UNIVERSIDADE FEDERAL DE CIÊNCIAS DA SAÚDE DE PORTO ALEGRE – UFCSPA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA SAÚDE

Thiago Galvão da Silva Paim

Análise Genômica de Uropatógenos: Características Associadas à Adaptação ao Trato Urinário de Isolados de *Escherichia coli*, *Enterococcus faecalis* e *Staphylococcus saprophyticus*

Porto Alegre
2017

Catalogação na Publicação

Paim, Thiago Galvão da Silva
Análise Genômica de Uropatógenos: Características Associadas à Adaptação ao Trato Urinário de Isolados de Escherichia coli, Enterococcus faecalis e Staphylococcus saprophyticus. / Thiago Galvão da Silva Paim. -- 2017.
175 f. : il., graf., tab. ; 30 cm.

Tese (doutorado) -- Universidade Federal de Ciências da Saúde de Porto Alegre, Programa de Pós-Graduação em Ciências da Saúde, 2017.

Orientador(a): Pedro Alves d'Azevedo.

1. Uropatógenos. 2. Infecções do Trato Urinário. 3. Genômica Comparativa. I. Título.

Sistema de Geração de Ficha Catalográfica da UFCSPA com os dados fornecidos pelo(a) autor(a).

Thiago Galvão da Silva Paim

Análise Genômica de Uropatógenos: Características Associadas à Adaptação ao Trato Urinário de Isolados de Escherichia coli, Enterococcus faecalis e Staphylococcus saprophyticus

Tese submetida ao Programa de Pós-Graduação em Ciências da Saúde da Fundação Universidade Federal de Ciências da Saúde de Porto Alegre como requisito para a obtenção do grau de Doutor.

Orientador: Dr. Pedro Alves d'Azevedo

Porto Alegre
2017

Agradecimentos

Gostaria de agradecer a minha esposa, Camila, meu amor. Esteve sempre ao meu lado, tanto nos momentos difíceis quanto nos bons. Te amo muito.

Gostaria de agradecer à minha mãe e irmã – Rita e Tamirez – pela força, companheirismo e apoio nestes anos. Um agradecimento especial ao meu pai, Nelson, sei que está olhando por mim e me guiando nas conquistas. Amo vocês.

Gostaria de agradecer ao professor Pedro d'Azevedo pelo apoio e conhecimentos proporcionados nesses anos de minha formação acadêmica. Apesar de não estar tão frequentemente no laboratório, sempre prontamente esteve disposto a me ajudar. Obrigado.

Agradecer a todos do Laboratório de Cocos Gram-positivos que contribuíram com este trabalho, além de colegas, são amigos que levarei para toda a vida. Abraços especiais ao Gustavo Sambrano, Renata Soares, Adriana Rossato e para Mariana Mott.

Agradecer aos técnicos de laboratório, pela ajuda, amizade e paciência no decorrer desses anos.

A todos, um caloroso abraço.
Obrigado.

SUMÁRIO

RESUMO

Introdução. As infecções do trato urinário (ITU) são fontes de morbidade principalmente em mulheres e extremos de idade. Os principais micro-organismos agentes etiológicos das ITUs são *Escherichia coli*, *Enterococcus faecalis* e *Staphylococcus saprophyticus*. Esses uropatógenos apresentam um repertório de fatores de virulência que conferem vantagem adaptativa ao trato urinário como aderência e invasão do uroepitélio, além de evasão do sistema imune do hospedeiro. Quanto ao hospedeiro, todas as espécies apresentam como reservatório o trato gastrointestinal humano. **Objetivos.** O presente estudo tem por objetivo analisar as principais características associadas à adaptação ao trato urinário de isolados de *E. coli*, *E. faecalis* e *S. saprophyticus* comparados à representantes do trato gastrointestinal por meio de estudo de genômica comparativa. **Métodos.** Sequenciamento de genoma completo foi realizado em três representantes dos gêneros em estudo (*E. coli* E2, *E. faecalis F165* e *S. saprophyticus* SS545). Além disso, genomas disponíveis em banco de dado do Genbank foram triados conforme sítio de isolamento e agrupados para análise comparativa. Foram realizadas análise de pan-genoma dos grupos, anotação funcional dos genes, predição de profagos e resistoma. **Resultados**. As espécies em estudo apresentaram um pan-genoma aberto, característico de micro-organismo adaptativos, com o repertório de genes para o grupo gastroinstestinal de *E. coli* superior a 90.000. Uma proporção significativa de genes associados à motilidade celular, ciclo celular e metabolismo secundários foi superior em uropatógenos de *E. coli*, enquanto para *E. faecalis* genes homólogos a transportadores transmembrana associados ao elemento traço molibdênio foram frequentemente encontrados. Em comparação com isolados do trato urinário das outras duas espécies, a espécie *S. saprophyticus* apresenta uma proporção superior do pan-genoma associado ao transporte e metabolismo de aminoácidos, coenzimas e íons inorgânicos. Profagos são importante fonte de variabilidade em *E. coli*, onde isolados urinários apresentaram uma densidade superior de genes de resistência e virulência sendo carreados por esses elementos genéticos. **Conclusão.** Uropatógenos apresentam características genômicas relevantes à adaptabilidade ao trato urinário, embora muitos isolados apresentem determinantes genéticos que permitem a transição entre nichos.

Palavras-chave: Uropatógenos. Infecções do Trato Urinário. Genômica Comparativa.

ABSTRACT

Introduction. The Urinary Tract Infections (UTIs) are the most common bacterial infection affecting humans, mainly women. *Escherichia coli*, *Enterococcus faecalis* and *Staphylococcus saprophyticus* are uropathogens often recovered from UTIs and several virulence factors are known for urinary tract adaptation, allowing adherence and invasion of the bladder mucosa, besides evasion of immune host defenses. The main reservoir of these strains is the gastrointestinal tract of the host. **Objectives**. The aim of the study was evaluate genomic features for niche adaptation of urinary and gastrointestinal strains of these species by comparative genomics. **Methods**. Draft genome sequences was obtained from three uropathogens (*E. coli* E2, *E. faecalis F165* e *S. saprophyticus* SS545). In addition, genomes from Genbank database was recovered by isolation source and pan-genome analysis, functional annotation, prophages and resistome prediction was performed for comparative analysis. **Results.** In *E. coli* strains, the repertoire of genes of gastrointestinal isolates was higher than urinary, with genes number superior of 90,000. The ratio of genes associated to cell cycle-division, cell motility and secondary metabolite metabolism was superior in urinary isolates of *E. coli*, and *E. faecalis* had homologous genes of molybdenum transport system often found in urinary isolates. *S. saprophyticus* showing significant ratio of gene repertoire associated to amino acid, coenzyme and inorganic ion transport / metabolism compared to others uropathogens. Urinary *E. coli* strains had superior density of virulence factors and resistance genes carried by prophages compared to gastrointestinal strains. **Conclusion.** The uropathogens showed important features for urinary tract adaptation, although the gene repertoire of some isolates allows the transition between niche.

Keywords: Uropathogens. Urinary Tract Infections. Comparative Genome.

INTRODUÇÃO

Infecções do Trato Urinário

Manifestações Clínicas

As manifestações clínicas das Infecções do Trato Urinário (ITU) podem ser classificadas conforme o comprometimento anatômico do sistema urinário ou defesas naturais do hospedeiro. Para aquelas ITUs ditas não-complicadas, os indivíduos acometidos se apresentam anteriormente de boa saúde e sem anormalidades anatômicas ou neurológicas que acarretem em comprometimento do sistema urinário. Alguns fatores de risco podem desencadear os quadros agudos de ITU baixa (cistite), a exemplo de doença prévia como o diabetes ou atividade sexual prévia. ITU ascendente, como a pielonefrite, pode ser conseqüência de ascensão do patógenos via ureteres, ou mesmo a instalação de infecção em nível de pelve renal mediante disseminação hematogênica (1).

Para quadros de cistite, os sintomas mais comumente associados são disúria (dor no ato miccional), dor suprapúbica, frequência e urgência urinária além de hematúria em alguns casos (1). Para diferentes faixas etárias, as manifestações para ITU não-complicadas podem diferir. Crianças podem não apresentar os sintomas clássicos, sendo mais comumente visto manisfestações comportamentais como irritabilidade e dificuldade de alimentação além de sistêmicos inespecíficos como vômitos ou náuseas (2). Aproximadamente 3% das crianças abaixo de 5 anos de idade, quando atendidas em unidades de emergência com doença febril, apresentam como diagnóstico de causa primária as ITU (3). Em idosos, os sintomas são semelhantes aos de jovens adultos, podendo apresentar manifestações clínicas adicionais como dor abdominal baixa, dor nas costas e até constipação (4).

Em estudo conduzido em mulheres adultas com ITU, os sintomas mais frequentemente relatados no período pré-menopausa foram aumento da frequência miccional, dor e sensação de queimação ao urinar, enquanto em mulheres pós-menopausa foram relatadas queixas mais inespecíficas, como dor abdominal e lombar, calafrios e náuseas. Os autores relatam a importância desses achados no grupo pós-menopausa, uma vez que acarreta em um diagnóstico e tratamento tardio das ITU (5).

Epidemiologia

As ITU são de elevada prevalência no mundo, sendo o acometimento infeccioso de etiologia bacteriana mais comum. Dentre o espectro de hospedeiros humanos de maior incidência para ITU, destaca-se principalmente homens em extremos de idade (crianças e idosos) e mulheres em todas as faixas etárias (1).

Segundo dados obtidos do DATASUS, em 2016 no Brasil houve a notificação de 17.234 internações por cistite (CID-10: capítulo XIV. Doenças do aparelho geniturinário – Morbidade CID-10: Cistite) em unidades hospitalares, com média de permanência da internação de 4,4 dias e 390 óbitos (Fonte: Ministério da Saúde - Sistema de Informações Hospitalares do SUS (SIH/SUS)). Comparada à série histórica, internações por ITU caracterizadas clinicamente por cistite teve um aumento dos anos de 2011 a 2015, bem como mesma tendência para o número de óbitos notificados. Constatação importante é verificada em relação à epidemiologia local de ITU não-complicadas no estado do Rio Grande do Sul, onde no mesmo intervalo a taxa de mortalidade associada à cistite está em ascensão e aproximadamente 100% acima das taxas nacionais para o ano de 2016 (Figura 1).

Vale ressaltar que os casos de internações hospitalares por ITU baixa, como cistite, é infrequente. Em estudo conduzido em instituição de saúde, Cervellin e colaboradores (2016) avaliaram a epidemiologia e desfechos de pacientes adultos admitidos na Emergência Hospitalar com queixa de dor abdominal aguda, no qual a prevalência varia entre 7 e 10%. Desses pacientes, uma pequena parcela apresentou diagnóstico de ITU (homens: 0,76% versus mulheres: 2,66%) (6). Os dados disponíveis em banco de dados DATASUS evidenciam somente um pequeno grupo de paciente com a morbidade, pois a incidência mundial da doença é estimada em 150 milhões casos ao ano, principalmente determinada em consultas ambulatoriais (1).

As ITUs se apresentam como importante causa de morbidade entre crianças até 2 anos de idade. Além de alta prevalência, as complicações associadas podem acarretar desde bacteremia a disfunções renais e choque séptico, geralmente acompanhadas por quadro febril (7). A incidência de ITU é mais alta no primeiro ano de vida independente do gênero, diminuindo no sexo masculino com o decorrer da infância (8).

Evidência de distribuição de ITUs dependentes de variáveis como faixa etária e sexo é um importante parâmetro epidemiológico para a morbidade. A prevalência das ITUs em relação ao sexo do paciente apresenta-se com comportamento linear negativo, ou seja, em indivíduos do sexo masculino a prevalência de ITUs não-complicadas tende a diminuir com a progressão da idade, o mesmo não ocorrendo para o sexo feminino, no qual aproximadamente 90% das infecções ocorrem em indivíduos acima de 2 anos de idade (diferentemente do sexo masculino, onde apenas 10% são acometidos nessa faixa etária) (9).

Em adultos, a incidência estimada de ITU em mulheres é na ordem de quatro vezes a incidência em homens de mesma faixa etária, sendo que em pacientes do sexo feminino é estimado que pelo menos 50% tenham um episódio de ITU até a terceira década de vida (10).

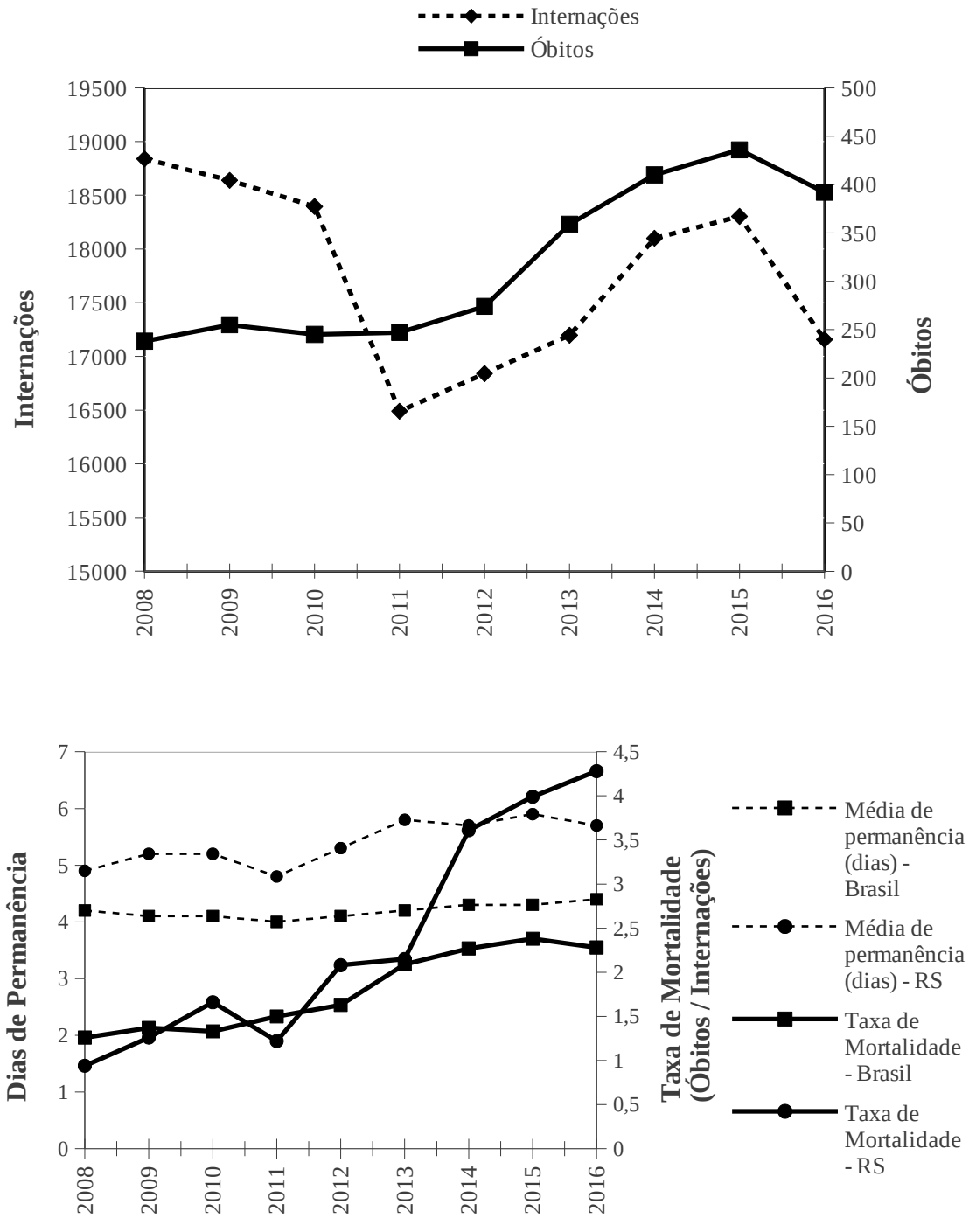


Figura 1: Notificações de infecções do trato urinário (cistite) no Brasil, período 2008 – 2016. Fonte: DATASUS.

Uropatógenos de Importância Médica

Escherichia coli

Escherichia coli é uma bactéria gram-negativa pertencente à família *Enterobacteriaceae*. Apresenta-se como uma espécie com habilidades tanto para colonizar em mutualismo com seu hospedeiro quanto para persistir em diferentes ambientes abióticos. Isolados bacterianos de *E. coli* são comensais do trato gastrointestinal de mamíferos desempenhando papel importante na regulação da microbiota humana por inibir a colonização por outras bactérias patogênicas (11).

Características moleculares de alguns isolados permitem que a espécie apresente potencial patogênico ao hospedeiro, com isolados associados à doença diarreicogênica ou de infecções extraintestinais. O espectro de doenças se restringe principalmente aos quadros de gastroenterites, onde isolados bacterianos são classificados segundo patotipos, como *E. coli* enteropatogênicas (EPEC), enterohemorrágicas (EHEC), enterotoxigênicas (ETEC), enteroagregativas (EAEC), enteroinvasiva (EIEC) e difusamente aderente (DAEC) (12), no qual esses isolados raramente causam doença extraintestinal. Contrapondo a cepas diarreicogênicas, isolados extraintestinais (ExPEC) colonizam o trato gastrointestinal de humanos como membro normal da microbiota. Entretanto, existe um espectro de determinantes genéticos que permite a disseminação e colonização de diferentes nichos do hospedeiro, como acesso ao sistema circulatório e ao trato urinário (11).

Isolados uropatogênicos (UPEC – *uropathogenic E. coli*) são comumente associados à doença em humanos, sendo responsável pela maioria dos casos de infecções do trato urinário adquiridos na comunidade, acarretando em elevados custos para os sistemas de saúde (11).

O processo infeccioso associado aos isolados uropatogênicos de *E. coli* é intermediado pela expressão de fatores de virulência como Pili tipo P, fímbrias tipo I, sideróforos e toxinas. Em etapas, as células bacterianas ganham acesso ao trato urinário, permitindo o processo de invasão da mucosa de transição do epitélio da bexiga. Esse fenômeno está correlacionado com características epidemiológicas das ITUs recorrentes no hospedeiro (13).

Após estabelecimento na mucosa do trato urinário, a resposta inflamatória do hospedeiro está associada à secreção de citocinas pró-inflamatórias e quimiocinas, permitindo

o recrutamento de células de defesa – principalmente neutrófilos – e indução de processo esfoliativo das células epiteliais da bexiga. Esses mecanismos fazem parte da resposta imunológica do hospedeiro para combater o processo infeccioso induzido por *E. coli* uropatogênica. Entretanto, características moleculares importantes da espécie no que tange o estabelecimento de doença são a presença de fatores de virulência que permitem a persistência das células bacterianas na mucosa, manutenção do metabolismo bacteriano em ambiente do trato urinário e secreção de fatores locais ao sítio infeccioso (14).

Fatores de Virulência associados a Aderência ao Uro-epitélio

Para a espécie *Escherichia coli*, a aderência das células bacterianas está associada principalmente pela presença de pili. O pili bacteriano é caracterizado como estruturas apendiculares de superfície, de natureza proteica, não pertencente à estrutura flagelar. Essas estruturas desempenham papel importante nos processos de adesão às células do hospedeiro, superfícies ambientais e até adesão entre células bacterianas, conferindo aos micro-organismos capacidade de colonização por tropismo a determinados nichos, invasão tecidual e formação de biofilme (15).

A fim de permanecer como colonizante no trato urinário, isolados uropatogênicos devem ser capazes de aderir ao urotélio e tecidos associados, uma vez que este nicho apresenta-se como um micro-ambiente hidrodinamicamente ativo (16) que previne a estase da urina, no qual o fluxo diminuído potencialmente beneficia a instalação de ITUs (17).

Historicamente, a expressão de estruturas tipo fímbrias em isolados de *Escherichia coli* foram reportadas como importantes fatores de virulência em isolados uropatogênicos (18), sendo caracterizados quanto a sua capacidade de hemaglutinação e estrutura, dividindo-os principalmente em fímbrias tipo 1 e P (pili P) (19). Em isolados UPEC, esses dois sistemas de síntese de pili é realizado via chaperona-proteína transportadora (*usher*) (15).

Fímbria tipo I é codificada pelo operon *fim*, com especificidade para receptores ricos em manose. O urotélio é constituído – além de estrutura celular característica do epitélio de transição – de estruturas proteicas especializadas na membrana celular que desempenham funções de impermeabilidade e flexibilidade ao epitélio, chamadas placas uroteliais. No que tange a característica molecular dessas estruturas, são ricas em estruturas proteicas uroplaquina (20). No epitélio renal, a presença de receptores de manose associados à

uroplaquina é diminuída, diferentemente do epitélio da bexiga. Essa característica desempenha papel importante na participação da fímbria tipo I em infecções do trato urinário baixo como cistite, especificamente pela adesina FimH (14), embora esteja associada também na facilitação da aderência entre células bacterianas e a formação de biofilme em nível de túbulo renal (16).

A adesina FimH liga-se aos receptores ricos em manose nas células epiteliais da bexiga, sendo importante também no estabelecimento de comunidades bacterianas intracelulares após processo invasivo do epitélio urinário. Entretanto, mutações no domínio proteico de ligação resultam na atenuação da virulência de isolados uropatogênicos em modelo animal (15). Apesar de ser reconhecida como um fator de virulência em UPEC, a presença de fímbrias tipo I é amplamente distribuída também em isolados comensais (14), embora o seu papel em isolados enteropatogênicos (EPEC) não seja claro (12).

O operon que codifica para fímbria tipo I está localizado no cromossomo bacteriano e pode estar sujeito a variação de fase devido a presença de segmentos de DNA invertidos entre o promotor. Sobre ação de recombinases, o operon pode ser transcrito ou não dependendo da orientação da sequência promotora (21). O mesmo processo já foi descrito para fímbrias tipo P em cepas uropatogênicas (22).

Fímbrias tipo P são codificadas pelo operon *pap* (*pyelonephritis-associated pilus*), organizado em cinco genes estruturais, uma proteína periplasmática da classe das chaperonas, uma proteína transportadora e dois genes regulatórios (22). Esse operon geralmente se encontra integrado em ilhas de patogenicidade em isolados UPEC e a subunidade PapG desempenha função de adesina por reconhecer glicoesfingolipídeos do epitélio renal (15).

O processo de variação de fase na expressão de fímbrias tipo I e pili P confere vantagem evolutiva para isolados UPEC, uma vez que as adesinas são antígenos de superfície e a modulação da apresentação dessas proteínas limita a exposição ao sistema imune. Essa estratégia acaba por evadir a resposta imune celular e humoral do hospedeiro, além de permitir que sub-populações possam colonizar diferentes porções do trato urinário (22).

Existe uma diversidade de fímbrias em isolados de *Escherichia coli* uropatogênicas (16 sistemas baseados em chaperona-proteína transportadora), possibilitando o ancoramento das células bacterianas a diversos tipos teciduais. Como o trato gastrointestinal é o reservatório de isolados UPEC, determinadas fímbrias participam também na colonização deste epitélio. Em estudo no qual foi avaliado o papel de nove operons associados à expressão

de fímbrias em isolado de *E. coli* recuperado de cistite humano (cepa UTI89), a deleção de sete operons não influenciou nas propriedades adesivas da UPEC sobre o epitélio intestinal no modelo animal usado no estudo. Entretanto, isolado mutante para o operon *fim* (codificador da fímbria tipo I) apresentou significativa diminuição da colonização, o mesmo ocorrendo para mutantes para o operon *ucl* (codificador para fímbrias F17). Além disso, a administração de um inibidor de manosídeo reduziu a colonização do trato gastrointestinal por isolados UPEC principalmente por saturar a ligação da subunidade FimH, sem que desencadeasse modificações na microbiota intestinal do modelo (23).

Íons inorgânicos como o Ferro (Fe) são essenciais para os micro-organismos, uma vez que esse átomo desempenha importante papel como cofator de diversas enzimas do metabolismo primário e secundário. A atuação do íon Fe está na participação de reações redox e na cadeia de transferência de elétrons (24). Isolados patogênicos extraintestinais de *E. coli* (ExPEC) apresentam inúmeros sistemas para a aquisição de ferro a partir do hospedeiro. Em sítios extraintestinais, a concentração do íon ferro é limitada e sistemas acoplados sideróforo/receptor de sideróforo tornam-se importantes fatores de virulência nesses isolados. Foi demonstrado que receptor de sideróforo *iroN* em *E. coli* apresenta sua expressão aumentada em urina humana e sangue, com elevada prevalência em amostras isoladas de ITUs comparado a amostras fecais. Além disso, *iroN* mostrou-se como um fator de uro-virulência em modelo animal de ITU, embora diferenças no crescimento celular não tenham sido evidenciadas quando em condições de cultura em urina humana comparado a cepa sem o determinante genético. Como não desempenha função como adesina, a aquisição de ferro provavelmente é realizada em associação com a colonização e invasão tecidual (diretamente das células do hospedeiro) (25).

Para proteínas receptoras de sideróforos, deve haver moléculas que desempenham a função de sequestro do íon a partir do micro-ambiente ou das células do hospedeiro. Essas moléculas são denominadas sideróforos, componentes quelantes de ferro inorgânico de baixo peso molecular, sendo a principal denominada enterobactina. Entretanto, outros três sistemas de sideróforos produzidos por *E. coli* foram descritos, como salmoquelina, yersiniabactina e aerobactina. Esses sideróforos são negativamente regulados pelo ferro, tendo a sua produção aumentada em condições de baixa disponibilidade do íon a exemplo do que ocorre no trato urinário. A vantagem para micro-organismos uropatogênicos de possuir múltiplos sistemas de aquisição de ferro é a habilidade de competir de maneira eficaz contra o hospedeiro e outras

bactérias para manter um metabolismo celular competente em um micro-ambiente hostil como o trato urinário (26).

Enterococcus faecalis

As espécies representantes do gênero *Enterococcus* são cocos gram-positivos arranjados aos pares ou cadeias curtas, catalase-negativos, podendo crescer em uma ampla faixa de temperatura (10°C a 45°C) e pH (4,6 a 9,9), tolerar sais de bile e concentração elevada de cloreto de sódio (27). Sorologicamente fazem parte do grupo D de Lancefield e metabolicamente são classificados como quimioorganotróficos e homofermentativos, produzindo ácido lático. Em relação à ecologia do gênero, há uma diversidade de habitats onde representantes podem ser encontrados no ambiente aquático, sedimentos e até hospedeiros invertebrados. Contudo, o reservatório das principais espécies é no trato gastrointestinal de animais homeotérmicos (28).

Os principais micro-organismos do gênero associados ao hospedeiro humano são *Enterococcus faecalis* e *Enterococcus faecium*. Essas duas espécies são muito comuns no trato gastrointestinal, fazendo parte da microbiota normal (29). Os isolados de *E. faecalis*, apesar de serem comensais, apresentam um repertório vasto de fatores de virulência, propiciando a adaptação a diferentes nichos. Quando existe a translocação desses isolados para sítios geralmente estéreis do corpo humano, ele são denominados de patógenos oportunistas (30). Determinantes genéticos para colonização e adesão a mucosas e superfícies abióticas (além de invasão tecidual), formação de biofilme bacteriano, proteases e bactericinas são exemplos de mecanismos utilizados pela espécie a fim de garantir aporte energético, manutenção e proteção das células bacterianas. Todos esses fatores, além de contribuir para a patogênese, são passíveis de transferência horizontal, somado a aquisição de novos determinantes. Isso permite que esta espécie seja um dos principais agentes infecciosos contra a saúde humana (31), podendo causar ITUs, infecções de feridas, endocardites e até sepse (30).

Fatores de Virulência Associados às ITUs em *E. faecalis*

Muitos dos fatores de virulência associados às infecções do trato urinário em *Enterococcus faecalis* são também para a formação de biofilme bacteriano, principalmente sendo induzido em superfícies inertes como cateteres urinários. O real papel desses determinantes genéticos de virulência não é claro em relação à especificidade para ITUs ou como consequência da formação de biofilme (27), mesmo que isolados uropatogênicos tenham tropismo ao tecido renal em modelos animais (32).

Os principais determinantes de adesão e colonização de isolados de *E. faecalis* são proteínas de superfície, como Esp (*Enterococcus surface protein*), Ace (*adhesion of collagen of E . faecalis*), EfbA (*enterococcal fibronectin binding protein*), Ebp (*endocarditis and biofilm-associated pilus*), Bop (*biofilm on plastic operon*) e Agg (*aggregation substance*) (27, 33).

Esp é uma proteína de superfície celular no qual apresenta múltiplas regiões repetitivas (motivos proteicos) frequentemente encontrados em isolados virulentos, como àqueles associados à bacteremia e ITUs em comparação aos isolados comensais (trato gastrointestinal), sendo uma característica importante em isolados clínicos de *E. faecalis* (34). A estrutura é semelhante à adesinas com potencial para ancoramento em diferentes ligantes no hospedeiro (27). Especificamente ao trato urinário, estudos em modelo animal reforçam o papel desse fator de virulência na aderência e colonização do epitélio da bexiga (35).

O reconhecimento de proteínas extracelulares de hospedeiros vertebrados, como fibrinogênio, fibronectina e colágeno, são os principais alvos para adesão mediada por uma subfamília de adesinas em bacterias gram-positivas, as MSCRAMM (do inglês *microbial surface components recognizing adhesive matrix molecules*). Essas adesinas estão ancoradas ao peptidioglicano por um motivo proteico de sequência LPXTG (36). Em *Enterococcus faecalis*, Ace é o representante das MSCRAMM no qual media o ancoramento da célula bacteriana no tecido do hospedeiro via colágeno tipo I e IV, laminina e dentina (37). Isolados mutantes para *ace* mostraram-se atenuados quanto a sua patogenicidade, demonstrando o papel dessa adesina como importante fator de virulência na espécie (38), exemplificado pela perca da capacidade adesiva de isolados mutantes comparado com cepa isogênica selvagem em modelo animal para endocardite (39).

EfbA desempenha função como adesina em isolados de *Enterococcus faecalis*, principalmente por adesão à fibronectina do hospedeiro, onde mutantes para a região codificadora apresentaram diminuição do potencial de adesão à fibronectina humana

imobilizada e apresentaram virulência atenuada em modelo animal de ITU ascendente. Além disso, EfbA apresentou tropismo acentuado para o epitélio renal em detrimento do epitélio da bexiga (40). Esse fator de virulência apresenta aderência a outras proteínas da matriz extracelular como colágeno tipo I e V e, em menor nível, à laminina, ácido hialurônico, fibrinogênio entre outros. Sua contribuição para a patogenicidade de isolados associados a endocardites foi corroborada em modelo animal, no qual foi verificado aumento da colonização das válvulas cardíacas, indicando um importante papel nas endocardites infecciosas por *E. faecalis* (41).

Ebp (*endocarditis and biofilm-associated pilus*) está localizado no cromossomo e faz parte de um operon composto por 3 genes estruturais para pilina (*ebpA*, *ebpB* e *ebpC*) e altamente conservado na espécie *E. faecalis*. A designação deste fator de virulência foi dada após a evidência de que isolados mutantes para o *locus* apresentaram patogênese atenuada para endocardite em modelo animal e diminuição da formação de biofilme bacteriano. Além disso, em modelos animais para ITUs mesma evidência foi encontrada. A arquitetura e funcionalidade do pili associado ao operon *ebpABC* é dada pela presença de um arcabouço de pilina organizada em fibras (EbpA, EbpC – extensão da fibra) ancoradas por uma pilina na parede celular bacteriana (EbpB), com participação importante de EbpA+C na adesão ao epitélio renal (42). Além disso, a presença do pili sintetizado pelo operon *ebpABC* contribui para a transferência de material genético entre as células bacterianas, principalmente por facilitar a aproximação, ancoramento e estabilização do par conjugante em *E. faecalis* (43).

Agg contribui para a virulência de *E. faecalis* por aumentar a hidrofobicidade da superfície celular, permitindo a agregação entre os isolados bacterianos. É uma glicoproteína de superfície induzível por ferormônio, permitindo a agregação das células bacterianas no processo de conjugação e aderência nas células do hospedeiro (29).

Diversos outros fatores de virulência foram descritos para isolados de *Enterococcus sp*. participando no processo de adesão e colonização do hospedeiro, como ElrA (*enterococcal leucine-rich-repeat-containing protein*) e Bee (*biofilm enhancer in Enterococcus*), ou na evasão do sistema imune e estimulação de citocinas pró-inflamatórias – locus *cps* para produção de cápsula polissacarídea em *Enterococcus sp*.; Epa (e*nterococcal polysaccharide antigen*) – antígeno de parede celular (44). Fatores secretados no meio extracelular por isolados de *Enterococcus sp*. aumentam o potencial patogênico das espécies, como, por exemplo toxina Cyl (*Haemolysin–cytolysin*) no qual acarreta na lise celular de

eritrócitos humanos e algumas linhagens celulares de leucócitos; e proteases como GelE (*gelatinase*) com ação sobre o sistema imune, degradação de tecidos do hospedeiro e participação na formação de biofilme (44).

Staphylococcus saprophyticus

Dentre os cocos Gram-positivos, as infecções por *S. saprophyticus* é a segunda causa de infecções de trato urinário adquiridas na comunidade, afetando principalmente um grupo restrito de pacientes: mulheres jovens e sexualmente ativas (45). Esse micro-organismo apresenta características singulares que permitem que seja descrito como um dos únicos representantes do gênero *Staphylococcus* capaz de levar a quadros de cistite por seus mecanismos de virulência. Outros isolados bacterianos, principalmente do grupo dos Staphylococcus Coagulase-negativos (ScoN) podem causar ITU, entretanto, o processo infeccioso é freqüentemente correlacionado com pacientes hospitalizados que estão em uso de dispositivos médicos, como o uso de cateteres urinários (46).

A espécie em questão é frequentemente isolada da região perianal, o que faz com estes micro-organismos estejam mais envolvidos em infecções do trato urinário, principalmente em mulheres jovens e sexualmente ativas (47). Além disso, é o segundo mais importante, seguido somente por *Escherichia coli*, causador de ITUs não complicadas em mulheres. Algumas complicações mais severas podem ocorrer como pielonefrite aguda, septicemia, nefrolitíase e endocardite. No sexo masculino, ITUs por *S. saprophyticus* são menos frequentes, entretanto, podem acometer homens de todas as idades. Nestes pacientes, podem ocorrer uretrite, epididimite e prostatite (48).

Historicamente, os SCoN eram considerados contaminantes quando isolados a partir de material clínico, analogamente aos isolados de *S. saprophyticus*. Pouco se sabia dos mecanismos patogênicos desse micro-organismo, sabendo-se apenas que era recuperado de ITUs não complicadas como patógenos primários e que atingiam um grupo restrito de pacientes, a saber, mulheres jovens (49). Com a evolução dos estudos científicos desse uropatógeno, diversos fatores foram correlacionados para colonização por esse micro-organismo, como intercurso sexual recente, variação sazonal e colonização do trato gastro-intestinal (50). Essas infecções também se assemelhavam às infecções por bactérias Gram-

negativas, na qual os sujeitos de pesquisa experimentavam colonização recorrente por diferentes cepas de *S. saprophyticus* (51). Esses estudos, entretanto, pouco elucidaram os mecanismos e determinantes de virulência envolvidos no processo infeccioso por esse patógeno.

O potencial patogênico de isolados de *S. saprophyticus* é decorrente da presença de diversas proteínas de superfície que medeiam a aderência da célula bacteriana à célula hospedeira. Essa interação permite que o micro-organismo se ancore principalmente nas células uro-epiteliais, induzindo um processo inflamatório no tecido, caracterizando o quadro de cistite.

Diversos determinantes de virulência foram descritos para a espécie. Na análise comparativa do genoma de *S. saprophyticus* com outros dois principais representantes do gênero *Staphylococcus* (*S. aureus* e *S. epidermidis*), foi verificado que este uropatógeno possui em seu cromossomo informações que revelam a presença de proteínas ancoradas à parede celular e um sistema de transporte de íons que facilita a adaptação ao ambiente urinário. Além disso, a atividade da enzima urease de isolados de *S. saprophyticus* é significativamente maior comparada às outras espécies de *Staphylococcus*, sendo considerada um importante fator de virulência para a persistência de infecções no trato urinário. Portanto, o *S. saprophyticus* se adaptou especificamente para o ambiente urinário, expressando fatores de aderência e um notável crescimento nesse meio (46).

Fatores de Virulência Associados à Aderência ao Uro-epitélio e às Proteínas da Matriz Extracelular do Hospedeiro

Sendo de fundamental importância ao processo infeccioso à aderência dos isolados bacterianos aos tecidos do hospedeiro, especificamente ao uro-epitélio, diversas proteínas bacterianas de superfície foram descritas como determinantes de aderência e virulência de *S. saprophyticus*, como UafA, Aas, Ssp e SdrI (52).

O processo de aderência da célula bacteriana ao tecido do hospedeiro é mediado por proteínas de superfície celular (MSCRAMMs). Dessas moléculas adesivas, são descritas aquelas que estão covalentemente ancoradas ao peptidioglicano da parede celular de

patógenos Gram-positivos por um motivo de aminoácidos LPXTG. No mesmo estudo que realizou a análise genômica comparativa do *S. saprophyticus* com *S. aureus* e *S. epidermidis*, somente uma proteína foi encontrada tendo essas características, a *uro-adherence factor* A (UafA), sugerindo que esse isolado apresenta características distintas de colonização ao tecido do hospedeiro (46, 53).

A proteína UafA é uma hemaglutinina que contribui significativamente à aderência de isolados de *S. saprophyticus* às células uro-epiteliais do trato urinário, uma vez que nesse micro-ambiente as células bacterianas estão submetidas a intenso estresse devido ao fluxo de urina. Entretanto, os mecanismos moleculares de interação entre receptor e ligante do UafA não foram bem elucidados, acreditando-se que fosse uma molécula que agia como receptor de fibronectina no hospedeiro humano (54). Posteriormente, por meio de ensaio de aderência e análise cristalográfica, a UafA foi descrita como uma molécula consistindo de 3 domínios, na qual não medeia a aderência por moléculas de colágeno, fibronectina ou de proteínas, estando sua molécula ligante provavelmente composta de baixo peso molecular como sacarídeos e lipídios (52, 53, 55).

A elucidação do papel de outras proteínas de superfície na patogênese dos isolados de *S. saprophyticus* tornou-se importante após a evidência de que isolados clínicos que não expressam hemaglutinação ainda mantém seu potencial virulento de causar infecções e ligar-se a moléculas de colágeno. Outra proteína MSCRAMM com motivo LPXTG foi descrita para *S. saprophyticus*, com repetições de serina/aspartato (*serine-aspartate repeat*-SdrI), na qual desempenha essas características citadas. Essa molécula de superfície apresenta estrutura similar a outras proteínas da família Sdr, com características típicas de proteínas de superfície encontradas em micro-organismos Gram-positivos. Em mutantes *knockout* para SdrI, foi evidenciado decréscimo significativo na ligação a moléculas de colágeno, comparado com a cepa selvagem. Esse resultado indica que o SdrI seria o responsável pela ligação a moléculas de colágeno em *S. saprophyticus*, uma proteína da matriz extracelular. Contudo, devido a baixa prevalência do gene *sdr*I em isolados clínicos, não ficou evidente se verdadeiramente a presença dessa proteína de superfície aumentaria a virulência ou apenas seria mais um caso de uma adesina redundante presente em isolados de *Staphylococcus* spp. (56).

Isolados SdrI positivos apresentam capacidade de ligação a moléculas de fibronectina, mesmo que a estrutura dessa proteína de superfície não contenha uma região típica das

proteínas ligantes dessa proteína da matriz extracelular (57). Em estudo de modelo animal, foi verificado que, embora a presença de SdrI não seja necessária para a colonização inicial no trato urinário, essa proteína é requerida para a persistência dos isolados de *S. saprophyticus* em órgãos como a bexiga e os rins (58).

Em bactérias de importância clínica como as Gram-negativas, *Streptococcus* e *S. aureus*, é reconhecido o papel de proteínas de superfície na aderência em células eucarióticas. Já em micro-organismos pertencentes ao grupo do SCoN, estudos tem sido conduzidos principalmente a fim de verificar a contribuição da formação de biofilme e produção de cápsula na virulência dessas cepas, fatores que estruturalmente são constituídos de polímeros de carboidratos. Contudo, em isolados de *S. saprophyticus*, foi descrito estruturas filamentares protéicas associadas à superfície celular em grande quantidade, de 95 kDa denominada de proteína associada a superfície de *S. saprophyticus* (do inglês *S. saprophyticus surface-associated protein* - Ssp) que poderia ser um candidato adicional a fator de aderência em células eucarióticas de hospedeiros humanos, mesmo não apresentando atividade de hemaglutinação (59).

Embora tenha sido sugerido que Ssp apresentasse função de aderência, estudos posteriores demonstraram que esse polipeptídeo não apresenta propriedades adesivas às proteínas da matriz extracelular como fibronectina, fibrinogênio, colágeno e a laminina. Em estudo que realizou clonagem, sequenciamento e ensaio de atividade de Ssp, Sakinc e colaboradores (2005) concluíram que a maior proteína de superfície de *S. saprophyticus* é uma enzima da família das lipases, uma vez que a sua seqüência gênica apresentou homologia com lipases estafilocócicas. Além disso, é produzida em grandes quantidades por essa espécie de *Staphylococcus* e não apresenta uma função específica de adesina. Segundo os autores, pode contribuir para a persistência do micro-organismo por promover uma fonte de energia ou facilitar a aderência, já que as evidências mostram que é expressa *in vivo* e pode apresentar função promovendo o crescimento bacteriano ou como um fator de patogênese em isolados de *S. saprophyticus* (60).

Semelhantemente ao Ssp, foi descrito em isolados de *S. saprophyticus* outra proteína expressa em grande quantidade na superfície das células bacterianas, a Aas (*adhesin-autolysin of S. saprophyticus*). Em estudo conduzido por Gatermann, Meyer e Wanner (1992), foi estudada uma cepa bacteriana de *S. saprophyticus* Ssp-negativa, mas com atividade de

hemaglutinação em eritrócitos de carneiro. Uma proteína de 160 kDa foi identificada e caracterizada funcionalmente pelos pesquisadores, apresentando atividade semelhante às hemaglutininas, contudo, não sendo inibida por anticorpos dirigidos contra a proteína Ssp (54). Caracterização posterior dos tipos moleculares em que essa hemaglutinina poderia utilizar como receptor foram estudados em eritrócitos de carneiro.Ao contrário de muitas outras hemaglutininas, o receptor é do tipo protéico, uma vez que hemaglutinação não foi evidenciada em membranas de eritrócitos tratadas com proteinase (61).

O conhecimento atual sobre a hemaglutinina de *S. saprophyticus* revela uma molécula multifuncional. Além de mediar a ligação a fibronectina, essa proteína apresenta forte homologia com autolisinas presentes em outras espécies bacterianas importantes do gênero, como *S. aureus* (*atl*) e *S. epidermidis* (*atlE*). Contudo, a caracterização dessa proteína de superfície mostrou diferenças importantes se comparadas com outras presentes em micro-organismos Gram-positivos. Essa molécula apresenta duas atividades distintas, de dispersão durante a divisão celular e de aderência a moléculas presentes em organismos hospedeiros. A combinação dessas atividades (de autolisina e de aderência) poderia contribuir para a virulência dos micro-organismos por liberar componentes da parede celular imunologicamente ativos e permitir a colonização e o desenvolvimento do processo infeccioso a partir da adesão a células uro-epiteliais do hospedeiro. Portanto, a proteína Aas – adesina/autolisina – de *S. saprophyticus* é um potencial fator de virulência dessa espécie, representando uma nova classe de adesinas estafilocócicas (62).

Genômica Comparativa de Uropatógenos de Importância Médica

Com o advento das novas tecnologias de sequenciamento, o número de genomas bacterianos sequenciados aumentou de maneira exponencial. Estudos comparativos entre micro-organismos patogênicos e comensais, evolução, e interação entre patógeno e hospedeiro são os principais eixos norteadores que alimentam as descobertas científicas associadas à biologia dos procariotos (63).

Em 2015, os filos *Proteobacteria* e *Firmicutes* totalizavam aproximadamente 25.000 genomas depositados no GenBank. A análise de um número muito grande de genomas mostra que espécies bacterianas como *Escherichia coli* apresentam uma ampla diversidade de

famílias gênicas, o que caracteriza a plasticidade genômica para grupos específicos de procariotos (64).

Em estudo conduzido com 312 isolados clínicos de *E. coli* extraintestinal (ExPEC) recuperados de sangue e urina, Salipante e colaboradores (2015) verificaram que não existe uma estruturação bem delimitada entre populações com capacidade invasiva sobre um dos sítios anatômicos, além de os isolados se apresentarem genomicamente heterogêneos (65). Além disso, em comparação com o pan-genoma, o pan-metabolismo de *E. coli* mostrou-se menos diversificado, compartilhando processos metabólicos independente do sítio de isolamento (isolados comensais e patogênicos recuperados de sítios extraintestinais e intestinais). Os autores comentam que os fenótipos de patogenicidade aparentemente não acarretam em grandes mudanças nas reações metabólicas compartilhadas entre as cepas (66).

Em estudo de genômica comparativa por metodologia de hibridização por micro-arranjos de DNA, isolados de *E. coli* recuperados de ITUs como urosepse, pielonefrite e cistite, bem como de bacteriúria assintomática (BA), apresentaram poucas diferenças associadas ao quadro infeccioso. Especificamente, duas ilhas de patogenicidade (IP) importantes em isolado uropatogênico *E. coli* CFT073 foram significativamente associados a quadros de pielonefrite e urosepse, comparado aos isolados de BA e cistite. Fímbria tipo P foi comumente carreada dentro dessas IP. As características de uro-virulência fazem parte, provavelmente, de um espectro de genes que conferem vantagens adaptativas no trato urinário (67).

Semelhantemente em *Enterococcus faecalis*, estudo de hibridização em isolados comensais recuperados do trato gastrointestinal de recém-nascidos mostrou que a divergência dos genomas estava associada principalmente a elementos genéticos móveis (EGM) – como profagos – e proteínas hipotéticas. Além disso, variação genética entre os isolados do estudo foi comparável ao obtido de outras fontes como amostras ambientais, clínicas, alimentos e animais. Muitos dos micro-organismos apresentaram fatores de virulência bem descritos para a espécie, como *ace*, *agg*, *esp* e citolisina *cylL*, evidenciados tanto na análise de hibridização quanto por PCR. Os dados sugerem que a o potencial de virulência dos isolados é um evento complexo, não de característica singular de determinados genes (68). O papel dos EGM na evolução de isolados de *E. faecalis* foi descrito em estudo posterior, uma vez que esses elementos contribuem principalmente com a diversidade de genes e o aumento do tamanho médio do genoma da espécie (69).

O pangenoma de 168 isolados clínicos de *E. faecalis* foi determinado em estudo prévio. O genoma conservado entre os isolados foi aproximadamente 1.700 genes e genoma acessório de 6.200 genes. Os genes acessórios mais comumente recuperados foram proteínas hipotéticas, elementos de inserção ou transposons e genes associados a fagos ou plasmídeos, corroborando o importante papel de elementos genéticos móveis em *E. faecalis* (70).

Estudo conduzido por Kuroda e colaboradores (2005) determinou a sequência genômica do isolado *Staphylococcus saprophyticus ATCC 15305*, marco no estudo dessa espécie bacteriana. Esse uropatógeno apresenta um genoma de tamanho médio de 2,5 Mb e dois plasmídio. Além disso, comparativamente com genomas de *S. aureus* e *S. epidermidis*, há genes presentes somente na espécie *S. saprophyticus*, principalmente associados a transcrição, produção e conversão de energia, e transporte e metabolismo de carboidratos e aminoácidos. Sistemas moleculares associados a transporte e regulação são numerosos, não compartilhando genes de virulência comumente encontrados nas espécies patogênicas de *Staphylococcus*. A característica de expansão de genes parálogos para o transporte de íons confere uma vantagem seletiva no micro-ambiente urinário, onde o pH, osmolaridade, concentração de uréia, entre outros são abundantes e instáveis. A osmotolerância é uma capacidade única de *S. saprophyticus*, permitindo que as células bacterianas mantenham sua integridade e metabolismo no trato urinário (46).

Visto que o volume de dados gerados e depositados em banco de dados de genomas está em ascensão, a análise estruturada em genômica comparativa aprimora o conhecimento sobre a biologia de micro-organismos patogênicos. Em uropatógenos, existe uma lacuna sobre as características moleculares que permitam que um isolado normalmente representante da microbiota gastrointestinal desencadeie um processo infeccioso no trato urinário. Mesmo que determinados fatores de virulência apresentem frequência maior sobre micro-organismos recuperado de ITUs, esses não permitem uma estruturação populacional desses isolados.

REFERÊNCIAS

1. Flores-Mireles AL, Walker JN, Caparon M, Hultgren SJ. Urinary tract infections: epidemiology, mechanisms of infection and treatment options. Nat Rev Microbiol. 2015 May;13(5):269-84.

2. Becknell B, Schober M, Korbel L, Spencer JD. The Diagnosis, Evaluation and Treatment of Acute and Recurrent Pediatric Urinary Tract Infections. Expert Rev Anti Infect Ther. 2015 Jan;13(1):81-90.

3. Craig JC, Williams GJ, Jones M, Codarini M, Macaskill P, Hayen A, et al. The accuracy of clinical symptoms and signs for the diagnosis of serious bacterial infection in young febrile children: prospective cohort study of 15 781 febrile illnesses BMJ. 2010 Apr 20;340:c1594.

4. Rowe TA, Juthani-Mehta M. Diagnosis and Management of Urinary Tract Infection in Older Adults. Infect Dis Clin North Am. 2014 Mar;28(1):75-89.

5. Arinzon Z, Shabat S, Peisakh A, Berner Y. Clinical presentation of urinary tract infection (UTI) differs with aging in women. Arch Gerontol Geriatr. 2012 Jul-Aug;55(1):145-7.

6. Cervellin G, Mora R, Ticinesi A, Meschi T, Comelli I, Catena F, et al. Epidemiology and outcomes of acute abdominal pain in a large urban Emergency Department: retrospective analysis of 5,340 cases. Ann Transl Med. 2016 Oct;4(19):362.

7. Doern CD, Richardson SE. Diagnosis of Urinary Tract Infections in Children. Kraft CS, ed. J Clin Microbiol. 2016;54(9):2233-2242.

8. Zorc JJ, Kiddoo DA, Shaw KN. Diagnosis and Management of Pediatric Urinary Tract Infections. Clin Microbiol Rev. 2005 Apr;18(2):417-22.

9. Hanna-Wakim RH, Ghanem ST, El Helou MW, Khafaja SA, Shaker RA, Hassan SA, et al. Epidemiology and characteristics of urinary tract infections in children and adolescents. Front Cell Infect Microbiol. 2015 May 26;5:45.

10. Geerlings SE. Clinical Presentations and Epidemiology of Urinary Tract Infections. Microbiol Spectr. 2016 Oct;4(5).

11. Wiles TJ, Kulesus RR, Mulvey MA. Origins and Virulence Mechanisms of Uropathogenic Escherichia coli. Exp Mol Pathol. 2008 Aug;85(1):11-9.

12. Torres AG, Zhou X, Kaper JB. Adherence of Diarrheagenic Escherichia coli Strains to Epithelial Cells. Infect Immun. 2005;73(1):18-29.

13. Andersen TE, Khandige S, Madelung M, Brewer J, Kolmos HJ, Møller-Jensen J. Escherichia coli Uropathogenesis In Vitro: Invasion, Cellular Escape, and Secondary Infection Analyzed in a Human Bladder Cell Infection Model. Infect Immun. 2012 May;80(5):1858-67.

14. Bien J, Sokolova O, Bozko P. Role of Uropathogenic Escherichia coli Virulence Factors in Development of Urinary Tract Infection and Kidney Damage. Int J Nephrol. 2012;2012:681473.

15. Kline KA, Dodson KW, Caparon MG, Hultgren SJ. A tale of two pili: assembly and function of pili in bacteria. Trends Microbiol. 2010 May;18(5):224-32.

16. Melican K, Sandoval RM, Kader A, Josefsson L, Tanner GA, Molitoris BA, et al. Uropathogenic Escherichia coli P and Type 1 Fimbriae Act in Synergy in a Living Host to Facilitate Renal Colonization Leading to Nephron Obstruction. PLoS Pathog. 2011 Feb;7(2):e1001298.

17. Hickling DR, Sun T-T, Wu X-R. Anatomy and Physiology of the Urinary Tract: Relation to Host Defense and Microbial Infection. Microbiol Spectr. 2015 Aug;3(4).

18. Edén CS, Hansson HA. Escherichia coli pili as possible mediators of attachment to human urinary tract epithelial cells. Infect Immun. 1978 Jul;21(1):229-37.

19. Salit IE, Vavougios J, Hofmann T. Isolation and characterization of Escherichia coli pili from diverse clinical sources. Infect Immun. 1983;42(2):755-762.

20. Wu X-R, Kong X-P, Pellicer A, Kreibich G, Sun T-T. Uroplakins in Urothelial Biology, Function and Disease. Kidney Int. 2009 Jun;75(11):1153-65.

21. Spurbeck RR, Stapleton AE, Johnson JR, Walk ST, Hooton TM, Mobley HLT. Fimbrial Profiles Predict Virulence of Uropathogenic Escherichia coli Strains: Contribution of Ygi and Yad Fimbriae. Infect Immun. 2011;79(12):4753-4763.

22. Holden N, Totsika M, Dixon L, Catherwood K, Gally DL. Regulation of P-Fimbrial Phase Variation Frequencies in Escherichia coli CFT073 . Infect Immun. 2007 Jul;75(7):3325-34.

23. Spaulding CN, Klein RD, Ruer S, Kau AL, Schreiber HL, Cusumano ZT, et al. Selective depletion of uropathogenic E. coli from the gut by a FimH antagonist. Nature. 2017 Jun 22;546(7659):528-532.

24. Miethke M, Marahiel MA. Siderophore-Based Iron Acquisition and Pathogen Control. Microbiol Mol Biol Rev. 2007 Sep;71(3):413-51.

25. Russo TA, McFadden CD, Carlino-MacDonald UB, Beanan JM, Barnard TJ, Johnson JR. IroN Functions as a Siderophore Receptor and Is a Urovirulence Factor in an Extraintestinal Pathogenic Isolate of Escherichia coli. Infect Immun. 2002 Dec;70(12):7156-60.

26. Watts RE, Totsika M, Challinor VL, Mabbett AN, Ulett GC, De Voss JJ, et al. Contribution of Siderophore Systems to Growth and Urinary Tract Colonization of Asymptomatic Bacteriuria Escherichia coli. Infect Immun. 2012 Jan;80(1):333-44.

27. Kline KA, Lewis AL. Gram-Positive Uropathogens, Polymicrobial Urinary Tract Infection, and the Emerging Microbiota of the Urinary Tract. Microbiol Spectr. 2016 Apr;4(2).

28. Byappanahalli MN, Nevers MB, Korajkic A, Staley ZR, Harwood VJ. Enterococci in the Environment. Microbiol Mol Biol Rev. 2012;76(4):685-706.

29. Fisher K, Phillips C. The ecology, epidemiology and virulence of Enterococcus. Microbiology. 2009 Jun;155(Pt 6):1749-57.

30. Van Tyne D, Gilmore MS. A Delicate Balance: Maintaining Mutualism to Prevent Disease. Cell Host Microbe. 2014 Oct 8;16(4):425-7.

31. Anderson AC, Jonas D, Huber I, Karygianni L, Wölber J, Hellwig E, et al. Enterococcus faecalis from Food, Clinical Specimens, and Oral Sites: Prevalence of Virulence Factors in Association with Biofilm Formation. Front Microbiol. 2016 Jan 11;6:1534.

32. Kau AL, Martin SM, Lyon W, Hayes E, Caparon MG, Hultgren SJ. Enterococcus faecalis Tropism for the Kidneys in the Urinary Tract of C57BL/6J Mice .Infect Immun. 2005 Apr;73(4):2461-8.

33. Hashem YA, Amin HM, Essam TM, Yassin AS, Aziz RK. Biofilm formation in enterococci: genotype-phenotype correlations and inhibition by vancomycin. Sci Rep. 2017 Jul 18;7(1):5733.

34. Medeiros AW, Pereira RI, Oliveira DV, Martins PD, d'Azevedo PA, Van der Sand S, et al. Molecular detection of virulence factors among food and clinical Enterococcus faecalis strains in South Brazil. Braz J Microbiol. 2014 Apr 18;45(1):327-32.

35. Shankar N, Lockatell CV, Baghdayan AS, Drachenberg C, Gilmore MS, Johnson DE. Role of Enterococcus faecalis Surface Protein Esp in the Pathogenesis of Ascending Urinary Tract Infection. Infect Immun. 2001;69(7):4366-4372.

36. Vengadesan K, Narayana SVL. Structural biology of gram-positive bacterial adhesins. Protein Sci. 2011 May;20(5):759-72.

37. Roh JH, Singh KV, La Rosa SL, Cohen ALV, Murray BE. The Two-Component System GrvRS (EtaRS) Regulates ace Expression in Enterococcus faecalis OG1RF. Infect Immun. 2015 Jan;83(1):389-95.

38. Cohen ALV, Roh JH, Nallapareddy SR, Hook M, Murray BE. Expression of the Collagen Adhesin ace by Enterococcus faecalis Strain OG1RF is not Repressed by Ers but Requires the Ers box. FEMS Microbiol Lett. 2013 Jul;344(1):18-24.

39. Singh KV, Nallapareddy SR, Sillanpää J, Murray BE. Importance of the Collagen Adhesin Ace in Pathogenesis and Protection against Enterococcus faecalis Experimental Endocarditis. PLoS Pathog. 2010 Jan 8;6(1):e1000716.

40. Torelli R, Serror P, Bugli F, Paroni Sterbini F, Florio AR, Stringaro A, et al. The PavA-like fibronectin-binding protein of Enterococcus faecalis, EfbA, is important for virulence in a mouse model of ascending urinary tract infection. J Infect Dis. 2012 Sep 15;206(6):952-60.

41. Singh KV, La Rosa SL, Somarajan SR, Roh JH, Murray BE. The Fibronectin-Binding Protein EfbA Contributes to Pathogenesis and Protects against Infective Endocarditis Caused by Enterococcus faecalis. Infect Immun. 2015;83(12):4487-4494.

42. Sillanpää J, Chang C, Singh KV, Montealegre MC, Nallapareddy SR, Harvey BR, et al. Contribution of Individual Ebp Pilus Subunits of Enterococcus faecalis OG1RF to Pilus Biogenesis, Biofilm Formation and Urinary Tract Infection. PLoS One. 2013 Jul 11;8(7):e68813.

43. Leanti La Rosa S, Camila Montealegre M, Singh KV, Murray BE. Enterococcus faecalis Ebp pili are important for cell-cell aggregation and intraspecies gene transfer. Microbiology. 2016 May;162(5):798-802.

44. Arias CA, Murray BE. The rise of the Enterococcus: beyond vancomycin resistance. Nat Rev Microbiol. 2012 Mar 16;10(4):266-78.

45. Minardi D, d' Anzeo G, Cantoro D, Conti A, Muzzonigro G. Urinary tract infections in women: etiology and treatment options. Int J Gen Med. 2011;4:333-43.

46. Kuroda M, Yamashita A, Hirakawa H, Kumano M, Morikawa K, Higashide M, et al.. Whole genome sequence of Staphylococcus saprophyticus reveals the pathogenesis of uncomplicated urinary tract infection. Proc Natl Acad Sci U S A. 2005 Sep 13;102(37):13272-7.

47. Widerström M, Wiström J, Sjöstedt A, Monsen T. Coagulase-negative staphylococci: update on the molecular epidemiology and clinical presentation, with a focus on Staphylococcus epidermidis and Staphylococcus saprophyticus. Eur J Clin Microbiol Infect Dis. 2012 Jan;31(1):7-20.

48. Raz R, Colodner R, Kunin CM. Who are you--Staphylococcus saprophyticus? Clin Infect Dis. 2005 Mar 15;40(6):896-8.

49. Gillespie WA, Sellin MA, Gill P, Stephens M, Tuckwell LA, Hilton AL. Urinary tract infection in young women, with special reference to Staphylococcus saprophyticus.J Clin Pathol. 1978 Apr;31(4):348-50.

50. Rupp ME, Soper DE, Archer GL. Colonization of the female genital tract with Staphylococcus saprophyticus. J Clin Microbiol. 1992 Nov;30(11):2975-9.

51. Rupp ME, Han J, Goering RV. Repeated Recovery of Staphylococcus saprophyticus From the Urogenital Tracts of Women: Persistence Vs. Recurrence. Infect Dis Obstet Gynecol. 1995;2(5):218-22.

52. Kleine B, Gatermann S, Sakinc T. Genotypic and phenotypic variation among Staphylococcus saprophyticus from human and animal isolates. BMC Res Notes. 2010 Jun 10;3:163.

53. Matsuoka E, Tanaka Y, Kuroda M, Shouji Y, Ohta T, Tanaka I, et al. Crystal structure of the functional region of Uro-adherence factor A from Staphylococcus saprophyticus reveals participation of the B domain in ligand binding. Protein Sci. 2011 Feb;20(2):406-16.

54. Gatermann S, Meyer HG, Wanner G. Staphylococcus saprophyticus hemagglutinin is a 160-kilodalton surface polypeptide. Infect Immun. 1992;60(10):4127-4132.

55. Meyer HG, Wengler-Becker U, Gatermann SG. The hemagglutinin of Staphylococcus saprophyticus is a major adhesin for uroepithelial cells. Infect Immun. 1996 Sep;64(9):3893-6.

56. Sakinc T, Kleine B, Gatermann SG. SdrI, a Serine-Aspartate Repeat Protein Identified in Staphylococcus saprophyticus Strain 7108, Is a Collagen-Binding Protein. Infect Immun. 2006 Aug;74(8):4615-23.

57. Sakinç T, Kleine B, Michalski N, Kaase M, Gatermann SG. SdrI of Staphylococcus saprophyticus is a multifunctional protein: localization of the fibronectin-binding site. FEMS Microbiol Lett. 2009 Nov;301(1):28-34.

58. Kline KA, Ingersoll MA, Nielsen HV, et al. Characterization of a Novel Murine Model of Staphylococcus saprophyticus Urinary Tract Infection Reveals Roles for Ssp and SdrI in Virulence . Infect Immun. 2010 May;78(5):1943-51.

59. Gatermann S, Kreft B, Marre R, Wanner G. Identification and characterization of a surface-associated protein (Ssp) of Staphylococcus saprophyticus. Infect Immun. 1992 Mar;60(3):1055-60.

60. Sakinc T, Woznowski M, Ebsen M, Gatermann SG. The Surface-Associated Protein of Staphylococcus saprophyticus Is a Lipase. Infect Immun. 2005;73(10):6419-6428.

61. Meyer HG, Müthing J, Gatermann SG. The hemagglutinin of Staphylococcus saprophyticus binds to a protein receptor on sheep erythrocytes. Med Microbiol Immunol. 1997 Jun;186(1):37-43.

62. Hell W, Meyer HG, Gatermann SG. Cloning of aas, a gene encoding a Staphylococcus saprophyticus surface protein with adhesive and autolytic properties. Mol Microbiol. 1998 Aug;29(3):871-81.

63. Prentice MB. Bacterial comparative genomics. Genome Biol. 2004;5(8):338.

64. Land M, Hauser L, Jun SR, Nookaew I, Leuze MR, Ahn TH, et al. Insights from 20 years of bacterial genome sequencing. Funct Integr Genomics. 2015 Mar;15(2):141-61.

65. Salipante SJ, Roach DJ, Kitzman JO, Snyder MW, Stackhouse B, Butler-Wu SM, et al. Large-scale genomic sequencing of extraintestinal pathogenic Escherichia coli strains. Genome Res. 2015 Jan;25(1):119-28.

66. Vieira G, Sabarly V, Bourguignon PY, Durot M, Le Fèvre F, Mornico D, et al. Core and Panmetabolism in Escherichia coli. J Bacteriol. 2011 Mar;193(6):1461-72.

67. Vejborg RM, Hancock V, Schembri MA, Klemm P. Comparative Genomics of Escherichia coli Strains Causing Urinary Tract Infections. Appl Environ Microbiol. 2011 May;77(10):3268-78.

68. Solheim M, Aakra Å, Snipen LG, Brede DA, Nes IF. Comparative genomics of Enterococcus faecalis from healthy Norwegian infants. BMC Genomics. 2009 Apr 24;10:194.

69. Palmer KL, Godfrey P, Griggs A, Kos VN, Zucker J, Desjardins C, et al. Comparative Genomics of Enterococci: Variation in Enterococcus faecalis, Clade Structure in E. faecium, and Defining Characteristics of E. gallinarum and E. casseliflavus. MBio. 2012 Mar 1;3(1):e00318-11.

70. Raven KE, Reuter S, Gouliouris T, Reynolds R, Russell JE, Brown NM, et al. Genome-based characterization of hospital-adapted Enterococcus faecalis lineages. Nat Microbiol. 2016 Feb 8;1:15033.

OBJETIVOS

Principal

Analisar características associadas à adaptação de uropatógenos ao trato urinário por genômica comparativa.

Secundários

Conhecer a estruturação genômica de representantes locais de uropatógenos isolados na cidade de Porto Alegre (*Escherichia coli*, *Enterococcus faecalis* e *Staphylococcus saprophyticus*).

Automatizar processo de análise em larga escala de genomas depositados em banco de dados públicos utilizando ferramentas de bioinformática.

MANUSCRITO I – Formatado para submissão no periódico *Genome Biology*.

Evaluation of Niche Adaptation Features of *Escherichia coli, Enterococcus faecalis and Staphylococcus saprophyticus* Species: Genome Data Mining Approach of Urinary and Gastrointestinal Strains.

Thiago Galvão da Silva Paim[1], Gustavo Enck Sambrano[1], Renata Soares[1], Mariana Mott[1], Samuel Paulo Cibulski[2], Pedro Alves d'Azevedo[1].

[1] Universidade Federal de Ciências da Saúde de Porto Alegre, UFCSPA, Brazil.

[2] Universidade Federal do Rio Grande do Sul, UFRGS, Brazil.

Abstract

Background

The Urinary Tract Infections (UTIs) are the most common bacterial infection affecting humans and *Escherichia coli*, *Enterococcus faecalis* and *Staphylococcus saprophyticus* are uropathogens often responsible for these clinical manifestations. Although several virulence factors are known to be present on these species, allowing adherence and colonization of the bladder mucosa, the reservoir of these strains is the gastrointestinal tract of the host. The aim of the study was evaluate genomic features for niche adaptation of urinary and gastrointestinal strains of these species by data mining approach.

Results

In *E. coli* strains, the repertoire of genes is higher than previous studies and the ratio of genes associated to primary metabolism does not depend of bacteria niche, with exception of Cell cycle-division, Cell Motility and Secondary Metabolite Metabolism. Urinary tract isolates of *E. coli* had superior density of virulence and resistance genes carried by prophages, differently of gram-positive strains. *E. faecalis* genomes from urinary tract are more divergent based on pangenome analysis, with *S. saprophyticus* showing significant ratio of gene repertoire associated to ion transport and metabolism compared to others uropathogens.

Conclusion

The urinary and gastrointestinal strains of the species evaluated in the study have an open pangenome, with groups of functional annotation genes associated to specific niches. In addition, gastrointestinal isolates of *E. coli* are important reservoir of resistance genes.

Keywords

niche adaptation, pangenome analysis, uropathogens, urinary tract infections

Background

The urinary tract infections (UTIs) are common bacterial infections that affect individuals of all ages [1] and the recurrence can affect 40% of female patients [2]. The cost associated to UTIs are high in developed countries, approximately US$ 3 billions per year in the USA [1] and €58 million in France [3].

Escherichia coli is the most frequent microorganism recovered from UTIs, with pathotypes associated to enteric and extraintestinal infections [4]. However, gram-positive species are also considered etiological agents of UTIs, mainly *Enterococcus faecalis* and *Staphylococcus saprophyticus* [5].

One important characteristic of these species is that the major niche reservoir is the gastrointestinal tract. The *Escherichia coli* strains are members of human colon microbiota at low abundance, but in the infant gut this genus is rich [6]. *S. saprophyticus* frequently colonizes the rectum of human and it is considered medium-pathogenic species of staphylococci. In addition, this specie is the only true uropathogen of the genus [7], and *Enterococcus faecalis* is an adapted member of gastrointestinal tract of mammals and others hosts, with pathogenic strains associated to acquisition of diversified mobile elements integrated on genome [8].

There are well characterized genes associated to pathogenesis of these strains, which confer host adherence, colonization and bacterial survival in urinary tract [4, 5]. However, molecular features present in both enteroaggregative and uropathogenic strains of *E. coli* have been shown in same isolate [9], supporting the hypothesis that others genetic determinants play important role on bacterial persistence in different host niche.

The aim of the study was analyze strains recovered from gastrointestinal and urinary tract of same species for data mining genomic features to better understand niche-specific adaptation and selection.

Methods

Bacterial strains and Genomes

This study was performed with bacterial strains from Laboratory of Molecular Microbiology - UFCSPA and genomes deposited on NIH-NCBI database. Two samples of uropathogenic bacteria had their genome sequence determinated previously - *Enterococcus faecalis* F165 [10] and *Escherichia coli* E2 [9] and whole genome sequencing was performed in a isolate of *Staphylococcus saprophyticus* strain *SS545* recovered from a female patient with significant bacteriuria. Phenotypically, this uropathogen was strong biofilm producing by microtiter plate method. The sequencing was performed in a next-generation Illumina MiSeq platform, paired-end and read length of 150 bp. The draft genome was assembled with SPAdes [11] and annotated by Prokka v.1.12 with database created from *Staphylococcus saprophyticus* genomes [12]. *Staphylococcus saprophyticus* SS545 had 414 contigs, 2,492 coding sequences (CDS) and 2,597,312 bp.

Representative genomes of *Enterococcus faecalis*, *Escherichia coli* and *Staphylococcus saprophyticus* strains were obtained from NCBI Microbial Genomes (https://www.ncbi.nlm.nih.gov/genomes/MICROBES/microbial_taxtree.html). A total of 5,295 genomes of *Escherichia coli*, 432 of *Enterococcus faecalis* and 21 *Staphylococcus saprophyticus* strains were downloaded to create a local database for bioinformatics analysis.

Only bacterial genomes containing information about source of isolation and recovered from human host were analyzed in the study. Among these, the genomes were grouped according to site of isolation: *Escherichia coli* (Urinary Tract - UT (n=274), Gastrointestinal Tract – GI (n=1,190) and E2 cluster (n=382), *Enterococcus faecalis* (UT - n=7, GI - n=6) and *Staphylococcus saprophyticus* (all belonging to UT - n=19).

Genome clusterization of *Escherichia coli*

As genome analysis is limited due to computational performance, the *Escherichia coli* genomes were clustered using data matrix generated by Tetranucleotide correlation (TETRA) using the software pyani (available on https://pypi.python.org/pypi/pyani/). Bi-dimensional matrix of pairwise TETRA correlations was clustered by the hierarchical method Simple K-means using the python scientific computing package Scikit-Learn [13] and the software Weka [14]. To find the number of cluster that minimize the sum of squared errors (SSE) generated by the algorithm, numerical derivatives were calculated by both methods and the

genomes belonging to same cluster of *Escherichia coli* E2 were recovered to pan-genome analysis.

Genome Groups

The strains belonging to genomes groups (*Escherichia coli*: UT; *Enterococcus faecalis:* UT+GI+Others; and *Staphylococcus saprophyticus*: UT) were confirmed at species level by Average Nucleotide Identity (ANI). The method was computed among microorganisms of the database using the module *pyani* for python computer programming language (available on https://pypi.python.org/pypi/pyani/) using BLASTN algorithm with fragment size of 1020 bp, with the exception of *E. coli* GI group that tetranucleotide frequency correlation was computed due to computational limitation. Heatmaps were generated with command-line matplotlib package of python language for data visualization.

Pan-Genome Analysis

The genomes sequences were submitted to pan-genome analysis using the ROARY command-line software [15]. Briefly, the respective genomes were converted from GENBANK to GFF3 format file containing annotated coding regions and these were grouped in clusters based on homology. Representative sequence from each cluster was used for gene distribution among the isolates and to build the pan-genome. Core and accessory genome was defined by genes present in $\geq$ 99% and < 99% of the strains, respectively. In addition, core sequences were concatenated and aligned with MAFFT [16] to deduce phylogenetic tree.

Functional Annotation of Pan-Genome

The pan-genome sequences were analyzed by functional characterization of the genes using the Clusters of Orthologous Groups of proteins (COGs) database (http://www.ncbi.nlm.nih.gov/COG/) [17]. The genes were *in silico* translated and a python script was written to perform a BLASTP search (cutoff: e-value $\leq 10^{-10}$) of each CDS from a local COG database. The functional assignments of the most significant sequences were extracted to compute the pan-genome functional profile of each isolates clusters.

Phylogenetic Analysis

Two approaches were used to perform phylogenetic analysis. First, the reconstruction of phylogenetic tree among isolates of the same group was performed by gene-content information from pan-genome analysis. Briefly, the output from ROARY was converted to binary matrix. The transpose of this matrix was used to convert it to a distance-based matrix, followed by neighbor-joining (NJ) tree estimation [18] by '*ape*' package from R statistical computing software (https://www.r-project.org) [19]. The second approach for phylogenetic tree inference was the sequence-based method from alignment of core pan-genome after concatenation of individual genes (supermatrix). NJ method was performed by '*ape*' [19] and '*phangorn*' packages from R [20]. The model of nucleotide substitution used in the study was Kimura 2-parameter (K80), and inference of phylogeny by NJ was performed with bootstrap of 1000 replicates (with the exception of *E. coli* E2 cluster which bootstrap was 100 replicates due computational limitation) and both tree was drawn by '*ggtree*' R package [21].

Prediction of Prophages Sequences

To verify the contribution of integrative elements in the structure of bacterial genomes, prophages sequences were predicted using PHAge Search Tool (PHASTER), available on www.phaster.ca [22].

Resistome

To evaluate the distribution of resistance-associated genes from genomes and prophage sequences, the command-line software ResFinder [23] was used with the following parameters: at least alignment length of 60% and nucleotide identity of 95%. Informations about gene and antimicrobial class as well as frequency of the resistance genes were recovered for data analysis.

Virulence Factors Analysis

The prophage sequences were analyzed by presence of virulence factors (VF) using the Virulence Factor Database (VFDB) [24]. The search of VF's was performed by an in-house python script using BLASTP algorithm (parameters: e-value $\leq 10^{-3}$, alignment coverage $\geq$ 50% and identity $\geq$ 30%), with CDS recovered from prophages homologous sequences reported after PHASTER analysis.

Results

Pan-Genome analysis and Functional Annotation

The pan-genome analysis demonstrated different numbers of genes among the species verified (Table 1, Figure 1). The biggest pan-genome was from the *E. coli* gastrointestinal tract, with > 93,000 genes, but core/pan-genome ratio of 1.29%. From urinary tract genomes, pan-genome presented a total of 34,936 genes and ratio of 7.14%. The ratio of pan/core genes in gram-positive species was elevated, superior to 35% (*E. faecalis*: GI = 46.5% *versus* UT = 37.07%; *S. saprophyticus*: 47.79%), with approximately 4,200 genes in pan-genome.

Genes from pangenome were grouped in twenty-five functional annotations clusters, as Amino acid Transport and Metabolism, Carbohydrate Transport and Metabolism, Cell cycle-division, Cell Motility, Cell Wall or Membrane Biogenesis, Coenzyme Transport and Metabolism, Cytoskeleton, Defense Mechanisms, Energy Production and Conversion, Extracellular Structures, Function Unknown, General Function Prediction Only, Inorganic Ion Transport and Metabolism, Intracellular Traffic, Lipid Transport and Metabolism, Mobilome, Nucleotide Transport and Metabolism, Pos-translational Modification, Replication and Recombination and Repair, RNA Processing, Secondary Metabolite Metabolism, Signal Transduction, Transcription, Translation, and Without Functional Annotation.

An elevated number of genes did not have homologous sequences in COG database, therefore, they were categorized as 'without functional annotation', ranging from approximately 25% in gram-positive isolates to superior to 40% in *E. coli* strains (Figure 2). The association between the ratio of functional annotation and the site of isolation of each strains was analyzed by Pearson's Chi-squared test. There was a significant association on

Escherichia coli genomes ($p < 0.001$), in which urinary tract isolates (UT) had a superior ratio of genes associated to Cell cycle-division, Cell Motility and Secondary Metabolite Metabolism; and a lower percentage of genes associated to Mobilome compared to gastrointestinal isolates.

Genome clusterization of *Escherichia coli*

From 2,254 *E. coli* strains belonging to diversified sources of isolation, the cluster that include *E. coli* E2 strain [9] by simple K-means method recovered 382 genomes (cut-off number of clusters that minimizes the SSE = 10) (Figures 3). The most frequent source of isolation was GI tract (n=250), following by UT (n=56) and bloodstream (n= 37).

Prophages Analysis

A total of 13,233 prophage sequences were predicted for all genomes belonging to GI (n=1196) and UT (300), with median of 8 prophages per genome [1st quartile: 6; 3rd quartile: 11]. For *Escherichia coli* strains belonging to GI group (n=1,190), the median was 9 prophages, followed by *E. coli* UT (n=274, median = 7), *Enterococcus faecalis* UT (n=7, median=6), *E. faecalis* GI (n=6, median=3) and *Staphylococcus saprophyticus* (n=19, median=2). According to isolation tract, GI isolates had more prophages predicted that UT [GI – median=9 *versus* UT – median=7; p-value <0.001 by Wilcoxon rank sum test], and according to bacteria species only *E. coli* from GI Tract had a higher number of prophages predicted than UT [GI – median=9 *versus* UT – median=7; p-value <0.001 by Wilcoxon rank sum test; *Enterococcus faecalis*: GI – median=3 *versus* UT – median=6; p-value=0.07018] (Figure 4).

Genes encoding resistance to antibiotics located inside of prophages sequences were predicted only in 144 of 1,496 genomes (9.6%, all in *E. coli* strains). There was no significant correlation between prophages number and acquisition of resistance genes by bacterial genomes [Pearson correlation coefficient (r) = 0.0955, p-value = 0.2545, CI95% (-0.0691 to 0.2552)], therefore the GI group had a significant, but weak correlation [r = 0.2178, p-value = 0.0183, CI95% (0.0378 to 0.3841)] (Figure 5). Regarding density of resistance genes by prophage, UT isolates had a superior median (Figure 6 and 7). A total of 409 genes were

found, containing genetic determinants that confer resistance to trimethoprim and sulphonamide most frequently found between prophages (GI – 103 and 84, UT – 23 and 16, respectively) (Figure 8). From these, co-occurrence of *dfrA7* gene (dihydrofolate reductase - trimethoprim resistance) and *sul1* gene (dihydropteroate synthase - sulphonamide resistance) in same prophage was frequently recovered from *E. coli* GI group (45 co-occurrences).While in *E. coli* UT group the most common co-ocurrence found was *dfrA14* and *mphA* gene (macrolide 2'-phosphotransferase I – macrolide resistance) (5 co-occurrence). To both groups, co-occurrence of three resistance genes in a same prophage predicted was 14 (*dfrA7*, *sul1* and *catA1* - chloramphenicol acetyltransferase) (Figure 9).

A total of 13,661 virulence genes were found inside of prophage sequences, and these were recovered in 1,460 of 1,496 genomes (97.6%). The median of virulence factors in prophages was 7 [1st quartile: 4; 3rd quartile: 11] and the density of virulence genes per prophage was equal 0.86 [1st quartile: 0.50; 3rd quartile: 1.29]. By isolation source, the median of density was 0.82 and 1.10 to GI and UT, respectively (p-value < 0.001 by Wilcoxon rank sum test). *E. coli* UT cluster had significant higher density that GI cluster (p-value <0.001 by Wilcoxon rank sum test; *E. faecalis* – p-value = 0.2192). Strong and moderate correlation between number of prophages and virulence factors carried by these integrative elements was found in GI and UT groups, respectively (Figure 10). The most frequent genes recovered by *E. coli* genomes were *rck*, *ipaH*, *ipaH2.5*, *ompD*, *perC/bfpW* and *flmH*, while in *E. faecalis* were *feoB*, *mgtC* and *mll6359*. For *S. saprophyticus*, *msrA/B*, *lisR* and *mga* were found in prophages (Table 2).

Resistome

For *Escherichia coli* isolates, 771 of 1,475 (52.3%) had resistance genes predicted by ResFinder tool. Regarding to genome groups, 635 of 1,199 (53%) and 136 of 276 (49.3%) showed at least one resistance determinant in GI and UT clusters, respectively. To both groups, the most prevalent phenotype class which confers resistance to antimicrobial agents were aminoglycoside, sulphonamide and beta-lactam (Figure 11 and 12). All *Enterococcus faecalis* strains also showed genes associated to antimicrobial resistance in which vancomycin, aminoglycoside and tetracycline were the most prevalent (Figure 13 and 14). Seven of nineteen *Staphylococcus saprophyticus* genomes (37%) had resistance genes

predicted, with four genes that confer phenotype resistance to macrolide, lincosamide and streptogramin B (MLS-B, *msr(A)* gene), three genes related to macrolide resistance (*mph(C)*), two of tetracycline resistance (*tet(K)*) and one for each resistance phenotype (aminoglycoside – *str*, trimethoprim – *dfrG*, beta-lactam – *blaZ*).

There was a significant correlation between isolation date and the number of resistance genes predicted from genomes of *E. coli* (Figure 15). The analysis by class of resistance genes, showed significant correlations among aminoglycoside (r = 0.1657, p-value < 0.001, CI95% (0.0686 to 0.2596)), phenicol (r = 0.2486, p-value = 0.0055, CI95% (0.0749 to 0.4077)), sulphonamide (r = 0.1455, p-value = 0.0027, CI95% (0.0749 to 0.4077)) and beta-lactam (r = 0.1258, p-value = 0.0210, CI95% (0.0191 to 0.2297)). For the variables, correlation was examined separately for *E. coli* group (GI and UT). Although significant correlation was demonstrated on the phenotype of aminoglycoside and phenicol resistance for all genomes of *E. coli*, only GI group was significant. In addition, only in the GI group a significant negative correlation was found to phenotype of trimethoprim resistance (Figure 16, 17 and 18).

For *Enterococcus faecalis* genomes, there was only significant negative correlation between isolation date and number of macrolide resistance genes (r = -0.5523, p-value = 0.0328, CI95% (-0.8298 to -0.0558) (Figures 19, 20 and 21).

Discussion

Uropathogenic microorganisms are an important cause of morbidity in humans, affecting female and male of all ages. Although gram-negative bacteria (*Escherichia coli*) are the most common agents of these infections, gram-positive microorganisms like *Enterococcus faecalis* and *Staphylococcus saprophyticus* are frequently recovered from UTI. The gene repertoire that confers adherence, colonization and ability to survive at the urinary tract it is important to uropathogens, and some genetic determinants are known to execute these functions [1].

Pangenome analysis is defined by the repertoire of genes based on set of genomes, and information about closed or open pangenome can be determined analysing the pattern of acquisition of new genetic elements [25]. Furthermore, the bacteria lifestyle is affected by capacity of niche adaptation, which sympatric species develops in microbial community and

has an open pangenome [26]. The *Escherichia coli* genomes of the study had low ratio core/pan for both isolation sources (GI or UT), typical of sympatric species. In addition, based on pangenome size, it increased when new genomes were added, showing that *Escherichia coli*, *Enterococcus faecalis* and *Staphylococcus saprophyticus* species represents an open pangenome model independently of the niche. Similar conclusion of pangenome model was obtained by a previous study that analyzed seventeen *E. coli* genomes from different pathovars and one commensal isolate [27]. However, isolates from urinary and gastrointestinal tract had pangenomes containing more than 30,000 and 90,000 genes respectively, that represents 2.7x to 7.2x of the pangenome size reported by Rasko et al (2008) [27], although the number of core genes from UT strains were similar (2,493 *versus* ~2,200).

In a previous study, the pangenome and the metabolic network was obtained from twenty-three *E. coli* strains isolated from various niches, as commensal, intraintestinal and extraintestinal. The core genome of these strains had 1,957 genes that represent ~ 10% of the total pangenome, but the core reactions were conserved, representing ~ 70% of panmetabolism [28]. Only *E. coli* strains had a significant association between the functional class of pangenome genes and the isolation tract, with genes annotated to cell cycle-division, cell motility and secondary metabolite metabolism significantly higher in urinary tract isolates. Consequently, genes associated to amino acid, lipid, carbohydrate, inorganic ion and nucleotide metabolism, as well as energy production, translation and replication did not have difference in proportion with gastrointestinal strains. Metabolic categories associated to biosynthetic processes and energy metabolism were superior in core reactions of *E. coli* strains [28], that corroborate the hypothesis which these metabolic processes do not vary significantly because there are basic metabolites as precursors, differently from functions related to metabolites of the environment [28].

Analysing the gene-content of secondary metabolite metabolism from *E. coli* in this study, the urinary tract strains had a high frequency of genes associated to non-ribosomal peptide synthetase component F, which is a homologous gene to the enterobactin component F by COG id. Iron uptake is essential for bacterial metabolism as a biocatalyst or as an electron carrier in reactions associated to carbon metabolism, replication and DNA repair and energy production [29]. Enterobactin system is enconded by operon *entCDEBAHF*, that synthesizes siderophore protein from shikimic acid pathway [30, 31]. In mammalian hosts, iron is available to bacteria strains from various sources, but dietary iron in the

gastrointestinal tract is important to colonization and commensalism and is limited in extraintestinal infection sites [29, 32]. The presence of sixty-four homologous *entF* genes shows the diversity of this enzime in the pangenome of *E. coli* urinary tract isolates. This enzime and others of operon convert 2,3-dihydroxybenzoate to enterobactin, which is secreted by bacteria [30]. Currently, is not well clear if variant proteins of EntF modify enterobactin secreted by these strains, unless the chemical structure and compositon of this siderophore are experimentaly determinated. However, in *Salmonella* species the production of modified enterobactins is relevant to infection processes by preserving the iron-binding activity but evading the siderocalin activity. This protein is expressed by immune cells of the host that bind to enterobactin-iron complex and present an antibacterial activity [33]. In addition, in urinary tract infections the oxidative stress is elevated [34], which induces the up-regulation of enterobactin synthesis [31]. These findings suggest that homologous enterobactin F genes have a relevant role on uropathogenesis of *E. coli* strains.

Although the proportion of genes of the functional class ‘cell motility’ in UT *E. coli* group was higher that GI strains, the gene-content for both groups did not have significant difference. The pilin (type I fimbrae) was the most frequent protein recovered in the pangenome, with 383 homologous genes in urinary tract strains. These peritrichous fimbriae are coded by operon *fim*. The FimH protein has an adhesin function of adherence to host cells and invasion of bladder epithelium [35]. In addition, pilin allows adhesion of enterotoxigenic strains (ETEC) in intestinal epithelial surface and the release of toxin, both essentials for virulence of these isolates [36]. Although the diversity of homologous gene sequence have been found in pangenome of *Escherichia coli* species, the protein sequences were clustered in nine delimited branches, independently if recovered from urinary or gastrointestinal tract (with exception of one branch) (Figure 22). The *E. coli* genome showed a high frequency of fimbrial and fimbrial-like adhesin proteins, with some strains (CFT073) with more than ten fimbrial operons. It was demonstrated that the number of fimbrial type is significantly high in uropathogenic *E. coli* and these strains have a superior number of virulence factors compared to commensal isolates [37]. The genetic variation found in fimbrial genes in the study can be related to elevated putative fimbrial-like homologous or/and diversity of *fimA-H* alleles. The last hypothesis is supported by higher frequency of non-synonymous mutations of *fimH* locus, allows that this adhesin has strong structural positive selection in extraintestinal *E. coli* [38] and evidence of genetic variation of *fimA* gene in atypical enteropathogenic strains [39].

It was not found differences in gene-content between UT and GI *E. coli* genomes to Cell cycle control, cell division, chromosome partitioning (COG 1196). The chromosome organization is crucial for microbial cell replication, and there are different molecular systems that execute the bacteria chromosome segregation [40]. Although the proportion of genes from functional annotation 'cell cycle-division' has been superior in urinary tract isolates, there was no evidence that any particular genetic determinant is unique to UT group.

The isolates of *Enterococcus faecalis* had the pangenome analysis performed clustering the genomes by isolation source, differently of previous studies [41, 42]. First, relevant properties about pangenome computation should be available. As more divergent the genome – based on gene-content, less homologous sequences are shared. UT genomes have a core genome with 1,559 CDS and GI *E. faecalis* with 2,126, even if they have similar pangenome size (Table 1). This finding suggests that urinary tract isolates share less homologous genes compared to gastrointestinal strains. In addition, all unique COG genes description in UT genomes belong to accessory genome (Table 3). Although these genes are part of pangenome of urinary strains, they are not spread through all genomes; therefore they are not a common feature for UT. The computational analysis of pangenome was performed with core gene threshold of 99%. Therefore, for genomes of *Enterococcus faecalis*, if a homologous gene was not present in all isolates it was computed as accessory gene. However, there was a unique COG gene often found in urinary strains, the ABC-type molybdenum transport system/ATPase component/photorepair protein PhrA (six of seven genomes). ABC-transporters constitute a paralogous family with ATP-hydrolyzing or nucleotide-binding domain widespread in archeae, bacteria and eukarya. It belongs to AAA+ superfamily which uses the energy from ATP to molecular process [43]. The function of these proteins are mainly transmembrane transport and several proteins were found in bacteria performing uptake of nutrients and export of toxins and metabolites (functional categorie of importers) and DNA repair [44]. In addition, molybdenum is a trace element cofactor for redox reactions associated to metalloenzymes in nitrogen, sulfur and carbon cycles [45, 46]. Compared to gastrointestinal tract, urine environment has high concentration of nitrogen compounds (amino acids, creatinine and urea), organic acids and inorganic ions (sodium, potassium and ammonia), and the major metabolites in urinary tract are amino acids and carbohydrates [47]. The presence of homologous gene for ABC-type molybdenum transport system in pangenome of UT *E. faecalis* apparently confers selective advantages to use this trace element because the

primary route of molybdenum excretion is the urine [48], allowing the use of diversified metabolic pathway for maintenance of celular metabolism.

Compared to *Escherichia coli* and *Enterococcus faecalis* strains recovered from urinary tract, the pangenome of *Staphylococcus saprophyticus* species revealed significant association among proportion of functional categories of amino acid, coenzyme and inorganic ion transport and metabolism (Pearson's Chi-squared test, $p < 0.001$, Figure 2). The metabolic capacities of these gram-positive bacteria were demonstrated in previous studies [49, 50]. The reference study of *Staphylococcus saprophyticus* genome was performed by Kuroda et al (2005) with the uropathogenic strain ATCC 15305. Although *S. saprophyticus* ATCC 15305 has specific adhesin for adherence to urinary tract host tissue, this strain possess relevant paralogs genes expansion associated to osmotolerance and inorganic ions transporter, as sodium antiporters, sulfate and divalent transporters, as well as systems for nitrogen metabolism [51]. Differently to *S. aureus* and *S.epidermidis* species, *Staphylococcus saprophyticus* pangenome has a high number of genes associated to transport metabolites, also superior in proportion compared to others uropathogenic strains of the study.

Prophage sequences, besides mobile DNA elements, are common source of variability among isolates of same species. Integrated in bacterial chromosome, genetic elements as resistance genes, virulence factor and genes for niche adaptation can be interchanged among different strains or replicons (ex. plasmids) [52]. The elevated diversity of prophage number predicted in genomes of *Escherichia coli* (Figure 4) shows that these genetics elements are important for chromosome structure and potential selective adaptation.

In the bacterial genomes of this study, the distribution of prophages was evaluated by isolation source feature as isolates recovered from gastrointestinal or urinary tract. The genome database had a high number of GI strains of *Escherichia coli* species, alike UT strains. For all, there was an important variability of prophage recovered, with strains showing number of predicted sequences higher than twenty-five integrative elements. In addition, the median of prophages in GI group was superior of UT for all genomes and particularly to *E. coli* species, although for *Enterococcus faecalis* this difference was not significant.

The presence of elevated number of prophage in gastrointestinal strains of *E. coli* is a common feature of enterohemorrhagic O157:H7 serovar. A previous study found an average number of sixteen prophages in these pathogenic isolates, many carrying the virulence-associated gene of Shiga Toxin *stx* [53]. Considering that GI isolates are more susceptible to

prophage integration, logically it is expected that strains recovered from gastrointestinal tract exhibit, based on chance, more prophages carrying genetic determinants that enable selective advantages. These include genes for antimicrobial resistance and tolerance, adhesion, immune evasion mechanism and nutrition acquisition strategies, all virulence factors candidates to an evolutionary history of success inside of prophage sequences [54].

There was an increase of genes that confer antimicrobial resistance at the proportion that the number of prophages incorporated in bacterial genome of *E. coli* recovered from gastrointestinal tract increased, although for urinary tract strains this correlation was not valid. However, for GI group, it was found a higher number of integrative elements carrying genetic determinants other than resistance genes. These facts are straightforward for the analysis of density of resistance genes by prophage sequences predicted in the study. Even though strains isolated from GI tract may have a superior genomic plasticity to acquisition of prophage sequences, these elements did not show often carry genes that confer selective advantages as antimicrobial resistance. There are a significant difference in genes density associated to antimicrobial resistance and virulence between *E. coli* groups, with strains recovered from UT showing also superior frequency of homologous genes of virulence factors carried by prophages. Therefore, isolates of *E. coli* from urinary tract were more cost-effective in relation to the acquisition of integrative elements, but they showed genetic elements for selective advantages per prophage superior compared to strains isolated from gastrointestinal tract.

The Figure 7 shows an example of density of resistance genes carried by urinary tract *E. coli* strain MGH 58. In the complete prophage sequence that was predicted (Figure 7 b), there was eight genes encoding resistance to trimethoprim, macrolide, aminoglycoside and beta-lactam antibiotics. Although the co-occurrence of different resistance genes was not often found among the prophage sequences, there were integrative elements that carried two or more.

Analysing the gene-content of virulence factors homologous predicted inside of prophage sequences, there were not significant differences among *E. coli* groups. The most common genes shared by prophage sequences between urinary and gastrointestinal strains was *rck*, *ipaH* and *mlr6326* (putative DNA invertase). Resistance to complement killing protein (Rck) is found often in virulence plasmids of *Salmonella* species. This is an outer membrane beta-barrel protein that allows tolerance to microbial cell death by complement

immune system of host, adherence binding to fibronectin and laminin, as well as tissue invasion [55]. In addition, *ipaH* – invasion plasmid antigen – codes for a protein with repetitive regions of leucine amino acid. This protein is related to type III secretion system in gram-negative bacteria and the symbiosis process between host and pathogen. It is often found in genomic elements as pathogenicity island or plasmids and can be shared among different strains by horizontal gene transfer [56]. Interestingly, DNA invertase are recombinase genes found in bacteria and phages genomes with low frequency. In addition, the main role of these genes is coupling or switching promoter sequence to CDS, modulating the gene expression [57]. In *Salmonella* species, the flagellar phase variation is controlled by invertion of DNA segment of promoter region of flagellar protein, occurring in a sub-set of bacterial population. This mechanism is important when changes in environment happen, giving selective advantages to the pathogen [58]. It is known these genes are important to bacterial virulence. They were common between gastrointestinal and urinary tract isolates of *E. coli*, and only the presence of these genetic elements it is not enough to evaluate the role of prophage frequencies in niche adaptation.

The resistome recovered from the study demonstrated that the number of resistance genes carried by isolates of *E. coli* increased over the last decades. Independently of isolation source, the *E. coli* isolates had an important increase of genes that confer resistance to aminoglycoside, phenicol, sulphonamide and beta-lactam antimicrobial agents. In a retrospective study that evaluated the historical changes of *E. coli* resistance in US, Tadesse et al (2012) sampled strains isolated from humans and animals during 1950-2002 and performed phenotypic antimicrobial susceptibility test (AST) to 15 antimicrobial agents. From human source, the study demonstrated that 65% of the strains were pan-susceptible [59]. Our data had genomes isolated from 1970 to 2016, and the antimicrobial susceptibility based on absence of resistance gene predicted was 48%. In addition, our data supports the increasing trend in resistance to ampicillin (beta-lactam) and sulphonamide, but not to tetracycline [59]. The most frequent beta-lactamase gene found in genomes of *E. coli* was *blaTEM-1b*, which is often associated to ampicillin resistance in this species [60].

Although significant correlation has been found for acquisition of aminoglycoside resistance genes over time in all *E. coli* genomes, only for gastrointestinal strains the trend was significant. There were a diversified number of resistance genes for aminoglycoside class and these are at same proportion in UT or GI strains (Figure 12). There were genes found

only in gastrointestinal isolates, as *aph(3')-Ic*, *aph(3')-IIa*, *aph(3')-XV*, *aph(4)-Ia*, *aac(3)-IVa* and *aadA12*, but there were genes also found only in urinary strains as *aadA16*, *rmtC*, *aph(3')-VIa*, *rmtE* and *aadD*, that corroborate the hypothesis that gastrointestinal isolates have an important role as reservoir to resistance genes [61, 62] and the presence of these molecular determinants are increasing on *E. coli* genome, but some selective pressure are fixing determined genes based on adaptation to niche of *E. coli* isolates.

Quinolones (ciprofloxacin and norfloxacin) are often suggested as antibiotics prophylaxis for recurrent urinary tract infections in women [63]. Interestingly, a significant proportion of UT *E. coli* showed genes associated to Fluorquinolones and Aminoglycoside resistance. An unique resistance gene was recovered by *in silico* approach, the *aac(6')Ib-cr*. This gene is a variant of an aminoglycoside acetyl transferase that confers decreased susceptibility by chemical modification of quinolone molecule and can be mainly transfered by mobile elements as plasmids [64], although chromosomal location was found associated to integrons [65]. Although the main mechanism of quinolone resistance is mutation on bacterial DNA gyrase or topoisomerase IV [66], the significant presence of *aac(6')Ib-cr* in urinary strains of *E. coli* could be associated to beneficial adaptation due selective pressure, even the gastrointestinal isolates pursue a superior proportion of genes associated to Quinolone Resistance (Figure XX) as q*nrS1* (decrease the binding to target), *qepA, oqxB* and *oqxA* (efflux pumps) [66]. This evidence shows the role of gastrointestinal strains as reservoir of resistance genes.

There is an important difference on evolutionary history between *Escherichia coli* and *Enterococcus faecalis* strains when it is computed the phylogeny based on core genes (supermatrix) and gene-content. Although both methods are phylogenomic approaches [67], they did not show the same history for *E. faecalis* Strain F165 [10] and *E. coli* Strain E2 [9] (Figure 23, 24 and 25 respectively).The exception was *Staphylococcus saprophyticus* Strain SS545 (Figure 26).

The Figure 23a shows phylogenetic tree based on alignment of concatenated 1,396 core genes of 36 *Enterococcus faecalis* strains isolated from diverse sources. In addition, tree based on gene-content matrix was computed to same strains in Figure 23b. The clade of *E. faecalis* F165 does not share the same strains between phylograms. Considering that phylogeny based on alignment is the true evolutionary history, the F165 strain is related to isolates from blood, urine and faeces (Figure 23a). Similarly, F165 is related to isolates from

wound exudate, blood, sputum and urine (Figure 24b) when gene-content is evaluated. The cluster of *E. coli* E2 was built from 2,254 genomes by the clusterization method using tetranucleotide correlation. The cluster of strain E2 had the pangenome analyzed and same method of phylogenomic reconstruction was performed similarly to *E. faecalis* F165. The Figure 24 and 25 shows similar pattern found in *Enterococcus* strains, in which the clades generated by core and gene-content do not share same strains. Differently of *E. faecalis*, all strains of *E. coli* were recovered from gastrointestinal source. The gene gain – mainly by horizontal gene transfer – and gene loss are important source of variability in pathogenic bacteria [68]. The data demonstrated that the evolutionary relationship among strains is not associated to niche adaptation.

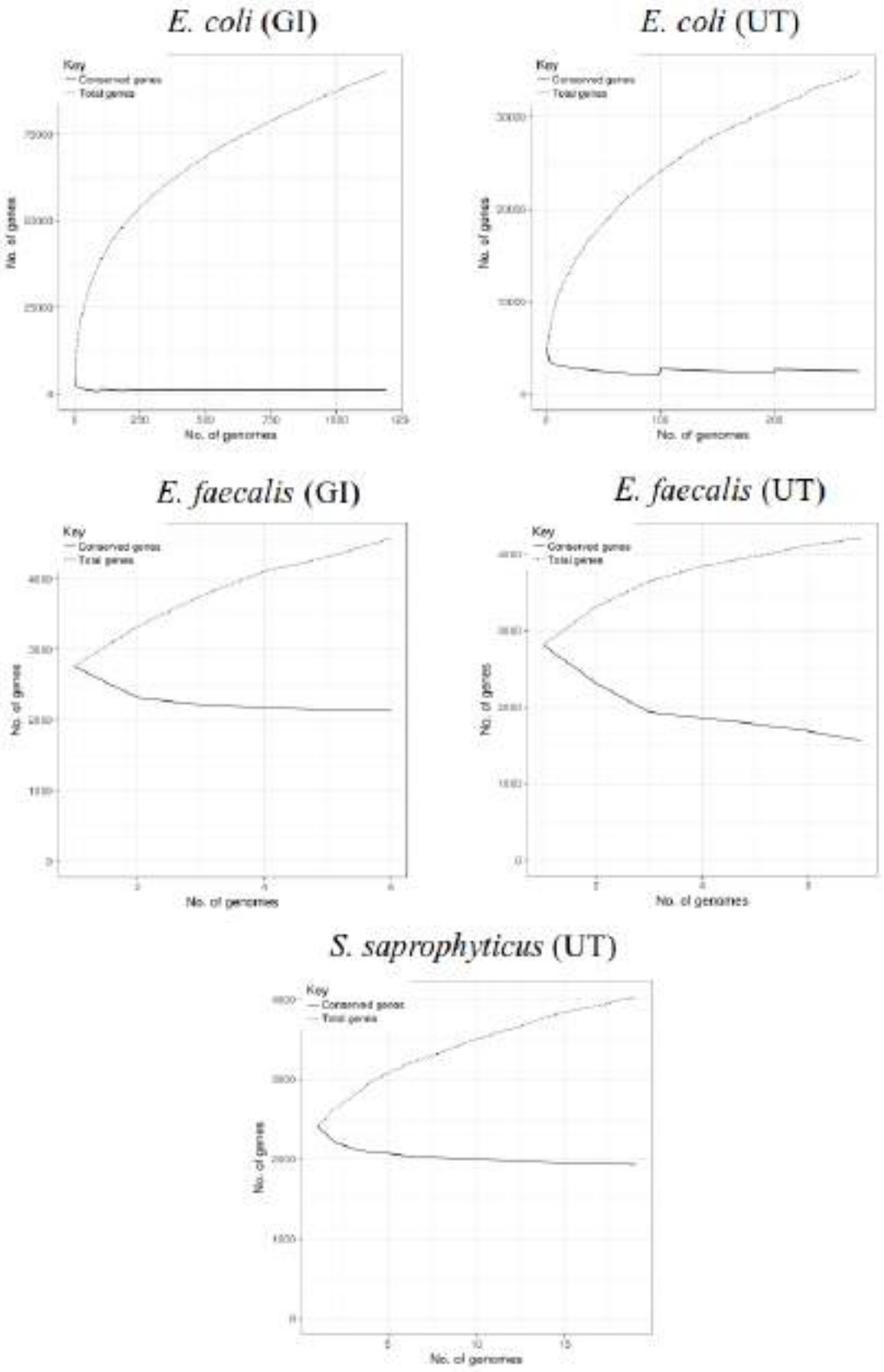
Conclusion

The study showed that all species have an open pangenome independently of isolation source. For *E. coli* strains, the repertoire of genes is higher than previous studies and the ratio of genes associated to primary metabolism does not depend of bacteria niche, however, homologous genes to siderophore proteins in urinary tract isolates could contribute – as fimbrial-like proteins – for uropathogenesis of *E. coli* isolates. These same genomes showed often prophages sequences carrying genetic determinants of resistance and virulence, although the gene-content of these mobile elements did not show differences with gastrointestinal isolates. In addition, *E. faecalis* genomes from urinary tract are more divergent based on pangenome analysis, with *S. saprophyticus* showing significant ratio of gene repertoire associated to ion transport and metabolism compared to others uropathogens.

543 Table 1. Summary of pangenome analysis among bacterial genomes of the study.

Specie	Genome Groups	Number of genes			Rat (core/
		Pan-genome	Core Genes	Accessory Genes	
richia coli	GI	93,222	1,200	92,022	1.29
	UT	34,936	2,493	32,443	7.14
ococcus faecalis	GI	4,572	2,126	2,446	46.5
	UT	4,205	1,559	2,646	37.0
ylococcus saprophyticus	UT	4,034	1,928	2,106	47.7

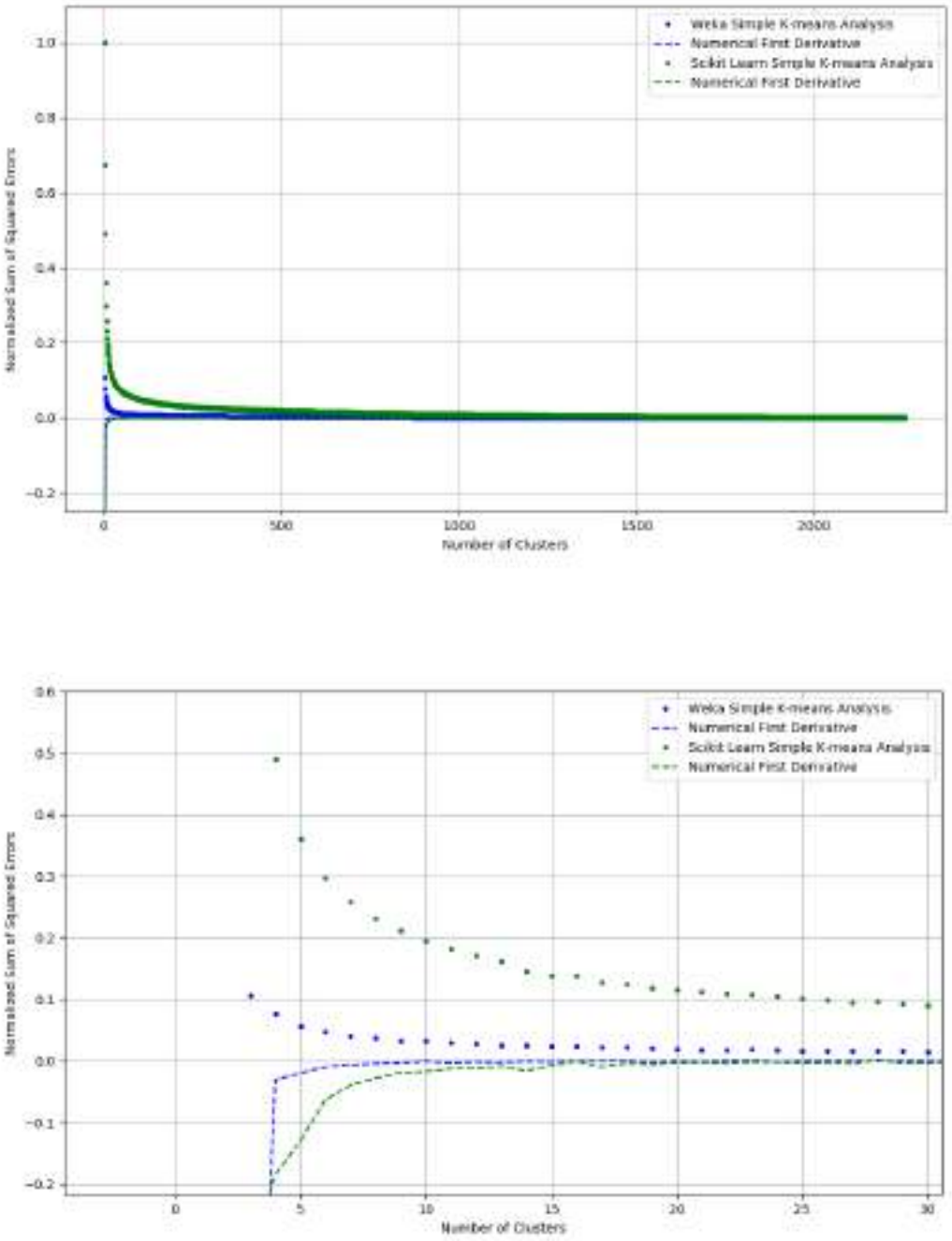
545



547 Figure 1. Core and pangenome from bacteria groups of the study.

Figure 2. Functional annotation of pan-genome sequences. Legend: EC-GI (*Escherichia coli* genomes from gastrointestinal tract), EC-UT (*Escherichia coli* genomes from urinary tract), EF (*Enterococcus faecalis* genomes) and SSAP (*Staphylococcus saprophyticus* genomes).

554



557 Figure 3. Genome clusterization of *Escherichia coli* strains based on sum of squared errors
558 (SSE) by simple K-means method. Euclidean distance was computed from pairwise
559 tetranucleotide correlation index and the SSE was determined for each cluster number. The
560 rate of decay of SSE was obtained from numerical first derivative method (dashed lines).

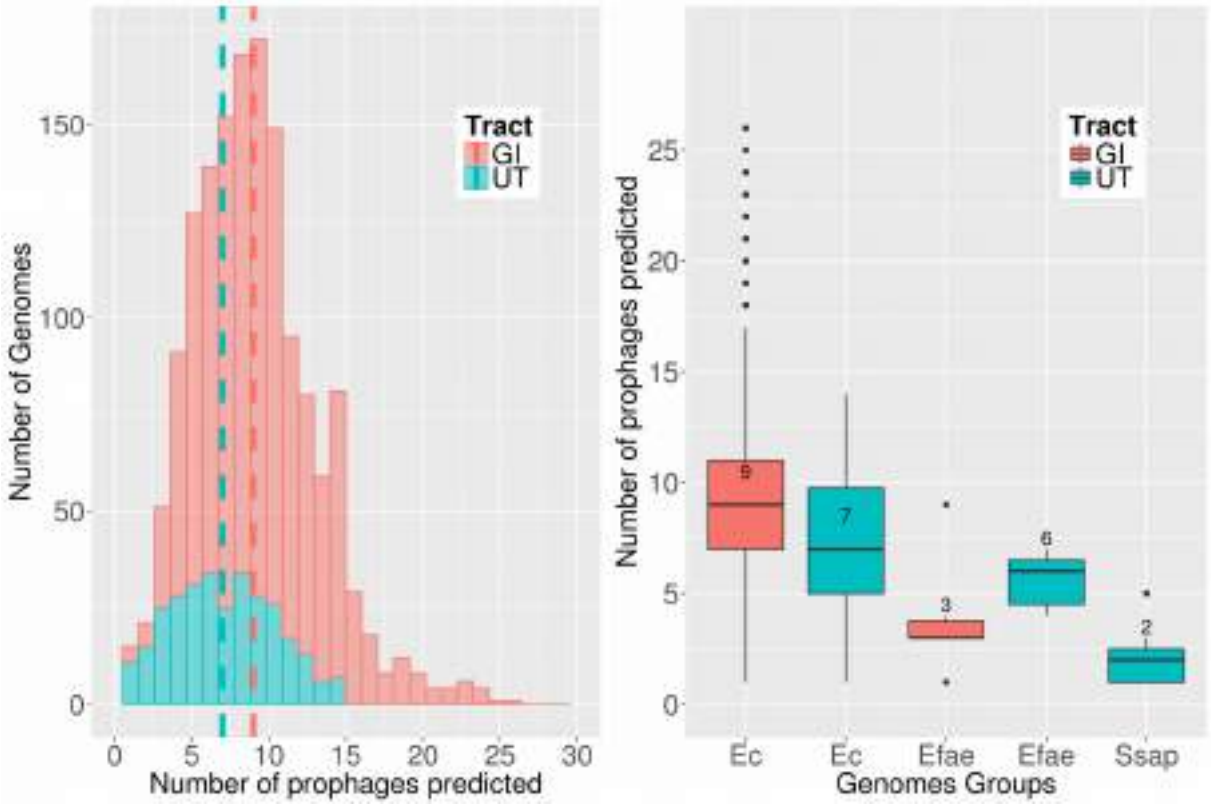
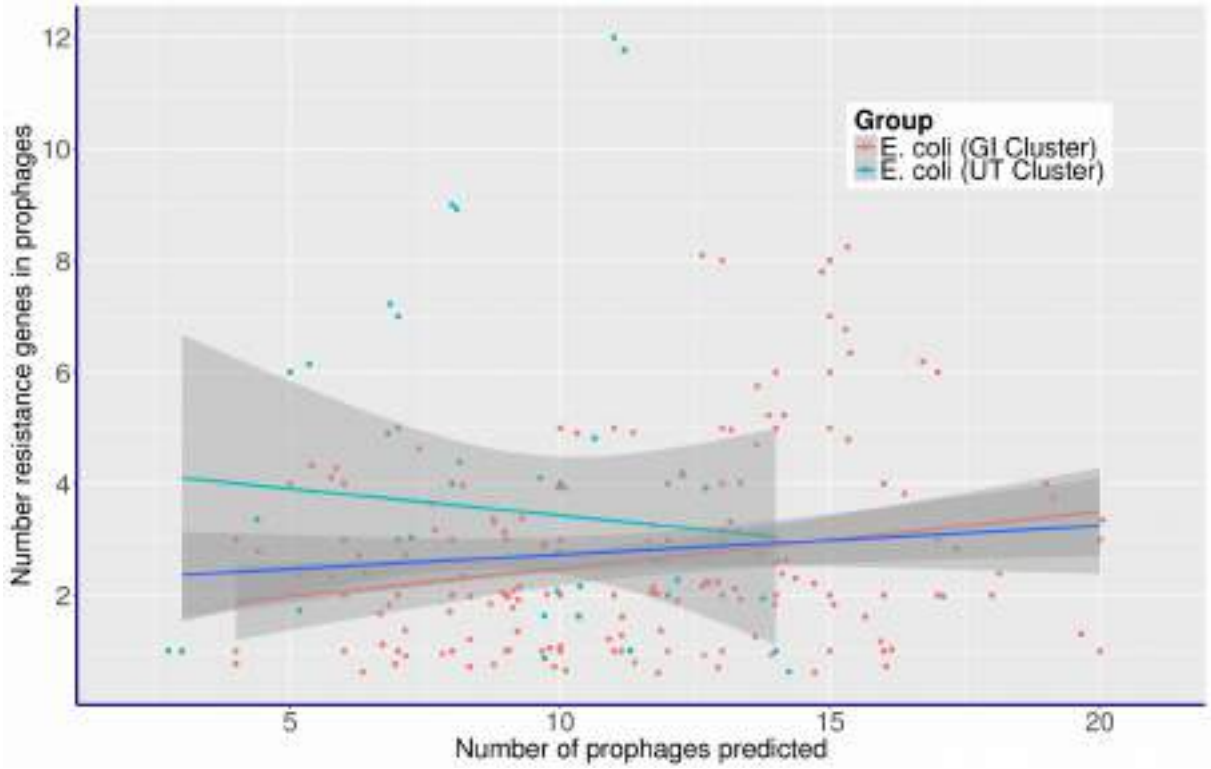
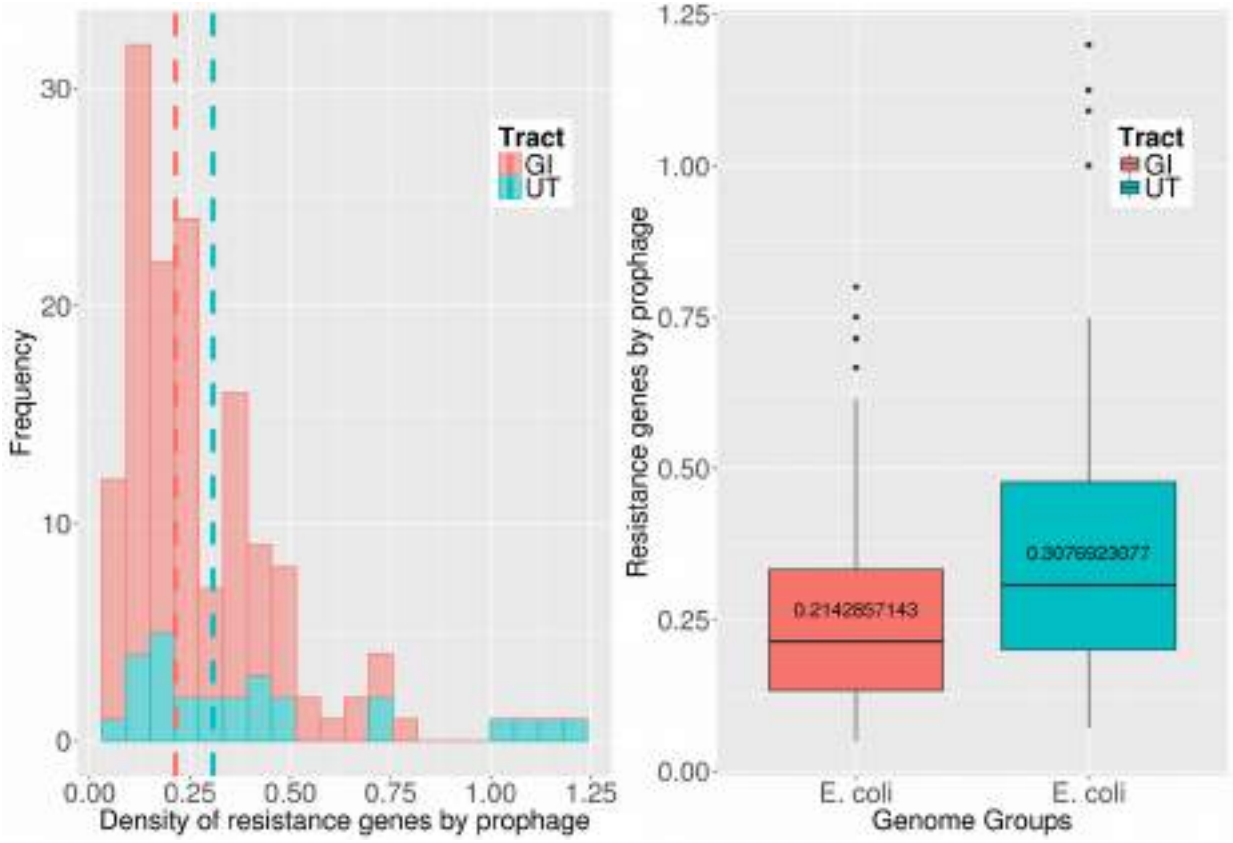


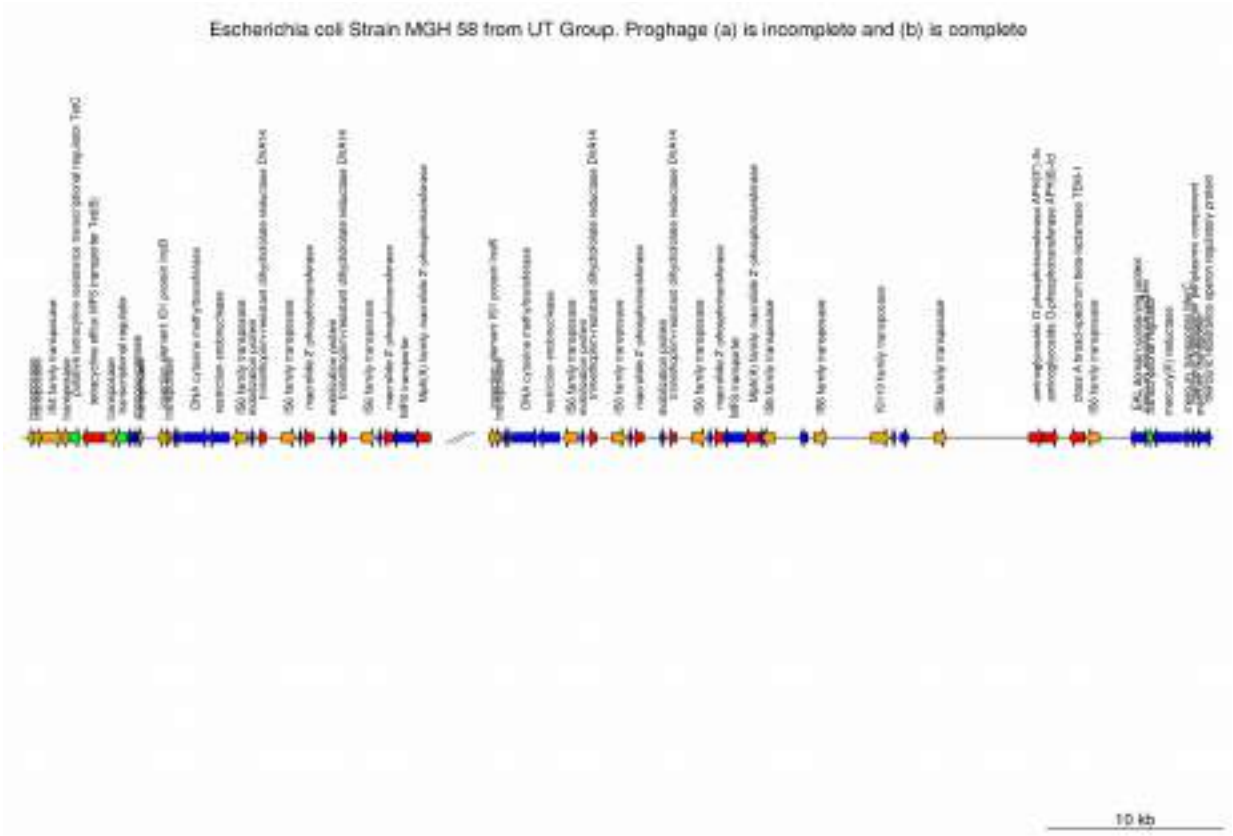
Figure 4. Distribution of prophages predicted *in silico* according to tract and bacterial species of the study. Legend. Lines: median of prophage in Gastroinstestinal isolates (GI) [9, red line] and Urinary Tract (UT) [7, blue line]. Legend: Ec (*Escherichia coli*), Efae (*Enterococcus faecalis*) and Ssap (*Staphylococcus saprophyticus*).



569 Figure 5. Scatter plot and linear regression between number of prophages and resistance genes
570 carried by these integrative elements.
571

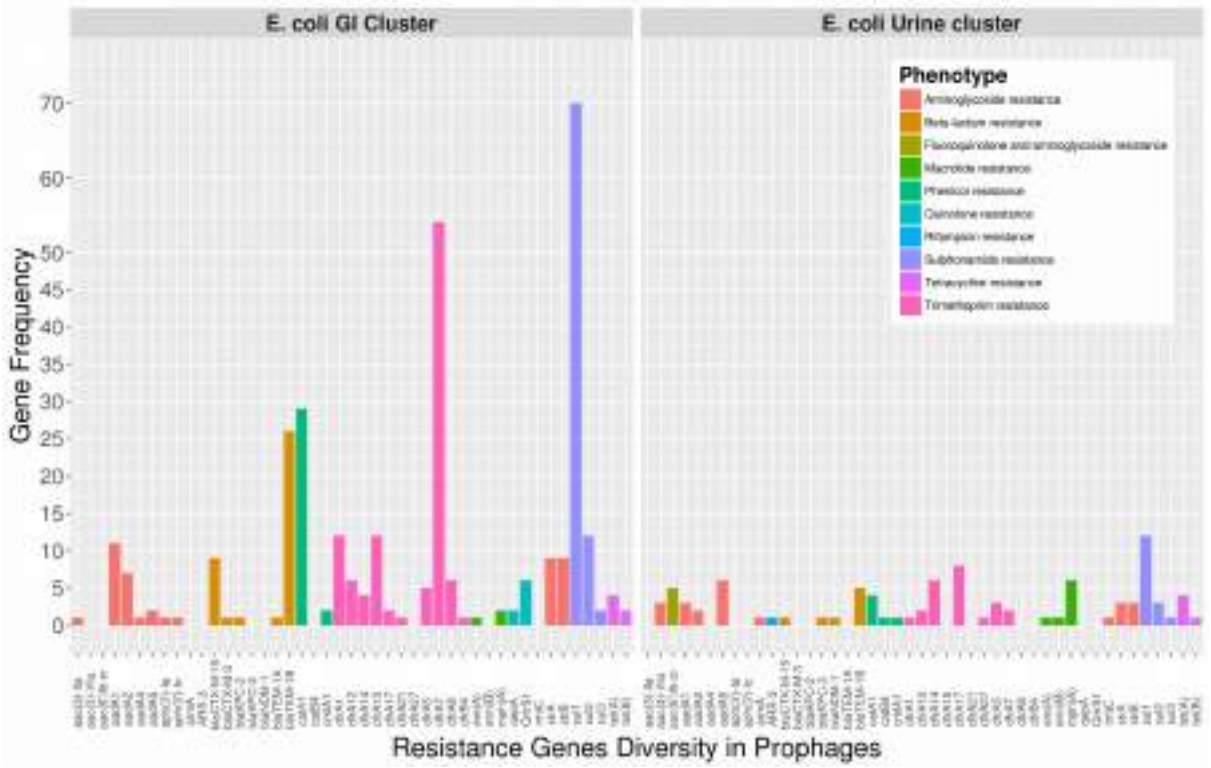


573 Figure 6. Density of resistance genes by prophage. There was significant difference between
574 GI and UT isolates (Wilcoxon rank sum test, p = 0.01746).
575
576



577 Figure 7. Structural representation of two prophage sequences from genome of urinary tract
578 strain *E. coli*. Prophage (a – left) has 23.6 kb and (b – right) has 41.8 kb of sequence length.
579 Color legend: Orange – transposase genes; Green – transcriptional regulator genes; Red –
580 antimicrobial resistance genes (ResG); Blue: genes other than ResG.

581



582 Figure 8. Diversity of resistance genes (n=409) recovered inside of prophage sequences, all in
583 *Escherichia coli* genomes.

584

585

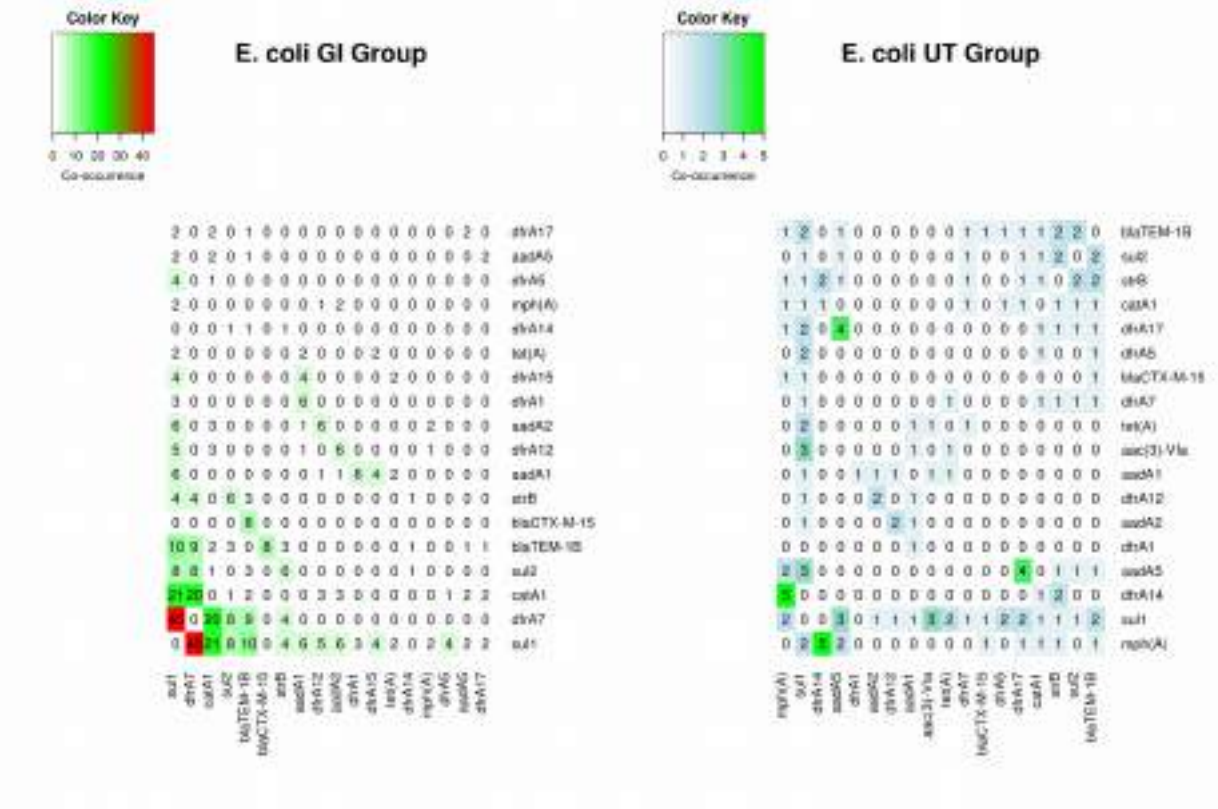


Figure 9. Co-occurrence of resistance genes carried by same prophage sequence.

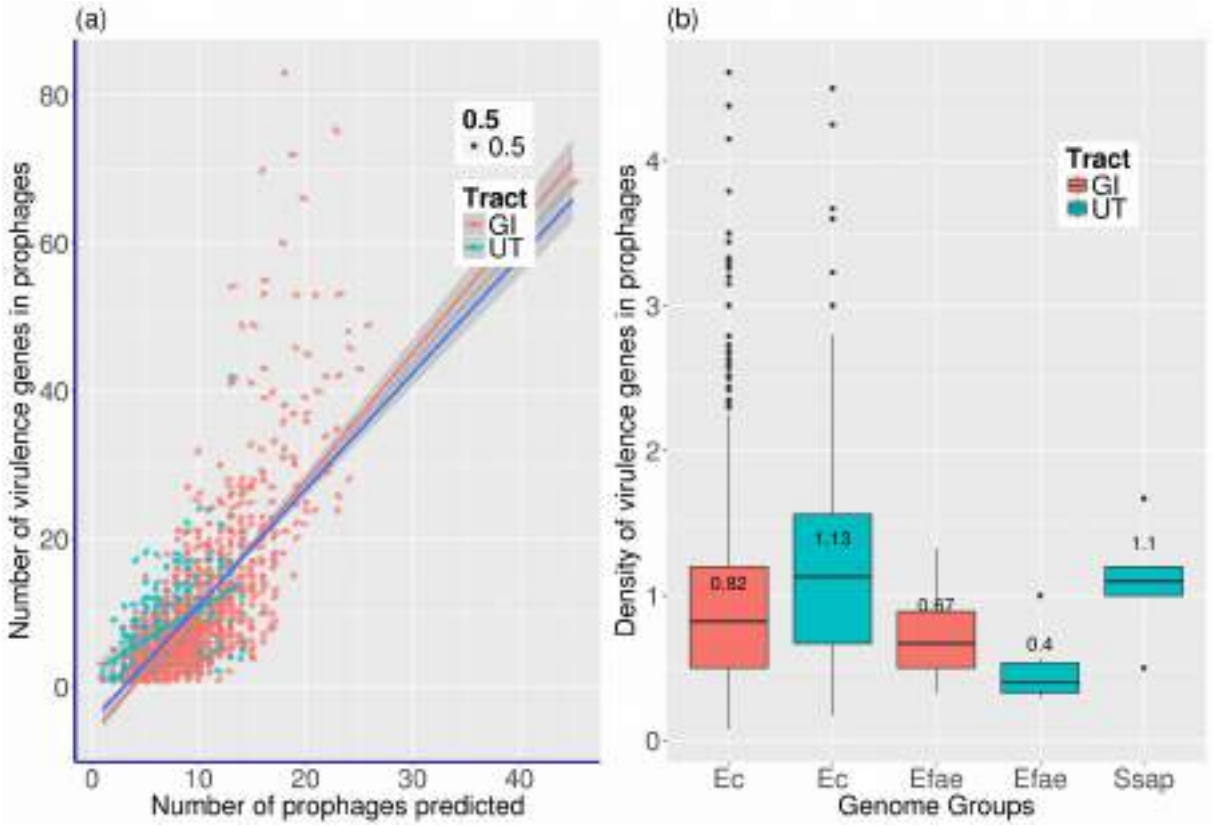


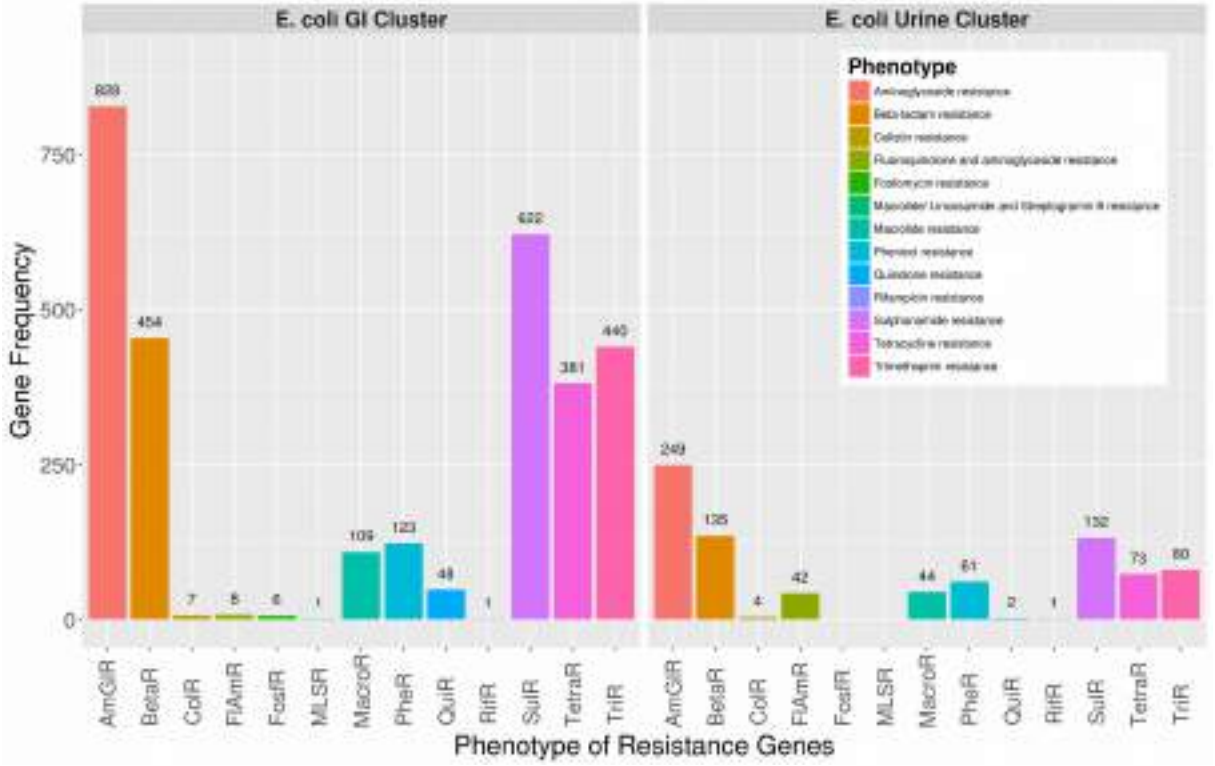
Figure 10. Distribution of virulence factors carried by prophage sequences. Density is the ratio of number of virulence genes to number of prophage for each genome. Legend: GI (Gastrointestinal Tract), UT (Urinary Tract), Ec (*Escherichia coli*), Efae (*Enterococcus faecalis*), Ssap (*Staphylococcus saprophyticus*). Lines: Red line – Pearson's product-moment correlation for GI isolates [r = 0.7269, CI95% (0.6988 to 0.7528), p-value < 0.001]; Light blue line – Pearson's product-moment correlation for UT isolates [r = 0.4867, CI95% (0.3929 to 0.5705), p-value < 0.001]; Dark blue line – Pearson's product-moment correlation for all isolates [r = 0.6983, CI95% (0.6710 to 0.7236), p-value < 0.001].

Table 2. Virulence factor *in silico* predicted and carried by prophage sequences. The most ten frequent genes by genome groups were recovered with respective number of occurrences. The diversity of virulence factors by genome group was: *E. coli* GI = 518; *E. coli* UT = 213; *E. faecalis* GI = 12; *E. faecalis* UT = 4 and *S. saprophyticus* UT = 9.

E. coli (GI Cluster)	(ipaH2.5) invasion plasmid antigen, fragment [Mxi-Spa TTSS effectors controlled by MxiE (CVF465)]	732	6,8%
	(ompD) outer membrane porin precursor [Hek (AI384)]	560	5,2%
	(perC/bfpW) transcriptional regulator BfpW [Per (VF0150)]	475	4,2%
	(sclB) Collagen-like surface protein [Streptococcal collagen-like proteins (CVF116)]	232	2,0%
	(lpg2644) hypothetical protein [Legionella collagen-like protein (Lcl) (AI343)]	176	1,6%
	(flmH) 3-oxoacyl-ACP reductase [Polar flagella (VF0473)]	157	1,4%
	(eps4) exopolysaccharide biosynthesis protein [Capsule (CVF186)]	153	1,3%
. coli (UT Cluster)	(mlr6326) putative DNA invertase [T3SS (SS026)]	313	13,8%
	(rck) resistance to complement killing [Rck (VF0106)]	298	13,1%
	(ipaH) hypothetical prophage protein [Mxi-Spa TTSS effectors controlled by MxiE (CVF465)]	189	8,3%
	(ompD) outer membrane porin precursor [Hek (AI384)]	153	6,7%
	(ipaH2.5) invasion plasmid antigen, fragment [Mxi-Spa TTSS effectors controlled by MxiE (CVF465)]	152	6,7%
	(nleK) hypothetical protein [T3SS (SS022)]	97	4,3%
	(flmH) 3-oxoacyl-ACP reductase [Polar flagella (VF0473)]	65	2,9%
	(mlr6352) transposase [T3SS (SS026)]	64	2,8%
	(perC/bfpW) transcriptional regulator BfpW [Per (VF0150)]	55	2,4%
	(aaiW) transposase [AAI/SCI-II (SS182)]	44	1,9%
faecalis (GI Cluster)	(feoB) ferrous iron transporter B [FeoAB (VF0160)]	3	17,6%
	(mgtC) hypothetical protein [Mg2+ transport (CVF005)]	3	17,6%
	(mlr6359) transposase [T3SS (SS026)]	2	11,8%
	(lpg2644) hypothetical protein [Legionella collagen-like protein (Lcl) (AI343)]	1	5,9%
	(iaprotehA) invasion associated secreted endopeptidase [Cell wall hydrolase (CVF234)]	1	5,9%
	(bsh) conjugated bile acid hydrolase [Bile-salt hydrolase (CVF250)]	1	5,9%
	(fbpC) iron-uptake permease ATP-binding protein [ABC transporter (CVF197)]	1	5,9%
	(mrpJ) fimbrial operon regulator [Proteus-like (MR/P) fimbriae, mannose resistant (biofilm formation) (AI072	1	5,9%
	(nuc) staphylococcal thermonuclease precursor [Thermonuclease (CVF093)]	1	5,9%
	(sclA) Collagen-like surface protein [Streptococcal collagen-like proteins (CVF116)]	1	5,9%
aecalis (UT Cluster)	(feoB) ferrous iron transporter B [FeoAB (VF0160)]	8	44,4%
	(mgtC) hypothetical protein [Mg2+ transport (CVF005)]	8	44,4%
	(flmE) tyrosine recombinase [type 1 fimbriae (AI075)]	1	5,6%
	(mlr6359) transposase [T3SS (SS026)]	1	5,6%
S. saprophyticus	(msrA/B/pilB)) trifunctional thioredoxin/methionine sulfoxide reductase A/B protein [(CVF762)]	6	30,0%
	(lisR) two-component response regulator [LisR/LisK (CVF253)]	3	15,0%
	(MGA_1142) organic hydroperoxide resistance protein [MG1142 (AI361)]	3	15,0%
	(clpP) ATP-dependent Clp protease proteolytic subunit [ClpP (VF0074)]	2	10,0%
	(hlyB) alpha-hemolysin translocation ATP-binding protein HlyB [Alpha-hemolysin (CVF453)]	2	10,0%

609

610



611 Figure 11. Frequency of resistance genes predicted in *Escherichia coli* genomes. A total of
612 3,851 gene sequences were recovered by *in silico* search (GI – n=3,028; UT – n=823).

613

614

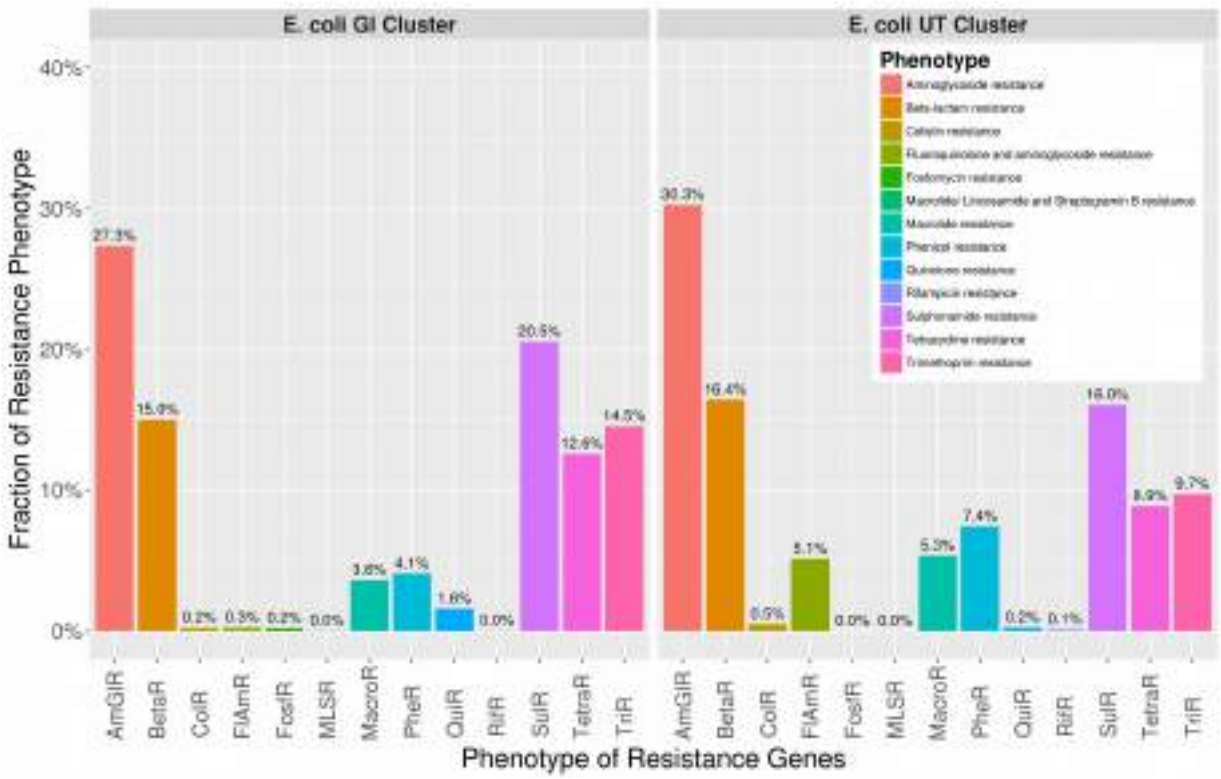
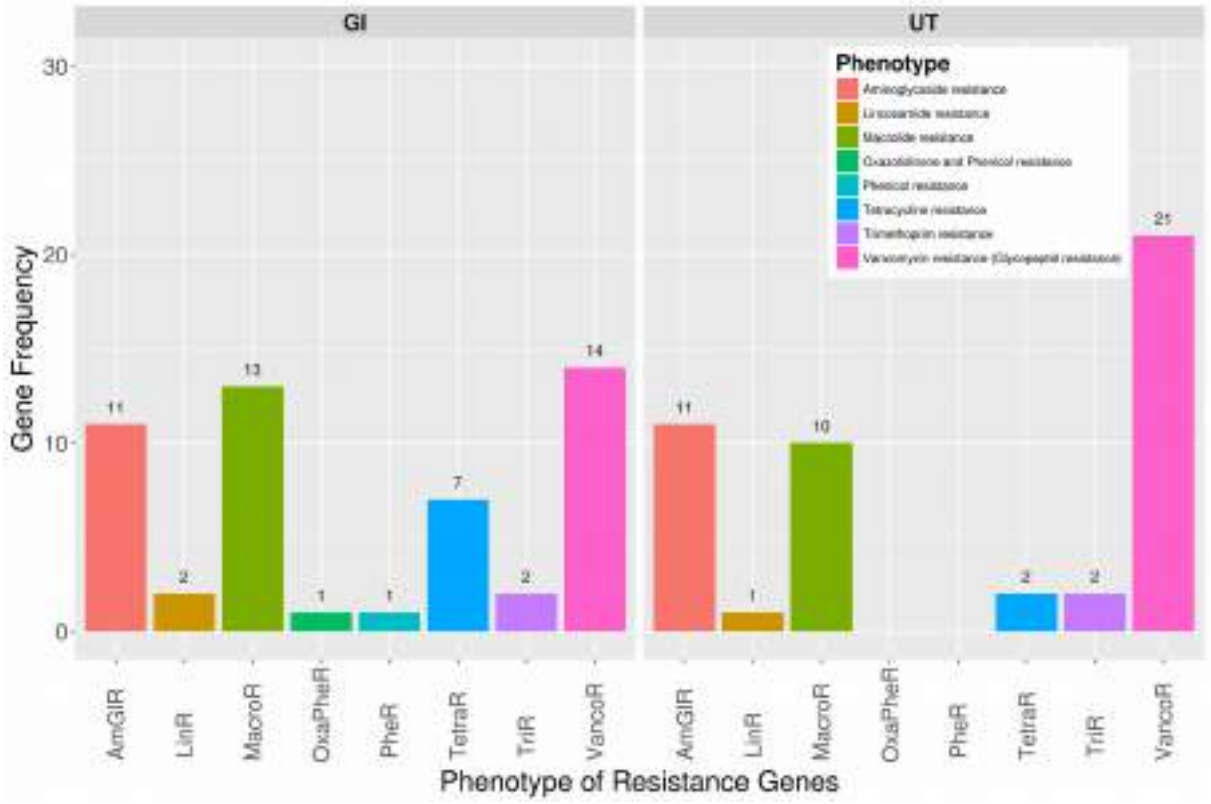
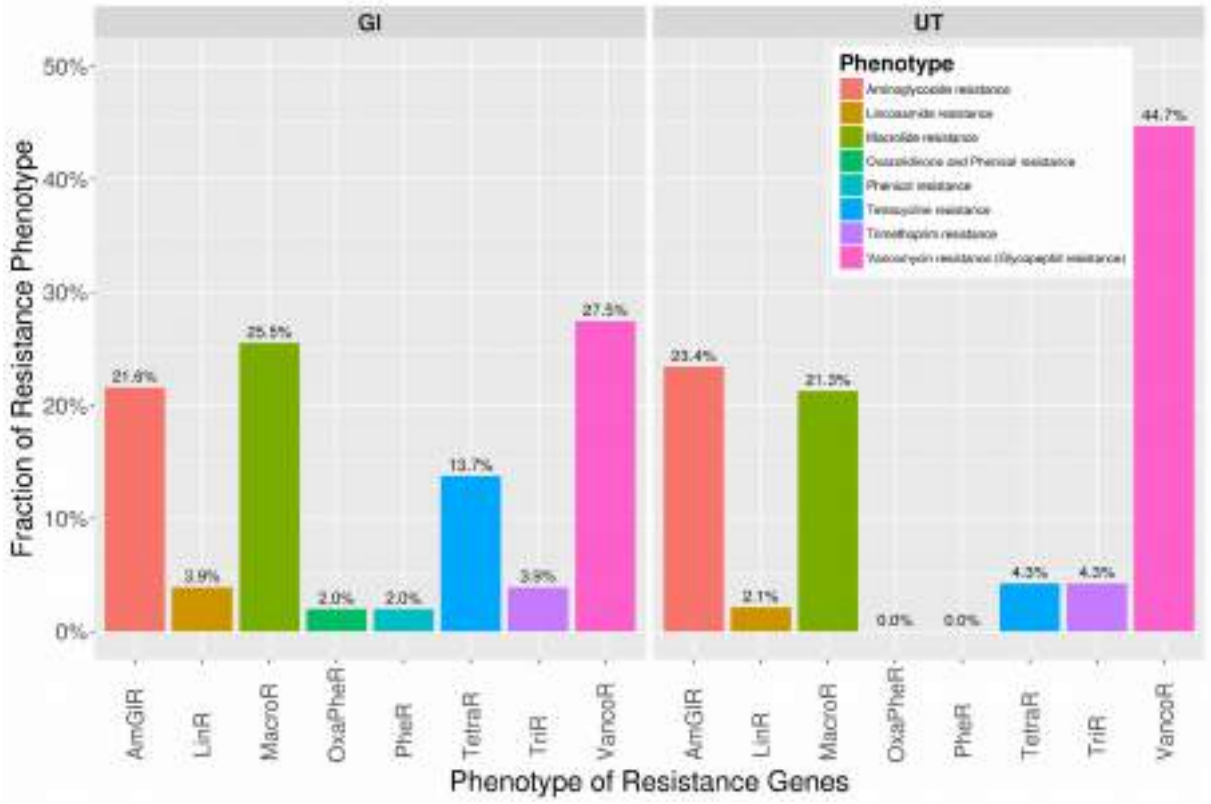


Figure 12. Prevalence of genes predicted in *Escherichia coli* genomes by resistance phenotype. A total of 3,851 gene sequences were recovered by *in silico* search (GI – n=3,028; UT – n=823). There was significant association between class of resistance genes between groups by Pearson's Chi-squared test ($p < 0.001$), which urinary tract isolates had superior ratio of resistance genes of Fluoroquinolones and Aminoglycoside Resistance (FlAmR), Macrolide Resistance (MacroR) and Phenicol Resistance (PheR); and low ratio of Quinolone Resistance (QuiR), Sulphonamide Resistance (SulR), Tetracycline Resistance (TetraR) and Trimethoprim Resistance (TriR).



627 Figure 13. Frequency of resistance genes predicted in *Enterococcus faecalis* genomes. A total
628 of 230 gene sequences were recovered by *in silico* search (GI – n=51; UT – n=47; Others sites
629 – n=132). There was not significant association between among classes of resistance genes
630 and isolation source (Pearson's Chi-squared test, p = 0.4553).
631
632
633



635 Figure 14. Prevalence of genes predicted in *Enterococcus faecalis* genomes by resistance
636 phenotype. A total of 230 gene sequences were recovered by *in silico* search (GI – n=51; UT –
637 n=47; Others sites – n=132).
638
639

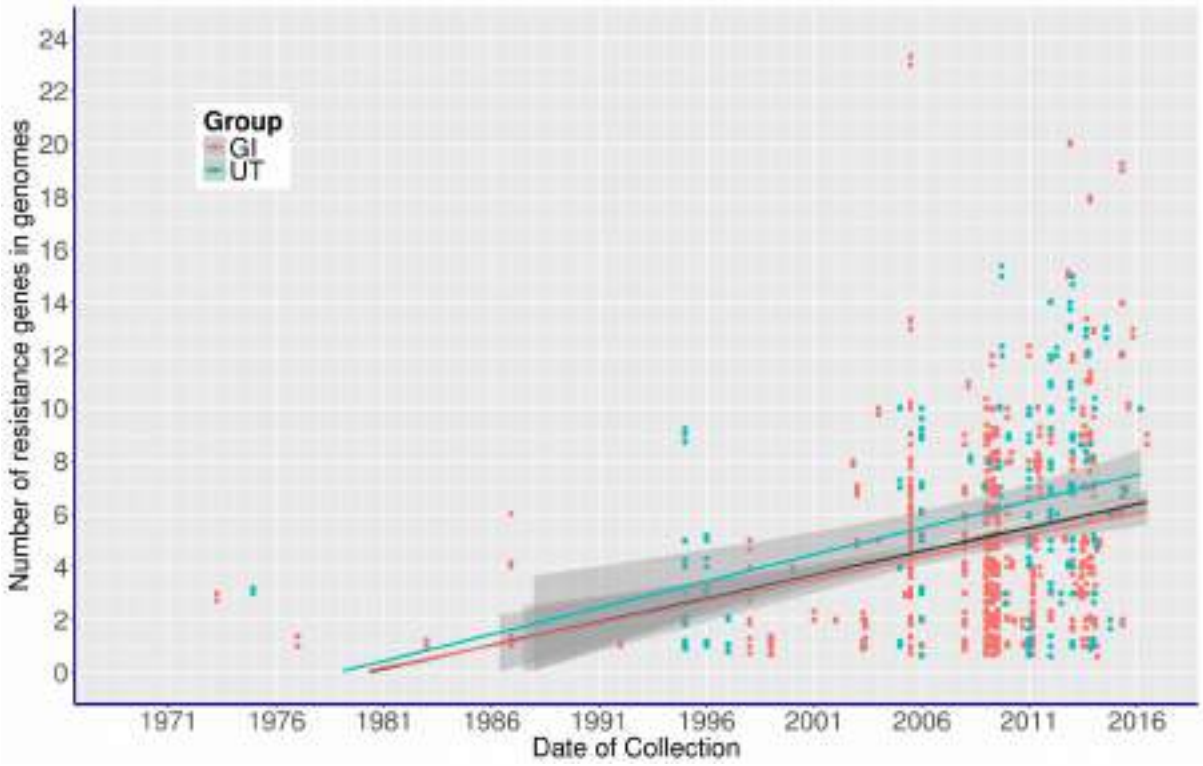
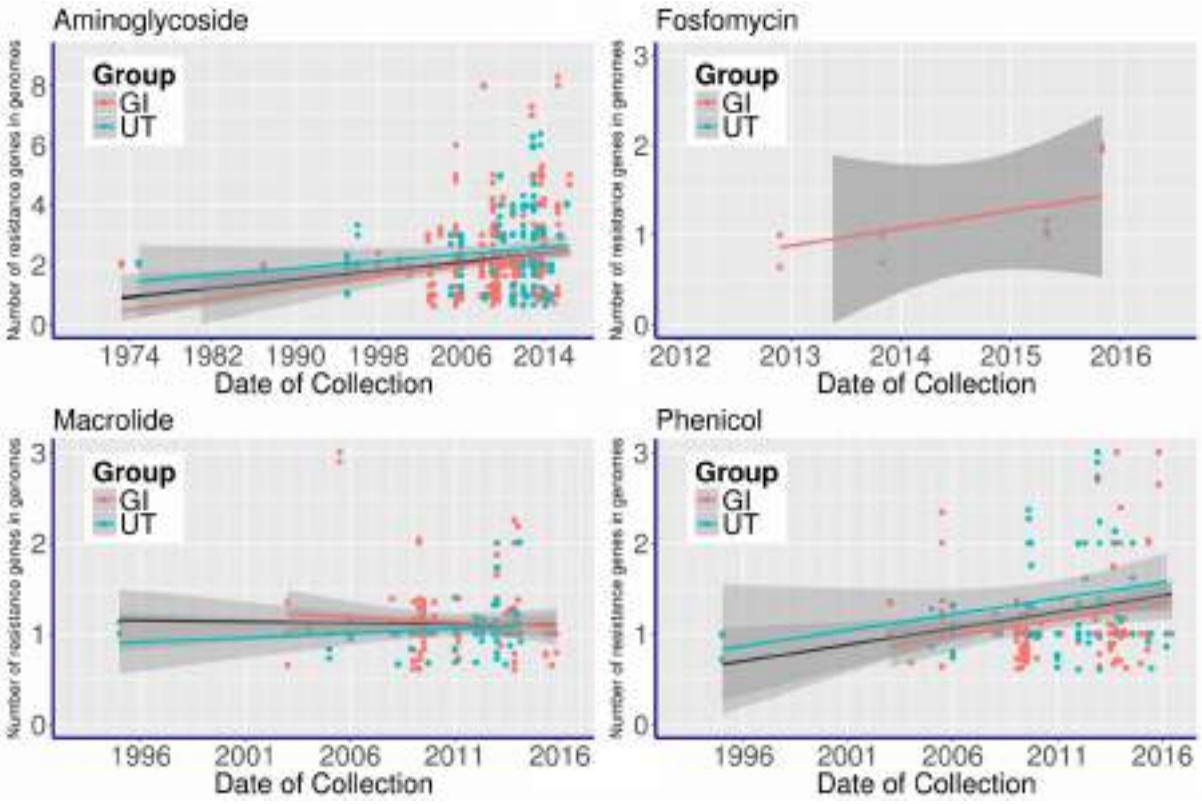
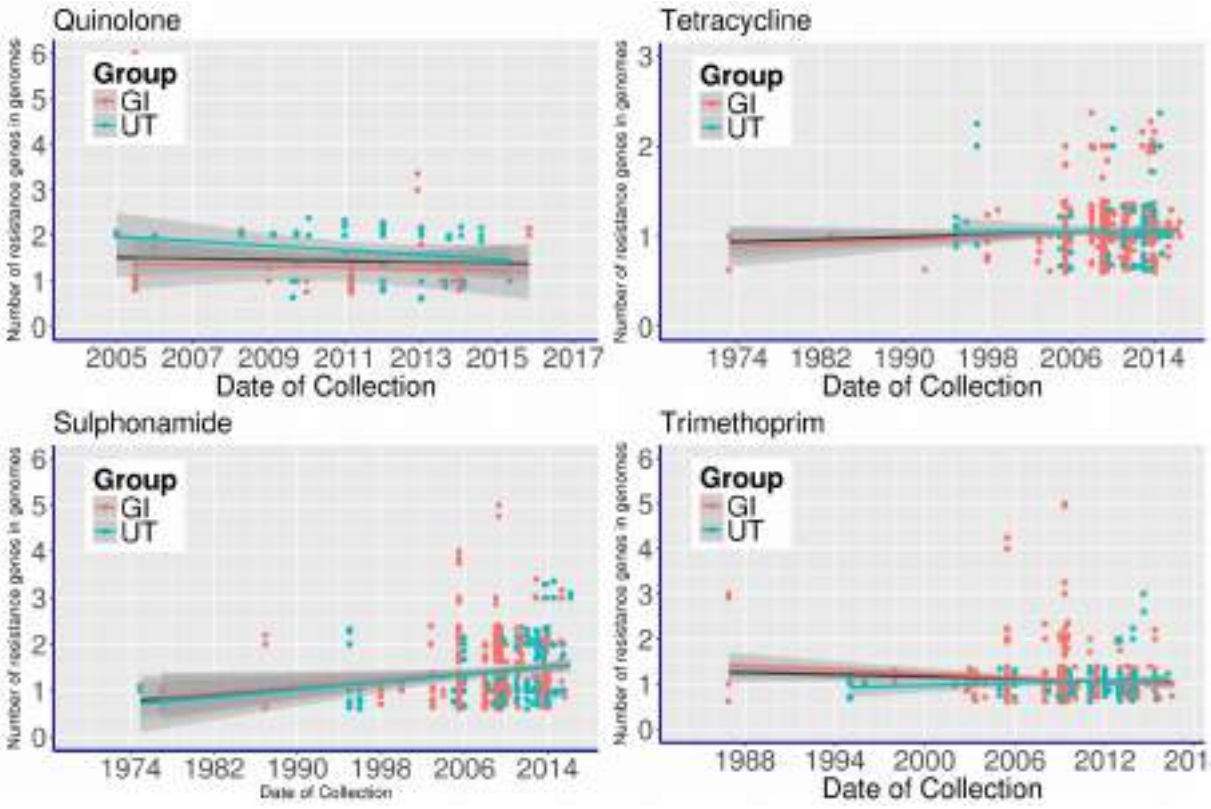


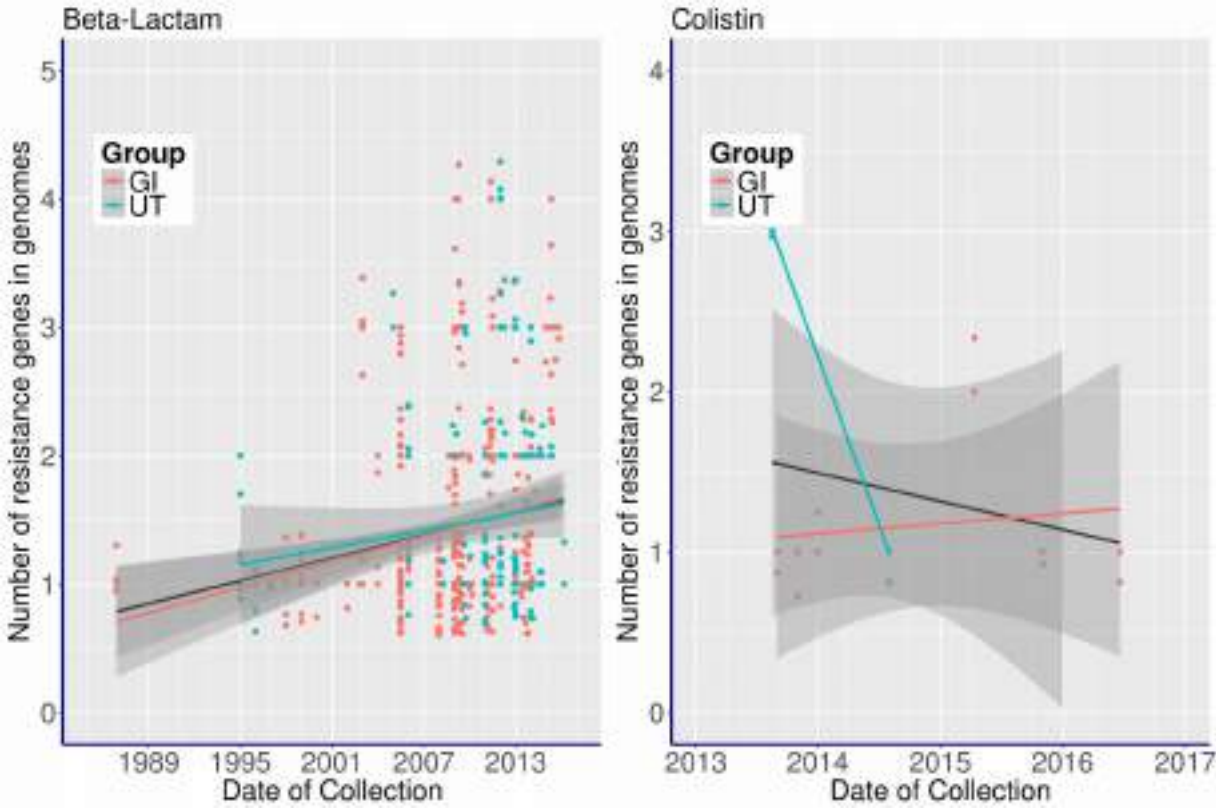
Figure 15: Scatter plot between number of resistant genes predicted in genomes of *Escherichia coli* and date of collection of isolates. Red points: Gastrointestinal Tract (GI); Blue points: Urinary Tract (UT); Lines: Linear regression for both groups (GI + UT) (black); GI (red) and UT (blue). Statistics: GI + UT – r = 0.2835, p-value < 0.001, CI95% (0.2097 to 0.3541); GI – r = 0.1140, p-value < 0.001, CI95% (0.0496 to 0.1774); UT – r = 0.4688, p-value < 0.001, CI95% (0.3701 to 0.5570).



649 Figure 16. Scatter plots of number of resistance genes by class (aminoglycoside, fosfomycin,
650 macrolide and phenicol) and temporal isolation of *E. coli* strains. Red points: Gastrointestinal
651 Tract (GI); Blue points: Urinary Tract (UT); Lines: Linear regression for both groups (GI +
652 UT) (black); GI (red) and UT (blue).
653



655 Figure 17: Scatter plots of number of resistance genes by class (quinolone, tetracycline,
656 sulphonamide and trimethoprim) and temporal isolation of *E. coli* strains. Red points:
657 Gastrointestinal Tract (GI); Blue points: Urinary Tract (UT); Lines: Linear regression for both
658 groups (GI + UT) (black); GI (red) and UT (blue). Statistics: (r = -0.1193, p-value = 0.0352,
659 CI95% (-0.2273 to -0.0084).



661 Figure 18: Scatter plots of number of resistance genes by class (beta-lactam and colistin) and
662 temporal isolation of *E. coli* strains. Red points: Gastrointestinal Tract (GI); Blue points:
663 Urinary Tract (UT); Lines: Linear regression for both groups (GI + UT) (black); GI (red) and
664 UT (blue).
665
666

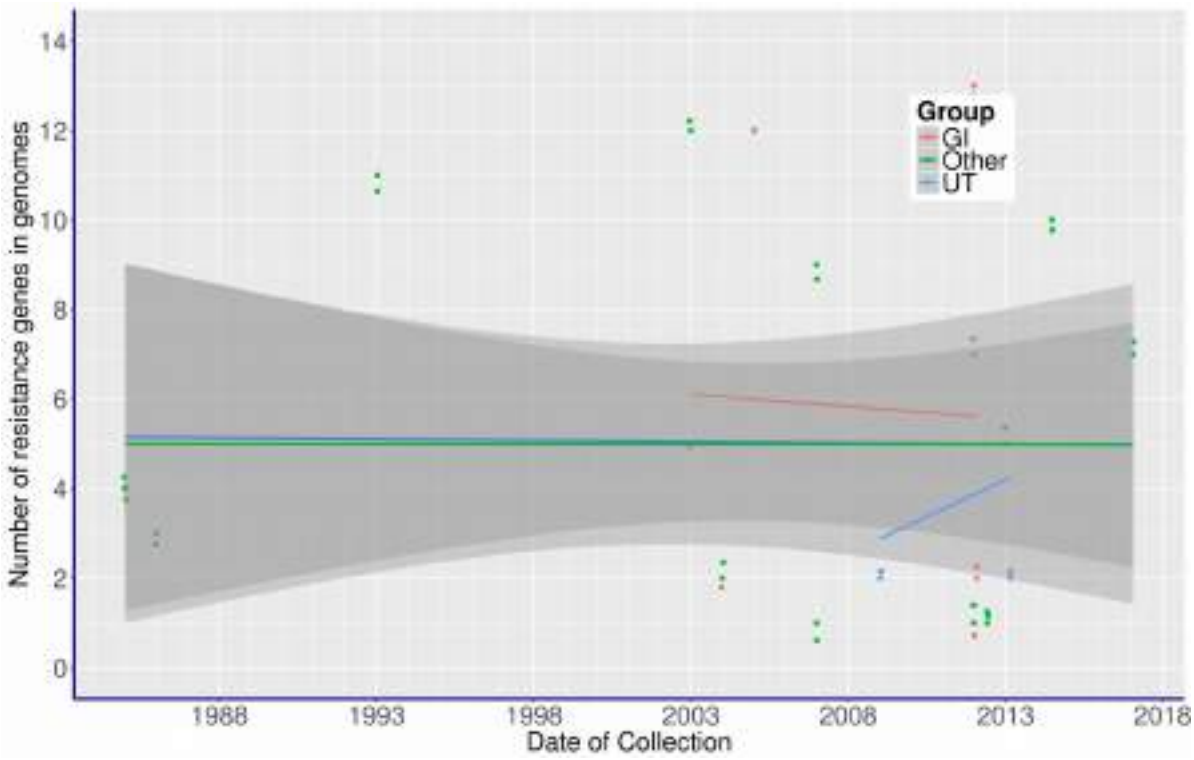


Figure 19: Scatter plot between number of resistant genes predicted in genomes of *Enterococcus faecalis* and date of collection of isolates. Red points: Gastrointestinal Tract (GI); Light Blue points: Urinary Tract (UT); Green points: Other sites (Other), Lines: Linear regression for both groups (GI + UT + Other) (dark blue); GI (red); UT (light blue) and Other (green).

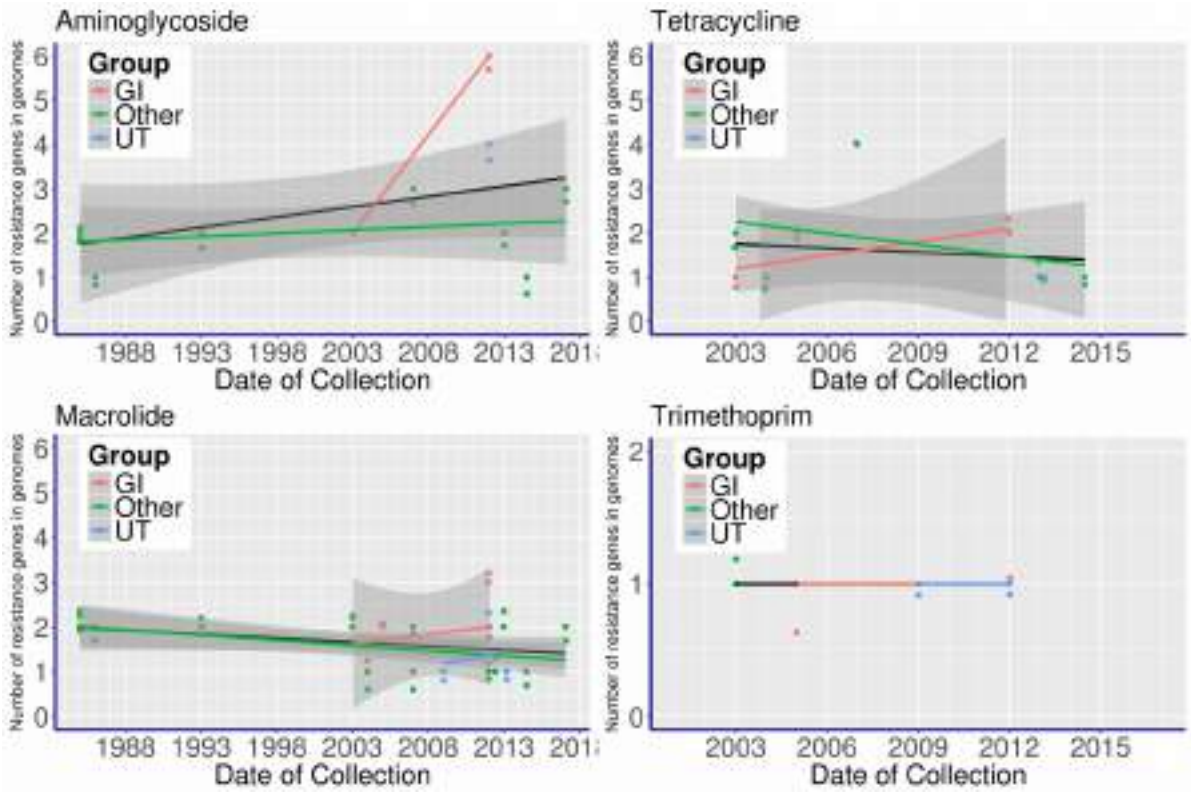


Figure 20: Scatter plots of number of resistance genes by class (aminoglycoside, tetracycline, macrolide and trimethoprim) and temporal isolation of *E. faecalis* strains. Red points: Gastrointestinal Tract (GI); Light Blue points: Urinary Tract (UT); Green points: Other sites (Other), Lines: Linear regression for both groups (GI + UT + Other) (black); GI (red); UT (light blue) and Other (green).

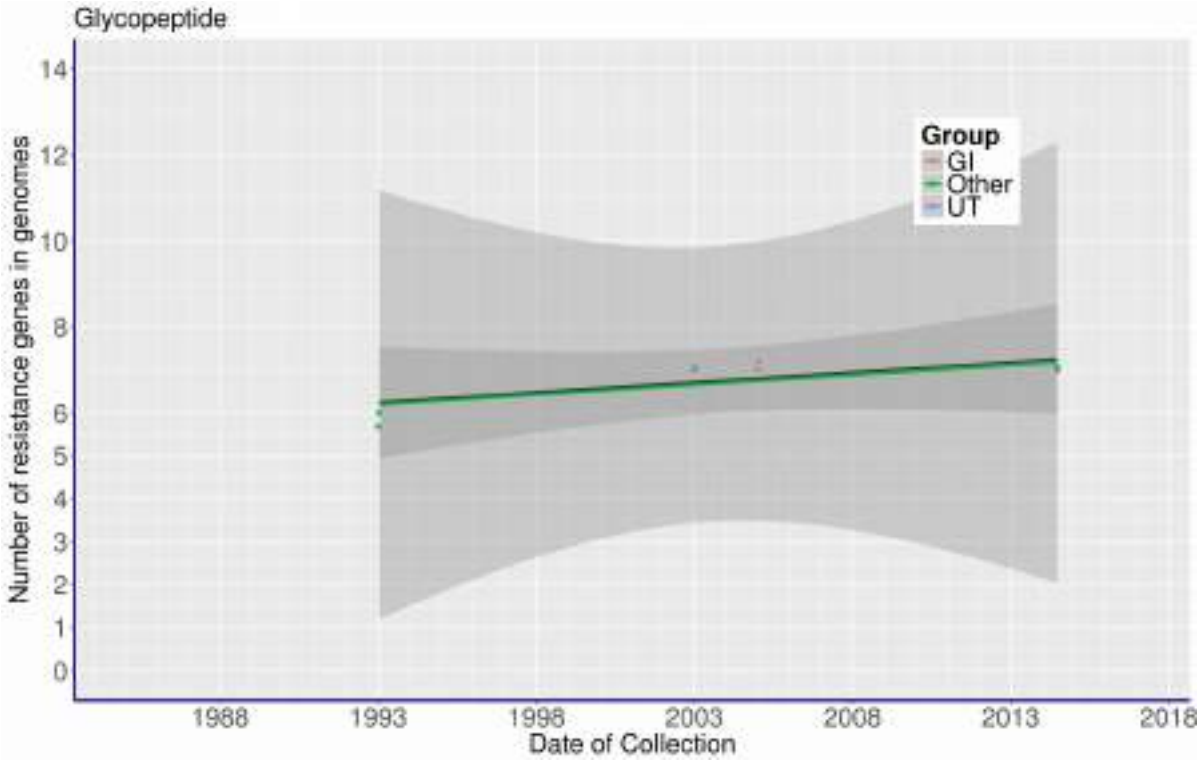


Figure 21: Scatter plot of number of resistance genes belonging to glycopeptide and temporal isolation of *E. faecalis* strains. Red points: Gastrointestinal Tract (GI); Light Blue points: Urinary Tract (UT); Green points: Other sites (Other), Lines: Linear regression for both groups (GI + UT + Other) (black); GI (red); UT (light blue) and Other (green).

Figure 22. Diversity of homologous protein sequence of pilin (type I fimbrae) recovered from functional annotation of pangenome of UT and GI *E. coli* isolates. Multiple sequence alignment was performed by ClustalW software [69] and neighbor-joining method using BLOSUM62 score was used to build the tree in MEGA v.6 software [70]. The tree branches shows at least one homologous pilin sequence belonging to GI pangenome, in exception red branch.

Table 3: Unique COG proteins descriptions recovered from urinary tract genomes of *Enterococcus faecalis.*

nylacetate-coenzyme A ligase PaaK/ adenylate-forming ain family	Coenzyme transport and metabolism
riction endonuclease S subunit	Defense mechanisms
comycin resistance protein YoaR (function unknown)/ ains peptidoglycan-binding and VanW domains	Defense mechanisms
haracterized conserved protein YukE	Function unknown
haracterized membrane protein YukC	Function unknown
haracterized membrane protein/ UPF0182 family	Function unknown
C-type ATPase fused to a predicted acetyltransferase domain	General function prediction only
icted double-glycine peptidase	General function prediction only
icted phosphoadenosine phosphosulfate sulfurtransferase/ ains C-terminal DUF3440 domain	General function prediction only
cal SAM superfamily enzyme/ MoaA/NifB/PqqE/SkfB ly	General function prediction only
C-type molybdenum transport system/ ATPase ponent/photorepair protein PhrA	Inorganic ion transport and metabolism
h-affinity K+ transport system/ ATPase chain B	Inorganic ion transport and metabolism
transporting ATPase/ A chain	Inorganic ion transport and metabolism
transporting ATPase/ c chain	Inorganic ion transport and metabolism
eriophage DNA transposition protein/ AAA+ family ATPase	Mobilome: prophages/ transposons
ge- or plasmid-associated DNA primase	Mobilome: prophages/ transposons
element-mobilizing transposase RayT	Mobilome: prophages/ transposons
on-type reverse transcriptase	Mobilome: prophages/ transposons
dependent protease with chaperone function	Posttranslational modification/ protein turnover/ chaperones
A gyrase inhibitor GyrI	Replication/ recombination and repair
A mismatch repair protein MutH	Replication/ recombination and repair
A polymerase elongation subunit (family B)	Replication/ recombination and repair
mismatch repair DNA endonuclease/ very short patch repair ein	Replication/ recombination and repair
iday junction resolvase/ archaeal type	Replication/ recombination and repair
cin large subunit	Secondary metabolites biosynthesis/ transport and catabolism
nylate cyclase/ class 3	Signal transduction mechanisms

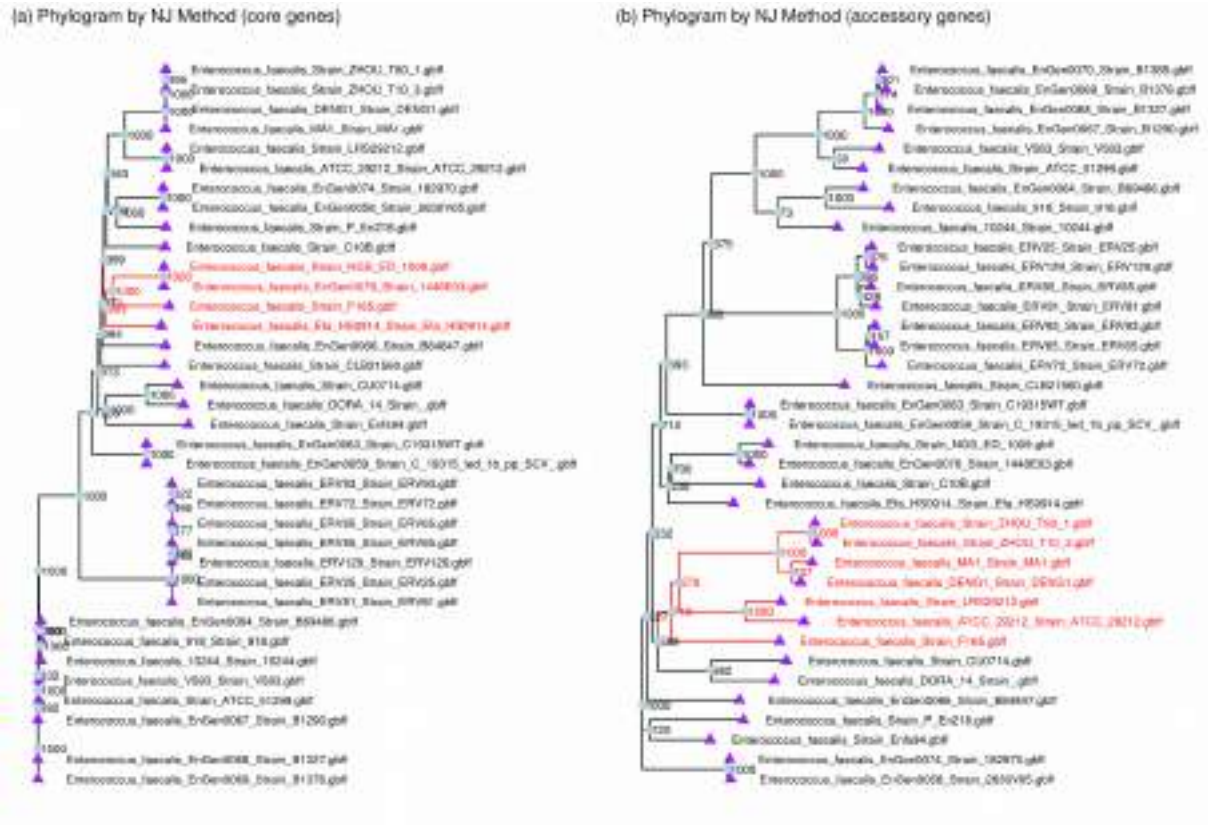
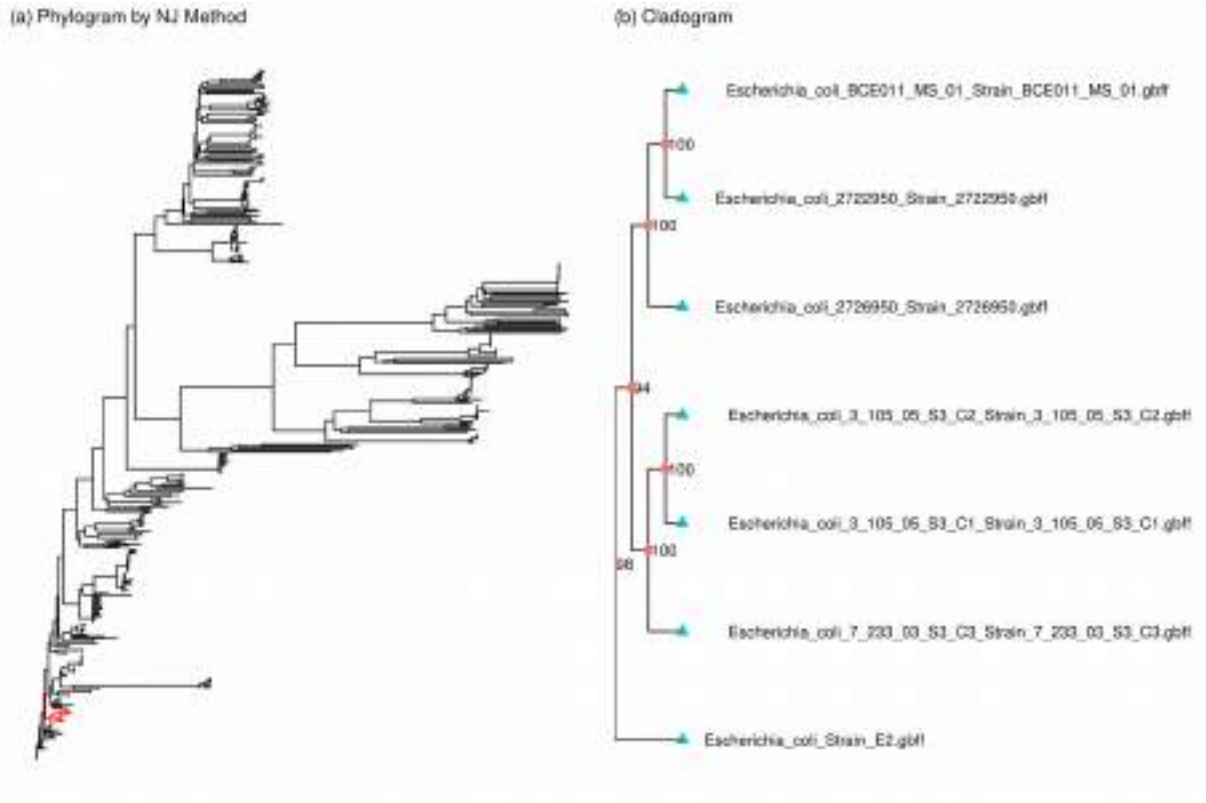
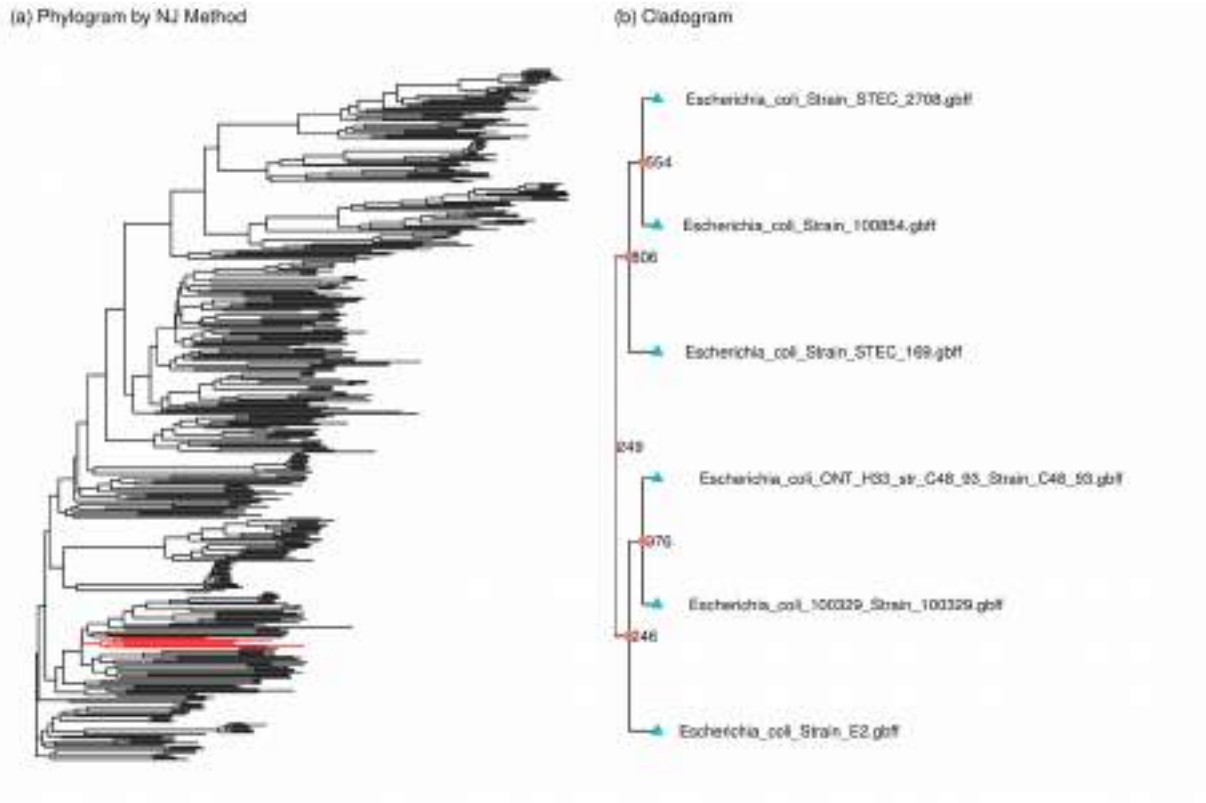


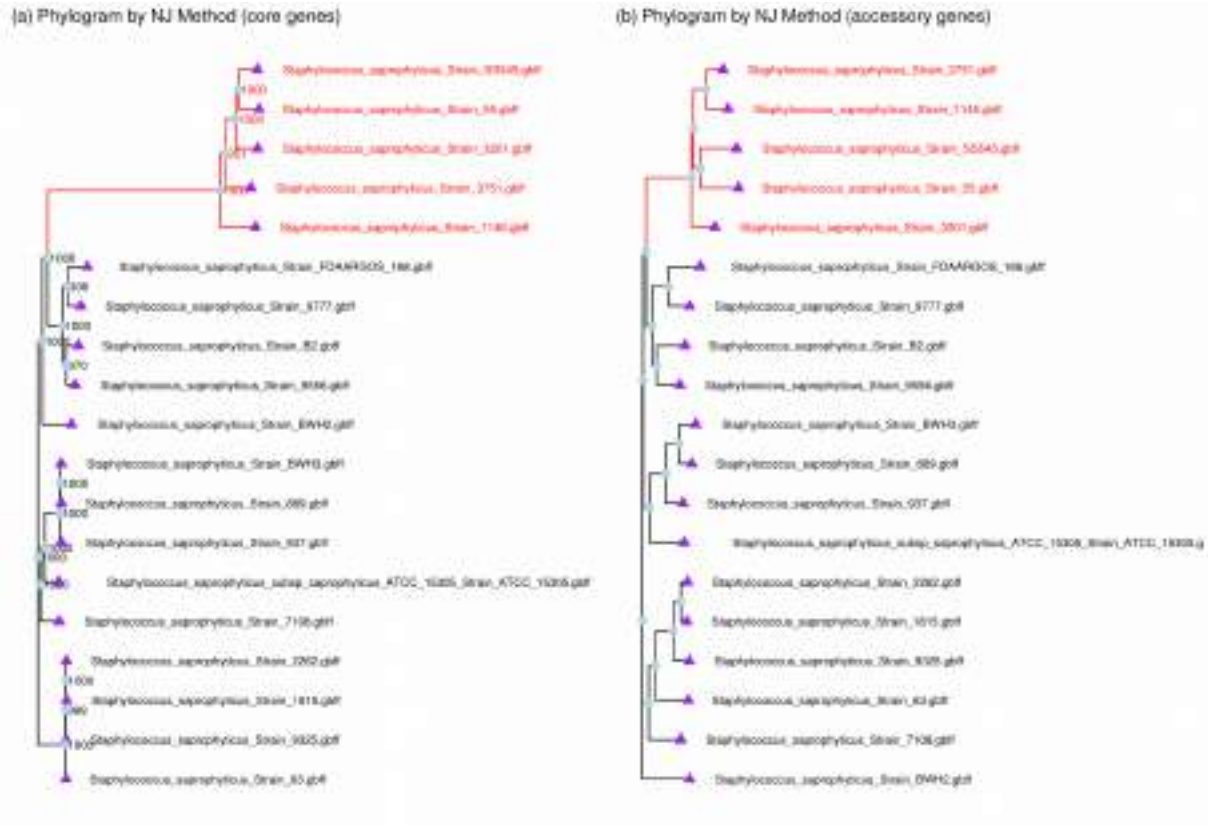
Figure 23. Phylogenetic tree of 36 strains of *Enterococcus faecalis* isolated from diverse source. Red names belongs the same clade of *E. faecalis* F165. (a) Tree based on neighbor-joining method of core genes alignment – Strains: NGS ED 1009 (blood); EnGen0076 (feaces); HS0914 (urine). (b) Tree based on neighbor-joining method of gene-content matrix – Strains: ZHOU T60 and T10 (wound exudate); MA1 (blood); DENG1 (sputum); LRS29212 and ATCC 29212 (urine). Node number: Bootstrap of 1000 replicates.



714 Figure 24. Phylogenetic tree of 382 strains of *Escherichia coli* E2 cluster isolated from
715 diverse source. Red names belongs the same clade of E2. (a) Tree based on neighbor-joining
716 method of core genes alignment. (b) Strains: BCE011 MS 01, 2722950, 2726950, S3 C2, S3
717 C1, S3 C3 (all from stool samples). Node number: Bootstrap of 100 replicates.
718



720 Figure 25. Phylogenetic tree of 382 strains of *Escherichia coli* E2 cluster isolated from
721 diverse source. Red names belongs the same clade of E2. (a) Tree based on neighbor-joining
722 method of gene-content matrix. (b) Strains: STEC 2708, 100854, STEC 169, C48 93, 100329
723 (all from stool samples). Node number: Bootstrap of 1000 replicates.
724



726 Figure 26. Phylogenetic tree of *Staphylococcus saprophyticus* strains isolated from urinary
727 tract. Red names belongs the same clade of *S. saprophyticus* SS545. (a) Tree based on
728 neighbor-joining method of core genes alignment. (b) Tree based on neighbor-joining method
729 of gene-content matrix. Node number: Bootstrap of 1000 replicates.
730

731 REFERENCES

732

733 1. Flores-Mireles AL, Walker JN, Caparon M, Hultgren SJ. Urinary tract infections:
734 epidemiology, mechanisms of infection and treatment options. Nat Rev Microbiol. 2015
735 May;13(5):269-84.

736

737 2. Ciani O, Grassi D, Tarricone R. An economic perspective on urinary tract infection: the
738 "costs of resignation". Clin Drug Investig. 2013 Apr;33(4):255-61.

739

740 3. François M, Hanslik T, Dervaux B, Le Strat Y, Souty C, Vaux S, et al. The economic
741 burden of urinary tract infections in women visiting general practices in France: a cross-
742 sectional survey. BMC Health Serv Res. 2016 Aug 9;16(a):365.

743

744 4. Bien J, Sokolova O, Bozko P. Role of Uropathogenic *Escherichia coli* Virulence Factors in
745 Development of Urinary Tract Infection and Kidney Damage. Int J Nephrol.
746 012;2012:681473.

747

748 5. Kline KA, Lewis AL. Gram-Positive Uropathogens, Polymicrobial Urinary Tract Infection,
749 and the Emerging Microbiota of the Urinary Tract. Microbiol Spectr. 2016 Apr;4(2).

750

751 6. Jandhyala SM, Talukdar R, Subramanyam C, Vuyyuru H, Sasikala M, Reddy DN. Role of
752 the normal gut microbiota. World J Gastroenterol. 2015 Aug 7;21(29):8787-803.

753

754 7. Becker K, Heilmann C, Peters G. Coagulase-Negative Staphylococci. Clin Microbiol Rev.
755 2014 Oct;27(4):870-926.

756

757 8. Gilmore MS, Lebreton F, van Schaik W. Genomic Transition of Enterococci from Gut
758 Commensals to Leading Causes of Multidrug-resistant Hospital Infection in the Antibiotic
759 Era. Curr Opin Microbiol. 2013 Feb;16(1):10-6.

760

9. Paim TG, Pieta L, Prichula J, Sambrano GE, Soares R, Caierão J, et al. Draft Genome Sequence of Brazilian Escherichia coli Uropathogenic Strain E2. Genome Announc. 2016 Oct 6;4(5). pii: e01085-16.

10. Paim TG, Pieta L, Prichula J, Sambrano GE, Soares R, Bello AD, et al. Draft Genome Sequence of Enterococcus faecalis Strain F165 Isolated from a Urinary Tract Infection. Genome Announc. 2016 Oct 6;4(5). pii: e01084-16.

11. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. J Comput Biol. 2012 May;19(5):455-77.

12. Seemann T. Prokka: rapid prokaryotic genome annotation. Bioinformatics. 2014 Jul 15;30(14):2068-9.

13. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12:2825–2830.

14. Frank E, Hall M, Trigg L, Holmes G, Witten IH. Data mining in bioinformatics using Weka. Bioinformatics. 2004 Oct 12;20(15):2479-81.

15. Page AJ, Cummins CA, Hunt M, Wong VK, Reuter S, Holden MT, et al. Roary: rapid large-scale prokaryote pan genome analysis. Bioinformatics. 2015 Nov 15;31(22):3691-3.

16. Katoh K, Misawa K, Kuma K, Miyata T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 2002 Jul 15;30(14):3059-66.

17. Galperin MY, Makarova KS, Wolf YI, Koonin EV. Expanded microbial genome coverage and improved protein family annotation in the COG database. Nucleic Acids Res. 2015 Jan;43(Database issue):D261-9.

18. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol. 1987 Jul;4(4):406-25.

19. Paradis E, Claude J, Strimmer K. APE: Analyses of Phylogenetics and Evolution in R language. Bioinformatics. 2004 Jan 22;20(2):289-90.

20. Schliep KP. phangorn: phylogenetic analysis in R. Bioinformatics. 2011 Feb 15;27(4):592-3.

21. G Yu, DK Smith, H Zhu, Y Guan, TTY Lam*. ggtree: an R package for visualization and annotation of phylogenetic trees with their covariates and other associated data. Methods Ecol Evol. 2017, 8(1):28-36.

22. Arndt D, Grant JR, Marcu A, Sajed T, Pon A, Liang Y, et al. PHASTER: a better, faster version of the PHAST phage search tool. Nucleic Acids Res. 2016 Jul 8;44(W1):W16-21.

23. Zankari E, Hasman H, Cosentino S, Vestergaard M, Rasmussen S, Lund O, et al. Identification of acquired antimicrobial resistance genes. J Antimicrob Chemother. 2012 Nov;67(11):2640-4.

24. Chen L, Xiong Z, Sun L, Yang J, Jin Q. VFDB 2012 update: toward the genetic diversity and molecular evolution of bacterial virulence factors. Nucleic Acids Res. 2012 Jan;40(Database issue):D641-5.

25. Rouli L, Merhej V, Fournier P-E, Raoult D. The bacterial pangenome as a new tool for analysing pathogenic bacteria. New Microbes New Infect. 2015 Jun 26;7:72-85.

26. Georgiades K, Raoult D. Defining Pathogenic Bacterial Species in the Genomic Era. Front Microbiol. 2011 Jan 17;1:151.

27. Rasko DA, Rosovitz MJ, Myers GS, Mongodin EF, Fricke WF, Gajer P, et al. The Pangenome Structure of Escherichia coli: Comparative Genomic Analysis of E. coli Commensal and Pathogenic Isolates. J Bacteriol. 2008 Oct;190(20):6881-93.

28. Vieira G, Sabarly V, Bourguignon PY, Durot M, Le Fèvre F, Mornico D, et al. Core and Panmetabolism in *Escherichia coli* . J Bacteriol. 2011 Mar;193(6):1461-72.

29. Caza M, Kronstad JW. Shared and distinct mechanisms of iron acquisition by bacterial and fungal pathogens of humans. Front Cell Infect Microbiol. 2013 Nov 19;3:80.

30. Ma L, Payne SM. AhpC Is Required for Optimal Production of Enterobactin by Escherichia coli. J Bacteriol. 2012 Dec;194(24):6748-57.

31. Peralta DR, Adler C, Corbalán NS, Paz García EC, Pomares MF, Vincent PA. Enterobactin as Part of the Oxidative Stress Response Repertoire. PLoS One. 2016 Jun 16;11(6):e0157799.

32. Russo TA, McFadden CD, Carlino-MacDonald UB, Beanan JM, Barnard TJ, Johnson JR. IroN Functions as a Siderophore Receptor and Is a Urovirulence Factor in an Extraintestinal Pathogenic Isolate of *Escherichia coli* . Infect Immun. 2002 Dec;70(12):7156-60.

33. Cherayil BJ. The role of iron in the immune response to bacterial infection. Immunol Res. 2011 May;50(1):1-9.

34. Belge Kurutas E, Ciragil P, Gul M, Kilinc M. The Effects of Oxidative Stress in Urinary Tract Infection. Mediators Inflamm. 2005 Aug 31;2005(4):242-4.

35. Schilling JD, Mulvey MA, Hultgren SJ. Structure and function of Escherichia coli type 1 pili: new insight into the pathogenesis of urinary tract infections. J Infect Dis. 2001 Mar 1;183 Suppl 1:S36-40.

36. Sheikh A, Rashu R, Begum YA, Kuhlman FM, Ciorba MA, Hultgren SJ, et al. Highly conserved type 1 pili promote enterotoxigenic *E. coli* pathogen-host interactions. PLoS Negl Trop Dis. 2017 May 22;11(5):e0005586.

37. Spurbeck RR, Stapleton AE, Johnson JR, Walk ST, Hooton TM, Mobley HLT. Fimbrial Profiles Predict Virulence of Uropathogenic Escherichia coli Strains: Contribution of Ygi and Yad Fimbriae. Infect Immun. 2011 Dec;79(12):4753-63.

38. Weissman SJ, Chattopadhyay S, Aprikian P, Obata-Yasuoka M, Yarova-Yarovaya Y, Stapleton A, et al. Clonal analysis reveals high rate of structural mutations in fimbrial adhesins of extraintestinal pathogenic *Escherichia coli*. Mol Microbiol. 2006 Feb;59(3):975-88.

39. Hernandes RT, Velsko I, Sampaio SC, Elias WP, Robins-Browne RM, Gomes TA, et al. Fimbrial Adhesins Produced by Atypical Enteropathogenic Escherichia coli Strains . Appl Environ Microbiol. 2011 Dec;77(23):8391-9.

40. Toro E, Shapiro L. Bacterial Chromosome Organization and Segregation. Cold Spring Harb Perspect Biol. 2010 Feb;2(2):a000349.

41. Bakshi U, Sarkar M, Paul S, Dutta C. Assessment of virulence potential of uncharacterized Enterococcus faecalis strains using pan genomic approach – Identification of pathogen–specific and habitat-specific genes. Sci Rep. 2016 Dec 7;6:38648.

42. Raven KE, Reuter S, Gouliouris T, Reynolds R, Russell JE, Brown NM, et al. Genome-based characterization of hospital-adapted Enterococcus faecalis lineages. Nat Microbiol. 2016 Feb 8;1:15033.

43. Snider J, Thibault G, Houry WA. The AAA+ superfamily of functionally diverse proteins. Genome Biol. 2008 Apr 30;9(4):216.

885 44. Davidson AL, Dassa E, Orelle C, Chen J. Structure, Function, and Evolution of Bacterial
886 ATP-Binding Cassette Systems. Microbiol Mol Biol Rev. 2008 Jun;72(2):317-64.
887
888 45. Wiethaus J, Wirsing A, Narberhaus F, Masepohl B. Overlapping and Specialized
889 Functions of the Molybdenum-Dependent Regulators MopA and MopB in Rhodobacter
890 capsulatus. J Bacteriol. 2006 Dec;188(24):8441-51
891
892 46. Zhang Y, Rump S, Gladyshev VN. Comparative Genomics and Evolution of Molybdenum
893 Utilization. Coord Chem Rev. 2011 May;255(9-10):1206-1217.
894
895 47. Mann R, Mediati DG, Duggin IG, Harry EJ, Bottomley AL. Metabolic Adaptations of
896 Uropathogenic E. coli in the Urinary Tract. Front Cell Infect Microbiol. 2017 Jun 8;7:241.
897
898 48. Institute of Medicine (US) Panel on Micronutrients. Dietary Reference Intakes for
899 Vitamin A, Vitamin K, Arsenic, Boron, Chromium, Copper, Iodine, Iron, Manganese,
900 Molybdenum, Nickel, Silicon, Vanadium, and Zinc. Washington (DC): National Academies
901 Press (US); 2001. 11, Molybdenum. Available from:
902 https://www.ncbi.nlm.nih.gov/books/NBK222301/
903
904 49. Chan K-G, Sulaiman J, Yong DA, Tee KK, Yin W-F, Priya K. Draft Genome Perspective
905 of Staphylococcus saprophyticus Strain SU8, an N-Acyl Homoserine Lactone-Degrading
906 Bacterium. Genome Announc. 2015 Sep 24;3(5). pii: e01097-15.
907
908 50. Bertuzzi AS, Guinane CM, Crispie F, Kilcawley KN, McSweeney PLH, Rea MC.
909 Genome Sequence of Staphylococcus saprophyticus DPC5671, a Strain Isolated from
910 Cheddar Cheese. Genome Announc. 2017 Apr 20;5(16). pii: e00193-17.
911
912 51. Kuroda M, Yamashita A, Hirakawa H, Kumano M, Morikawa K, Higashide M, et al.
913 Whole genome sequence of Staphylococcus saprophyticus reveals the pathogenesis of
914 uncomplicated urinary tract infection. Proc Natl Acad Sci U S A. 2005 Sep 13;102(37):13272-
915 7.
916

52. Canchaya C, Proux C, Fournous G, Bruttin A, Brüssow H. Prophage Genomics. Microbiol Mol Biol Rev. 2003 Jun;67(2):238-76.

53. Shaaban S, Cowley LA, McAteer SP, Jenkins C, Dallman TJ, Bono JL, et al. Evolution of a zoonotic pathogen: investigating prophage diversity in enterohaemorrhagic *Escherichia coli* O157 by long-read sequencing. Microb Genom. 2016 Dec 12;2(12):e000096.

54. Fortier L-C, Sekulovic O. Importance of prophages to evolution and virulence of bacterial pathogens. Virulence. 2013 Jul 1;4(5):354-65.

55. Soares JA, Ahmer BMM. Detection of acyl-homoserine lactones by Escherichia and Salmonella. Curr Opin Microbiol. 2011 Apr;14(2):188-93.

56. Klein JA, Dave BM,Raphenya AR,McArthur AG,Knodler LA. Functional relatedness in the Inv/Mxi-Spa type III secretion system family. Mol Microbiol. 2017 Mar;103(6):973-991.

57. Johnson RC. Site-specific DNA Inversion by Serine Recombinases. Microbiol Spectr. 2015 Feb 19;3(3):1-36.

58. Kutsukake K, Nakashima H, Tominaga A, Abo T. Two DNA Invertases Contribute to Flagellar Phase Variation in Salmonella enterica Serovar Typhimurium Strain LT2. J Bacteriol. 2006 Feb;188(3):950-7.

59. Tadesse DA, Zhao S, Tong E, Ayers S, Singh A, Bartholomew MJ, et al. Antimicrobial Drug Resistance in *Escherichia coli* from Humans and Food Animals, United States, 1950–2002. Emerg Infect Dis. 2012 May;18(5):741-9.

60. Briñas L, Zarazaga M, Sáenz Y, Ruiz-Larrea F, Torres C. β-Lactamases in Ampicillin-Resistant *Escherichia coli* Isolates from Foods, Humans, and Healthy Animals. Antimicrob Agents Chemother. 2002 Oct;46(10):3156-63.

948 61. Huddleston JR. Horizontal gene transfer in the human gastrointestinal tract: potential
949 spread of antibiotic resistance genes. Infect Drug Resist. 2014 Jun 20;7:167-76.
950
951 62. Van Schaik W. The human gut resistome. Philos Trans R Soc Lond B Biol Sci. 2015 Jun
952 5;370(1670):20140087.
953
954 63. Dason S, Dason JT, Kapoor A. Guidelines for the diagnosis and management of recurrent
955 urinary tract infection in women. Can Urol Assoc J. 2011 Oct;5(5):316-22.
956
957 64. Kao CY, Wu HM, Lin WH, Tseng CC, Yan JJ, Wang MC, et al. Plasmid-mediated
958 quinolone resistance determinants in quinolone-resistant *Escherichia coli* isolated from
959 patients with bacteremia in a university hospital in Taiwan, 2001–2015. Sci Rep. 2016 Aug
960 30;6:32281.
961
962 65. Ruiz E, Sáenz Y, Zarazaga M, Rocha-Gracia R, Martínez-Martínez L, Arlet G, et al. qnr,
963 aac(6')-Ib-cr and qepA genes in Escherichia coli and Klebsiella spp.: genetic environments
964 and plasmid and chromosomal location. J Antimicrob Chemother. 2012 Apr;67(4):886-97.
965
966 66. Aldred KJ, Kerns RJ, Osheroff N. Mechanism of Quinolone Action and Resistance.
967 Biochemistry. 2014 Mar 18;53(10):1565-74.
968
969 67. Delsuc F, Brinkmann H, Philippe H. Phylogenomics and the reconstruction of the tree of
970 life. Nat Rev Genet. 2005 May;6(5):361-75.
971
972 68. Bolotin E, Hershberg R. Gene Loss Dominates As a Source of Genetic Variation within
973 Clonal Pathogenic Bacterial Species. Genome Biol Evol. 2015 Jul 10;7(8):2173-87.
974
975 69. Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: improving the sensitivity of
976 progressive multiple sequence alignment through sequence weighting, position-specific gap
977 penalties and weight matrix choice. Nucleic Acids Res. 1994 Nov 11;22(22):4673-80.
978

70. Tamura K, Stecher G, Peterson D, Filipski A, Kumar S. MEGA6: Molecular Evolutionary Genetics Analysis Version 6.0. Mol Biol Evol. 2013 Dec;30(12):2725-9.

MANUSCRITO II – Publicado no periódico *Genome Announcements*

AMERICAN SOCIETY FOR MICROBIOLOGY | genomeAnnouncements

Draft Genome Sequence of Brazilian *Escherichia coli* Uropathogenic Strain E2

Thiago G. S. Paim,[a] Luiza Pieta,[b] Janira Prichula,[a] Gustavo E. Sambrano,[a] Renata Soares,[a] Juliana Caierão,[a] Jeverson Frazzon,[b] Pedro A. d'Azevedo[a]

Laboratory of Molecular Microbiology, Federal University of Health Sciences of Porto Alegre, Porto Alegre, Brazil[a]; Institute of Food Sciences and Technology, Federal University of Rio Grande do Sul, Porto Alegre, Brazil[b]

***Escherichia coli* is a common pathogen recovered from cystitis infections. In this report, we announce the draft genome sequence of strain E2 isolated from the urine specimen from a female patient in South Brazil. The genome assembly has 5,081,209 bp, a G+C content of 50.57%, and virulence factors associated with both enteroaggregative and uropathogenic *E. coli* strains.**

Received 13 August 2016 **Accepted** 17 August 2016 **Published** 6 October 2016

Citation Paim TGS, Pieta L, Prichula J, Sambrano GE, Soares R, Caierão J, Frazzon J, d'Azevedo PA. 2016. Draft genome sequence of Brazilian *Escherichia coli* uropathogenic strain E2. Genome Announc 4(5):e01085-16. doi:10.1128/genomeA.01085-16.

Address correspondence to Thiago G. S. Paim, thiagog@ufcspa.edu.br.

Escherichia coli is the most common uropathogen isolated from urinary tract infections (UTIs). This member of the *Enterobacteriaceae* family colonizes the gastrointestinal tract as commensal microbiota and protects against other enteropathogens (1). However, extraintestinal *E. coli* infection, mainly UTI, is a public health problem due to high prevalence, antimicrobial resistance, and health care costs. Although *E. coli* species are well studied, knowledge about this species is continually changing (2).

Uropathogenic *E. coli* (UPEC) strains are examples of opportunistic pathogens. The genome plasticity of UPEC strains increases the pathogenic potential of these bacteria, represented by virulence-specific adaptation mechanisms, such as horizontal gene transfer and gene mutations (3).

Here, we present a draft genome sequence of the *E. coli* E2 strain, isolated from a uroculture from young woman, who is 19 years old, in a tertiary hospital in South Brazil.

Genomic DNA from *E. coli* E2 was extracted with the Wizard genomic DNA purification kit (Promega). After DNA quantification by the Qubit double-stranded DNA (dsDNA) high-sensitivity (HS) assay kit, according to the manufacturer's instructions (Life Technologies), the library was generated using the Nextera XT DNA sample preparation kit and Nextera XT index primers (Illumina). The genome sequence was determined using shotgun sequencing (MiSeq reagent kit version 2 with 300 cycles, paired-end read length of 2 × 150 bp) on an Illumina MiSeq platform. The quality metrics of reads was performed by FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/), which revealed a total of 767,716 reads and quality scores >30 across all bases. Preprocessing of reads was performed with FASTX-Toolkit (http://hannonlab.cshl.edu/fastx_toolkit/index.html), removing sequencing artifacts. A shell script command was carried out to *de novo* assembly using the following assemblers: ABySS (4), SPAdes (5), SOAP*denovo*2 (6), and Velvet (7), with k-mer sequence lengths ranging from 20 to 63. The best k-mer was 47-mer for ABySS. The genome sequence was annotated via NCBI Prokaryotic Genome Annotation Pipeline (8), consisting of 222 contigs (5,081,209 bases), a G+C content of 50.57%, N_{50} of 74,212 bases, 4,746 protein-coding sequences (CDSs), 51 tRNAs, and five rRNAs. The genome coverage was 45×.

Two plasmids were predicted using the PlasmidFinder (9): conjugative *Escherichia coli* K-12 plasmid F, and pRSF1010_SL1344 from *Salmonella enterica* subsp. *enterica* serovar Typhimurium (accession numbers AP001918 and HE654726, respectively). Three genes that confer sulfonamide and trimethoprim resistance (*sul1*, *sul2*, and *dfrA1*, respectively) were found using ResFinder (10). In addition, the phenotype of beta-lactam resistance using ampicillin was confirmed by the presence of the $bla_{\text{TEM-1B}}$ gene encoding beta-lactamase. VirulenceFinder (11) found genes related to aggregative adhesion fimbria (AAF) type III (*agg3RABCD*), which is typical of enteroaggregative *Escherichia coli* (12–14). This pathotype is suggested by the presence of diagnostic gene of dispersin transporter (*aatA*) (15). However, E2 strain carries typical UPEC virulence factors, such as secreted autotransporter toxin (*sat*) (16). These results suggest that the *E. coli* E2 strain shows the ability to colonize different host niches and has genome plasticity, as demonstrated in a previous study (17).

Accession number(s). This whole-genome shotgun project has been deposited at DDBJ/ENA/GenBank under the accession no. MBRL00000000. The version described in this paper is version MBRL01000000.

ACKNOWLEDGMENTS

We thank the Institute of Food Sciences and Technology (ICTA-UFRGS) for providing technical support to use the MiSeq Illumina sequencing platform.

FUNDING INFORMATION

This work, including the efforts of Pedro Alves d'Azevedo, was funded by MCTI | Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

REFERENCES

1. **Hudault S, Guignot J, Servin AL.** 2001. *Escherichia coli* strains colonising the gastrointestinal tract protect germfree mice against *Salmonella* Typhimurium infection. Gut **49:**47–55. http://dx.doi.org/10.1136/gut.49.1.47.

2. **Vila J, Sáez-López E, Johnson JR, Römling U, Dobrindt U, Cantón R, Giske CG, Naas T, Carattoli A, Martínez-Medina M, Bosch J, Retamar P, Rodríguez-Baño J, Baquero F, Soto SM**. 2016. *Escherichia coli*: an old friend with new tidings. FEMS Microbiol Rev, in press. http://dx.doi.org/10.1093/femsre/fuw005.
3. **Sokurenko E**. 2016. Pathoadaptive mutations in uropathogenic *Escherichia coli*. Microbiol Spectr **4:**UTI-0020-2015. http://dx.doi.org/10.1128/microbiolspec.UTI-0020-2015.
4. **Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I**. 2009. ABySS: a parallel assembler for short read sequence data. Genome Res **19:**1117–1123. http://dx.doi.org/10.1101/gr.089532.108.
5. **Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, Pyshkin AV, Sirotkin AV, Vyahhi N, Tesler G, Alekseyev MA, Pevzner PA**. 2012. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol **19:**455–477. http://dx.doi.org/10.1089/cmb.2012.0021.
6. **Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, Tang J, Wu G, Zhang H, Shi Y, Liu Y, Yu C, Wang B, Lu Y, Han C, Cheung DW, Yiu SM, Peng S, Xiaoqian Z, Liu G, Liao X, Li Y, Yang H, Wang J, Lam TW, Wang J**. 2012. SOAP*denovo*2: an empirically improved memory-efficient short-read *de novo* assembler. Gigascience **1:**18. http://dx.doi.org/10.1186/2047-217X-1-18.
7. **Zerbino DR, Birney E**. 2008. Algorithms for *de novo* short read assembly using de Bruijn graphs. Genome Res **18:**821–829. http://dx.doi.org/10.1101/gr.074492.107.
8. **Tatusova T, DiCuccio M, Badretdin A, Chetvernin V, Nawrocki EP, Zaslavsky L, Lomsadze A, Pruitt KD, Borodovsky M, Ostell J**. 2015. NCBI prokaryotic genome annotation pipeline. Nucleic Acids Res **43:**D599–D605. http://dx.doi.org/10.1093/nar/gku1062.
9. **Carattoli A, Zankari E, García-Fernández A, Voldby Larsen M, Lund O, Villa L, Møller Aarestrup F, Hasman H**. 2014. *In silico* detection and typing of plasmids using PlasmidFinder and plasmid multilocus sequence typing. Antimicrob Agents Chemother **58:**3895–3903. http://dx.doi.org/10.1128/AAC.02412-14.
10. **Zankari E, Hasman H, Cosentino S, Vestergaard M, Rasmussen S, Lund O, Aarestrup FM, Larsen MV**. 2012. Identification of acquired antimicrobial resistance genes. J Antimicrob Chemother **67:**2640–2644. http://dx.doi.org/10.1093/jac/dks261.
11. **Joensen KG, Scheutz F, Lund O, Hasman H, Kaas RS, Nielsen EM, Aarestrup FM**. 2014. Real-time whole-genome sequencing for routine typing, surveillance, and outbreak detection of verotoxigenic *Escherichia coli*. J Clin Microbiol **52:**1501–1510. http://dx.doi.org/10.1128/JCM.03617-13.
12. **Farfan MJ, Inman KG, Nataro JP**. 2008. The major pilin subunit of the AAF/II fimbriae from enteroaggregative *Escherichia coli* mediates binding to extracellular matrix proteins. Infect Immun **76:**4378–4384. http://dx.doi.org/10.1128/IAI.00439-08.
13. **Berry AA, Yang Y, Pakharukova N, Garnett JA, Lee WC, Cota E, Marchant J, Roy S, Tuittila M, Liu B, Inman KG, Ruiz-Perez F, Mandomando I, Nataro JP, Zavialov AV, Matthews S**. 2014. Structural insight into host recognition by aggregative adherence fimbriae of enteroaggregative *Escherichia coli*. PLoS Pathog **10:**e1004404.
14. **Hebbelstrup Jensen B, Olsen KE, Struve C, Krogfelt KA, Petersen AM**. 2014. Epidemiology and clinical manifestations of enteroaggregative *Escherichia coli*. Clin Microbiol Rev **27:**614–630. http://dx.doi.org/10.1128/CMR.00112-13.
15. **Lima IF, Boisen N, Quetz Jda S, Havt A, de Carvalho EB, Soares AM, Lima NL, Mota RM, Nataro JP, Guerrant RL, Lima AÂ**. 2013. Prevalence of enteroaggregative *Escherichia coli* and its virulence-related genes in a case–control study among children from north-eastern Brazil. J Med Microbiol **62:**683–693.
16. **Bien J, Sokolova O, Bozko P**. 2012. Role of uropathogenic *Escherichia coli* virulence factors in development of urinary tract infection and kidney damage. Int J Nephrol **2012:**681473. http://dx.doi.org/10.1155/2012/681473.
17. **Toval F, Köhler CD, Vogel U, Wagenlehner F, Mellmann A, Fruth A, Schmidt MA, Karch H, Bielaszewska M, Dobrindt U**. 2014. Characterization of *Escherichia coli* Isolates from hospital inpatients or outpatients with urinary tract infection. J Clin Microbiol **52:**407–418.

MANUSCRITO III – Publicado no periódico *Genome Announcements*

AMERICAN SOCIETY FOR MICROBIOLOGY | genomeAnnouncements

Draft Genome Sequence of *Enterococcus faecalis* Strain F165 Isolated from a Urinary Tract Infection

Thiago G. S. Paim,[a] Luiza Pieta,[b] Janira Prichula,[a] Gustavo E. Sambrano,[a] Renata Soares,[a] Aline Dall Bello,[a] Jeverson Frazzon,[b] Pedro A. d'Azevedo[a]

Laboratory of Molecular Microbiology, Federal University of Health Sciences of Porto Alegre, Porto Alegre, Brazil[a]; Institute of Food Sciences and Technology, Federal University of Rio Grande do Sul, Porto Alegre, Brazil[b]

We report here a draft genome sequence of *Enterococcus faecalis* strain F165 isolated from a urine specimen in South Brazil. The genome size was 3,049,734 bp, with a G+C content of 37.38%, and genes related to antimicrobial resistance and adherence were found in the strain. These findings are consistent with pathogenesis of *E. faecalis* species.

Received 13 August 2016 **Accepted** 17 August 2016 **Published** 6 October 2016

Citation Paim TGS, Pieta L, Prichula J, Sambrano GE, Soares R, Bello AD, Frazzon J, d'Azevedo PA. 2016. Draft genome sequence of *Enterococcus faecalis* strain F165 isolated from a urinary tract infection. Genome Announc 4(5):e01084-16. doi:10.1128/genomeA.01084-16.

Address correspondence to Thiago G. S. Paim, thiagog@ufcspa.edu.br.

Enterococcus faecalis strains are the Gram-positive cocci often recovered from urinary tract infections (UTIs), particularly among individuals that have risk factors such as advanced age, pregnancy, or urinary catheterization. These Gram-positive cocci are isolated from polymicrobial communities on the surface of indwelling urinary devices, causing at least 30% of catheter-associated UTIs (1). Although enterococci are commensals in the gastrointestinal tract, *E. faecalis* is known as an opportunistic pathogen causing infections related to health care, mainly due to the expression of virulence factors associated with adherence of mucosal and abiotic surfaces (2).

We report here a draft genome sequence of a clinical isolate of vancomycin-susceptible *Enterococcus faecalis*, namely, F165, recovered from positive uroculture in a tertiary care hospital of South Brazil.

Genomic DNA was extracted using the Wizard genomic DNA purification kit (Promega). The quality and yield were assessed by agarose gel electrophoresis and QuBit double-stranded DNA (dsDNA) high-sensitivity (HS) assay kit (Life Technologies), respectively. The library for sequencing was prepared using the Nextera XT DNA kit and index primers (Illumina), and the reads were generated by MiSeq reagent kit version 2 with 300 cycles on an Illumina MiSeq platform. A total of 757,793 paired-end reads were recovered, and the sequences were adapter and quality trimmed using FASTX-Toolkit (http://hannonlab.cshl.edu/fastx_toolkit/index.html). All the reads showed quality score $Q > 30$. A shell script command for Linux/Unix was carried out to *de novo* assembly using the following assemblers: ABySS (3), SPAdes (4), SOAP*denovo*2 (5), and Velvet (6), with k-mer sequence length ranging from 20 to 63. The best assembly had a 37-mer and was performed on ABySS. Then, the NCBI Prokaryotic Genome Annotation Pipeline (7) was used to annotate the DNA sequences. The draft genome has 3,049,734 bp in a total of 194 contigs, with an N_{50} of 37,159 bp and G+C content of 37.38%. The annotation found 2,886 coding sequences, 29 tRNAs, and six rRNAs.

The identification of acquired antibiotic resistance and virulence factor genes were performed with the Web tool ResFinder and VirulenceFinder, respectively (8, 9). Four genes to aminoglycoside resistance were identified [*ant(6)-Ia, aac(6′)-aph(2′′), aph(3′)-III,* and *str*], which was in agreement with the phenotype screening for high-level aminoglycoside resistance. Ciprofloxacin and norfloxacin resistance have been found in the strain, and mutations that confer amino acid substitutions in housekeeping genes, such as DNA gyrase A (*gyrA*), are the main mechanism of antimicrobial nonsusceptibility to fluoroquinolone agents (10). Therefore, pairwise alignment was performed with Clustal W algorithm between *gyrA*-translated genes of *Enterococcus faecalis* ATCC 29212 and *E. faecalis* F165 strains. The serine at position 84 (relative to amino acid coordinates of GyrA from F165 strain) was changed to isoleucine, similar to a previous study in resistant strains (11). Fifteen virulence factors were found, such as aggregation substance (*agg*), collagen adhesin (*ace*), hyaluronidase (*hylA* and *hylB*), endocarditis- and biofilm-associated pili (*ebpA*, *ebpB*, and *ebpC*), and *E. faecalis* endocarditis antigen A (*efa*Afs) (2). All these genes contribute to adherence, aggregation, and invasion of the host tissue for enterococcal strains.

Accession number(s). This Whole Genome Shotgun project has been deposited at DDBJ/ENA/GenBank under the accession MBRC00000000. The version described in this paper is version MBRC01000000.

ACKNOWLEDGMENTS

We thank the Institute of Food Sciences and Technology (ICTA-UFRGS) for providing MiSeq Illumina sequencing technology.

FUNDING INFORMATION

This work, including the efforts of Pedro Alves d'Azevedo, was funded by MCTI | Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

REFERENCES

1. **Kline KA, Lewis AL.** 2016. Gram-positive uropathogens, polymicrobial urinary tract infection, and the emerging microbiota of the urinary tract.

Microbiol Spectr **4**. http://dx.doi.org/10.1128/microbiolspec.UTI-0012-2012.

2. **Anderson AC, Jonas D, Huber I, Karygianni L, Wölber J, Hellwig E, Arweiler N, Vach K, Wittmer A, Al-Ahmad A**. 2015. *Enterococcus faecalis* from Food, clinical specimens, and oral sites: prevalence of virulence factors in association with biofilm formation. Front Microbiol **6:**1534. http://dx.doi.org/10.3389/fmicb.2015.01534.
3. **Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I**. 2009. ABySS: a parallel assembler for short read sequence data. Genome Res **19:**1117–1123. http://dx.doi.org/10.1101/gr.089532.108.
4. **Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, Pyshkin AV, Sirotkin AV, Vyahhi N, Tesler G, Alekseyev MA, Pevzner PA**. 2012. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol **19:**455–477. http://dx.doi.org/10.1089/cmb.2012.0021.
5. **Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, Tang J, Wu G, Zhang H, Shi Y, Liu Y, Yu C, Wang B, Lu Y, Han C, Cheung DW, Yiu SM, Peng S, Xiaoqian Z, Liu G, Liao X, Li Y, Yang H, Wang J, Lam TW, Wang J**. 2012. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience **1:**18. http://dx.doi.org/10.1186/2047-217X-1-18.
6. **Zerbino DR, Birney E**. 2008. Algorithms for de novo short read assembly using de Bruijn graphs. Genome Res **18:**821–829. http://dx.doi.org/10.1101/gr.074492.107.
7. **Tatusova T, DiCuccio M, Badretdin A, Chetvernin V, Nawrocki EP, Zaslavsky L, Lomsadze A, Pruitt KD, Borodovsky M, Ostell J**. 2015. NCBI prokaryotic genome annotation pipeline. Nucleic Acids Res **43** (Database issue):D599-605. http://dx.doi.org/10.1093/nar/gku1062.
8. **Zankari E, Hasman H, Cosentino S, Vestergaard M, Rasmussen S, Lund O, Aarestrup FM, Larsen MV**. 2012. Identification of acquired antimicrobial resistance genes. J Antimicrob Chemother **67:**2640–2644. http://dx.doi.org/10.1093/jac/dks261.
9. **Joensen KG, Scheutz F, Lund O, Hasman H, Kaas RS, Nielsen EM, Aarestrup FM**. 2014. Real-time whole-genome sequencing for routine typing, surveillance, and outbreak detection of verotoxigenic *Escherichia coli*. J Clin Microbiol **52:**1501–1510. http://dx.doi.org/10.1128/JCM.03617-13.
10. **Aldred KJ, Kerns RJ, Osheroff N**. 2014. Mechanism of quinolone action and resistance. Biochemistry **53:**1565–1574. http://dx.doi.org/10.1021/bi5000564.
11. **El Amin NA, Jalal S, Wretlind B**. 1999. Alterations in GyrA and ParC associated with fluoroquinolone resistance in *Enterococcus faecium*. Antimicrob Agents Chemother **43:**947–949.

CONCLUSÃO

O presente estudo abordou sistematicamente a análise de genomas bacterianos das principais espécies de uropatógenos, avaliando-se características associadas à adaptação ao trato urinário e pontualmente informações genômicas de três isolados bacterianos recuperados de infecções do trato urinário da região sul do país.

Para as três espécies de uropatógenos analisadas, todas foram caracterizadas como tendo um pan-genoma aberto, propício a aquisição de elementos genéticos à medida que novos genomas são computados. Um repertório amplo de genes é característico de bactérias com elevado valor adaptativo ao nicho em que se estabelecem, sendo que, independente do nicho ao qual os genomas do estudo foram agrupados (trato gastrointestinal e urinário, com exceção de *S. saprophyticus*) o pan-genoma menteve a característica.

Devido ao elevado número de genomas representantes de *E. coli*, os genes conservados entre os isolados fazem parte de uma pequena fração do pan-genoma. Os genes do conjunto acessório representam mais de 90% do repertório de genes do gênero, sendo que a proporção de genes em categorias funcionais associadas ao metabolismo primário e transporte de íons não difere entre nichos. Já para categorias como motilidade celular, ciclo celular e metabolismo secundário, isolados do trato urinário apresentaram uma proporção superior de genes. Genes importantes como homólogos à enterobactina (sideróforo) e à fímbria tipo I podem conferir vantagem seletiva para isolado do trato urinário principalmente por variabilidade gênica, uma vez que também estão presentes em isolados gastrointestinais.

Em relação a espécie *Enterococcus faecalis*, não houve diferenças significativas entre as categorias funcionais de agrupamento dos genes presente nos pan-genomas (gastrointestinal e urinário), provavelmente devido ao número reduzido de genomas computados. Mesmo assim, é visto que há genes no pan-genoma recuperado do trato urinário com descritores únicos baseados no banco de dados do COG. Todos eles fazem parte do conjunto acessório de genes e frequentemente encontrado nos genomas analisados. Fazem parte de uma família de transportadores transmembrana tendo como cofator o micro-elemento molibdênio. A presença desses homólogos poderia conferir vantagem adaptativa desses isolados no trato urinário por redundância na maquinaria celular para obter e exportar moléculas, a exemplo de nutrientes e metabólitos.

Todos os isolados de *Staphylococcus saprophyticus* fazem parte do grupo do trato urinário. Em comparação com os outros uropatógenos do estudo, essa espécie apresenta proporcionalmente mais genes associados ao metabolismo e transporte de aminoácidos, coenzimas e íons inorgânicos. Como o único uropatógeno verdadeiro do gênero *Staphylococcus*, esses dados corroboram para a adaptabilidade da espécie S. saprophyticus para o trato urinário.

A presença de elementos genéticos integrativo é uma importante fonte de variabilidade em micro-organismos patogênicos. No presente estudo, somente diferenças significativas foram encontradas para a espécie *E. coli*. Embora em isolados gastrointestinais a presença de profagos seja superior, em isolados urinários mais frequentemente os profagos carreiam genes de resistência e virulência.

De posse das informações sobre os genomas no banco de dados GenBank, uma série histórica pôde ser traçada sobre a presença de genes de resistência. Para *E. coli*, há um aumento desses genes nos genomas no decorrer do tempo. Além disso, isolados gastrointestinais mostraram-se como um importante reservatório de genes de resistência, e em isolados urinários a presença um gene para aminoglicosídeo acetiltransferase (*aac(6')Ib-cr*) que confere resistência também às fluorquinolonas foi frequentemente encontrado.

Filogeneticamente, os isolados não se agrupam conforme o nicho. Há uma variabilidade importante sobre o conteúdo gênico de cada micro-organismo. O isolado *E. faecalis* F165 compartilha um ancestral comum com isolados relacionados tanto ao trato urinário, como micro-organismo comensais (trato gastrointestinal) e patogênicos (recuperado de bacteremia). O mesmo ocorre para o isolado *E. coli* E2. Entretanto, diferentemente para *E. faecalis*, o clado apresenta todos os representantes a partir do trato gastrointestinal. A exemplo dos outros uropatógenos, para o isolado *S. saprophyticus* SS545 não se pôde correlacionar o agrupamento conforme nicho de isolamento.

Por conseguinte, os micro-organismo uropatogênicos apresentam características importantes associadas ao nicho, comparados aos seus representantes do nicho reservatório (gastrointestinal). Mesmo assim, cada isolado apresenta um repertório singular, como *E. coli* E2 no qual apresenta determinantes genéticos típicos de *E. coli* enteroagregativa bem como de UPEC. Nesse contexto, a análise em grande escala de dados disponíveis em banco de dados, aliado ao conhecimento de ferramentas de bioinformática e linguagem de programação,

propicia um melhor entendimento sobre a biologia dos micro-organismos e potencial uso para o desenvolvimento de novas metodologias diagnósticas e terapêuticas.

ANEXO I – Parecer Consubstanciado CEP-UFCSPA

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: DESENVOLVIMENTO DE UM ENSAIO DE PCR EM TEMPO REAL PARA IDENTIFICAÇÃO E AVALIAÇÃO DE MECANISMOS DE RESISTÊNCIA EM MICRO-ORGANISMOS DE IMPORTÂNCIA MÉDICA: ÊNFASE EM COCOS GRAM-

Pesquisador: Pedro Alves d'Azevedo

Área Temática:

Versão: 1

CAAE: 23971913.6.0000.5345

Instituição Proponente: Universidade Federal de Ciências da Saúde de Porto Alegre

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 502.474

Data da Relatoria: 19/12/2013

Apresentação do Projeto:

O presente projeto faz parte de tese de doutorado em Programa de Pós-graduação em Ciências da Saúde da UFCSPA e consta da análise in silico para pesquisa de assinaturas genômicas das principais espécies bacterianas de cocos Gram-positivos através de ensaio de PCR multiplex em tempo real para identificação em nível de espécie dos micro-organismos. O mesmo será realizado utilizando-se um banco de micro-organismos pertencente ao Laboratório de Cocos Gram-positivos da UFCSPA, caracterizados previamente. Determinações de sensibilidade e especificidade diagnóstica e analítica serão conduzidas para avaliar o desempenho da metodologia proposta.

Objetivo da Pesquisa:

O projeto tem como objetivo principal o uso de PCR em tempo real para identificação de cocos Gram-positivos de importância médica. Como objetivos secundários
estão o desenvolvimento de ensaio multiplex para identificação em nível de espécie para cocos Gram-positivos pela metodologia de PCR em tempo real com marcação por fluorescência; desenvolvimento de ensaio de PCR em tempo real seguido de análise HRM (High Resolution Melting) da região genômica 16S rRNA para identificação de patógenos cocos Gram-positivos; avaliação de mecanismos de resistência aos antimicrobianos por ensaio multiplex de PCR em tempo real dos principais cocos Gram-positivos.

Endereço: Rua Sarmento Leite ,245
Bairro: **CEP:** 90.050-170
UF: RS **Município:** PORTO ALEGRE
Telefone: (513)303 -8804 **E-mail:** cep@ufcspa.edu.br

Avaliação dos Riscos e Benefícios:

Este projeto não utilizará sujeitos humanos para seu desenvolvimento, uma vez que o banco de micro-organismos presentes no laboratório foi obtido de projetos anteriores do grupo de pesquisa, contemplados os requisitos de ética em pesquisa. Está referido que todas as boas práticas de biossegurança serão contempladas no projeto, como manipulação de isolados bacterianos em cabines de biossegurança de fluxo laminar, uso de

EPI adequados, descarte e/ou esterilização do material infectante por autoclavação, entre outros.

Os benefícios esperados para o desenvolvimento da metodologia proposta pelo projeto são maior rapidez para o diagnóstico microbiológico, maiores sensibilidade e especificidade diagnóstica e, portanto, maior acurácia na identificação de micro-organismos de importância médica, especificamente para cocos Gram-positivos, isolados alvo do estudo.

Comentários e Considerações sobre a Pesquisa:

Trata-se de um projeto baseado em um banco de micro-organismos e que possui grande relevância pela possibilidade do desenvolvimento de novas técnicas diagnósticas.

Considerações sobre os Termos de apresentação obrigatória:

Como referido o presente projeto de pesquisa não utilizará sujeitos humanos para seu desenvolvimento utilizando um banco de micro-organismos presentes no laboratório.

Não há necessidade de termo de consentimento informado.

Recomendações:

Conclusões ou Pendências e Lista de Inadequações:

O projeto preenche requisitos éticos necessários para seu desenvolvimento e a recomendação é de aprovação do mesmo pelo CEP.

Situação do Parecer:

Aprovado

Necessita Apreciação da CONEP:

Não

Considerações Finais a critério do CEP:

De acordo com o parecer do Relator.

PORTO ALEGRE, 19 de Dezembro de 2013

Assinador por:
José Geraldo Vernet Taborda
(Coordenador)

Endereço: Rua Sarmento Leite ,245
Bairro: **CEP:** 90.050-170
UF: RS **Município:** PORTO ALEGRE
Telefone: (513)303 -8804 **E-mail:** cep@ufcspa.edu.br

ANEXO II – Instruções para submissão de artigo científico para publicação no periódico *Genome Biology*

https://genomebiology.biomedcentral.com/submission-guidelines

APÊNDICE – Programas escritos em linguagem de programação Python e Shell Script utilizados no estudo.

```
#resfinder python script
#This script perform a compilation of data from ResFinder software
#Input is the genbank file

import FeaturesGenbankBiopython as FGB
import os
import sys, getopt


database = ["aminoglycoside","beta-
lactam","colistin","fosfomycin","fusidicacid","macrolide","nitroimidazole","oxazolidinone","pheni


def results_all_tab(list_out, statistics):

    out = open(str(statistics)+".tab","w")
    out.write("Organism\tResistance gene\tIdentity\tQuery/HSP\tContig\tPosition in
contig\tPhenotype\tAccession no.\n")


    for dir in list_out:

        aux = dir.split("/")
        aux2 = aux[-1]
        organism = aux2.replace("_resfinder","")
        try:
            fh = open(str(dir)+"/results_tab.txt","r")
            file = fh.readlines()
            fh.close()

            for line in file[1:]:
                try:
                    out.write(organism+"\t")
                    out.write(line)

                except:
                    None
        except:
            None
    out.close()


def all_atb():

    all = ",".join(database)

    return str(all)


def genbank_to_fasta(path_gb):

    genome = FGB.Genbank(str(path_gb))
    path_fasta = genome.genome_fasta()

    return path_fasta

def resfinder(path_fasta, atb, ident, min_lenght):

    out = path_fasta.replace(".fasta","_resfinder")

    if atb == "all":
        all = all_atb()

        os.system("perl /home/thiagopaim/resfinder/resfinder.pl -d /home/
thiagopaim/resfinder/ResFinderDB -i " + str(path_fasta) + " -a " + str(all) + " -k
" + str(ident) + " -l " + str(min_lenght) + " -o " + str(out))
```

```
    else:
        try:
            if atb in database:

                os.system("perl resfinder.pl -d ResFinderDB -i " + str(path_fasta)
+ " -a " + str(atb) + " -k " + str(ident) + " -l " + str(min_lenght) + " -o " + str
(out))
            else:
                None
        except:

            print("\nAntimicrobial class not valid!\n")
            sys.exit(2)


    return out



def main(argv):

    inputdir = ""
    ident = ""
    min_len = ""
    atb = ""
    statistics = ""

    try:
        opts, args = getopt.getopt(argv,"hd:i:l:a:s:",
["inputdir=","ident=","min_len=","antib=","statistics="])# o ':' informa q passa
argumento
    except getopt.GetoptError:
        print("Resfinder.py -d <inputfile_folder> -i <identity threshold> -l
<minimun lenght> -a <antimicrobial class> -s <name file for statistics output>")
        sys.exit(2)

    for opt, arg in opts:
        if opt == "-h":
            print("\n\nResfinderScript.py \n-d <inputfile_folder_gb>\n-i <identity
threshold>\n-l <minimun lenght>\n-s <name file for statistics output>\n-a
<antimicrobial class:\n\t\taminoglycoside\n\t\tbeta-lactam\n\t\tcolistin\n\t
\tfosfomycin\n\t\tfusidicacid\n\t\tmacrolide\n\t\tnitroimidazole\n\t\toxazolidinone
\n\t\tphenicol\n\t\tquinolone\n\t\trifampicin\n\t\tsulphonamide\n\t\ttetracycline\n
\t\ttrimethoprim\n\t\tglycopeptide\n\t\tall\n\n")
            sys.exit()
        elif opt in ("-d", "--inputdir"):
            inputdir = arg
            print(arg)
        elif opt in ("-i", "--ident"):
            ident = arg
            print(arg)
        elif opt in ("-l","--min_len"):
            min_len = arg
            print(arg)
        elif opt in ("-a","--antib"):
            atb = arg
            print(arg)
        elif opt in ("-s","--statistics="):
            statistics = arg
            print(arg)

    list_dir = os.listdir(str(inputdir))
    list_output = []
```

```
    for file in list_dir:
        if file.endswith(".gb") is True or file.endswith(".gbf") is True or
file.endswith(".gbff") is True or file.endswith(".gbk") is True:

            path_fasta = genbank_to_fasta(str(inputdir)+"/"+str(file)) #convert gb
to fasta

            out = resfinder(str(path_fasta),str(atb),str(ident),str(min_len))
            list_output.append(str(out)) #Add path to output diretory

            os.system("rm "+str(path_fasta)) #Remove fasta file

    print("\n")
    print(list_output)
    print("\n")

    results_all_tab(list_output, statistics)

    print("\n\t\tDone!\n\n")

if __name__ == "__main__":
    main(sys.argv[1:])
else:
    None
```

```
#Extract statistics from Resfinder output

import os
import sys, getopt


def compose_file(inputfile, outfile, type_molecule, organism):

    if type_molecule in ["prophage","genome"]:
        aux_tuple = ()
        string = ""

        out_name = inputfile.replace(".txt","")
        fh1 = open(str(out_name)+"_"+str(outfile)+".csv","w")
        fh1.write("Organism,Resistance gene,Identity,Query/HSP,Contig,Position in
contig,Phenotype,Accession no.,Feature\n")

        fh2 = open(str(inputfile), "r")
        file = fh2.readlines()

        control = len(file)

        if control > 1:
            for elem in file[1:]:
                aux1 = elem.replace(",","/")
                aux2 = aux1.replace("\t",",")
                line = aux2.replace("\n",",")

                string = str(organism)+","+line+str(type_molecule)+"\n"
                fh1.write(string)
        else:
                string = str(organism)+",,,,,,,,"+str(type_molecule)+"\n"
                fh1.write(string)
        fh2.close()
        fh1.close()

        return str(out_name)+"_"+str(outfile)+".csv"
    else:
        print("ResfinderStat.py \n\n-i <inputfile_result_tab.txt from resfinder>
\n-o <outfile>\n-t <type_molecule 'prophage' or 'genome'>\n-s <organism name>\n\n")
        sys.exit(2)


def main(argv):

    inputfile = ""
    outfile = ""
    type_molecule = ""
    organism = ""

    try:
        opts, args = getopt.getopt(argv,"hi:o:t:s:",
["ifile=","outfile=","type_molecule=","organism="])# o ':' informa q passa
argumento
    except getopt.GetoptError:
        print("ResfinderStat.py \n\n-i <inputfile_result_tab.txt from resfinder>
\n-o <outfile>\n-t <type_molecule 'prophage' or 'genome'>\n-s <organism name>\n\n")
        sys.exit(2)

    for opt, arg in opts:
        if opt == "-h":
            print("ResfinderStat.py \n\n-i <inputfile_result_tab.txt from
resfinder> \n-o <outfile>\n-t <type_molecule 'prophage' or 'genome'>\n-s <organism
name>\n\n")
            sys.exit(2)
        elif opt in ("-i", "--ifile"):
```

```
            inputfile = arg
            print(arg)
        elif opt in ("-o", "--outfile"):
            outfile = arg
            print(arg)
        elif opt in ("-t", "--type_molecule"):
            type_molecule = arg
            print(arg)
        elif opt in ("-s", "--organism"):
            organism = arg
            print(arg)

    try:

        out = compose_file(str(inputfile), str(outfile), str(type_molecule), str
(organism))

        print("\nFile saved in -> "+str(out)+"\n")

        return out

    except:
        None


if __name__ == "__main__":
    main(sys.argv[1:])
else:
    None
```

```
#statisticts resfinder genome


def tabular_statistics_resfinder(path_file_tab):

    fh = open(str(path_file_tab), 'r')

    e_c_urine = fh.readlines()

    fh.close()

    organism = []

    for elem in e_c_urine[1:]:
        split = elem.split(",")
        organism.append(split[0])




    org = list(set(organism))


    new_file = path_file_tab.replace(".csv","_statistics.csv")
    fh = open(str(new_file),'w')

    fh.write("organism, n_resist_genes,aminoglycoside,beta-
lactam,colistin,fosfomycin,fusidicacid,macrolide,nitroimidazole,oxazolidinone,phenicol,quinolone,
\n")


    aminog = 0

    beta = 0

    colis = 0

    fosf = 0

    fusid = 0

    macro = 0

    nitro = 0

    oxaz = 0

    pheni = 0

    qui = 0

    rif = 0

    tetra = 0

    tri = 0

    glic = 0

    sulf = 0

    n_resist = 0

    for elem1 in org:
        for elem2 in e_c_urine[1:]:
            split = elem2.split(",")
```

```
            resistance = str(split[6])
            if elem1 == split[0]:
                if resistance == "Aminoglycoside resistance":
                    aminog += 1
                elif resistance == "Beta-lactam resistance":
                    beta += 1
                elif resistance == "Colistin resistance":
                    colis += 1
                elif resistance == "Fosfomycin resistance":
                    fosf += 1
                elif resistance == "Fusidic acid resistance":
                    fusid += 1
                elif resistance == "Macrolide resistance":
                    macro += 1
                elif resistance.find("itroimidazole") != -1:
                    nitro += 1
                elif resistance.find("xazolidinone") != -1:
                    oxaz += 1
                elif resistance == "Phenicol resistance":
                    pheni += 1
                elif resistance.find("uinolone") != -1:
                    qui += 1
                elif resistance == "Rifampicin resistance":
                    rif += 1
                elif resistance == "Sulphonamide resistance":
                    sulf += 1
                elif resistance == "Tetracycline resistance":
                    tetra += 1
                elif resistance == "Trimethoprim resistance":
                    tri += 1
                elif resistance == "Vancomycin resistance (Glycopeptid
resistance)":
                    glic += 1
                else:
                    None


        n_resist = aminog + beta + colis + fosf + fusid + macro + nitro + oxaz +
pheni + qui + rif + sulf + tetra + tri + glic

        fh.write(str(elem1)+","+str(n_resist)+","+str(aminog)+","+str(beta)+","+str
(colis)+","+str(fosf)+","+str(fusid)+","+str(macro)+","+str(nitro)+","+str(oxaz)
+","+str(pheni)+","+str(qui)+","+str(rif)+","+str(sulf)+","+str(tetra)+","+str(tri)
+","+str(glic)+"\n")

        n_resist = 0
        aminog = 0
        beta = 0
        colis = 0
        fosf = 0
        fusid = 0
        macro = 0
        nitro = 0
        oxaz = 0
        pheni = 0
        qui = 0
        rif = 0
        sulf = 0
        tetra = 0
        tri = 0
        glic = 0

    fh.close()

def add_date_collection_to_resfinder_stat(path_csv_resfinder, path_info_genomes):
```

```
    fh1 = open(str(path_csv_resfinder),"r")
    resfinder = fh1.readlines()
    fh1.close()

    fh2 = open(str(path_info_genomes),"r")
    info = fh2.readlines()
    fh2.close()

    new_file = path_csv_resfinder.replace(".csv","_added_info.csv")
    fh = open(new_file, "w")

    fh.write("Organism_strain,n_resist_genes,aminoglycoside,beta-
lactam,colistin,fosfomycin,fusidicacid,macrolide,nitroimidazole,oxazolidinone,phenicol,quinolone,
\n")

    for elem1 in resfinder[1:]:
        split1 = elem1.split(",")
        org1 = split1[0]
        for elem2 in info[1:]:
            split2 = elem2.split(",")
            aux = split2[-1]
            file_org = aux.replace(".gbff\n","")

            if org1 == file_org:
                resf = elem1.replace("\n",",")
                fh.write(str(resf)+str(elem2))
            else:
                None

    fh.close()
```

```
class BlastAnalysis():
    """Enter na PATH of FASTA file to create object"""
    name_file = ""

    def __init__(self, name_file):

        self.name_file = name_file

    def create_database_blast(self, db_type, title):
        """Enter the PATH to FASTA file to create a blast database"""

        try:

            import os
            os.system('makeblastdb -in "'+self.name_file+'" -dbtype '+str(db_type)
+' -title '+str(title)+' -out '+str(title))

            #os.system('C:/Python31/ncbi-blast-2.2.31+/bin/makeblastdb.exe -in
"{0}" -dbtype nucl -title {1} -out "{2}"'.format(str(self.name_file),str(title),
str(output)))

            self.title = title

            print("\nDatabase created!")

        except:

            print("\nError, try again!")


    def create_fasta_file(self, name_seq, sequence):
        import os
        seq = ""

        try:

            fh = open("fasta_file_"+name_seq+".fasta", "w")

            fh.write(">"+name_seq+"\n")

            for letter in sequence:
                seq += letter
                if len(seq) == 80:
                    fh.write(seq)
                    fh.write("\n")
                    seq = ""
                else:
                    None
            fh.write(seq)

            dirlist = os.listdir(".")
            for name in dirlist:
                if name.find("fasta_file_"+name_seq+".fasta") != -1:
                    #return os.path.abspath(name)

                    self.name = name

                    return name
                else:
                    None

            fh.close()

        except:

            print("\nError")
```

```
    def blastn(self, name_file, path_fasta_file):

        try:
            import os
            os.system("blastn -db "+self.title+" -query "+str(path_fasta_file)+" -
outfmt 10 -out "+str(name_file)+".csv")

            dirlist = os.listdir(".")
            for name in dirlist:
                if name.find(str(name_file)+".csv") != -1:
                    #return os.path.abspath(name)
                    return name
                else:
                    None
        except:
            print("\nError")

    def blastp(self, name_database, evalue, format, num_cpu):

        if format == "xml":
            try:
                aux1 = str(self.name_file)
                file_query = aux1.replace(".fasta","_blastpOutput")

                import os
                os.system("blastp -db "+str(name_database)+" -query "+str
(self.name_file)+" -outfmt 5 -num_threads "+str(num_cpu)+" -evalue "+str(evalue)+"
-out "+str(file_query)+".xml")

                dirlist = os.listdir(".")
                for name in dirlist:
                    if name.find(str(file_query)+".xml") != -1:
                        #return os.path.abspath(name)
                        return name
                    else:
                        None
            except:
                print("\nError!")

        elif format == "csv_default":

            try:
                aux1 = str(self.name_file)
                file_query = aux1.replace(".fasta","_blastpOutput")

                import os
                os.system("blastp -db "+str(name_database)+" -query "+str
(self.name_file)+" -outfmt 10 -num_threads "+str(num_cpu)+" -evalue "+str(evalue)
+" -out "+str(file_query)+".csv")

                dirlist = os.listdir(".")
                for name in dirlist:
                    if name.find(str(file_query)+".csv") != -1:
                        #return os.path.abspath(name)
                        return name
                    else:
                        None
            except:
                print("\nError!")

        elif format == "csv_modified":

            try:
                aux1 = str(self.name_file)
```

```
                file_query = aux1.replace(".fasta","_blastpOutput")

                import os
                os.system('blastp -db '+str(name_database)+' -query '+str
(self.name_file)+' -outfmt "10 qseqid sseqid pident evalue sallseqid" -num_threads
'+str(num_cpu)+' -evalue '+str(evalue)+' -out '+str(file_query)+'.csv')

                dirlist = os.listdir(".")
                for name in dirlist:
                    if name.find(str(file_query)+".csv") != -1:
                        #return os.path.abspath(name)
                        return name
                    else:
                        None
            except:
                print("\nError!")
        else:
            print("\nError: format not found. Try again!")



#RUN WITHOUT CLASS INSTANCE

def random_ID():
    import time
    import random

    aux1 = str(time.clock())
    aux2 = aux1.replace(".","")
    aux3 = str(random.randint(0,100))

    id = aux2 + aux3

    return id


def create_database_blast(name_file, db_type, title):
    """Enter the PATH to FASTA file to create a blast database. DB_TYPE -> 'nucl'
or 'prot'"""

    try:

        import os
        os.system('makeblastdb -in "'+str(name_file)+'" -dbtype '+str(db_type)+' -
title '+str(title)+' -out '+str(title))

        #os.system('C:/Python31/ncbi-blast-2.2.31+/bin/makeblastdb.exe -in "{0}" -
dbtype nucl -title {1} -out "{2}"'.format(str(self.name_file),str(title), str
(output)))

        print("\nDatabase created!")

        return title

    except:

        print("\nError, try again!")

def all_database_in_one(list_name_database, new_database):
    """Enter the list of names of database to create a new database with all"""
    try:
        import os
        all_name_database = " ".join(list_name_database)
        os.system('blastdb_aliastool -dblist "'+str(all_name_database)+'" -dbtype
prot -title "'+str(new_database)+'" -out '+str(new_database))
```

```
        print("\nDatabase with all sequences created!")

    except:

        print("\nError in create to database!")

def BlastN(name_file, path_fasta_file, database):

    try:
        import os
        os.system("blastn -db "+str(database)+" -query "+str(path_fasta_file)+" -
outfmt 10 -out "+str(name_file)+".csv")

        dirlist = os.listdir(".")
        for name in dirlist:
            if name.find(str(name_file)+".csv") != -1:
                #return os.path.abspath(name)
                return name
            else:
                None
    except:
        print("\nError")


def BlastP(fasta_file, name_database, evalue, format, num_cpu):

    if format == "xml":
        try:
            aux1 = str(fasta_file)
            id_random = str(random_ID())
            file_query = aux1.replace(".fasta",str("_blastpOutput"+id_random)) or
aux1.replace(".fa",str("_blastpOutput"+id_random))

            import os
            os.system("blastp -db "+str(name_database)+" -query "+str(fasta_file)
+" -max_target_seqs 1 -outfmt 5 -num_threads "+str(num_cpu)+" -evalue "+str(evalue)
+" -out "+str(file_query)+".xml")

            output = str(file_query)+".xml"
            return output

        except:
            print("\nError!")

    elif format == "csv_default":

        try:
            aux1 = str(fasta_file)
            id_random = random_ID()
            file_query = aux1.replace(".fasta","_blastpOutput"+id_random)

            import os
            os.system("blastp -db "+str(name_database)+" -query "+str(fasta_file)
+" -outfmt 10 -num_threads "+str(num_cpu)+" -evalue "+str(evalue)+" -out "+str
(file_query)+".csv")

            output = str(file_query)+".csv"
            return output

        except:
            print("\nError!")

    elif format == "csv_modified":

        try:
            aux1 = str(fasta_file)
```

```
            id_random = random_ID()
            file_query = aux1.replace(".fasta","_blastpOutput"+id_random)

            import os
            os.system('blastp -db '+str(name_database)+' -query '+str(fasta_file)
+' -outfmt "10 qseqid sseqid pident evalue sallseqid" -num_threads '+str(num_cpu)
+' -evalue '+str(evalue)+' -out '+str(file_query)+'.csv')

            output = str(file_query)+".csv"
            return output

        except:
            print("\nError!")
    else:
        print("\nError: format not found. Try again!")
```

```python
import os
from Bio import SeqIO

def translate_format(seq_translated):
    aa = ""
    sequence = ""
    for letter in seq_translated:
        aa += letter
        if len(aa) == 80:
            sequence += (aa+"\n")
            aa = ""
        else:
            None
    sequence += aa
    return sequence


class Genbank(object):

    def __init__(self, genbank_file):

        records = SeqIO.parse(str(genbank_file), "genbank")

        self.records = records

        self.genbank_file = genbank_file

    def find_locus_tag(self, locus_tag):

        for record in self.records:

            for elem in record.features:
                try:
                    if elem.type == "CDS":
                        locus = elem.qualifiers["locus_tag"]

                        if  locus[0] == str(locus_tag):

                            return elem

                except:
                    return "erro"

    def list_tRNA(self):
        """Return a list of tRNA"""

        list_tRNA = []

        for record in self.records:

            for elem in record.features:
                try:
                    if elem.type == "tRNA":

                        list_tRNA.append(elem)

                except:

                    return "erro"


        return list_tRNA


    def list_rRNA(self):
        """Return a list of rRNA"""
```

```
        list_rRNA = []

        for record in self.records:

            for elem in record.features:
                try:
                    if elem.type == "rRNA":

                        list_rRNA.append(elem)

                except:

                    return "erro"


        return list_rRNA


    def locus_id(self):
        """return a list of sequences identification from a genbank file"""
        list_id = []

        for record in self.records:

            list_id.append(record.id)

        return list_id

    def find_annotation_gene(self, gene_name):
        """Find the gene name in features 'gene' """
        list_features = []
        aux_tuple = ()

        for record in self.records:

            for elem in record.features:

                try:
                    if elem.type == "CDS":

                        gene = elem.qualifiers["gene"]

                        try:
                            if gene[0] == str(gene_name):

                                aux_tuple = record.id, elem
                                list_features.append(aux_tuple)

                        except UnboundLocalError:
                            None
                except:
                    None

        return list_features


    def find_annotation_CDS(self, string):
        """Find the string in features 'product' or 'note'"""

        list_features = []
        aux_tuple = ()

        for record in self.records:
            for elem in record.features:
                try:
```

```
                    if elem.type == "CDS":
                        product = elem.qualifiers
["product"]
                        try:
                            note = elem.qualifiers["note"]
                        except KeyError:
                            None
                        try:
                            if product[0] == str(string) or note[0] == str(string):
                                aux_tuple = record.id, elem
                                list_features.append(aux_tuple)
                        except UnboundLocalError:
                            None
                except:
                    None
        return list_features



    def extract_sequence_from_coordenate(self, sequence_id, start, end):

        for record in self.records:

            if record.id == str(sequence_id):
                sequence_dna = str(record.seq)
                start_ = int(start)
                stop_ = int(end + 1)

                sequence = sequence_dna[start_:stop_]

                return sequence

            else:
                None




    def sequence_locus(self, id_sequence):
        """return the chromosome or contig sequence"""

        for record in self.records:

            try:
                if record.id == str(id_sequence):

                    return record.seq

            except:
                None

    def genome_fasta(self):

        aux_name = str(self.genbank_file)

        try:

            if aux_name.endswith(".gb") is True:
                new_name = aux_name.replace(".gb", ".fasta")
            elif aux_name.endswith(".gbf") is True:
                new_name = aux_name.replace(".gbf", ".fasta")
            elif aux_name.endswith(".gbff") is True:
                new_name = aux_name.replace(".gbff", ".fasta")
            else:
                None
        except:
```

```
            None

        fh = open(str(new_name),"w")

        for record in self.records:

            try:
                fh.write(">" + str(record.id) + "\n")
                dna = str(record.seq)
                seq = translate_format(dna)
                fh.write(seq)
                fh.write("\n")
            except:
                None

        fh.close()



####functions####

def extract_coordenates(seq_feature):

    start = int(seq_feature.location.start)
    end = int(seq_feature.location.end)

    return start, end
```

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 19 21:53:04 2016

@author: thiago
"""
import os

class GenbankFile (object):

    def __init__ (self, genbank_file):

        self.genbank_file = genbank_file

    def read_genome(self):
        file_genome_system = open(str(self.genbank_file), "r")
        self.read = file_genome_system.readlines()
        file_genome_system.close()

    def feature_source(self, name_feature):
        """Sintaxe: var = feature_source(feature, list); feature: 'organism',
'mol_type','strain','db_xref','isolation_source', 'country', 'note',
'host','collection_date','lat_lon','mol_type','organelle' """

        string_control = str("                     /"+str(name_feature)+"=")
        start = 0
        stop = 0
        for line in self.read:

            if line.find("     source          ") != -1:
                start += 1
                break
            else:
                start += 1

        stop = start

        for line in self.read[int(start)+1:]:

            if line.find("..") != -1:
                stop += 1
                break
            else:
                stop += 1

        if name_feature in ['organism',
'mol_type','strain','db_xref','isolation_source', 'country', 'note',
'host','collection_date','lat_lon','mol_type','organelle']:

            for line in self.read[int(start):int(stop)+1]:

                if line.find(string_control) == 0:
                    aux1 = str(line)
                    aux2 = aux1.replace(string_control,"")
                    aux3 = aux2.replace('"','')
                    feature = aux3.replace("\n","")
                    return feature
                else:
                    None
        else:
            None

    def coord_features_translation(self):
        list_features = []
        count = 0
        for line in self.read:
```

```
            aux = line.find("/translation=")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features

    def feature_translation(self, translation_coord):
        aa = ["A", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N", "P",
"Q", "R", "S", "T", "V", "W", "Y"]
        sequence_translated = []
        for line in self.read[int(translation_coord): ]:
            aux = line.find("                     ") #a sequencia de espaco se
encontra sempre antes das seq dos aminoacidos
            if aux != -1:
                for letter in line:
                    if letter in aa:
                        sequence_translated.append(str(letter))
                    else:
                        None
            else:
                break
        sequence_aa = "".join(sequence_translated)
        return sequence_aa

    def coord_feature_gene(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     gene            ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features


    def feature_gene(self, name_feature, gene_coord):
        """features: 'gene', 'locus_tag', 'gene_synonym', 'db_xref'"""

        if self.read[int(gene_coord)].find("     gene            ") == 0:

            list_feature = []
            try:
                string = str("                     /"+name_feature+"=")
            except:
                print("\nFeature not found! Try again!\n")
            try:
                count = int(gene_coord)
                for line in self.read[int(gene_coord + 1): ]: #delimita busca ate
a proxima feature
                    stop = line.find("..") #esta sempre presente na ate a proxima
feature
                    if stop != -1:
                        count += 1
                        break
                    else:
                        count += 1

                for line in self.read[int(gene_coord):count]:
                    aux = line.find(string)
                    if aux != -1:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
```

```
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                    else:
                        None

            except:
                print("\nFeature not found in the file!")

            return list_feature

        else:

            print("\nCoordenate don't have GENE feature")


    def coord_feature_repeat_region(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     repeat_region   ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features

    def feature_repeat_region(self, repeat_region_coord):

        list_feature = []
        try:
            string = str("                     /note=")
        except:
            print("\nFeature not found! Try again!\n")
        try:
            count = int(repeat_region_coord)
            for line in self.read[int(repeat_region_coord + 1): ]: #delimita busca
ate a proxima feature
                stop = line.find("..") #esta sempre presente na ate a proxima
feature
                if stop != -1:
                    count += 1
                    break
                else:
                    count += 1

            for line in self.read[int(repeat_region_coord):count]:
                aux = line.find(string)
                if aux != -1:
                    aux2 = str(line)
                    aux3 = aux2.replace(string,"")
                    feature = aux3.replace("\n", "")
                    list_feature.append(feature)
                else:
                    None

        except:
            print("\nFeature not found in the file!")

        return list_feature

    def coord_feature_CDS(self):

        list_features = []
        count = 0
        for line in self.read:
```

```
            aux = line.find("     CDS             ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        control = str(list_features[0])
        if control.isdigit() is True:
            return list_features
        else:
            return "error"

    def feature_CDS(self, name_feature, CDS_coord):

        """Sintaxe: var = feature_CDS(feature, CDS_coord, list); feature:
'locus_tag',
'inference','note','codon_start','transl_table','product','protein_id','db_xref','translation'
"""

        if self.read[int(CDS_coord)].find("     CDS             ") == 0:

            list_feature = []
            try:
                string = str("                     /"+name_feature+"=")
            except:
                print("\nFeature not found! Try again!\n")
            try:
                count = int(CDS_coord)
                for line in self.read[int(CDS_coord + 1): ]: #delimita busca ate a
proxima feature

                    if line.find("..") != -1 or line.find("ORIGIN") != -1:
                        count += 1
                        break
                    else:
                        count += 1

        # Variaveis de controle

                count_2 = CDS_coord
                count_3 = 0
                number_line_feature = 0
                start_line_feature = CDS_coord -1

        #---------------------

        #Busca se existe mais de uma feature (string) dentro da CDS

                for line in self.read[int(CDS_coord):count]:
                    if line.find(string) == 0:
                        start_line_feature += 1
                        break
                    else:
                        start_line_feature += 1


                for line in self.read[int(CDS_coord):count]:
                    if line.find(string) == 0:
                        number_line_feature += 1
                    else:
                        None

        #----------------------

                if number_line_feature > 1:
```

```
                    for line in self.read[int(start_line_feature):int
(start_line_feature + number_line_feature)]:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                else:

                    for line in self.read[int(CDS_coord):count]:
                        aux = line.find(string)
                        count_2 += 1
                        if aux != -1:
                            aux2 = str(line)
                            aux3 = aux2.replace(string,"")
                            aux7 = aux3.replace('"',"")
                            feature = aux7.replace("\n", "")
                            list_feature.append(feature)

                            if name_feature == "translation":
                                for line in self.read[int(count_2):count]:
                                    aux3 = line.find("..")
                                    if line.find("..") != -1 or line.find
("ORIGIN") != -1:
                                    #if aux3 != -1:
                                        count_3 += 1
                                        break
                                    else:
                                        count_3 += 1
                                count_3 += 1

                            else:
                                for line in self.read[int(count_2):count]:
                                    aux3 = line.find("                     /")
                                    if aux3 != -1:
                                        count_3 += 1
                                        break
                                    else:
                                        count_3 += 1

                            for line in self.read[int(count_2):int(count_2 +
count_3 - 1)]:
                                aux4 = str(line)
                                aux5 = aux4.replace("                     ","")
                                aux6 = aux5.replace('"',"")
                                feature = aux6.replace("\n", "")
                                list_feature.append(feature)

                        else:
                            None


            except:
                print("\nFeature not found in the file!")

            if name_feature == "translation": #concatena a sequencia de aminoacidos
                seq = "".join(list_feature)
                seq_translated = ""
                AA =
["A","B","C","D","E","F","G","H","I","K","L","M","N","P","Q","R","S","T","V","W","X","Y","Z"]

                for letter in seq:
                    if letter in AA:
                        seq_translated += letter

                    else:
                        break
```

```
                aa = ""
                sequence = ""
                for letter in seq_translated:
                    aa += letter
                    if len(aa) == 80:
                        sequence += (aa+"\n")
                        aa = ""
                    else:
                        None
                sequence += aa
                return sequence
            else:

                return list_feature
        else:

            print("\nCoordenate don't have CDS feature")

    def coord_feature_mat_peptide(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     mat_peptide     ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features

    def feature_mat_peptide(self, name_feature, mat_pep_coord):

        if self.read[int(mat_pep_coord)].find("     mat_peptide     ") == 0:

            list_feature = []
            try:
                string = str("                     /"+name_feature+"=")
            except:
                print("\nFeature not found! Try again!\n")
            try:
                count = int(mat_pep_coord)
                for line in self.read[int(mat_pep_coord + 1): ]: #delimita busca
ate a proxima feature
                    stop = line.find("..") #esta sempre presente na ate a proxima
feature
                    if stop != -1:
                        count += 1
                        break
                    else:
                        count += 1

        # Variaveis de controle

                count_2 = mat_pep_coord
                count_3 = 0
                number_line_feature = 0
                start_line_feature = mat_pep_coord -1

        #---------------------

        #Busca se existe mais de uma feature (string) dentro da CDS

                for line in self.read[int(mat_pep_coord):count]:
                    if line.find(string) == 0:
                        start_line_feature += 1
```

```
                        break
                    else:
                        start_line_feature += 1


                for line in self.read[int(mat_pep_coord):count]:
                    if line.find(string) == 0:
                        number_line_feature += 1
                    else:
                        None

        #----------------------

                if number_line_feature > 1:

                    for line in self.read[int(start_line_feature):int
(start_line_feature + number_line_feature)]:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                else:

                    for line in self.read[int(mat_pep_coord):count]:
                        aux = line.find(string)
                        count_2 += 1
                        if aux != -1:
                            aux2 = str(line)
                            aux3 = aux2.replace(string,"")
                            aux7 = aux3.replace('"',"")
                            feature = aux7.replace("\n", "")
                            list_feature.append(feature)

                            for line in self.read[int(count_2):count]:
                                aux3 = line.find("                     /")
                                if aux3 != -1:
                                    count_3 += 1
                                    break
                                else:
                                    count_3 += 1

                            for line in self.read[int(count_2):int(count_2 +
count_3 -1)]:

                                aux4 = str(line)
                                aux5 = aux4.replace("                     ","")
                                aux6 = aux5.replace('"',"")
                                feature = aux6.replace("\n", "")
                                list_feature.append(feature)

                        else:
                            None


            except:
                print("\nFeature not found in the file!")

            return list_feature
        else:

            print("\nCoordenate don't have MAT_PEPTIDE feature")

    def coord_feature_tRNA(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     tRNA            ")
```

```
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features

    def feature_tRNA(self, name_feature, tRNA_coord):

        if self.read[int(tRNA_coord)].find("     tRNA            ") == 0:

            list_feature = []
            try:
                string = str("                     /"+name_feature+"=")
            except:
                print("\nFeature not found! Try again!\n")
            try:
                count = int(tRNA_coord)
                for line in self.read[int(tRNA_coord + 1): ]: #delimita busca ate
a proxima feature
                    stop = line.find("..") #esta sempre presente na ate a proxima
feature
                    if stop != -1:
                        count += 1
                        break
                    else:
                        count += 1

        # Variaveis de controle

                count_2 = tRNA_coord
                count_3 = 0
                number_line_feature = 0
                start_line_feature = tRNA_coord -1

        #---------------------

        #Busca se existe mais de uma feature (string) dentro da CDS

                for line in self.read[int(tRNA_coord):count]:
                    if line.find(string) == 0:
                        start_line_feature += 1
                        break
                    else:
                        start_line_feature += 1


                for line in self.read[int(tRNA_coord):count]:
                    if line.find(string) == 0:
                        number_line_feature += 1
                    else:
                        None

        #----------------------

                if number_line_feature > 1:

                    for line in self.read[int(start_line_feature):int
(start_line_feature + number_line_feature)]:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                else:

                    for line in self.read[int(tRNA_coord):count]:
```

```
                        aux = line.find(string)
                        count_2 += 1
                        if aux != -1:
                            aux2 = str(line)
                            aux3 = aux2.replace(string,"")
                            aux7 = aux3.replace('"',"")
                            feature = aux7.replace("\n", "")
                            list_feature.append(feature)

                            for line in self.read[int(count_2):count]:
                                aux3 = line.find("                     /")
                                if aux3 != -1:
                                    count_3 += 1
                                    break
                                else:
                                    count_3 += 1

                            for line in self.read[int(count_2):int(count_2 +
count_3 -1)]:
                                aux4 = str(line)
                                aux5 = aux4.replace("                     ","")
                                aux6 = aux5.replace('"',"")
                                feature = aux6.replace("\n", "")
                                list_feature.append(feature)

                        else:
                            None


            except:
                print("\nFeature not found in the file!")

            return list_feature
        else:

            print("\nCoordenate don't have tRNA feature")

    def coord_feature_ncRNA(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     ncRNA           ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features


    def feature_ncRNA(self, name_feature, ncRNA_coord):
        """Feature: 'ncRNA_class', 'gene', 'locus_tag', 'gene_synonym', 'product',
'db_xref' """
        if self.read[int(ncRNA_coord)].find("     ncRNA           ") == 0:

            list_feature = []
            string = str("                     /"+name_feature+"=")

            try:
                count = int(ncRNA_coord)
                for line in self.read[int(ncRNA_coord + 1): ]: #delimita busca ate
a proxima feature
                    stop = line.find("..") #esta sempre presente na ate a proxima
feature
                    if stop != -1:
                        count += 1
```

```
                        break
                    else:
                        count += 1

        # Variaveis de controle

                count_2 = ncRNA_coord
                count_3 = 0
                number_line_feature = 0
                start_line_feature = ncRNA_coord -1

        #---------------------

        #Busca se existe mais de uma feature (string) dentro da CDS

                for line in self.read[int(ncRNA_coord):count]:
                    if line.find(string) == 0:
                        start_line_feature += 1
                        break
                    else:
                        start_line_feature += 1


                for line in self.read[int(ncRNA_coord):count]:
                    if line.find(string) == 0:
                        number_line_feature += 1
                    else:
                        None

        #----------------------

                if number_line_feature > 1:

                    for line in self.read[int(start_line_feature):int
(start_line_feature + number_line_feature)]:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                else:

                    for line in self.read[int(ncRNA_coord):count]:
                        aux = line.find(string)
                        count_2 += 1
                        if aux != -1:
                            aux2 = str(line)
                            aux3 = aux2.replace(string,"")
                            aux7 = aux3.replace('"',"")
                            feature = aux7.replace("\n", "")
                            list_feature.append(feature)

                            for line in self.read[int(count_2):count]:
                                aux3 = line.find("                     /")
                                if aux3 != -1:
                                    count_3 += 1
                                    break
                                else:
                                    count_3 += 1

                            for line in self.read[int(count_2):int(count_2 +
count_3 -1)]:
                                aux4 = str(line)
                                aux5 = aux4.replace("                     ","")
                                aux6 = aux5.replace('"',"")
                                feature = aux6.replace("\n", "")
                                list_feature.append(feature)
```

```
                        else:
                            None


            except:
                list_feature.append("\nFeature not found in the file!")

            return list_feature
        else:

            print("\nCoordenate don't have ncRNA feature")

    def coord_feature_rep_origin(self):
        list_features = []
        count = 0
        for line in self.read:
            aux = line.find("     rep_origin      ")
            if aux != -1:
                list_features.append(str(count))
                count += 1
            else:
                count += 1
        return list_features


    def feature_rep_origin(self, name_feature, rep_ori_coord):
        """Feature: 'note', 'db_xref' """
        if self.read[int(rep_ori_coord)].find("     rep_origin      ") == 0:

            list_feature = []
            string = str("                     /"+name_feature+"=")

            try:
                count = int(rep_ori_coord)
                for line in self.read[int(rep_ori_coord + 1): ]: #delimita busca
ate a proxima feature
                    stop = line.find("..") #esta sempre presente na ate a proxima
feature
                    if stop != -1:
                        count += 1
                        break
                    else:
                        count += 1

        # Variaveis de controle

                count_2 = rep_ori_coord
                count_3 = 0
                number_line_feature = 0
                start_line_feature = rep_ori_coord -1

        #---------------------

        #Busca se existe mais de uma feature (string) dentro da CDS

                for line in self.read[int(rep_ori_coord):count]:
                    if line.find(string) == 0:
                        start_line_feature += 1
                        break
                    else:
                        start_line_feature += 1


                for line in self.read[int(rep_ori_coord):count]:
                    if line.find(string) == 0:
```

```
                        number_line_feature += 1
                    else:
                        None

        #----------------------

                if number_line_feature > 1:

                    for line in self.read[int(start_line_feature):int
(start_line_feature + number_line_feature)]:
                        aux2 = str(line)
                        aux3 = aux2.replace(string,"")
                        feature = aux3.replace("\n", "")
                        list_feature.append(feature)
                else:

                    for line in self.read[int(rep_ori_coord):count]:
                        aux = line.find(string)
                        count_2 += 1
                        if aux != -1:
                            aux2 = str(line)
                            aux3 = aux2.replace(string,"")
                            aux7 = aux3.replace('"',"")
                            feature = aux7.replace("\n", "")
                            list_feature.append(feature)

                            for line in self.read[int(count_2):count]:
                                aux3 = line.find("                     /")
                                if aux3 != -1:
                                    count_3 += 1
                                    break
                                else:
                                    count_3 += 1

                            for line in self.read[int(count_2):int(count_2 +
count_3 -1)]:
                                aux4 = str(line)
                                aux5 = aux4.replace("                     ","")
                                aux6 = aux5.replace('"',"")
                                feature = aux6.replace("\n", "")
                                list_feature.append(feature)

                        else:
                            None


            except:
                list_feature.append("\nFeature not found in the file!")

            return list_feature
        else:

            print("\nCoordenate don't have rep_origin feature")

    def fasta_seq(self):

        position = []
        start = 0

        aux = self.genbank_file

        if aux.endswith(".gb") is True:
            extension_file = ".gb"
        elif aux.endswith(".gbf") is True:
            extension_file = ".gbf"
        elif aux.endswith(".gbff") is True:
```

```
        extension_file = ".gbff"
    else:
        None

    name_file = aux.replace(extension_file,".fasta")
    name_contig = aux.replace(extension_file,"")

    fh = open(name_file,"w")

    for line in self.read: #coordenate of ORIGIN

        if line[0:6] == "ORIGIN":
            position.append(start)
        else:
            None

        start += 1

    contig_count = 1

    for coord in position:

        dna = ""

        for line in self.read[coord + 1:]:

            if line[0:2] != "//":
                aux1 = line.replace("\par","")
                aux2 = aux1.replace(" ","")
                aux3 = aux2.replace("\t","")
                elem = aux3.replace("\n","")

                for letter in elem:
                    if letter in ["a","c","t","g","n"]:
                        dna += letter

                    else:
                        None
            else:
                break

        fh.write(">"+name_contig+"_contig"+str(contig_count)+"\n")

        print(">"+name_contig+"_contig"+str(contig_count))

        fasta = ""
        seq = ""
        for nucl in dna:
            seq += nucl
            if len(seq) == 80:
                fasta += seq+"\n"
                fh.write(fasta)
                fasta = ""
                seq = ""
            else:
                None
        fasta += seq+"\n"
        fh.write(fasta)

        contig_count += 1

    fh.close()
```

```python
#WEKA command

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D


def pathVarWeka(path):

    try:

        path_weka = path

        return path_weka

    except:

        print("/nError in set environmental variable Weka. Try again!")

        return 0

def convert_tab_to_csv(tab_file):

    fh = open(str(tab_file), "r")

    read_file = fh.readlines()

    fh.close()

    aux = str(tab_file)
    aux_csv = aux.replace(".tab",".csv")

    fh = open(aux_csv,"w")

    for line in read_file:

        modified_line = line.replace("\t",",")
        line_split = modified_line.split(",")
        new_line = ",".join(line_split[1:])
        fh.write(new_line)

    fh.close()

    return aux_csv

def scatter3dplots(csvfile):

    fh = open(str(csvfile), "r")

    file = fh.readlines()
    data = []

    for elem in file:
        elem_data = elem.split(",")
        data.append(elem_data)

    x = []
    y = []
    id = []
    label = data[0]
```

```
    #array of x
    countx = 0
    for xe in np.arange(1, len(label)+1, 1):
        while countx < len(label):
            x.append(xe)
            countx += 1
        countx = 0

    #array of y
    county = 0
    while county < len(label):
        for ye in np.arange(1, len(label)+1, 1):
            y.append(ye)

        county += 1

    #array of z
    for line in data[1:]:
        for elem in line:
            tetra = elem.replace("\n","")
            id.append(float(tetra))

    #return x, y, id

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, id, c="r", marker="o", s=5, alpha=0.2)
    ax.set_xlabel("genomes")
    ax.set_ylabel("genomes")
    ax.set_zlabel("Tetranucleotides correlation")
    #plt.savefig(str(name_file)+"_3dScatter_TmF_TmR_Penalty.png")
    plt.show()

    #return data


def plots_poly_reg(weka_label, scikit_lean_label, n_clusters, sse1, pol_reg_eq1,
der_eq1, sse2, pol_reg_eq2, der_eq2):

    end = n_clusters[-1]

    xp = np.linspace(2, int(end), 200)

    fig, ax = plt.subplots()

    ax.plot(n_clusters, sse1, ".", c="b", label=str(weka_label))
    ax.plot(xp, pol_reg_eq1(xp), "-", c="b", label="Polinomial Adjust Equation" )
    ax.plot(xp, der_eq1(xp), "--", c="darkcyan", label="First Derivative Equation")

    ax.plot(n_clusters, sse2, ".", c="g", label=str(scikit_lean_label))
    ax.plot(xp, pol_reg_eq2(xp), "-", c="g", label="Polinomial Adjust Equation")
    ax.plot(xp, der_eq2(xp), "--", c="darkgreen", label="First Derivative
Equation")

    plt.legend()
    ax.grid(True)

    plt.show()

def plots_num_reg(weka_label, scikit_lean_label, n_clusters, sse1, num_deriv_sse1,
sse2, num_deriv_sse2):

    fig, ax = plt.subplots()

    ax.plot(n_clusters, sse1, ".", c="b", label=str(weka_label))
    ax.plot(n_clusters[1:], num_deriv_sse1, "--", c="b", label="Numerical First
```

```
Derivative" )

    ax.plot(n_clusters, sse2, ".", c="g", label=str(scikit_lean_label))
    ax.plot(n_clusters[1:], num_deriv_sse2, "--", c="g", label="Numerical First
Derivative")

    plt.legend()
    ax.grid(True)

    plt.show()


def polinomial_regression(list_n_cluster, list_SSE):

    x = np.array(list_n_cluster)

    y = np.array(list_SSE)

    z = np.polyfit(x, y, 10)

    polinomial = np.poly1d(z)

    return polinomial


def first_derivative(polinomial):

    f1 = np.polyder(polinomial, 1)

    return f1


def inflection(derivative):

    inflection_point = np.roots(derivative)

    return inflection_point




def csv_to_arff(path_weka, path_file):

    aux = str(path_file)
    output = aux.replace(".csv",".arff")

    os.system("java -cp "+str(path_weka)+" weka.core.converters.CSVLoader "+str
(path_file)+" > "+str(output))

    return output

def count_atributes_arff_file(arff_file):

    count = 0

    fh = open(str(arff_file), "r")

    for elem in fh:
        if elem.find("@attribute") != -1:
            count += 1
        else:
            None

    return count
```

```
def simpleKmeans(weka, n_clusters, iterations, seed, arff_file):

    aux = str(arff_file)
    aux2 = aux.replace(".arff",".outKmeans")
    output = "N_"+str(n_clusters)+"_I_"+str(iterations)+"_S_"+str(seed)+"_"+str
(aux2)

    os.system('java -cp '+str(weka)+' weka.clusterers.SimpleKMeans -V -N '+str
(n_clusters)+' -A "weka.core.EuclideanDistance -R first-last" -I '+str(iterations)
+' -S '+str(seed)+' -t '+str(arff_file)+' > '+str(output))

    return output, n_clusters, iterations, seed


def extract_SSE(output_kmeans):

    fh = open(str(output_kmeans),"r")

    file_read = fh.readlines()
    fh.close()

    for line in file_read:

        if line.find("Within cluster sum of squared errors:") != -1:
            aux = str(line)
            aux2 = aux.replace("Within cluster sum of squared errors:","")
            SSE = str(aux2)
            break

        else:
            None

    return SSE

def read_output_csv_file(csv_file):

    data = pd.read_csv(str(csv_file),sep=",", names=
["N_clusters","N_iterations","Seed","SSE"])

    list_aux = data["N_clusters"]
    N_clust = [int(elem) for elem in list_aux[1:]]

    list_aux = data["N_iterations"]
    N_iter = [int(elem) for elem in list_aux[1:]]

    list_aux = data["Seed"]
    seed = [elem for elem in list_aux[1:]]

    list_aux = data["SSE"]
    SSE = [float(elem) for elem in list_aux[1:]]

    return N_clust, N_iter, seed, SSE



def statistics_poly_reg(weka_label,file_csv1_weka, scikit_label, file_csv2_scikit):

    n1, i1, s1, sse1 = read_output_csv_file(file_csv1_weka)
    n2, i2, s2, sse2 = read_output_csv_file(file_csv2_scikit)
    #normalization SSE

    max_SSE1 = np.max(sse1)
    norm_sse1 = [float(elem / max_SSE1) for elem in sse1]

    max_SSE2 = np.max(sse2)
    norm_sse2 = [float(elem / max_SSE2) for elem in sse2]
```

```
    equation1 = polinomial_regression(n1, norm_sse1)
    print(equation1)

    der11 = first_derivative(equation1)
    print(der11)

    inflection_point1 = inflection(der11)
    print(inflection_point1)


    equation2 = polinomial_regression(n2, norm_sse2)
    print(equation2)

    der12 = first_derivative(equation2)
    print(der12)

    inflection_point2 = inflection(der12)
    print(inflection_point2)

    plots_poly_reg(str(weka_label), str(scikit_label), n1, norm_sse1, equation1,
der11, norm_sse2, equation2, der12)

def statistics_num_reg(weka_label,file_csv1_weka, scikit_label, file_csv2_scikit):

    n1, i1, s1, sse1 = read_output_csv_file(file_csv1_weka)
    n2, i2, s2, sse2 = read_output_csv_file(file_csv2_scikit)
    #normalization SSE

    max_SSE1 = np.max(sse1)
    norm_sse1 = [float(elem / max_SSE1) for elem in sse1]

    max_SSE2 = np.max(sse2)
    norm_sse2 = [float(elem / max_SSE2) for elem in sse2]

    diff_sse1 = np.diff(norm_sse1)
    diff_n1 = np.diff(n1)

    diff_sse2 = np.diff(norm_sse2)
    diff_n2 = np.diff(n2)

    num_deriv_sse1 = diff_sse1 / diff_n1
    num_deriv_sse2 = diff_sse2 / diff_n2

    plots_num_reg(weka_label, scikit_label, n1, norm_sse1, num_deriv_sse1,
norm_sse2, num_deriv_sse2)

def kmeans_analysis_range(ANI_csv):

    #weka = pathVarWeka("/usr/share/java/weka.jar")
    weka = pathVarWeka("/home/thiagopaim/weka-3-8-0/weka.jar")

    print("\nConverting CSV to ARFF file...\n")
    outArff = csv_to_arff(weka,str(ANI_csv))
    print("\nARFF file %s created!\n" % outArff)

    rand = str(random.randint(0,1000))

    csv_result = open(rand+"_cluster_output_analysis.csv","w")
    csv_result.write("N_clusters,N_iterations,Seed, SSE\n")

    print("\n\nPerforming Cluster analysis...\n")

    max_n_clusters = count_atributes_arff_file(outArff)
```

```
    for n in np.arange(2,(int(max_n_clusters)+1),1): #ate n clusters

        for i in [500,1000,1500,2000,3000,4000,5000,10000]: #iterations

            for s in np.arange(2,100,2):

                outKmeans, N, I, S = simpleKmeans(weka, n, i, s, outArff)
                #Sum of Square Erros
                sse = extract_SSE(outKmeans)

                print("\nNumber of Clusters: "+str(n))
                print("\nNumber of Iterations: %i" % i)
                print("\nNumber of Seed: %i" % s)
                print("SSE: "+str(sse)+"\n")

                csv_result.write(str(N)+","+str(I)+","+str(S)+","+str(sse))

                os.remove(str(outKmeans)) # remove arquivo temporario

    csv_result.close()

    print("\nDone!!")



def kmeans_analysis_weka(ANI_csv, iterations, seed):

    #weka = pathVarWeka("/usr/share/java/weka.jar")
    weka = pathVarWeka("/home/thiagopaim/weka-3-8-0/weka.jar")

    print("\nConverting CSV to ARFF file...\n")
    outArff = csv_to_arff(weka,str(ANI_csv))
    print("\nARFF file %s created!\n" % outArff)

    rand = str(random.randint(0,1000))
    output = rand+"_cluster_output_weka_analysis.csv"

    csv_result = open(str(output),"w")
    csv_result.write("N_clusters,N_iterations,Seed,SSE\n")

    print("\n\nPerforming Cluster analysis...\n")

    max_n_clusters = count_atributes_arff_file(outArff)

    for n in np.arange(2,(int(max_n_clusters)+1),1):

        outKmeans, N, I, S = simpleKmeans(weka, n, iterations, seed, outArff)
        #Sum of Square Erros
        sse = extract_SSE(outKmeans)

        print("\nNumber of Clusters: %i" % n)
        print("\nNumber of Iterations: %i" % iterations)
        print("\nNumber of Seed: %i" % seed)
        print("SSE: "+str(sse)+"\n")

        csv_result.write(str(N)+","+str(I)+","+str(S)+","+str(sse))

        os.remove(str(outKmeans)) # remove arquivo temporario

    csv_result.close()
    print("\nDone!!")

    return output

class KmeansScikitLearn (object):
```

```
    def __init__(self, csv_file):

        self.csv = csv_file

        data = np.loadtxt(str(csv_file), delimiter=",", skiprows=1)

        self.data = data

    def labels(self):

        fh = open(str(self.csv),"r")

        first_line = fh.readline()

        aux = first_line.replace("\n","")

        labels = aux.split(",")

        self.labels = labels #array of sample's name

        fh.close()

        return labels


    def kmeans(self, n_clusters):

        k = KMeans(int(n_clusters), max_iter=1000, init="random").fit(self.data)

        cc = k.cluster_centers_ #Coordinates of cluster centers --> array,
[n_clusters, n_features]
        l = k.labels_ #Labels (from clusters) of each point --> array
        i = k.inertia_ #SSE

        self.labels_cluster = l
        self.SSE = i

        return cc, l, i

def output_kmeans_scikit_learn(array_n_clusters, array_iter, array_seed,
array_SSE):

    fh = open("output_cluster_analysis_scikit_learn.csv","w")

    fh.write("N_clusters,N_iterations,Seed,SSE\n")

    for i in range(len(array_n_clusters)):

        fh.write(str(array_n_clusters[i])+","+str(array_iter[i])+","+str(array_seed
[i])+","+str(array_SSE[i])+"\n")


    fh.close()

    return "output_cluster_analysis_scikit_learn.csv"


def kmeans_analysis_scikit_learn(csv_file):

    analysis = KmeansScikitLearn(str(csv_file))

    labels_csv = analysis.labels()

    max_n_clusters = len(labels_csv)

    array_n_clusters = []
```

```
    array_iter = []
    array_seed = []
    array_SSE = []


    for n_clusters in np.arange(2,max_n_clusters+1,1):

        cc, l, i = analysis.kmeans(n_clusters)

        array_n_clusters.append(n_clusters)
        array_SSE.append(i)
        array_iter.append(1000)
        array_seed.append("-")

    output = output_kmeans_scikit_learn(array_n_clusters, array_iter, array_seed,
array_SSE)

    return output


def return_genomes_clusters(csv_file, n_cluster):

    object = KmeansScikitLearn(str(csv_file))

    labels = object.labels()

    cc, l, i = object.kmeans(n_cluster)

    fh = open("clusters_by_kmeans_scikit_learn_N_"+str(n_cluster)+".csv", "w")

    fh.write("Genome, clusters_number\n")

    for i in range(len(labels)):

        fh.write(str(labels[i])+","+str(l[i])+"\n")
        print(str(labels[i])+" --> cluster number: "+str(l[i])+"\n")

    fh.close()


def main():

    tab_to_csv = input("\nEnter a PATH of tab file (ANI or TETRA output):\t")

    fileName = convert_tab_to_csv(str(tab_to_csv))

    print("\nRunning scripts...wait...\n\n")

    outputScikitLearn = kmeans_analysis_scikit_learn(fileName)

    outputWeka = kmeans_analysis_weka(fileName, 500, 10)

    statistics("Weka",str(outputWeka),"Scikit-Learn",str(outputScikitLearn))

    print("\n\nDone!!")


#call functions
#statistics("96_cluster_output_analysis.csv")
#kmeans_analysis_range("ANIb_percentage_identity.csv")
#kmeans_analysis_weka("ANIb_percentage_identity.csv", 500, 10)
#kmeans_analysis_scikit_learn("ANIb_percentage_identity.csv")
#main()
```

```
import FeaturesGenbankClass as FGC
import os
import random

def show_statistics_for_hypothetical_protein():

    genome = FGC.GenbankFile("genoma.gb") #inicializa a classe
    genome.read_genome() #inicializa a leitura do arquivo

    organism = genome.feature_source("organism")
    print(organism)

    coord_CDS = genome.coord_feature_CDS()
    aux_tuple = ()
    list_hyp_prot = []

    for coordenate in coord_CDS:
        product = str(genome.feature_CDS("product", int(coordenate)))
        locus = str(genome.feature_CDS("locus_tag", int(coordenate)))
        if product.find("hypothetical protein") != -1:
            aux_tuple = str(locus), str(product)
            list_hyp_prot.append(aux_tuple)
        else:
            None
    number_of_hyp_protein = len(list_hyp_prot)
    number_CDS = len(coord_CDS)
    fraction = float(number_of_hyp_protein / number_CDS)

    print ("\nNumber of 'CDS' in "+str(organism)+" genome: "+str(number_CDS))
    print ("\nNumber of 'hypothetical protein' in "+str(organism)+" genome: "+str
(number_of_hyp_protein))
    print("\nFraction: "+str(fraction))

    for elem in list_hyp_prot:
        print("\n"+str(elem[0])+" --> "+str(elem[1])+"\n")
        #print(elem)

def csv_statistics_for_hypothetical_proteins():

    list_dir = os.listdir(".")

    fh = open("statistics_hyp_prot.csv","w")
    fh.write("organism,strain,number_CDS, number_hypothetical_protein, fraction\n")

    for genome_file in list_dir:

        if genome_file.endswith(".gb") is True or genome_file.endswith(".gbf") is
True:

            genome = FGC.GenbankFile(str(genome_file)) #inicializa a classe
            genome.read_genome() #inicializa a leitura do arquivo

            try:
                organism = genome.feature_source("organism")
                strain = genome.feature_source("strain")

                print("\nBacteria: "+str(organism)+" strain "+str(strain))
                print("\n...working...wait!\n")

                coord_CDS = genome.coord_feature_CDS()
                control = str(coord_CDS[0])

                if control.isdigit() is True: #garante que foi encontrada uma
coordenada

                    aux_tuple = ()
```

```
                    list_hyp_prot = []

                    for coordenate in coord_CDS:
                        product = str(genome.feature_CDS("product", int
(coordenate)))
                        locus = str(genome.feature_CDS("locus_tag", int
(coordenate)))

                        if product.find("hypothetical protein") != -1:
                            aux_tuple = str(locus), str(product)
                            list_hyp_prot.append(aux_tuple)
                        else:
                            None

                    number_of_hyp_protein = len(list_hyp_prot)
                    number_CDS = len(coord_CDS)
                    fraction = float(number_of_hyp_protein / number_CDS)

                    print ("\nNumber of 'CDS' in "+str(organism)+" genome: "+str
(number_CDS))
                    print ("\nNumber of 'hypothetical protein' in "+str(organism)
+" genome: "+str(number_of_hyp_protein))
                    print("\nFraction: "+str(fraction)+"\n")
                    print("******************")
                    fh.write(str(organism)+","+str(strain)+","+str(number_CDS)
+","+str(number_of_hyp_protein)+","+str(fraction)+"\n")

                else:
                    print("\nCDS error in "+str(organism))
            except:
                None
        else:
            None


    fh.close()

    print("\n\nDone!!")

def csv_statistics_fasta_for_hypothetical_proteins():

    list_dir = os.listdir(".")

    fh = open("statistics_hyp_prot.csv","w")
    fh.write("organism,strain,number_CDS, number_hypothetical_protein, fraction\n")

    for genome_file in list_dir:

        if genome_file.endswith(".gb") is True or genome_file.endswith(".gbf") is
True:

            genome = FGC.GenbankFile(str(genome_file)) #inicializa a classe
            genome.read_genome() #inicializa a leitura do arquivo

            try:
                organism = genome.feature_source("organism")
                strain = genome.feature_source("strain")

                print("\nBacteria: "+str(organism)+" strain "+str(strain))

                coord_CDS = genome.coord_feature_CDS()
                control = str(coord_CDS[0])

                if control.isdigit() is True: #garante que foi encontrada uma
coordenada
```

```
                    aux_tuple = ()
                    list_hyp_prot = []
                    fh_fasta = open("hyp_protein_seq_"+str(organism)+"_strain_"+str
(strain)+".fasta","w")
                    for coordenate in coord_CDS:
                        product = str(genome.feature_CDS("product", int
(coordenate)))
                        locus = str(genome.feature_CDS("locus_tag", int
(coordenate)))

                        if product.find("hypothetical protein") != -1:
                            aux_tuple = str(locus), str(product)
                            list_hyp_prot.append(aux_tuple)
                            translated = str(genome.feature_CDS("translation", int
(coordenate)))
                            fh_fasta.write(">"+str(aux_tuple)+"\n")
                            fh_fasta.write(str(translated))
                            fh_fasta.write("\n")
                        else:
                            None

                    number_of_hyp_protein = len(list_hyp_prot)
                    number_CDS = len(coord_CDS)
                    fraction = float(number_of_hyp_protein / number_CDS)

                    print ("\nNumber of 'CDS' in "+str(organism)+" genome: "+str
(number_CDS))
                    print ("\nNumber of 'hypothetical protein' in "+str(organism)
+" genome: "+str(number_of_hyp_protein))
                    print("\nFraction: "+str(fraction)+"\n")
                    print("******************")
                    fh.write(str(organism)+","+str(strain)+","+str(number_CDS)
+","+str(number_of_hyp_protein)+","+str(fraction)+"\n")
                    fh_fasta.close()
                else:
                    print("\nCDS error in "+str(organism))
            except:
                None
        else:
            None


    fh.close()

    print("\n\nDone!!")

def information_from_files(path_dir):

    list_dir = os.listdir(str(path_dir))

    fh = open(str(path_dir) + "/organism_information.csv","w")
    fh.write
("organism,strain,isolation_source,country,note,molecule_type,host,collection_date,lat_long,file_
\n")

    for genome_file in list_dir:

        if genome_file.endswith(".gb") is True or genome_file.endswith(".gbf") is
True or genome_file.endswith(".gbff") is True:

            genome = FGC.GenbankFile(str(path_dir) + "/" + str(genome_file))
#inicializa a classe
            genome.read_genome() #inicializa a leitura do arquivo


            aux1 = str(genome.feature_source("organism"))
```

```
            organism = aux1.replace(",","_")

            aux2 = str(genome.feature_source("strain"))
            strain = aux2.replace(",","_")

            aux3 = str(genome.feature_source("isolation_source"))
            isol_source = aux3.replace(",","_")

            aux4 = str(genome.feature_source("country"))
            country = aux4.replace(",","_")

            aux5 = str(genome.feature_source("note"))
            note = aux5.replace(",","_")

            aux6 = str(genome.feature_source("mol_type"))
            mol_type = aux6.replace(",","_")

            aux7 = str(genome.feature_source("host"))
            host = aux7.replace(",","_")

            aux8 = str(genome.feature_source("collection_date"))
            col_date = aux8.replace(",","_")

            aux9 = str(genome.feature_source("lat_lon"))
            latlong = aux9.replace(",","_")

            fh.write(str(organism)+","+str(strain)+","+str(isol_source)+","+str
(country)+","+str(note)+","+str(mol_type)+","+str(host)+","+str(col_date)+","+str
(latlong)+","+str(genome_file)+"\n")

            print("\nBacteria: "+str(organism)+" strain "+str(strain)+" --> File:
"+str(genome_file))

        else:

            None
    fh.close()

def rename_genbank_genome_files():

    list_dir = os.listdir(".")

    for genome_file in list_dir:

        if genome_file.endswith(".gb") is True or genome_file.endswith(".gbf") is
True or genome_file.endswith(".gbff") is True:

            if genome_file.endswith(".gb") is True:
                extension_file = ".gb"
            elif genome_file.endswith(".gbf") is True:
                extension_file = ".gbf"
            elif genome_file.endswith(".gbff") is True:
                extension_file = ".gbff"
            else:
                None

            genome = FGC.GenbankFile(str(genome_file)) #inicializa a classe
            genome.read_genome() #inicializa a leitura do arquivo

            aux1 = genome.feature_source("organism")
            aux2 = aux1.replace(" ","_")
            aux3 = aux2.replace(":","")
            organism = aux3.replace("/","")

            aux1 = genome.feature_source("strain")
```

```
            if aux1 != None:
                aux2 = aux1.replace(" ","_")
                aux3 = aux2.replace(":","")
                strain = aux3.replace("/","")
            else:
                aux1 = genome.feature_source("db_xref")
                aux2 = aux1.replace(":","")
                aux3 = aux2.replace("/","")
                strain = aux3.replace(" ","_")

            new_name = organism+"_strain_"+strain+extension_file

            os.rename(genome_file, new_name)

            print("\nRename files: "+str(genome_file)+" --> "+str(new_name))

        else:
            None

def rename_genbank_organelle_files():

    list_dir = os.listdir(".")

    for genome_file in list_dir:

        if genome_file.endswith(".gb") is True or genome_file.endswith(".gbf") is
True or genome_file.endswith(".gbff") is True:

            if genome_file.endswith(".gb") is True:
                extension_file = ".gb"
            elif genome_file.endswith(".gbf") is True:
                extension_file = ".gbf"
            elif genome_file.endswith(".gbff") is True:
                extension_file = ".gbff"
            else:
                None

            genome = FGC.GenbankFile(str(genome_file)) #inicializa a classe
            genome.read_genome() #inicializa a leitura do arquivo

            aux1 = genome.feature_source("organism")
            aux2 = aux1.replace(" ","_")
            aux3 = aux2.replace(":","")
            organism = aux3.replace("/","")

            aux1 = genome.feature_source("organelle")

            if aux1 != None:
                aux2 = aux1.replace(" ","_")
                aux3 = aux2.replace(":","")
                strain = aux3.replace("/","")
            else:
                aux1 = genome.feature_source("db_xref")
                aux2 = aux1.replace(":","")
                aux3 = aux2.replace("/","")
                strain = aux3.replace(" ","_")

            new_name = organism+"_strain_"+strain+extension_file

            os.rename(genome_file, new_name)

            print("\nRename files: "+str(genome_file)+" --> "+str(new_name))

        else:
            None
```

```
########### script from class FGBio  ###############
import FeaturesGenbankBiopython as FGBio
from FeaturesGenbankBiopython import extract_coordenates

def translate_format(seq_translated):
    aa = ""
    sequence = ""
    for letter in seq_translated:
        aa += letter
        if len(aa) == 80:
            sequence += (aa+"\n")
            aa = ""
        else:
            None
    sequence += aa
    return sequence


def CDS_fasta_file_search_string(name_file_gb, string):

    aux = string.replace(" ","_")
    name_file_fasta = name_file_gb.replace(".gb", str(aux)+".fasta")
    fh = open(str(name_file_fasta),"w")

    #initialize the class
    gb = FGBio.Genbank(str(name_file_gb))

    #find the CDS with annotation 'string' --> return a vector
    cds = gb.find_annotation_CDS(str(string))

    for elem in cds:
        start, stop = FGBio.extract_coordenates(elem[1])
        gb = FGBio.Genbank(str(name_file_gb))
        seq = gb.extract_sequence_from_coordenate(elem[0],start,stop)
        seq_format = translate_format(seq)
        fh.write(">"+str(elem[0])+"\n")
        fh.write(seq_format+"\n")

    fh.close()


#CALL FUNCTIONS
#rename_genbank_organelle_files()
#rename_genbank_genome_files()
#information_from_files()
#show_statistics_for_hypothetical_protein()
#csv_statistics_for_hypothetical_proteins()
#csv_statistics_fasta_for_hypothetical_proteins()
```

```
#PHASTER web
#Methods to upload genomes and save the files


import os
import time
from Bio import SeqIO


def input_PHAST_server(phast_in):

    aux = str(phast_in)
    out = aux.replace(".fasta","_out.json")

    os.system('wget --post-file "' + str(phast_in) + '" "http://phaster.ca/
phaster_api?contigs=1" -O ' + str(out))

    return out

def extract_job_id(out):

    fh = open(str(out),"r")
    file = fh.readlines()
    fh.close()

    aux = file[0].split(",")
    aux2 = aux[0].split(":")

    id = aux2[1].replace('"','')

    return id

def download_data(out, dir):

    #wget http://phaster.ca/submissions/ZZ_ed372851fc

    id = extract_job_id(str(out))

    #control = ""
    os.system("wget http://phaster.ca/submissions/" + str(id) + " -O " +str(dir))

    control = 0
    try:
        fh = open(dir,"r")
        test = fh.readlines()
        control = 0
        fh.close()
    except UnicodeDecodeError:
        control = 1

    while control == 0:
        time.sleep(30)
        print("\nTrying to download file..."+str(id)+" wait\n")
        os.system("wget http://phaster.ca/submissions/" + str(id) + " -O " +str
(dir))

        try:
            fh = open(dir,"r")
            test = fh.readlines()
            control = 0
            fh.close()
        except UnicodeDecodeError:
            control = 1

    os.system("unzip "+str(dir)+" -d "+str(dir)+"_PHASTER")
    os.system("rm "+str(dir))
```

```
import sys, getopt

def main(argv):

    inputfile = ""
    outdir = ""

    try:
        opts, args = getopt.getopt(argv,"hi:o:",["ifile=","outdir="])# o ':'
informa q passa argumento
    except getopt.GetoptError:
        print("PHASTERweb.py -i <inputfile_fasta> -o <out dir>")
        sys.exit(2)

    for opt, arg in opts:
        if opt == "-h":
            print("\n\nPHASTERweb.py \n-i <inputfile_fasta>\n-o <out dir>\n\n")
            sys.exit()
        elif opt in ("-i", "--ifile"):
            inputfile = arg
            print(arg)
        elif opt in ("-o", "--outdir"):
            outdir = arg
            print(arg)

    out = input_PHAST_server(str(inputfile))

    download_data(str(out), str(outdir))

    os.system("rm "+str(out))

if __name__ == "__main__":
    main(sys.argv[1:])
else:
    None
```

```
#Python script for data mining of phast outputs

def translate_format(seq_translated):
    aa = ""
    sequence = ""
    for letter in seq_translated:
        aa += letter
        if len(aa) == 80:
            sequence += (aa+"\n")
            aa = ""
        else:
            None
    sequence += aa

    return sequence

def remove_space_and_NewLine(string):

    aux1 = string.replace(" ","_")
    aux2 = aux1.replace(";","_")
    aux3 = aux2.replace(":","_")
    mod_string = aux3.replace("\n","")

    return mod_string

def extract_information(string):

    information_tuple = ()
    information_list = []

    split = string.split("   ")

    for elem in split:
        if elem != "":
            information_list.append(elem)
        else:
            None
    try:
        coord = information_list[0]
        hit = remove_space_and_NewLine(information_list[1])
        evalue = information_list[2]
        prophage_seq = remove_space_and_NewLine(information_list[3])

        information_tuple = coord, hit, evalue, prophage_seq

    except IndexError:

        None


    return information_tuple


def find_region (file_detail_phast):
    """return line coordenate of region"""

    list_region = []

    fh = open(str(file_detail_phast), "r")
    file = fh.readlines()
    fh.close()

    line_count = -1

    for line in file:
        line_count += 1
```

```
        if line[0:4] == "####":
            list_region.append(line_count)
        else:
            None

    return list_region


def extract_information_from_region (region_start, file_detail_phast):

    list_data = []

    fh = open(str(file_detail_phast), "r")
    file = fh.readlines()
    fh.close()

    for line in file[int(region_start)+1:]:
        if line[0:4] != "####":
            information_tuple = extract_information(line)
            list_data.append(information_tuple)
        else:
            break
    if list_data[-1] == ():
        del list_data[-1]

    return list_data


def tuple_information(organism, region, list_data):

    aux_tuple = ()
    list_tuple = []

    for elem in list_data:

        aux_tuple = organism, region, elem[0], elem[1], elem[2], elem[3]
        list_tuple.append(aux_tuple)


    return list_tuple



def tab_file(output, data_list):

    """the input is list of 'tuple_information'"""

    fh = open(str(output)+"_out.tab", "w")
    fh.write("organism\tregion\tphage_coordenate\tphage_hit\tevalue\tprophage_seq
\n")

    for data in data_list:
        for elem in data:
            for i in range(len(elem)):
                if i != 5:
                    fh.write(str(elem[i])+"\t")
                else:
                    fh.write(str(elem[i])+"\n")

    fh.close()

    return str(output)+"_out.tab"


def fasta_file(path_statistics):
```

```
    fh = open(str(path_statistics),"r")
    file = fh.readlines()
    fh.close()

    new_file = path_statistics.replace(".tab","_phast_seq.fasta")
    fh = open(str(new_file),"w")

    for line in file[1:]:
        split = line.split("\t")
        fh.write(">"+str(split[3])+"\n")
        aux1_seq = split[5].replace("\n","")
        seq = aux1_seq.replace("_","")
        sequence = translate_format(seq)
        fh.write(sequence)
        fh.write("\n")

    fh.close()

    return new_file

def organism_name (path_file):

    aux = str(path_file)

    organism = aux.replace("_detail.txt","")

    return organism


def statistics(path_file, outdir):

    organism = organism_name(str(path_file))

    list_region = find_region(str(path_file))


    region = 0
    list_data = []

    for coord in list_region:
        region += 1
        info = extract_information_from_region(int(coord),str(path_file))

        tuple_info = tuple_information(str(organism),int(region),info)

        list_data.append(tuple_info)

    output = tab_file(str(path_file),list_data)

    fasta_name = fasta_file(str(output))

    os.system("mv "+str(fasta_name)+" "+str(outdir))
    os.system("mv "+str(output)+" "+str(outdir))



import sys, getopt, os

def main(argv):

    inputfile = ""
    outdir = ""


    try:
        opts, args = getopt.getopt(argv,"hi:o:",["ifile=","outdir="])# o ':'
```

```
informa q passa argumento
    except getopt.GetoptError:
        print("\n\nPHASTstatistics.py\n -i <inputfile_fasta>\n -o <out dir>\n\n")
        sys.exit(2)

    for opt, arg in opts:
        if opt == "-h":
            print("\n\nPHASTstatistics.py\n -i <inputfile_fasta>\n -o <out dir>\n
\n")
            sys.exit()
        elif opt in ("-i", "--ifile"):
            inputfile = arg

        elif opt in ("-o", "--outdir"):
            outdir = arg



    os.system("mkdir "+str(outdir))

    statistics(inputfile, outdir)

    print("\n\nInput "+str(inputfile)+" -> Done!\n\n")

if __name__ == "__main__":
    main(sys.argv[1:])
else:
    None
```

```
#VFDB search script

import BlastAnalysis as blast
from Bio import SeqIO
import os


#GENOME DIAGRAM
from Bio.SeqFeature import SeqFeature, FeatureLocation
from Bio.Graphics import GenomeDiagram
from reportlab.lib import colors
from reportlab.lib.units import cm

def draw_genome_chromosome(name_file_genbank, type):
    """Type: 'linear', 'circular' """

    record = SeqIO.parse(str(name_file_genbank), "genbank")

    #inicializa a classe
    gd_diag = GenomeDiagram.Diagram("Chromosome")
    gd_track = gd_diag.new_track(1, name="Annotated Features")

    #gd_features = gd_track.new_set()

    length_draw = 0

    for records in record:

        gd_features = gd_track.new_set()

        length_draw += len(records)

        for elem in records.features:

            try:

                location = str(elem.location)

                if elem.type == "CDS" and location.endswith("(+)"):
                    color = colors.blue
                    gd_features.add_feature(elem, color=color, label=False,
sigil="ARROW", arrowshaft_height=1.0, label_position="middle", label_size=12)

                elif elem.type == "CDS" and location.endswith("(-)"):
                    color = colors.lightblue
                    gd_features.add_feature(elem, color=color, label=False,
sigil="ARROW", arrowshaft_height=1.0, label_position="middle", label_size=12)


                else:
                    None

            except:
                None


    gd_diag.draw(format="circular", start=0, end=int(length_draw),
circle_core=0.7, y=0.12)


    genbank = name_file_genbank.replace(".gb", "")


    gd_diag.write(str(genbank)+"_chromosome_genome_diagram.png","PNG", dpi=600)

##################################
```

```
def database_maker(path_database, name_database, type):
    """type: 'prot' or 'nucl' """

    if type == "prot":

        database = blast.create_database_blast(str(path_database), "prot", str
(name_database))

    elif type == "nucl":

        database = blast.create_database_blast(str(path_database), "nucl", str
(name_database))

    else:

        print("\nError")

    return database


def translate_format(seq_translated):
    aa = ""
    sequence = ""
    for letter in seq_translated:
        aa += letter
        if len(aa) == 80:
            sequence += (aa+"\n")
            aa = ""
        else:
            None
    sequence += aa
    return sequence

def convert_genbank_genome_to_protein_fasta_CDS(path_genbank_file):
    """return path of fasta file and features list - locus, start, end, strand,
contig """

    if path_genbank_file.endswith(".gb") is True:
        fasta_name = path_genbank_file.replace(".gb", "")
    elif path_genbank_file.endswith(".gbf") is True:
        fasta_name = path_genbank_file.replace(".gbf", "")
    elif path_genbank_file.endswith(".gbff") is True:
        fasta_name = path_genbank_file.replace(".gbff", "")
    elif path_genbank_file.endswith(".gbk") is True:
        fasta_name = path_genbank_file.replace(".gbk", "")

    name = str(fasta_name)+"_CDS_prot.fasta"
    fh = open(str(name),"w")


    record = SeqIO.parse(str(path_genbank_file), "genbank")

    features = []

    for records in record:

        for elem in records.features:

            try:
                if elem.type == "CDS":
                    try:
                        locus_tag = elem.qualifiers["locus_tag"]
                    except:
                        locus_tag = elem.qualifiers["db_xref"]
```

```
                    aux1 = elem.location
                    start = str(aux1.start)
                    stop = str(aux1.end)
                    strand = str(aux1.strand)
                    translation = elem.qualifiers["translation"]
                    translate = translate_format(translation[0])

                    tuple_features = str(locus_tag[0]),start,stop,strand, str
(records.id)

                    features.append(tuple_features)

                    fh.write(">"+str(locus_tag[0])+"\n")
                    fh.write(translate+"\n")
                else:
                    None
            except:
                None

    fh.close()

    return name, features

def blast_search(name_file_fasta, database, type_blast):
    """type_blast: 'BLASTP', 'BLASTN' """

    if type_blast == "BLASTP":
        xml_output = blast.BlastP(str(name_file_fasta), database, 1e-3, "xml", 2)

    #elif type_blast == "BLASTN":
        #xml_outpu = blast.BlastN(str(name_file_fasta), database

    return xml_output

def extract_significant_blast_results(xml_output):
    """query, subject, score, bitscore, align_length, evalue,gaps, identities """

    from Bio.Blast import NCBIXML

    aux_tuple = ()
    aux_list = []

    for record in NCBIXML.parse(open(str(xml_output))):#PATH do arquivo .xml
        if record.alignments :
            query_length = record.query_length
            print ("QUERY: %s" % record.query)
            print ("DATABASE: %s" % record.database)

            try:
                for align in record.alignments[:1]: #primeiro resultado
significativo
                    try:
                        subject = align.title

                        for hsp in align.hsps:
                            al_l = hsp.align_length
                            ex = hsp.expect
                            score = hsp.score
                            bitscore = hsp.bits
                            g = hsp.gaps
                            id = hsp.identities
                            ident_fraction = float(int(id)/int(al_l))
                            coverage = float(int(al_l) / int(query_length))
                            aux_tuple =
record.query,subject,query_length,score,bitscore,al_l,ex,g,id,ident_fraction,coverage
                            aux_list.append(aux_tuple)
```

```
                    except:
                        None
            except:
                None
        else:
            None
    #os.remove(str(outputBlast))

    tab_name = xml_output.replace(".xml","_VFDBscript.tab")

    fh = open(str(tab_name),"w")

    fh.write("CDS\tMatch_VF\tquery_length\tscore\tbit_score\talign_length\tevalue
\tgaps\tidentities\tid_fraction\tcoverage\n")

    for elem in aux_list:
        for i in range(len(elem)):

            if i == 10:
                fh.write(str(elem[i])+"\n")

            else:
                fh.write(str(elem[i])+"\t")

    fh.close()

    os.system("rm "+str(xml_output))

    return aux_list, tab_name #importante: a saida eh uma lista

def filter_HSP_percent_identities_coverage(name_file_tab, threshold_id,
threshold_cov):

    hsp_list = []

    fh = open(str(name_file_tab), "r")

    file = fh.readlines()
    fh.close()

    aux_tuple = ()
    try:
        for line in file[1:]:

            a, b, c, d, e, f, g, h, i, j, x = line.split("\t")
            y = x.replace("\n","")

            aux_tuple = a, b, c, d, e, f, g, h, i, j, y

            id_percent = float(j)
            cov_percent = float(y)

            if id_percent >= float(threshold_id) and cov_percent >= float
(threshold_cov):

                hsp_list.append(aux_tuple)
            else:
                None
    except:
        None

    return hsp_list

def create_tab_filter_file(hsp_list, output):

    fh = open(str(output)+".tab","w")
```

```
    fh.write("CDS\tMatch_VF\tquery_length\tscore\tbit_score\talign_length\tevalue
\tgaps\tidentities\tid_fraction\tcoverage\n")

    for elem in hsp_list:
        for i in range(len(elem)):
            if i != 10:
                fh.write(elem[i]+"\t")
            else:
                fh.write(elem[i]+"\n")
    fh.close()

    return str(output)+".tab"

def information_tab_file(features_list, hsp_list, name_file_to_save):

    #str(locus_tag[0]),start,stop,strand, str(records.id)
    #"CDS\tMatch_VF\tquery_length\tscore\tbit_score\talign_length\tevalue\tgaps
\tidentities\tid_fraction\tcoverage\n")
    name_file = str(name_file_to_save)+".tab"

    fh = open(name_file, "w")

    fh.write("CDS\tstart\tend\tstrand\tcontig\tMatch_VF\tquery_length\tscore
\tbit_score\talign_length\tevalue\tgaps\tidentities\tid_fraction\tcoverage\n")

    for blast_features in hsp_list:
        for features in features_list:
            if blast_features[0] == features[0]:

                for i in range(len(features)):
                    fh.write(str(features[i])+"\t")

                for i in range(len(blast_features)):
                    if i == 0:
                        None
                    elif i == 10:
                        fh.write(str(blast_features[i]+"\n"))
                    else:
                        fh.write(str(blast_features[i]+"\t"))
            else:
                None
    fh.close()

    return name_file

def find_database():

    list_dir = os.listdir(".")
    count = 0

    for elem in list_dir:

        if elem.endswith(".phr") is True or elem.endswith(".pin") is True or
elem.endswith(".psq") is True:
            count += 1

        else:
            None

    if count == 3:

        return True

    else:
```

```
        return False


def main_genbank(genbank):

    control = find_database()

    if control == False:

        print("\nBuilding database...wait...\n\n")

        database = database_maker("/home/thiago/VFDBSearch/
VFDB_setB_pro.fas","VFDB_setB_pro","prot")

        print("\nDatabase created: "+str(database)+"\n\n")

    elif control == True:

        print("\nDatabase ok!\n\n")

        database = "VFDB_setB_pro"

    print("\nGENBANK to FASTA CDS features...wait!\n")


    #genbank = "/home/thiago/Documentos/DOUTORADO/BIOINFORMATIC/MvirDBSearch/
MBRL01.1.gbff"

    path_fasta, features = convert_genbank_genome_to_protein_fasta_CDS(genbank)

    print("\nFasta file: "+str(path_fasta)+"\n")

    print("\nSearching the Database...wait!\n")

    xml = blast_search(str(path_fasta), str(database), "BLASTP")

    print("\nSearch done!!")

    print("\nSignificant blast results!!!! wait!!\n\n")

    list_features, name_file = extract_significant_blast_results(str(xml))

    print("\n Searching for identities of match sequence with threshlof of
80%...wait...\n\n")

    hsp_features = filter_HSP_percent_identities(str(name_file),0.8)

    name_file_information = str(genbank.replace(".gbk",""))

    name_file_complete_information = information_tab_file(features,hsp_features,
str(name_file_information)+"_VFDBscript_information_contigs_coverage")

    print("\n\nDone!!")


def main_fasta(path_fasta, outdir, name_out, threshold_id, lenght_coverage):

    control = find_database()

    if control == False:

        print("\nBuilding database...wait...\n\n")

        database = database_maker("/home/thiago/VFDBSearch/
VFDB_setB_pro.fas","VFDB_setB_pro","prot")
```

```
        print("\nDatabase created: "+str(database)+"\n\n")

    elif control == True:

        print("\nDatabase ok!\n\n")

        database = "VFDB_setB_pro"

    print("\nFasta file: "+str(path_fasta)+"\n")

    print("\nSearching the Database...wait!\n")

    xml = blast_search(str(path_fasta), str(database), "BLASTP")

    print("\nSearch done!!")

    print("\nSignificant blast results!!!! wait!!\n\n")

    list_features, name_file = extract_significant_blast_results(str(xml))


    hsp_features = filter_HSP_percent_identities_coverage(str(name_file),float
(threshold_id), float(lenght_coverage))

    name_out_file = create_tab_filter_file(hsp_features, str(name_out))

    os.system("mv "+str(name_out_file)+" "+str(outdir))

    print("\n\nDone!!")



import sys, getopt

def main(argv):

    inputfile = ""
    outdir = ""
    name_out = ""
    threshold_id = ""
    lenght_coverage = ""


    try:
        opts, args = getopt.getopt(argv,"hi:o:n:t:l:",["ifile=","outdir=",
"name_out=","threshold_id=","lenght_coverage="])# o ':' informa q passa argumento
    except getopt.GetoptError:
        print("\n\nVFDBSearchScript.py -i <inputfile_fasta> -o <out dir> -n <name
of output>\n\n")
        sys.exit(2)

    for opt, arg in opts:
        if opt == "-h":
            print("\n\nVFDBSearchScript.py -i <inputfile_fasta> -o <out dir> -n
<name of output>\n\n")
            sys.exit()
        elif opt in ("-i", "--ifile"):
            inputfile = arg
            print(arg)
        elif opt in ("-o", "--outdir"):
            outdir = arg
            print(arg)
        elif opt in ("-n", "--name_out"):
            name_out = arg
```

```
            print(arg)
        elif opt in ("-t", "--threshold_id"):
            threshold_id = arg
            print(arg)
        elif opt in ("-l", "--lenght_coverage"):
            lenght_coverage = arg
            print(arg)


    main_fasta(inputfile, outdir, name_out, float(threshold_id), float
(lenght_coverage))

if __name__ == "__main__":
    main(sys.argv[1:])
else:
    None


#main()
#test2()
#test3()
```

```python
#COG datamining
#importing modules
import os
import BlastAnalysis as blast
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from pylab import *
from Bio.Seq import Seq
from Bio import SeqIO
import matplotlib as mpl

def genbank_to_fasta(genbank_file):

    fasta_name = genbank_file.replace(".gb",".fasta")

    fh = open(str(fasta_name), "w")

    record = SeqIO.read(str(genbank_file),"genbank")

    for elem in record.features:
        try:
            if elem.type == "CDS":
                locus = elem.qualifiers["locus_tag"]
                aa_seq = elem.qualifiers["translation"]
                aa = ""
                sequence = ""
                for letter in aa_seq[0]:
                    aa += letter
                    if len(aa) == 80:
                        sequence += (aa+"\n")
                        aa = ""
                    else:
                        None
                sequence += aa

                fh.write(">"+str(locus[0])+"\n")
                fh.write(sequence+"\n")
        except:
            None

    fh.close()

def nucl_to_protein_fasta(path):

    transcribe = []
    tuple_aux = ()

    for fasta in SeqIO.parse(path, "fasta"): #especificar local e tipo de arquivo
        sequence = fasta.seq

        translated = sequence.translate()
        id = ">"+fasta.id

        tuple_aux = id, translated

        transcribe.append(tuple_aux)

    fh = open(str(path)+"_translated.fasta", "w")

    for elem in transcribe:

        fh.write(str(elem[0])+"\n")

        aa = ""
        sequence = ""
```

```
        for letter in elem[1]:
            aa += letter
            if len(aa) == 80:
                sequence += (aa+"\n")
                aa = ""
            else:
                None
        sequence += aa

        fh.write(sequence)
        fh.write("\n")

    fh.close()

def search_database():

    list_files = os.listdir(".")
    count = 0

    for elem in list_files:
        if elem.endswith(".phr") is True or elem.endswith(".pin") is True or
elem.endswith(".psq") is True:
             count += 1
        else:
            None
    if count == 3:
        return True
    else:
        return False

def extract_gi(string):

    aux1 = string.find("gi|") #procura e retorna coordenada do comeco de 'gi'
    aux2 = string[(aux1 + 3):] #conta mais tres posicoes da string, eh o comeco do
id

    aux3 = int(aux2.find("|")) #busca ate o final do id, onde ha '|'
    protein_id = aux2[:aux3]

    return protein_id

def build_database (path_database):

    """Build database from COG proteins"""

    database_name = blast.create_database_blast(str(path_database), "prot",
"prot2003-2014")

    return database_name


def data_compute_for_pie_chart (output_csv):

    data = pd.read_csv(str(output_csv),sep=",", names=["query", "protein_id",
"cog_id", "cog_annotation", "functional_annotation"])

    list_aux = data["functional_annotation"]
    Fun_An = [elem for elem in list_aux[1:]] #lista com as anotacoes funcionais

    list_aux_remove_dupl = list(set(Fun_An)) #remove elementos duplicados

    name_file = output_csv.replace(".csv","_statistics_functional_annotation.csv")

    fh = open(str(name_file),"w")
    fh.write("functional_annotation,count,fraction\n")
```

```
    count = 0
    aux_tuple = ()
    aux_list = []

    for elem1 in list_aux_remove_dupl: #conta eventos para cada anotacao presente
no arquivo
        for elem2 in Fun_An:
            if elem1 == elem2:
                count += 1
            else:
                None
        aux_tuple = elem1, count
        aux_list.append(aux_tuple)
        count = 0

    sum = 0
    for elem in aux_list: #soma dos eventos - total
        sum += elem[1]

    for elem in aux_list:
        fraction = float(elem[1] / sum)
        fh.write(str(elem[0])+","+str(elem[1])+","+str(fraction)+"\n")

    fh.close()

    return str(name_file)


def pie_chart (statistic_csv):

    from cycler import cycler
    mpl.rcParams['font.size'] = 18

    data = pd.read_csv(str(statistic_csv),sep=",", names=["functional_annotation",
"count", "fraction"])


    list_aux = data["functional_annotation"]
    names = [str(elem) for elem in list_aux[1:]]

    list_aux = data["fraction"]
    percent = [float(elem) for elem in list_aux[1:]]

    for i in range(len(names)):

        print(str(names[i])+" --> "+str(percent[i])+"\n")

    labels = names
    sizes = percent

    figure(1, figsize=(20,10))
    ax = axes([0.1, 0.1, 0.8, 0.8])

    #color_vals = np.arange(0,len(labels))
    color_vals = np.linspace(-1, 1, len(labels))
    #color_vals = range(len(labels))
    my_norm = matplotlib.colors.Normalize(-1, 1)
    my_cmap = matplotlib.cm.get_cmap("Paired")

    pie(sizes, colors=my_cmap(my_norm(color_vals)), pctdistance=1.12,
                    autopct='%1.1f%%', shadow=True, startangle=90)
    axis("equal")
    legend(labels, loc="right", fontsize=18)
    #title("Functional Annotation", bbox={'facecolor':'0.8', 'pad':5})
    #savefig("Annotation_plot.png")
    show()
```

```
class COG (object):

    def __init__(self, path_fasta_file, database, cogCSV, cogNAMES, FUN):

        """ Initialize object. Enter a PATH of FASTA file with sequences that you
want COG analyse"""
        self.path_fasta_file = path_fasta_file
        self.database = database
        self.cogCSV = cogCSV
        self.cogNAMES = cogNAMES
        self.FUN = FUN

    def initilize_cog_files(self):
        """ "cog2003-2014.csv", "cognames2003-2014.tab", "fun2003-2014.tab" """
        try:
            fh1 = open(str(self.cogCSV), "r")
            cog_csv = fh1.readlines()
            self.cog_csv = cog_csv
            fh1.close()

            fh2 = open(str(self.cogNAMES), "r", encoding='windows-1252')
            cognames_tab = fh2.readlines()
            self.cognames_tab = cognames_tab
            fh2.close()

            fh3 = open(str(self.FUN),"r")
            fun_tab = fh3.readlines()
            self.fun_tab = fun_tab
            fh3.close()

        except:
            print("\nError on open database COG files!\n")


    def blastPsearch(self):

        outputBlast = blast.BlastP(self.path_fasta_file, self.database, "1e-10",
"xml", "8") #The output is a name of .xml file

        self.outputBlast = outputBlast

    def extract_significant_blast_results(self):

        from Bio.Blast import NCBIXML

        aux_tuple = ()
        aux_list = []

        for record in NCBIXML.parse(open(str(self.outputBlast))):#PATH do
arquivo .xml
            if record.alignments :

                print ("QUERY: %s" % record.query)
                print ("DATABASE: %s" % record.database)

                try:
                    for align in record.alignments[0:1]: #primeiros resultados
significativos
                        try:
                            subject = align.title

                            for hsp in align.hsps:
                                al_l = hsp.align_length
                                ex = hsp.expect
                                g = hsp.gaps
```

```
                                id = hsp.identities
                                aux_tuple = record.query,subject, al_l, ex, g, id
                                aux_list.append(aux_tuple)
                            except:
                                None
                    except:
                        None
                else:
                    aux_tuple = "missing","missing"
                    aux_list.append(aux_tuple)
            #os.remove(str(outputBlast))

            #return aux_list #importante: a saida eh uma lista
            self.list_search = aux_list

    def extract_gi_protein_id_from_blast(self):

        """Return 'query_name' and 'protein_id' List"""

        QueryProteinIdList = []
        aux_tuple = ()

        fh = open("sequences_match_protein_id"+str(blast.random_ID())+".csv", "w")
        fh.write("query,protein_id\n")

        for elem in self.list_search:
            query = elem[0]
            subject = elem[1]

            if subject == "missing":
                protein_id = "not_found"
            else:
                protein_id = extract_gi(subject)

            aux_tuple = query, protein_id
            QueryProteinIdList.append(aux_tuple)

            fh.write(str(query)+","+str(protein_id)+"\n")
        fh.close()

        self.QueryProteinIdList = QueryProteinIdList

    def search_cog_description(self):

        fh = open("cog_functional_annotation.csv","w")
        fh.write("query, protein_id, cog_id, cog_annotation, functional_annotation
\n")

        aux_tuple = ()
        list_output = []

        for line in self.QueryProteinIdList:

            query = line[0]
            protein_id = line[1]

            if protein_id != "not_found":

                for elem in self.cog_csv:

                    split_line = elem.split(",") # desmembra a linha

                    if split_line[0] == str(protein_id):

                        cog_id = split_line[6] #campo que contem COG ID
```

```
                        for elem in self.cognames_tab:

                            split_line = elem.split("\t") # desmembra a linha
                            if split_line[0] == str(cog_id):

                                functional_class = split_line[1] #campo que contem
COG ID funcao
                                aux = split_line[2] #anotacao
                                aux2 = aux.replace("\n","")
                                cog_annotation = aux2.replace(",","/")

                                for elem in self.fun_tab:

                                    split_line = elem.split("\t")

                                    if elem[0] == functional_class:

                                        aux = split_line[1]
                                        aux2 = aux.replace("\n", "")
                                        description = aux2.replace(",","/")

                                        aux_tuple = query, protein_id, cog_id,
cog_annotation, description
                                        list_output.append(aux_tuple)

                                        fh.write(str(query)+","+str(protein_id)
+","+str(cog_id)+","+str(cog_annotation)+","+str(description)+"\n")
                                    else:
                                        None
                            else:
                                None
                    else:
                        None
            else:
                aux_tuple = "missing","missing","missing","missing","without
functional annotation"
                list_output.append(aux_tuple)

                fh.write("missing,missing,missing,missing,without functional
annotation\n")

            print("\n"+str(query)+" functional annotation finished!")

        fh.close()

        return "cog_functional_annotation.csv"


def figures (path_csv_file):

        print("\n**********Building Figure...wait...**********\n\n")

        path_statistic_file = data_compute_for_pie_chart(path_csv_file)

        pie_chart(path_statistic_file)

        print("\nDone!!")

def main():

    print("******************     COG Annotation   ******************\n\n")

    control = search_database()

    if control is False:
```

```
        print("\nProtein database not found. Building database...wait...\n\n")

        database = build_database("prot2003-2014.fa")

        print("\nDone!\n\n")

    else:
        None

    fasta_file = input("\nEnter a multi-FASTA file to performe COG annotation:\t")

    try:
        cogCSV = "cog2003-2014.csv"
        cogNAMES = "cognames2003-2014.tab"
        FUN = "fun2003-2014.tab"
        database = "prot2003-2014"

        print("\nInitialize Class COG...\n\n")
        cog = COG(str(fasta_file),str(database),str(cogCSV), str(cogNAMES), str
(FUN))

        print("Loading database COG files...wait!\n")
        cog.initilize_cog_files()
        print("Done!\n")

        print("\nSearching FASTA entries in the COG database...wait!\n")
        cog.blastPsearch()
        print("\nDone!\n")

        print("\nExtracting the most significant blast results from each query
fasta sequence...wait!\n")
        cog.extract_significant_blast_results()
        print("\nDone!\n")

        cog.extract_gi_protein_id_from_blast()

        print("Searching Functional Annotation...wait!\n")
        path_csv_file = cog.search_cog_description()
        print("\n\n********* Functional Annotation File created! *********\n")

        print("\n**********Building Figure...wait...**********\n\n")

        path_statistic_file = data_compute_for_pie_chart(path_csv_file)

        pie_chart(path_statistic_file)

        print("\nDone!!")

    except:

        print("\n\nError!!!Try again!\n\n")

def figures (path_csv_file):

        print("\n**********Building Figure...wait...**********\n\n")

        path_statistic_file = data_compute_for_pie_chart(path_csv_file)

        pie_chart(path_statistic_file)

        print("\nDone!!")

def splitPangenomeCOG(path_pangenome, number_of_iter):

    count = 0
```

```
    while count < int(number_of_iter):

        print("\n\nCreating a multifasta file with samples...wait\n")
        file_samples = span.main(str(path_pangenome),1000, count)

        print("\nCOG from "+str(count+1)+" of "+str(number_of_iter)+"\n")
        cog_annotation_fasta(file_samples)

        os.system("rm "+str(file_samples))

        count += 1

    print("\n\nDone")

#main()

#genbank_to_fasta("/home/thiagopaim/COGSoftware/Enterococcus_faecalis_V583.gb")
```

```
#!/bin/sh

export k
aux1=$(date +%d_%m_%Y_%r)
echo $aux1 > temporaryfile.txt
sed -i "s;$;;" temporaryfile.txt
aux2=$(head temporaryfile.txt -n 1)
j="_"
v_aux=$aux2$j
rm temporaryfile.txt


mkdir FASTQ_ASSEMBLER_$v_aux


echo "Processamento de dados..."
#inserir comando de repetição

read -p "Insira nome de arquivo .fastq read1: " sample1
read -p "Insira nome de arquivo .fastq read2: " sample2
read -p "Insira identificador da amostra:" sample_id

cd FASTQ_ASSEMBLER_$v_aux
mkdir Sample_$sample_id
cd Sample_$sample_id

cp -i $sample1 /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id.read1.fastq
cp -i $sample2 /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id.read2.fastq

mkdir FASTQC_without_clipper_trimmer
cd FASTQC_without_clipper_trimmer

fastqc /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id.read1.fastq /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id.read2.fastq -o /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/FASTQC_without_clipper_trimmer

cd .. #volta um diretorio acima

#processamento da read1
mkdir FASTQC_with_clipper_trimmer
echo "processando read1"
fastx_clipper -i $sample_id.read1.fastq -a CTGTCTCTTATACACATCT -M 10 |
fastx_trimmer -f 16 | fastx_trimmer -t 5 -o /home/thiagopaim/FASTQ_ASSEMBLER_
$v_aux/Sample_$sample_id/$sample_id._trim_clip_read1.fastq

fastqc -i /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read1.fastq -o /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/FASTQC_with_clipper_trimmer

#processamento da read2

echo "processando read2"
fastx_clipper -i $sample_id.read2.fastq -a CTGTCTCTTATACACATCT -M 10 |
fastx_trimmer -f 16 | fastx_trimmer -t 5 -o /home/thiagopaim/FASTQ_ASSEMBLER_
$v_aux/Sample_$sample_id/$sample_id._trim_clip_read2.fastq

fastqc -i /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read2.fastq -o /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/FASTQC_with_clipper_trimmer

read -p "Usar algoritmo Kmergenie para predizer melhor faixa de k-mer para
analise? [s] sim        [n] nao" kmer_control
```

```
while [ $kmer_control != "s" -a $kmer_control != "n" ]; do
        echo "Comando não encontrado. Repita operacao..."
        read -p "Usar algoritmo Kmergenie para predizer melhor faixa de k-mer para
analise? [s] sim        [n] nao" kmer_control
done


if [ $kmer_control = "s" ];then
        echo "gerando arquivo fastq para análise do melhor kmer pelo algoritmo
Kmergenie"

        cp -i /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read1.fastq /home/thiagopaim/kmergenie-1.6982/
$sample_id.kmer_sample.fastq
        cat /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read2.fastq >> /home/thiagopaim/kmergenie-1.6982/
$sample_id.kmer_sample.fastq

        echo "processado"

        cd #retorna ao diretorio home

        cd kmergenie-1.6982 #entra diretorio para executar busca de best kmer

        echo "Processando melhor kmer"

        ls -1 > /home/thiagopaim/$sample_id.temp_file_kmer.txt #grava o conteúdo
original do diretorio em arquivo temporario

        ./kmergenie $sample_id.kmer_sample.fastq #executa busca por melhor kmer

        best_k=$(grep 'Predicted best k' histograms_report.html) #le do report do
kmergie onde se encontra o melhor kmer
        echo $best_k > /home/thiagopaim/$sample_id.temp_file_best_kmer.txt #grava
em arquivo
        sed -i 's/<[^>]*>//g' /home/thiagopaim/$sample_id.temp_file_best_kmer.txt
# retira conteudo com < >
        sed -i 's/[^0-9]//g' /home/thiagopaim/$sample_id.temp_file_best_kmer.txt #
e mantem somente numeros, gravando em arquivo
        best_k_read=$(cat /home/thiagopaim/$sample_id.temp_file_best_kmer.txt)
        rm /home/thiagopaim/$sample_id.temp_file_best_kmer.txt #remove arquivo
temporario

        ls -1 >> /home/thiagopaim/$sample_id.temp_file_kmer.txt #grava o conteúdo
novo do diretorio no final do arquivo temporario

        echo Predicted best k to sample $sample_id is $best_k_read >> /home/
thiagopaim/FASTQ_ASSEMBLER_$v_aux/kmergenie_report.txt

        cat /home/thiagopaim/$sample_id.temp_file_kmer.txt | sort | uniq -u |
while read line2; do #remove arquivos gerados pelo kmergenie #le arquivo
temporario, ordena e remove duplicados
                rm /home/thiagopaim/kmergenie-1.6982/$line2

        rm /home/thiagopaim/$sample_id.temp_file_kmer.txt

        done


        cd #volta diretorio home

        #Processamento pelo ABYSS
        echo "Assembler de novo: ABYSS"
        echo "-------------------------"
```

```
        sup=`expr $best_k_read + 5`
        inf=`expr $best_k_read - 5`

        echo "Montagem do genoma para kmer incial de $inf até $sup"

        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        mkdir ABYSS
        cd ABYSS

        cp -i /home/thiagopaim/kmergenie-1.6982/$sample_id.kmer_sample.fastq /home/
thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/ABYSS/
$sample_id.reads_same_file.fastq

        rm /home/thiagopaim/kmergenie-1.6982/$sample_id.kmer_sample.fastq

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                mkdir k$k

                file_name=\'/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/ABYSS/$sample_id.reads_same_file.fastq\'

                abyss-pe -C k$k name=$sample_id in=$file_name k=$k

        done

        cd ..

        #Processamento pelo SPAdes -- somente executa para k ímpares
        echo "Assembler de novo: SPAdes"
        echo "-------------------------"

        mkdir SPAdes

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                cd SPAdes
                mkdir k$k
                cd
                cd SPAdes-3.7.0-Linux
                cd bin

                python spades.py --careful -1 /home/thiagopaim/FASTQ_ASSEMBLER_
$v_aux/Sample_$sample_id/$sample_id._trim_clip_read1.fastq -2 /home/thiagopaim/
FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/$sample_id._trim_clip_read2.fastq -k $k -
o /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/SPAdes/k$k

                cd
                cd FASTQ_ASSEMBLER_$v_aux
                cd Sample_$sample_id
        done

        #Processamento pelo SOAPdenovo
        echo "Assembler de novo: SOAPdenovo"
        echo "-----------------------------"

        mkdir SOAPdenovo

        #copia arquivo de configuração do SOAPdenovo para o diretório de trabalho
        cp -i /home/thiagopaim/SOAPdenovo2-bin-LINUX-generic-r240/
CONFIG_FILE.config /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
SOAPdenovo/CONFIG_$sample_id.config

        #adiciona ao arquivo de configuração os caminhos das arquivos .fastq
        echo q1=/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
```

```
$sample_id._trim_clip_read1.fastq >> /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config
        echo q2=/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read2.fastq >> /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config

        cd
        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        cd SOAPdenovo

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"

                mkdir k$k
                cd k$k

                SOAPdenovo-63mer all -s /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config -K $k -o $sample_id

                cd ..
        done

        #rm /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/SOAPdenovo/
CONFIG_$sample_id.config #deleta o arquivo config do SOAPdenovo

        #Processamento pelo VELVET
        echo "Assembler de novo: VELVET"
        echo "-------------------------"

        cd
        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        mkdir VELVET

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                cd
                cd FASTQ_ASSEMBLER_$v_aux
                cd Sample_$sample_id
                cd VELVET
                mkdir k$k
                cd
                cd velvet_1.2.10

                ./velveth /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/VELVET/k$k $k -shortPaired -fastq -interleaved /home/thiagopaim/
FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/ABYSS/$sample_id.reads_same_file.fastq

                ./velvetg /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/VELVET/k$k -read_trkg yes    -ins_length 300

        done


elif [ $kmer_control = "n" ];then
        cd #volta diretorio home

        read -p "Informe limite inferior de k-mer:" inf
        read -p "Informe limite superior de k-mer:" sup


        #Processamento pelo ABYSS
        echo "Assembler de novo: ABYSS"
        echo "-------------------------"
```

```
        echo "Montagem do genoma para kmer inicial de $inf até $sup"

        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        mkdir ABYSS
        cd ABYSS

        cp -i /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read1.fastq /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/ABYSS/$sample_id.reads_same_file.fastq
        cat /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read2.fastq >> /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/ABYSS/$sample_id.reads_same_file.fastq

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                mkdir k$k

                file_name=\'/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/ABYSS/$sample_id.reads_same_file.fastq\'

                abyss-pe -C k$k name=$sample_id in=$file_name k=$k

        done

        cd ..

        #Processamento pelo SPAdes -- somente executa para k ímpares
        echo "Assembler de novo: SPAdes"
        echo "-------------------------"

        mkdir SPAdes

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                cd SPAdes
                mkdir k$k
                cd
                cd SPAdes-3.7.0-Linux
                cd bin

                python spades.py --careful -1 /home/thiagopaim/FASTQ_ASSEMBLER_
$v_aux/Sample_$sample_id/$sample_id._trim_clip_read1.fastq -2 /home/thiagopaim/
FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/$sample_id._trim_clip_read2.fastq -k $k -
o /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/SPAdes/k$k

                cd
                cd FASTQ_ASSEMBLER_$v_aux
                cd Sample_$sample_id
        done

        #Processamento pelo SOAPdenovo
        echo "Assembler de novo: SOAPdenovo"
        echo "-------------------------"

        mkdir SOAPdenovo

        #copia arquivo de configuração do SOAPdenovo para o diretório de trabalho
        cp -i /home/thiagopaim/SOAPdenovo2-bin-LINUX-generic-r240/
CONFIG_FILE.config /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
SOAPdenovo/CONFIG_$sample_id.config

        #adiciona ao arquivo de configuração os caminhos das arquivos .fastq
        echo q1=/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read1.fastq >> /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config
```

```
        echo q2=/home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/
$sample_id._trim_clip_read2.fastq >> /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config

        cd
        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        cd SOAPdenovo

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"

                mkdir k$k
                cd k$k

                SOAPdenovo-63mer all -s /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/
Sample_$sample_id/SOAPdenovo/CONFIG_$sample_id.config -K $k -o $sample_id

                cd ..
        done

        #rm /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/SOAPdenovo/
CONFIG_$sample_id.config #deleta o arquivo config do SOAPdenovo

        #Processamento pelo VELVET
        echo "Assembler de novo: VELVET"
        echo "-------------------------"

        cd
        cd FASTQ_ASSEMBLER_$v_aux
        cd Sample_$sample_id
        mkdir VELVET

        for k in $(seq $inf $sup); do
                echo "Processando para kmer: $k"
                cd
                cd FASTQ_ASSEMBLER_$v_aux
                cd Sample_$sample_id
                cd VELVET
                mkdir k$k
                cd
                cd velvet_1.2.10

                ./velveth /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/VELVET/k$k $k -shortPaired -fastq -interleaved /home/thiagopaim/
FASTQ_ASSEMBLER_$v_aux/Sample_$sample_id/ABYSS/$sample_id.reads_same_file.fastq

                ./velvetg /home/thiagopaim/FASTQ_ASSEMBLER_$v_aux/Sample_
$sample_id/VELVET/k$k -read_trkg yes    -ins_length 300

        done

fi

echo "Fim da analise"


        #criar pasta QUAST com scaffold de cada assembler para posterior análise
comparativa
```