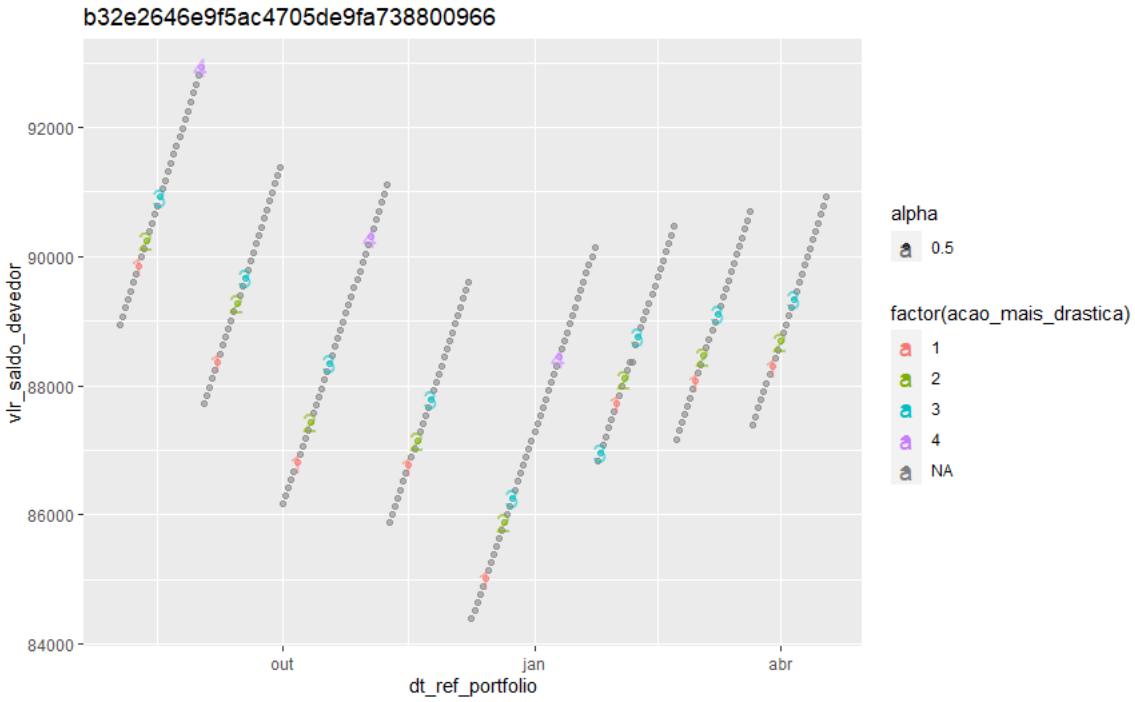
Esse cliente do gráfico, no período dados, se mostrou bastante reativo à negativação (número 4 em rosa).

4.3) Cliente aleatório 3 - vlr_saldo_devedor



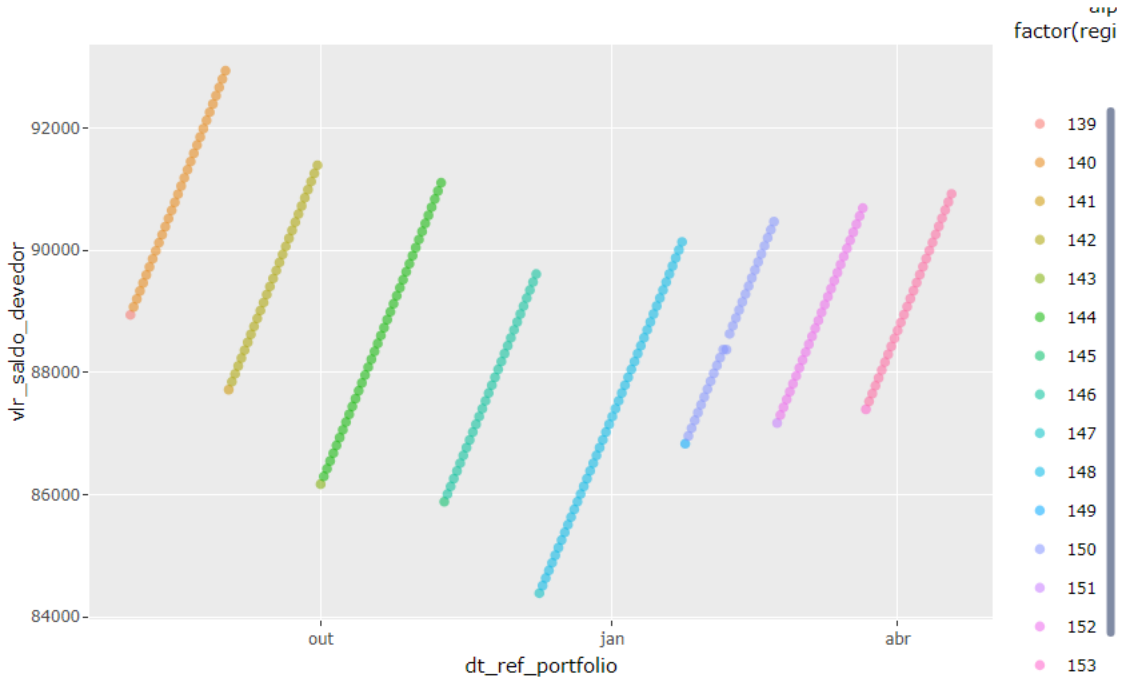
Cliente apresentou uma distorção no padrão de pagamento (regime mais longo que os anteriores) . Tivemos questionamentos repetidos por conta das duas réguas, o que poderia ser unificado. Nesse caso, poderíamos ter uma ação mais cedo com o cliente para entender o motivo do aumento da dívida e tomar alguma ação com antecedência à régua para ele liquidar sua dívida de forma sustentável.

4.3) Cliente aleatório 4 - Análise exploratória mais detalhada



Vamos delimitar os regimes de pagamento sempre que houver algum pagamento, ou seja, dsp = 0. Isso pode ser realizado simplesmente aplicando o script abaixo:

```
target[,var_created_pagamento_efetuado:=ifelse(dsp==0,1,0)]
target[,regime_ativo:=rleid(var_created_pagamento_efetuado), by = contrato_id]
```



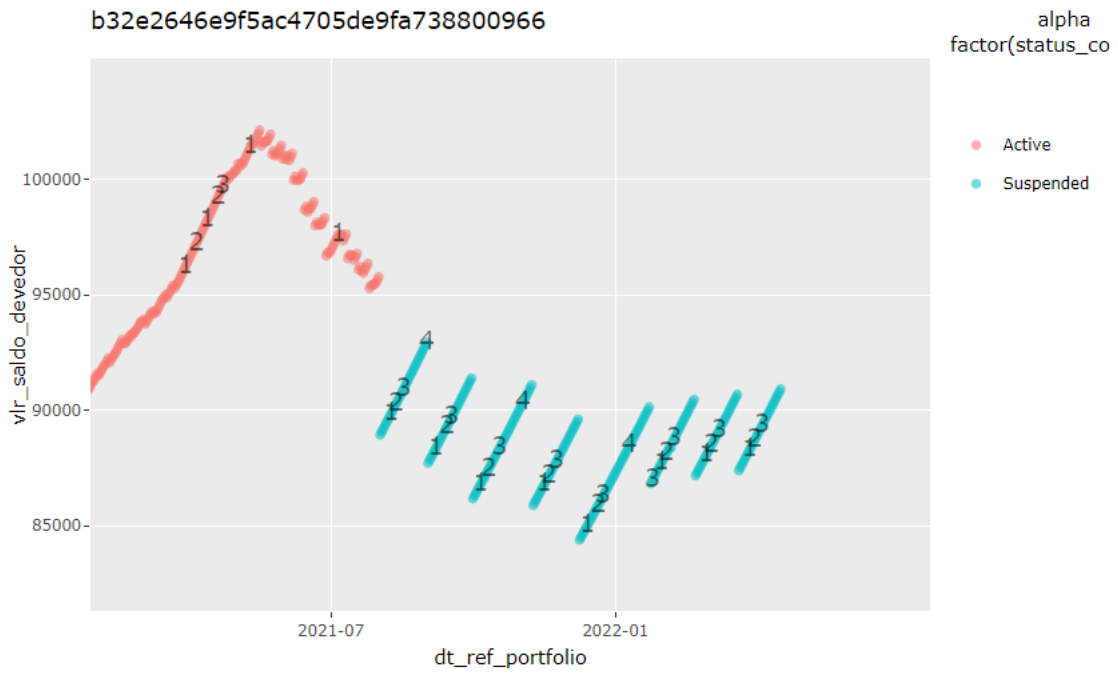
Pelo gráfico acima, notamos que a classificação em regimes de pagamento (números de 139 a 153) para o saldo devedor foi feita de forma satisfatória. Essa simples classificação nos dará uma grande vantagem ao analisar essa variável, pois podemos fazer diversas comparações entre os regimes que podem trazer valor para o negócio. como por exemplo:

1. Se nossa meta é reduzir o saldo devedor do cliente, podemos agora analisar quais são os fatores que fazem a transição de um estado para outro (dia antes do pagamento e dias após pagamento). Talvez a uma ação mais drástica (como a negativação) influenciou essa queda?
2. Dado um regime atual, calcular a probabilidade do cliente transitar para um outro regime (com maior ou menor dívida).
3. Prever se a dívida do cliente chegará a um limite irreversível, onde teremos de agir com o cliente para que as dívidas não atinjam o patamar de bola de neve, em que o pagamento se torna insustentável.
4. As possibilidades de análise são variadas, dependendo dos nossos objetivos.

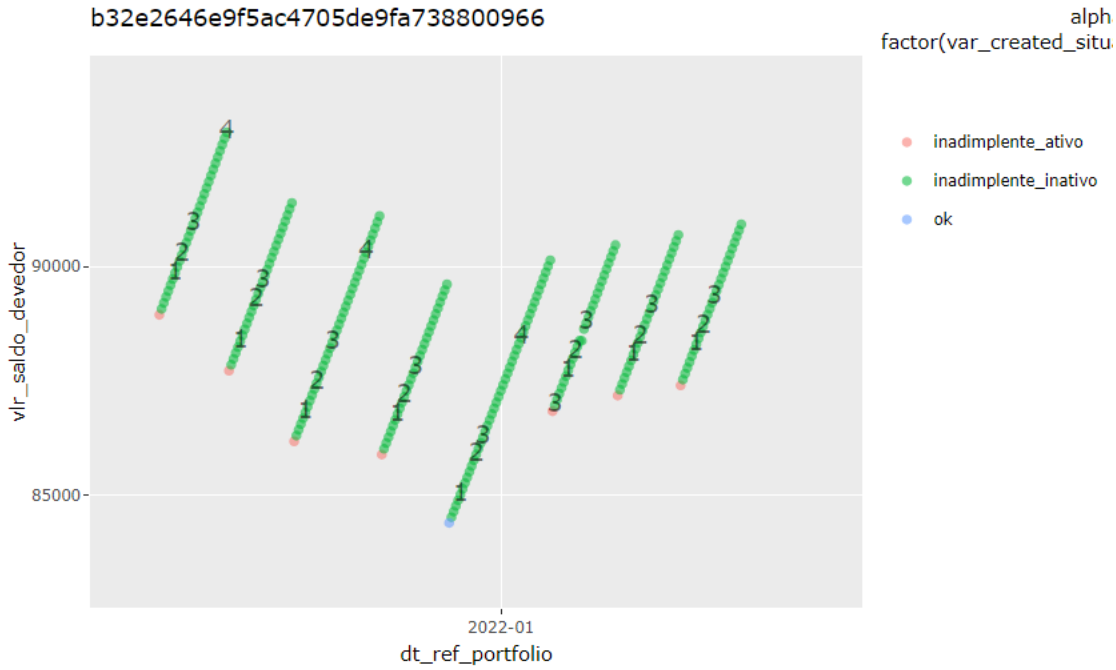
REGIME	INÍCIO REGIME	DURAÇÃO REGIME [DIAS]	ÚLTIMA CAMPANHA INFORMADA AO CLIENTE	DIA DO MÊS	MÊS	SEMANA ISO	DIA DA SEMANA
4	2021-08-02	30	4	2	8	31	segunda-feira
6	2021-09-02	28	3	2	9	35	quinta-feira
8	2021-10-01	38	4	1	10	39	sexta-feira
10	2021-11-09	29	3	9	11	45	terça-feira
12	2021-12-09	45	4	9	12	49	quinta-feira
14	2022-01-24	28	3	24	1	4	segunda-feira
16	2022-02-22	27	3	22	2	8	terça-feira
18	2022-03-22	27	3	22	3	12	terça-feira

A tabela acima mostra os regimes identificados. Note que temos diversos padrões que podem ser explorados e que podem nos ajudar como, a duração de cada regime, o dia do mês do início do regime etc. Dessa forma, para cada cliente individual, podemos comparar o regime atual e comparar com os regimes predecessores. Assim, entenderemos melhor cada cliente individual e poderemos regular a régua conforme os melhores critérios do nosso business.

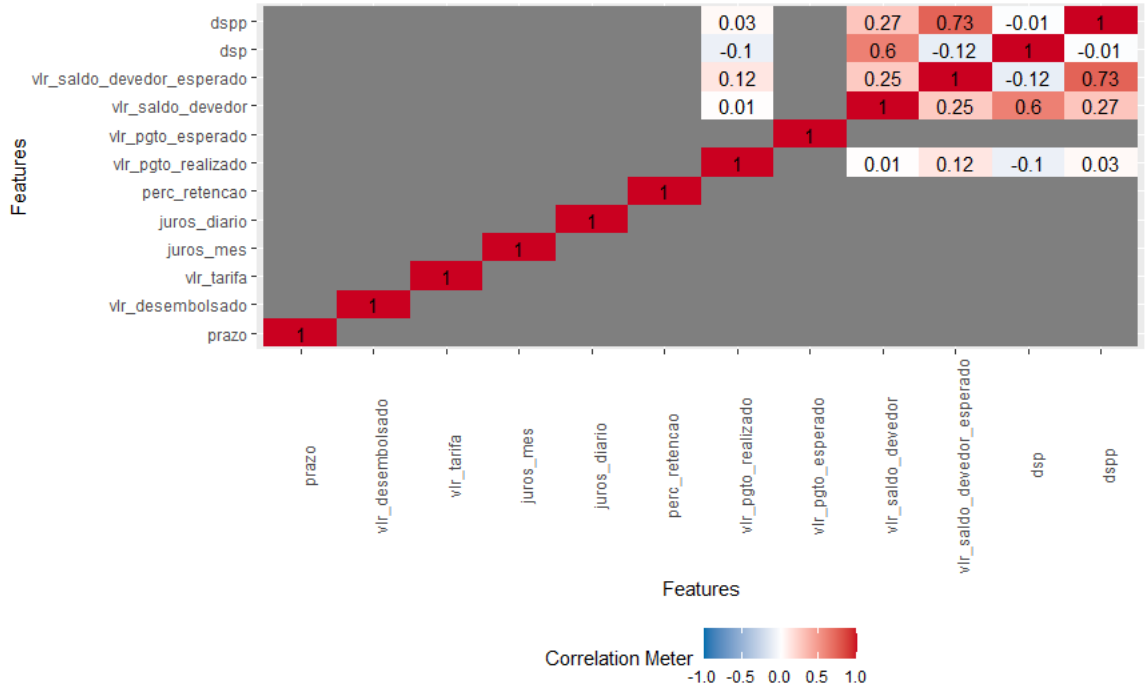
Após uma análise mais criteriosa, vimos que para a o período analisado o contrato estava suspenso., conforme podemos notar pela imagem abaixo. Será necessário fazer uma análise mais criteriosa e entender como funciona o fluxo de pagamento conforme o tipo de contrato.



Notamos também que sempre que podemos identificar a reversão do regime da dívida vigente simplesmente identificando se houve algum pagamento, pela DSP=0. Nesse caso, podemos simplesmente usar quando as DSP é nula para delimitar os regimes, sem a necessidade da complexidade que foi desenvolvida anteriormente. Ainda assim, podemos usar o número máximo de dias sem pagar como métrica do ciclo de pagamento do cliente, o que tentaremos reduzir o máximo possível.



Agora, iremos focar na seleção dos melhores atributos com base nesse cliente:



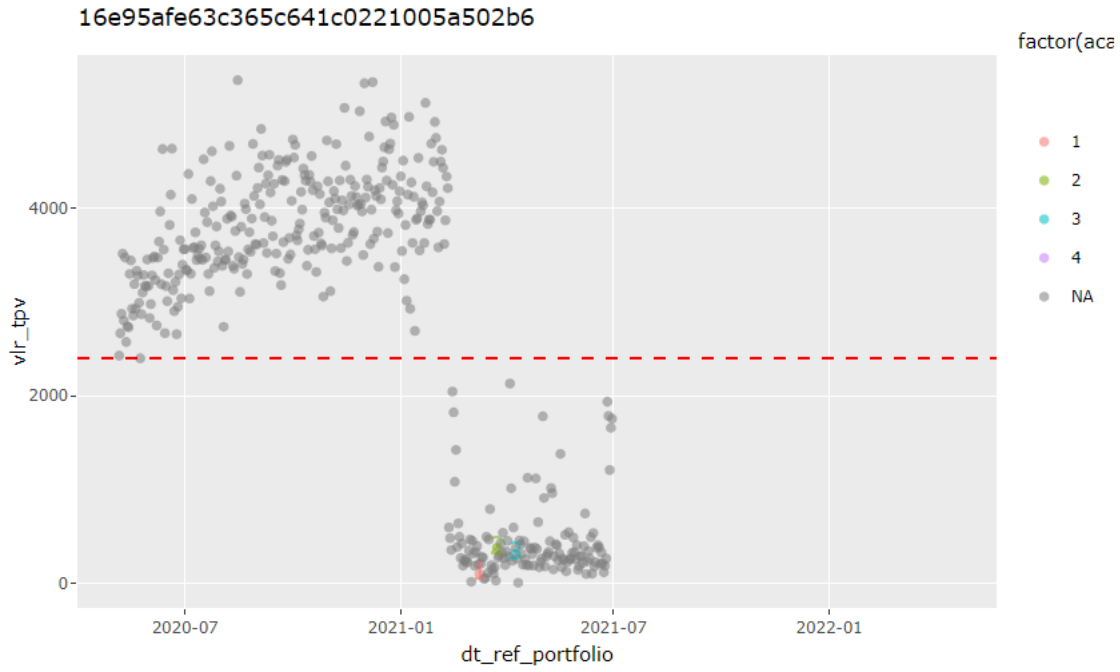
No gráfico acima, as variáveis que apresentam toda a linha/coluna em cinza apresentam variância zero, por isso provavelmente não serão úteis nas análises preditivas que serão desenvolvidas. Temos de averiguar se as variáveis estimadas (como a *vlr_pgto_esperado*) terão de ser removidas da análise, pois elas poderão enviesar o modelo.

5) Considerações sobre os objetivos secundários

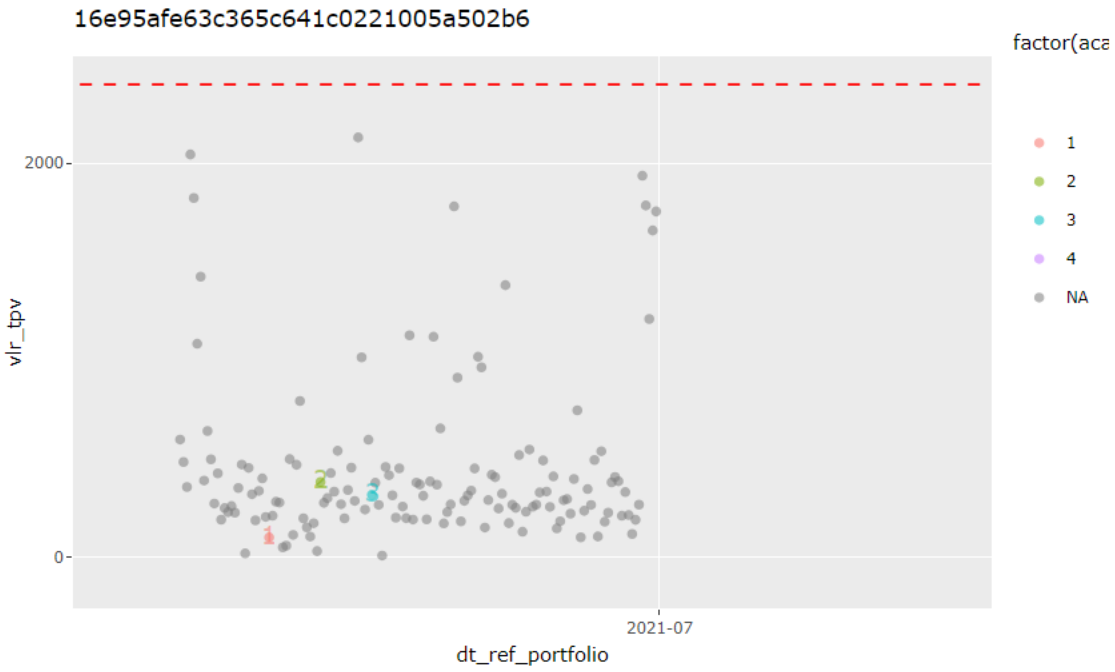
5.1) Determinação da sazonalidade via dados de TPV

A decisão sobre como identificar a sazonalidade não é complexa de ser implementada automaticamente, porém precisamos analisar alguns pontos.

Tomando como exemplo um cliente aleatório, temos as seguintes considerações:



Notamos que temos dois padrões nos dados bem definidos (delimitados pela linha tracejada em vermelho). Uma ideia seria aplicar um algoritmo de clusterização para identificar separar esses grupos antes de realizar a normalização dos dados, permitindo que a normalização seja feita somente dentro do cluster.



Considerando o segundo grupo, que está representado na figura acima, podemos facilmente identificar os pontos com anomalias nas vendas aplicando uma função para isso e, dessa forma, entender se se trata de algum feriado regional, incremento sazonal, influência das altas ou baixas temperaturas conforme o período do ano etc. Assim, poderemos identificar quais são os fatores que influenciam esse aumento/redução atípicos no fluxo de pagamento. Podemos depois definir se esse padrão se estende a demais categorias (cidade/segmento/feriado etc) e criar uma tabela padrão de sazonalidade. Dessa forma, será possível prever fluxos de caixa anômalos e tomar decisões (por exemplo, ajuste do percentual de retenção) com base nesse conhecimento. Essa informação pode ser usada também como insumo do modelo preditivo. Dessa forma, podemos ajustar o percentual de retenção.

5.2) Fontes de Dados Externas

Em um segundo momento, podemos pensar em incluir fontes de dados pública na tomada de decisão. Porém, para o desenvolvimento do nosso modelo de Machine Learning, esse não será o caminho ideal por dois motivos: Primeiro, porque vamos criar uma dependência externa em relação ao processo de tomada de decisão, o que pode ser prejudicial caso essas fontes não atualizem os dados. Segundo, porque iremos incluir mais estimadores no nosso modelo, o que pode enviesar ainda mais a nossa decisão, sendo ideal incluir o menor número de variáveis possíveis para tomarmos as decisões. Apenas complementar o processo de tomada de decisão conforme a previsão do modelo, podemos posteriormente validar algumas fontes de dados macroeconômicas, como por exemplo:

- Previsões da taxa Selic, IPCA, PIB informadas semanalmente no boletim Focus. Podemos automatizar a extração desses dados com linguagem R.
- Análise dos PMI.
- Fazer uma estimativa da necessidade de capital de giro para cada segmento, a fim suavizar o pagamento das empresas que têm ramos de atuação mais arriscados. Dados de balanço de empresas de capital aberto podem ser utilizados para isso.
- Taxa de desemprego do IBGE
- Monitoramento dos agregados monetários M1, M2 etc, fornecidos regularmente pelo Banco Central

6) Conclusões e Insights

Com base nisso, a proposta será primeiramente criar uma metodologia forma de classificar corretamente o regime de cada cliente. O script desenvonvido faz um bom trabalho, mas ainda precisa de alguns pequenos ajustes para alguns casos específicos. Após essa etapa, iremos fazer as devidas limpezas/transformações nos dados a fim de garantir que eles estejam preparados para a etapa de modelagem. Caso seguirmos por esse caminho, nossa proposta de solução será aplicar algum modelo de Machine Learning de Aprendizagem Supervisionada para prever a efetividade de contato com o cliente. Por exemplo, podemos identificar clientes que poderão ser acionados antes do que é expresso nas réguas. Dessa forma, teremos mais assertividade no contato com o cliente e poderemos acioná-lo em um momento crítico, no qual a comunicação será imprescindível. Poderemos também prever clientes que apresentaram uma padrão de inadimplência atípico, o que permitiria dar um enfoque mais para ele, sem depender da régua fixa.

Apesar de termos feito um progresso razoável no entendimento do problema e dos dados, ainda precisamos dedicar mais esforço para entender as relações entre as variáveis para propor a melhor solução possível.

Após o desenvolvimento e validação do modelo de Machine Learning, poderemos fazer a conexão com o banco de dados usando linguagem R e SQL para extrair novos dados e então alimentar nosso modelo. Assim, teremos uma forma ágil e eficaz de decidir quando e como se comunicar com o cliente.

Podemos criar, em paralelo, um outro algoritmo de Machine Learning para identificar atipicidade no ciclo/regime de pagamento de clientes semelhantes, a fim de atuar naqueles que apresentam um ciclo mais longo que outros clientes da mesma região, ramo etc.